Applied Intelligence (2023) 53:3183–3206
https://doi.org/10.1007/s10489-022-03412-8

Robust anomaly-based intrusion detection system for in-vehicle network by graph neural network framework

Junchao Xiao[1] · Lin Yang[2] · Fuli Zhong[1] · Hongbo Chen[1] · Xiangxue Li[3]

Accepted: 17 February 2022 / Published online: 24 May 2022

Abstract

With the development of Internet of Vehicles (IoVs) techniques, many emerging technologies and their applications are integrated with IoVs. The application of these new technologies requires vehicles to communicate with external networks frequently, which makes the in-vehicle network more vulnerable to hacker attacks. It is imperative to detect hacker attacks on in-vehicle networks. A control area network graph attention networks (CAN-GAT) model is proposed to implement the anomaly detection of in-vehicle networks, and a graph neural network (GNN) anomaly-based detection framework using graph convolution, graph attention and CAN-GAT network model for in-vehicle network based on CAN bus is presented. In this detection framework, a graph is designed with the traffic on the CAN bus to capture the correlation between the change of the traffic bytes and the state of other traffic bytes effectively and help improve the detection accuracy and efficiency. Compared simulation experiments are conducted to test the proposed model, and the obtained model performance metrics results show that the CAN-GAT-2 model based on two-layer CAN-GAT achieves better performance. In addition, the visualization and quantitative analysis methods are used to explain how can the attention mechanism of CAN-GAT-2 improve the performance, which can help to construct better GNNs in anomaly detection of in-vehicle network. The model performance evaluation results show that CAN-GAT-2 achieved improved accuracy among the compared baseline methods, and has good detection speed performance.

Keywords Anomaly detection · CAN · Graph neural network

1 Introduction

With the rapid development of modern automobiles, a variety of microprocessors with specific functions called Electronic Control Units (ECUs) [1] are embedded in the vehicles to achieve advanced functions such as forward collision warning, vehicular communication and intelligent parking assist. These microprocessors communicate through in-vehicle networks. ECUs connected by bus in the in-vehicle network send vehicle control commands to help complete the driver's operation. Since increasing intelligent technologies are integrated into vehicles, the communication between in-vehicle networks and external networks is becoming more frequent, resulting in more chances that the in-vehicle networks are exposed to attackers. When a vehicle is hacked, the driver's personal privacy will be exposed, and the driver's safety and others are in danger.

Due to the initial design flaws of in-vehicle networks, attackers can invade the in-vehicle network of actual vehicles

✉ Fuli Zhong
zhongfli@mail.sysu.edu.cn

✉ Hongbo Chen
chenhongbo@mail.sysu.edu.cn

Junchao Xiao
xiaojch3@mail2.sysu.edu.cn

Xiangxue Li
xxli@cs.ecnu.edu.cn

[1] School of Systems Science and Engineering, Sun Yat-Sen University, Guangzhou 510006, China

[2] National Key Laboratory of Science and Technology on Information System Security, Institute of System Engineering, Chinese Academy of Military Science, Beijing 100039, China

[3] School of Software Engineering, East China Normal University, Shanghai 200000, China

without too much effort, which attracts widespread concerns on their cybersecurity [2, 3]. An attacker may enter the in-vehicle network by accessing interfaces such as On-Board Diagnostics (OBD), Bluetooth and in-vehicle infotainment systems to implement attacks. If an in-vehicle network lacks a suitable authentication mechanism, once it is connected to an external network, the vulnerability of this vehicle network system is further broadened [4].

In recent years, researches on the vulnerability of in-vehicle networks and robust defense methods were presented [5–7], such as encryption and authentication technologies [8–10], firewall technology [11, 12], anomaly-based intrusion detection systems [13]. Encryption and authentication methods are very effective for protecting in-vehicle networks. However, it is generally necessary to add specific hardware to ensure the efficiency of encryption and decryption, which will result in increased costs for vehicle manufacturing [14, 15]. For the difficulty in defining specifications of potential attacks, firewall technology is tough to widely used in-vehicle networks. Anomaly-based intrusion detection systems for in-vehicle networks can make use of the information provided by intrusion prevention systems [16] and detect abnormal intrusion in time. It is an effective and backward compatible method to protect in-vehicle networks from hacker attacks, and can be applied to resource-constrained network systems to avoid the deficiencies of encryption, authentication and firewall methods.

Controller area network (CAN) bus is often used in-vehicle network for information communication among ECUs. Communication of information between ECUs which cannot effectively guarantee confidentiality, integrity and availability since the CAN bus has no security defense mechanism. Thus the majority of attacks can enter the in-vehicle network through CAN bus. Attackers may send well-designed commands through the CAN bus to achieve the purpose of controlling the vehicle. Detecting malicious behaviors of attackers on CAN bus is the bottom line to protect the in-vehicle network [17]. We study the anomaly detection on in-vehicle networks by using GNN framework [18]. GNN has achieved breakthroughs in natural language processing, recommendation systems and computer vision. It captures the correlation between the change of the traffic bytes and the state of other traffic bytes by learning the samples of the graph structure. Using GNN to capture the representation of CAN bus messages one can explore the relationship between adjacent CAN bus messages and understand the context.

The proposed anomaly intrusion detection system can capture the correlation between the change of the traffic bytes and the state of other traffic bytes to obtain the reciprocal relationship between adjacent CAN bus messages. In [19], physical topology features are used to detect anomalies, and quickly detect the event that attackers insert or disable an ECU. In [20], researchers analyzed the data fields of different CAN traffic and found that some complex changes in traffic bytes are related to each other. This topology structure idea is introduced to process CAN bus messages, so that the representation of CAN bus messages captures the correlation between the change of the traffic bytes and the state of other traffic bytes, contributing to the anomaly detection accuracy. Our method is different from those in [20] and [19]. In [20], the model is of linear structure, which can only identify whether the data is abnormal based on the characteristics of the CAN bus messages before the current ones. The method we propose uses a mesh structure. Based on the features before the current CAN bus messages, the model can judge and refer to several features afterward. In [19], the model identifies attacks that insert and disable ECUs based on the physical topology. However, it is ineffective against other attacks that inject CAN bus messages such as fuzzy and impersonation. Our method implements anomaly detection based on the context of CAN bus messages and can identify different types of attacks.

The contributions of this paper are summarized as follows.

- A new perspective: The normal driving of a vehicle relies on the cooperation of a variety of CAN bus messages, and each CAN bus message does not exist in isolation. CAN bus messages that are adjacent in time are more likely to have a cooperative relationship. We use edges to connect several CAN bus messages that are adjacent in time to find these relationships which is helpful for intrusion detection.
- A graph neural network architecture suitable for in-vehicle network anomaly detection is proposed. Through comparing experiments with a variety of classical GNN layer architectures, one found a variant GNN model based on graph attention mechanism for obtaining improved results than the compared GNN architectures.
- Based on GAT model, the CAN-GAT model is proposed. The presented model improves the feature extraction ability by adding more learnable weights and dot multiplication operations. Experimental results show that CAN-GAT has faster convergence speed and better classification performance compared with GAT model.

The remainder of this paper is organized as follows. In Section 2, related work on anomaly detection for in-vehicle networks is introduced. In Section 3, preliminaries of the work and the preprocessing process of the input data sample are presented. In Section 4, the proposed approach based on GNN used for CAN bus anomaly detection is explained in detail. Section 5 illustrates the performance of the proposed anomaly detection method and how the graph attention

mechanism uses the spatial characteristics of neighboring nodes. Followed by the conclusion in Section 6.

2 Related work

Previous researches on in-vehicle networks anomaly intrusion detection mainly focused on exploring the features of CAN bus messages in the normal state [13, 21, 22], and the correlation between the abnormal data features and the normal data features for detecting abnormalities on CAN bus. Based on the traditional methods of defining features, these works can be divided into three categories: Time sequence features analysis based anomaly intrusion detection, statistics frequency of CAN bus messages transmission based anomaly detection and specification based anomaly intrusion detection methods.

Time sequence features analysis based anomaly detection method usually extracts time features of normal CAN bus messages and compares those of abnormal CAN bus messages against hacker attacks [23–31]. Ref. [24] developed an anomaly detection system using autoencoder and support vector machine to learn the time sequece features of CAN messages. Refs. [20, 25] chose long-short-term-memory (LSTM) neural network to predict the next CAN bus messages and compared the correlation between the predicted next CAN bus messages and the next real CAN bus messages. If the correlation value exceeds a certain threshold, it is considered an anomaly. Ref. [23] proposed an improved LSTM neural network model which can better adapt to anomaly detection for in-vehicle networks. Ref. [26] introduced a multi-task LSTM neural network combining time-interval and data features for CAN bus messages anomaly detection, and designed an edge computing model to improve anomaly detection response time.

Statistics frequency of CAN bus messages transmission based anomaly intrusion detection explores the regular characteristics of CAN messages transmission on the bus. Because CAN bus messages are transmitted on the bus according to specific rules, the frequency feature can be used to defend against most threats [32]. The inserted attack and message drops attack can be effectively detected by these methods. Hoppe et al. [33] analyzed the frequency of CAN bus messages to implement anomaly intrusion detection for in-vehicle networks. Ref. [34] detected anomalous attacks through injecting false CAN messages on the bus to discover different distribution patterns of CAN messages.

Specification based anomaly detection schemes [35–42] explore the potential features of the physical properties of ECUs or CAN bus protocol under normal situations. These features clearly show different specifications in the normal state and the attacked state of the vehicle. Shin *et al*. Ref. [38] used the clock behaviors of ECUs as a specification for anomaly detection on the Internet of Vehicles (IoVs). Zhou *et al*. [36] selected the clock skew and other hardware characteristics in the ECUs as the detection specifications and illustrated the effect of temperature on the accuracy of the detection algorithm. Tariq *et al*. [43] proposed a CAN bus message attack detection framework using specification features and time sequence features learning schemes for anomaly detection.

Yu *et al*. [19] recently exploited network topology that consists of ECUs connected by CAN bus as detection features. Since this the method relies on the changes of the topology of the physical layer; It can only detect attacks those result in the increase and decrease of ECUs amount. Qin *et al*. [20] chose LSTM and different loss functions to predict carefully selected CAN message to achieve the purpose of intrusion detection. They analyzed the byte change range of the data fields of different CAN messages, and found that the change of bytes of some CAN message in the data field is related to other bytes. The traditional anomaly detection method based on the LSTM model is of a linear analysis structure, which can only judge whether the current data is abnormal based on the correlation of the previous data, and it does not consider the correlation after the current data. We introduce the concept of topology to CAN bus messages. This makes the data analysis method a network structure, considering both the previous and subsequent data correlations. This method allows the model to perceive the change of bytes in the data field of adjacent CAN bus messages.

In recent years, GNNs have been widely used in knowledge graphs, social networks and other research fields [44]. Processing these data of topology structure tasks can be better accomplished via GNNs [45, 46]. Based on the superior performance of GNN in these fields, we introduce GNN to process in-vehicle networks traffic data transformed into a topological structure.

3 Preliminaries

3.1 Controller Area Network background

As an essential bus in in-vehicle networks, CAN bus connects different kinds of ECUs in vehicles, Moreover, it supports the internal communication of ECUs. ECU is the control unit of vehicle and drives the activities of the vehicle. Fig. 1 shows a classic in-vehicle network connected by CAN bus in which OBD, infotainment systems and various types of ECUs are connected through gateways.

A general CAN bus message frame shown in Fig. 2, contains the start flag, ECU ID, control field, data field, etc. Especially, ID and data fields are the two most important

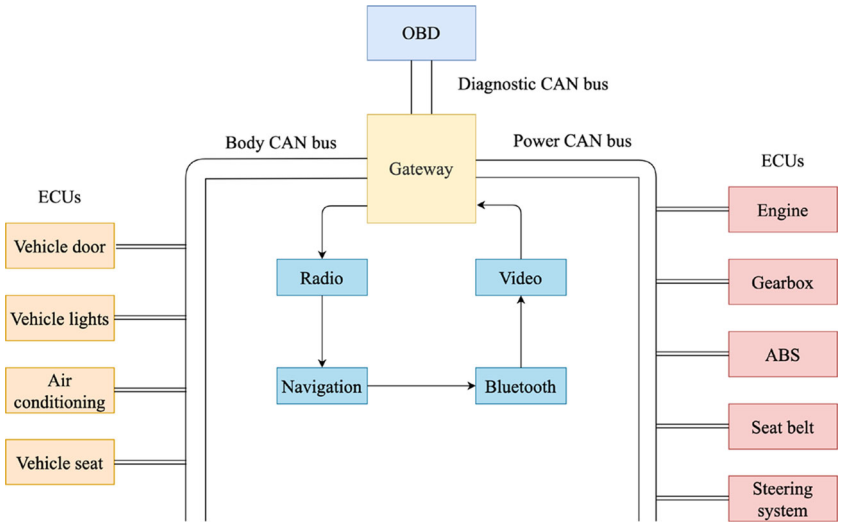


Fig. 1 A classic in-vehicle network model with CAN bus

fields. The ID is used to identify the ECU that sends CAN bus message, and determine the priority of sending CAN bus messages. If multiple ECUs are required to send messages to the CAN bus simultaneously, then the one with the smallest ID will be sent first. Instruction sent by the ECU to control vehicle behavior is filled in the data field of the CAN bus message frame. The Data field has 8 bytes of storage space, and vehicle manufacturers often use different methods to encode the instructions of ECU in the data field.

3.2 Attack scenarios

IoVs and Vehicle to Everything (V2X) technologies allow vehicles to communicate with external networks, which provides a favorable condition for hackers to invade in-vehicle networks. CAN bus protocol in in-vehicle networks does not include designed security mechanism. CAN bus messages are transmitted by broadcast on CAN bus, and any ECU on CAN bus can receive the sent message. If an attacker gains access to the CAN bus, he can pretend to be a legitimate ECU to control the vehicle. Typical attack modes on CAN bus include DoS, fuzzy, impersonation, RPM and GEAR.

DoS [47]: Attacker injects high-priority CAN bus messages periodically to prevent the vehicle from responding to legitimate CAN messages in time. In the example of this paper, the attacker injects the highest priority CAN messages (ID:0000).

Fuzzy [47]: Attacker randomly sends CAN bus messages which causes unexpected behavior of the vehicle. To successfully implement fuzzy attack, the attacker needs to sniff the CAN bus of the target vehicle to explore CAN bus messages that make the vehicle action unexpected. Different from DoS attack that delays legitimate CAN bus messages by occupying CAN bus, fuzzy attack causes abnormal vehicle behavior.

RPM [28]: Attacker periodically (e.g., 10ms) injects some CAN bus messages related to the RPM of the engine, which causes the engine to work abnormally.

GEAR [28]: Attacker periodically (e.g., 10ms) injects some CAN bus messages related to the GEAR of engine, and causes the engine to work abnormally.

Impersonation [38]: Attacker attacks the target ECU and makes the ECU unavailable, and then inserts his ECU, which disguises itself as the unavailable one. The ECU of the attacker can periodically respond to requests from other ECUs on the CAN bus [47]. Impersonation attack does not change the original time interval of CAN messages, so the anomaly detection system based on time interval specification cannot effectively detect this kind of attack. If the attacker does not change the content of CAN messages, the inserted ECU is like a legitimate ECU. However, the attacker may launch other types of attacks at any time.

3.3 Graph structure

To effectively capture the change of bytes of some CAN messages in the data field which is related to other bytes in adjacent CAN messages, the CAN messages are

Fig. 2 A general CAN bus message frame

transformed into graph structure form. Figure 3 shows some examples of the collected CAN bus messages. The timestamp represents the time point when the CAN message arrives at the collection device of the CAN bus message (OBD interface is generally used to connect to the collection device.). "ID" represents the ECU that sent CAN message, and "DLC" represents the length of the data field in CAN bus message frame. The last part represents the data field of CAN bus message frames.

Figure 4 shows the process of converting message frames into a graph structure. All the message frames are sorted according to timestamp from the smallest to the largest. Each message frame is regarded as a node in the graph. Based on the research in [24] which shows that each input sample containing 50 to 100 message frames can help improve the performance of the classifier, each graph contains 50 nodes herein. Considering the increase of the number of nodes will lead to the exponential increase of edges in the graph, and the computational cost, one takes 50 message frames with the smallest timestamp each time to construct a graph G_*, and then uses the $G_1, G_2, ..., G_{\lfloor n/50 \rfloor}$ to serve as an input data sample of the designed GNN framework.

The connection process of the edges of each node in the graph is shown in Algorithm 1. Each input data consists of 50 nodes that used to form the graph, and the output is the connection mode of the edges of each node. One empirically sets each node to be connected to 10 nodes adjacent in a time sequence. Taking nodes v_2 and v_6 as an example of constructing edges connected them, (see Fig. 5). Each node is connected to the 5 adjacent nodes before and after itself, respectively. If there are less than 5 adjacent nodes (for example, v_2), only the existing adjacent nodes are connected. In Fig. 5, the blue arrow represents the directed edge of node v_2, and the red arrow represents the directed edge of node v_6. Each node generates directed edges according to Algorithm 1, and an input sample generates 470 directed edges in total. The generative graph is shown in Fig. 6. The computational complexity exponentially increases with the number of edges for connection. If too many edges are connected, the computational complexity will become very high, while the number of connected edges is too small, the obtained model can not capture the relationship between nodes sufficiently. To explore the relationship between each CAN bus message (node), connecting edges between each CAN bus message is necessary. The adjacent CAN bus message in time has the most significant possibility of a relationship. Therefore, each node is connected to several nodes that are adjacent in time. Considering the computational cost and performance requirements, each node is connected to no more than 10 edges in this paper.

To construct a graph structure of CAN bus messages, each CAN bus message is described as one node, and each graph has $N_v = 50$ nodes. Let $V \in \mathbb{R}^{N_v}$ be a vector:

$$V = \{v_1, v_2, ..., v_{N_v}\} \tag{1}$$

where v_i is the i-th node in the graph, as shown in Fig. 4, the process of transforming CAN message into nodes.

Let a vector $E \in \mathbb{R}^{N_e}$ be the directed edge of the graph structure to represent the interaction relationship between nodes, and each graph has $N_e = 470$ directed edges.

$$E = \{e_1, e_2, ..., e_{N_e}\} \tag{2}$$

where $e_i (i = 1, 2, ..., N_e)$ is the i-th directed edge in the graph. For example, $e_1 = (v_1, v_2)$ represents a directed edge from node v_1 to node v_2. Let G represent a graph composed of N_v nodes and N_e directed edges, denoted as:

$$G = (V, E) \tag{3}$$

Graph G can be constructed with Algorithm 1. The overall graph structure of CAN bus messages is shown in Fig. 6. The N_v nodes represent the position of CAN bus messages in the graph, and the N_e directed edges represent the interaction relationship between CAN bus messages. As an important feature of the CAN bus protocol, the data field of CAN bus messages should be utilized properly. A vector $D \in \mathbb{R}^8$ represents the value of the data field, written as:

$$D = \{d_1, d_2, ..., d_8\} \tag{4}$$

where $d_i (i = 1, 2, ..., 8)$ is the i-th byte in the data field. If the length of the data field is less than 8, it is padded with 0. The byte of the data field of CAN bus messages is stored in the corresponding node as a node feature. Let the vector $\hat{V}_m \in \mathbb{R}^{N_v}$ represent these nodes in a graph, written as:

$$\hat{V}_m = \{\hat{v}_1(D_{m+1}), \hat{v}_2(D_{m+2}), ..., \hat{v}_{N_v}(D_{m+N_v})\} \tag{5}$$

```
Timestamp:        0.001928        ID: 0153    000    DLC: 8    00 80 10 ff 00 ff 40 ce
Timestamp:        0.002167        ID: 0260    000    DLC: 8    05 20 00 30 ff 93 5f 35
Timestamp:        0.002402        ID: 02a0    000    DLC: 8    62 00 60 9d db 0c ba 02
Timestamp:        0.002638        ID: 0370    000    DLC: 8    ff 20 00 80 ff 00 00 ec
Timestamp:        0.002882        ID: 0382    000    DLC: 8    40 fe 0f 00 00 00 00 08
Timestamp:        0.003123        ID: 043f    000    DLC: 8    10 50 60 ff 4b 98 09 00
Timestamp:        0.003370        ID: 0440    000    DLC: 8    ff f0 00 00 ff a0 09 00
Timestamp:        0.003617        ID: 0517    000    DLC: 8    00 00 00 00 00 00 00 00
Timestamp:        0.003852        ID: 0545    000    DLC: 8    d8 10 00 8c 32 00 32 00
Timestamp:        0.004095        ID: 059b    000    DLC: 8    00 00 00 00 00 00 00 c4
Timestamp:        0.005486        ID: 02b0    000    DLC: 5    15 00 00 07 30
Timestamp:        0.005713        ID: 0165    000    DLC: 8    0f e8 7f 00 00 00 00 98
```

Fig. 3 Collected CAN bus messages

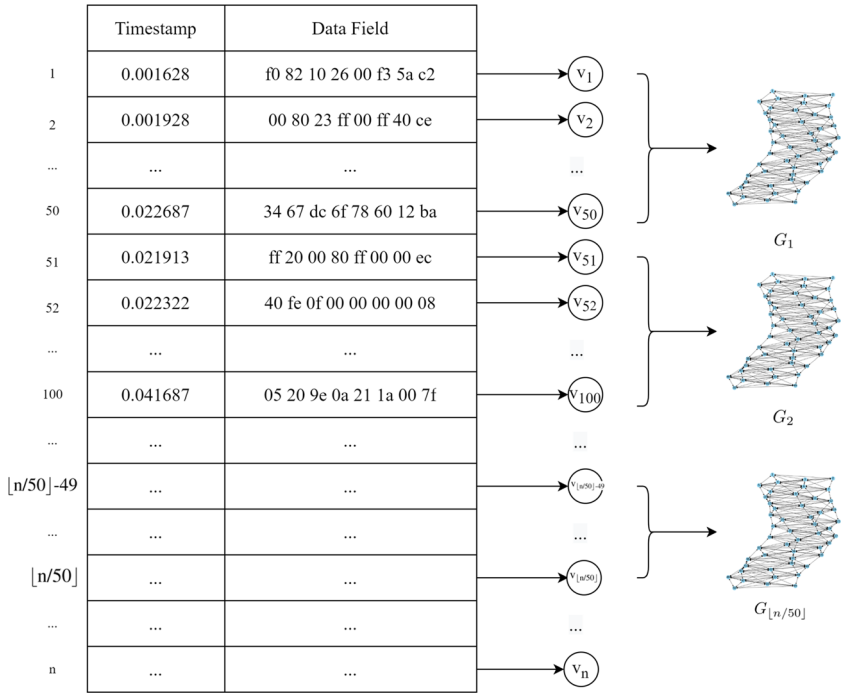


Fig. 4 The process of converting message frames into graph structure

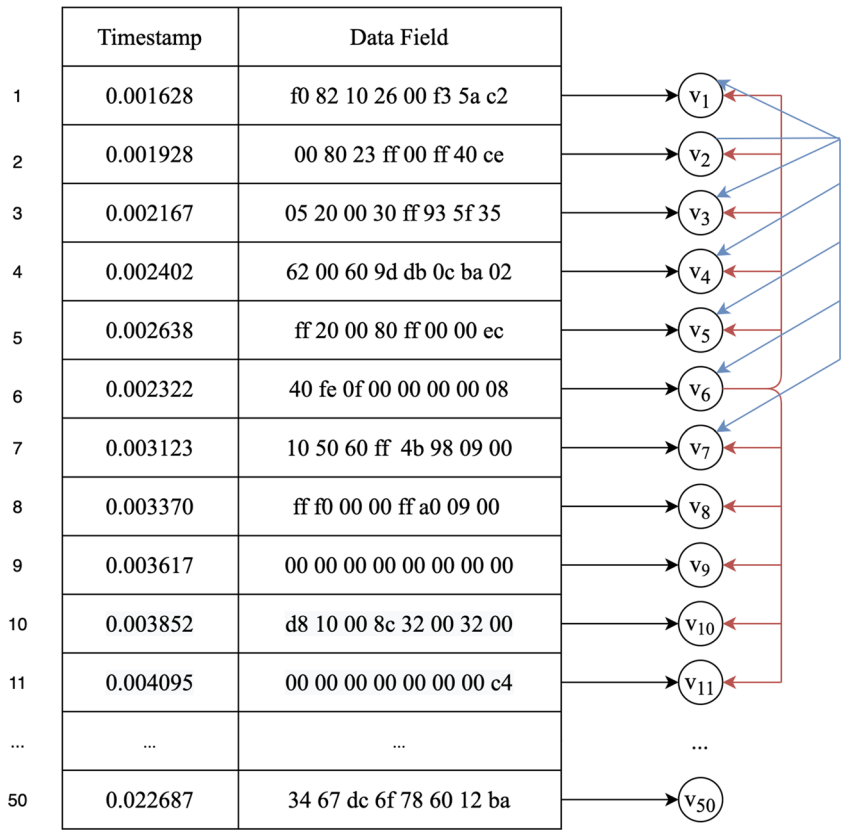


Fig. 5 An example of constructing edges between nodes

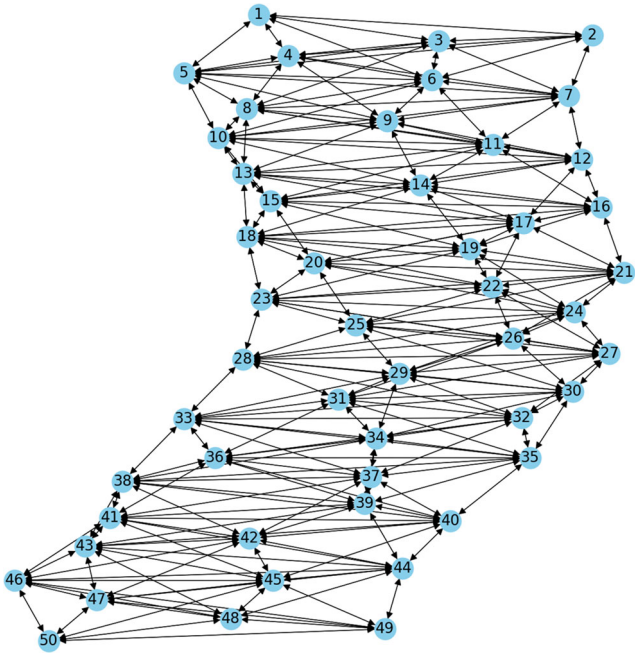


Fig. 6 Constructed graph structure of CAN bus messages

where $m = N_v \times (j-1)$, $j = 1, 2, ..., \lfloor n/N_v \rfloor$, n denotes the number of total CAN bus messages. $\hat{v}_i(D_{m+i})$ $(i = 1, 2, ..., N_v)$ is the i-th node of the $(m/N_v + 1)$-th graph. Node $\hat{v}_i(D_{m+i})$ stores the value of the data field of $(m + i)$-th CAN bus message, which acts as node features (see Fig. 4).

Let $\hat{G}_j$ represent the j-th graph with node features, written as:

$$\hat{G}_j = (\hat{V}_{(N_v \times (j-1))}, E), \tag{6}$$

which is composed of CAN bus messages from the $((N_v \times (j-1)) + 1)$-th one to the $(N_v \times j)$-th one. $\hat{G}_j$ as a sample is input into the GNN. As shown on the right side of Fig. 4, $\hat{G}_1, \hat{G}_2, ..., \hat{G}_{\lfloor n/N_v \rfloor}$ are used as input data samples.

Algorithm 1 Graph construction with CAN bus messages.

Input: V
Output: E

```
k ← 1
for i = 1 to N_v do
    for z = i − 1 to i − 5 do
        if z > 0 then
            E[k] ← e_k(v_i, v_z)
            k ← k + 1
    for z = i + 1 to i + 5 do
        if z ≤ N_v then
            E[k] ← e_k(v_i, v_z)
            k ← k + 1
return E
```

4 Anomaly detection using graph neural network

4.1 Basic approach

GNN is a valuable tool for graph structure data processing to facilitate extracting various data patterns in anomaly detection [18, 48]. It can achieve excellent performance in the classification of various graph structures [49, 50], and has significant potential for the anomaly detection of CAN bus message with graph structure [51]. To explore the relationship between each CAN bus message, we convert the CAN bus message into a graph structure and utilize multi-layer GNN to implement anomaly detection. The designed GNN framework is shown in Fig. 7, where the left part is the training process, and the right part presents the detection process. The training process captures the potential relationship features of CAN bus message with graph structure and forms a trained model. While the detection process determines whether the arriving CAN bus message is abnormal or not based on the trained model. Next, the GNN layer used to search for GNN framework with excellent performance for anomaly detection is discussed, as well as the proposed CAN-GAT network layer.

4.2 Graph convolutional networks

Graph convolutional networks (GCN) [49] is a classic GNN originally used for semi-supervised classification of graph structure data. The multi-layer GCN can be formulated as:

$$H^{(l+1)} = \sigma(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{7}$$

where $\sigma(\cdot)$ is the non-linearity activation function, $\widetilde{A} = A + I_N$ represents the adjacency matrix with added self-connections, I_N represents identity matrix, $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$, and $W^{(l)}$ represents the trainable weight matrix for layer-specific. $H^{(l)} \in \mathbb{R}^{N \times D}$ represents the matrix of activations in the l-th layer.

In [49], (7) is equivalent to

$$h_i^{(l+1)} = \sigma(\sum_{j \in \mathcal{N}(i)} \frac{1}{c_{ij}} W^{(l)} h_j^{(l)}) \tag{8}$$

where $c_{ij} = \sqrt{|\mathcal{N}(i)|}\sqrt{|\mathcal{N}(j)|}$ represents a normalization constant on the graph structure, $\mathcal{N}(i)$ denotes the set of one-hop neighbors of node v_i, $|\mathcal{N}(i)|$ is the number of one-hop neighbors of node v_i, $h_i^{(l)}$ represents a vector of features of node v_i in the l-th GCN layer, $W^{(l)}$ represents the shared trainable weight matrix for node-wise feature transformation, $\sigma(\cdot)$ is the non-linear activation function $ReLU(\cdot) = max(0, \cdot)$. The feature transformation process of multi-layer GCN is shown in Fig. 8, where the node

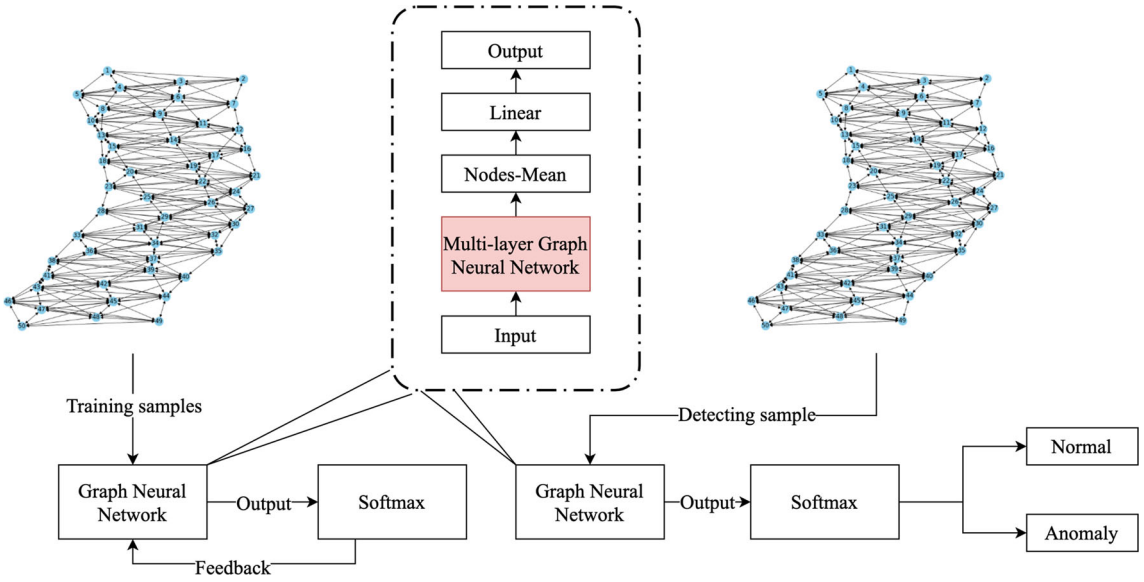


Fig. 7 The GNN framework with CAN bus message of graph structure input

features $(z_1, z_2, ..., z_5)$, are transformed into the node features $(h_1, h_2, ..., h_5)$, through the hidden layer. Y represents the graph label (normal or abnormal). If there is an abnormal CAN bus message in the composed graph, this graph will be marked as abnormal; otherwise, it is marked as normal.

4.3 GraphSAGE

GraphSAGE (GSAGE) [52] is a variant of GCN, its activation function is written as:

$$h_i^{(l+1)} = \sigma\Big(\sum_{j\in\mathcal{N}(i)} \frac{1}{|\mathcal{N}(i)|} W^{(l)} h_j^{(l)}\Big) \tag{9}$$

where $h_i^{(l)}$, $\sigma(\cdot)$, $W^{(l)}$, $|\mathcal{N}(i)|$ and $\mathcal{N}(i)$ have the same meanings as they do in the (8). In GCN, $c_{ij} = \sqrt{|\mathcal{N}(i)|}\sqrt{|\mathcal{N}(j)|}$ is used as the normalization constant, while in GSAGE, the normalization constant is $c_{ij} = |\mathcal{N}(i)|$. In this paper, the constructed graph structure of CAN bus message (Fig. 6) is relatively regular (Most nodes have 10 out-degrees and in-degrees, and each node is only connected to several nodes adjacent in time sequence.), one applies $\frac{1}{|\mathcal{N}(i)|}$ to calculate the average features of each node (CAN bus message).

4.4 Graph attention networks

Graph attention networks (GAT) [50] is an extension of the classic attention model [53]. Its formula is written as:

$$z_i^{(l)} = W^{(l)} h_i^{(l)} \tag{10}$$

$$e_{ij}^{(l)} = LeakyReLU\left(\vec{a}^{(l)^T}(z_i^{(l)} || z_j^{(l)})\right) \tag{11}$$

$$\alpha_{ij}^{(l)} = \frac{\exp(e_{ij}^{(l)})}{\sum_{k\in\mathcal{N}(i)}\exp(e_{ik}^{(l)})} \tag{12}$$

$$h_i^{(l+1)} = \sigma\Big(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(l)} z_j^{(l)}\Big) \tag{13}$$

where $h_i^{(l)}$ represents the node features of the i-th node of l-th layer, $h^{(0)} = \{\bar{h}_1^{(0)}, \bar{h}_2^{(0)}, ..., \bar{h}_N^{(0)}\}$, $\bar{h}_i^{(0)} \in \mathbb{R}^D$, represents

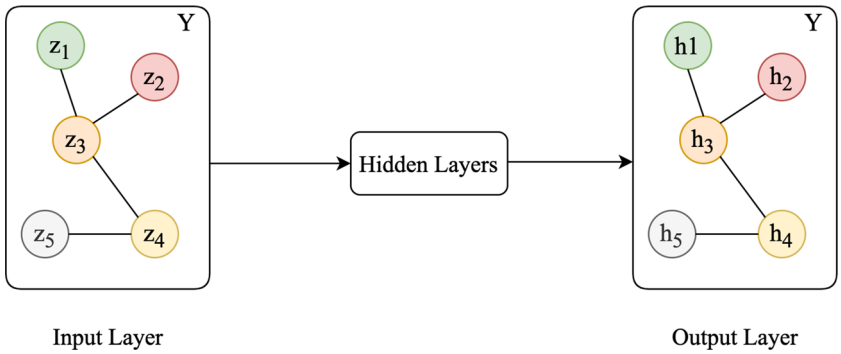


Fig. 8 Feature transformation process of multi-layer GCN

the set of node features in input layer, N represents the number of nodes and D represents the number of features in each node, $W^{(l)}$ is the learnable weight matrix of l-th layer. In (10), $z^{(l)} = \{\vec{z}_1^{(l)}, \vec{z}_2^{(l)}, ..., \vec{z}_N^{(l)}\}$, $\vec{z}_i^{(l)} \in \mathbb{R}^{D'}$, is calculated through linear transformation. In (11), the attention score of each pair of nodes connected by edges is calculated, $||$ represents the concatenation of two node features, $\vec{a}^{(l)^T}$ is a learnable weight vector, and $(\cdot)^T$ represents the transposition, and $LeakyReLU(\cdot)$ is a non-linear activation function,

$$LeakyReLU(x_i) = \begin{cases} x_i, & x_i \geq 0, \\ \frac{x_i}{b_i}, & x_i < 0. \end{cases} \tag{14}$$

To facilitate the comparison of different nodes, (12) is used to normalize the attention score on different edges. The normalized attention score is stored on each node's incoming edges and acts as the edge weight. (13) is used to calculate the linear combination of corresponding features with the normalized attention score as the new node feature to the next layer. $\sigma(\cdot)$ is a non-linear activation function.

The calculation process of the GAT is shown in Fig. 9, where the left hand side of softmax represents the calculation process of (10) and (11), and the right hand side represents the calculation process of (12).

4.5 The proposed Control Area Network-Graph Attention Networks (CAN-GAT)

The proposed Control Area Network-Graph Attention Networks model is written as:

$$z_i^{(l)} = ReLU(W^{(l)} h_i^{(l)}) \tag{15}$$

$$e_{ij}^{(l)} = LeakyReLU\left((\vec{V}_1^{(l)^T} z_i^{(l)}) \circ (\vec{V}_2^{(l)^T} z_j^{(l)})\right) \tag{16}$$

$$\alpha_{ij}^{(l)} = \frac{\exp(e_{ij}^{(l)})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik}^{(l)})} \tag{17}$$

$$h_i^{(l+1)} = \sigma\Big(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(l)} z_j^{(l)}\Big) \tag{18}$$

Compared with the GAT, (15) applies the $ReLU(\cdot)$ non-linear activation function to improve the model's ability to capture abstract features. In (16), $\vec{V}_1^{(\cdot)^T}$ and $\vec{V}_2^{(\cdot)^T}$ are the learnable weight vectors. $z_i^{(\cdot)}$ and $z_j^{(\cdot)}$ respectively represent the obtained node features. These two nodes are connected by the same edges. "$\circ$" represents the dot-product operation. In (16), the two node features are linearly transformed first, then are calculated with the dot-product operation, and applied to the $LeakyReLU(\cdot)$ activation function finally. (17) and (18) have the same meanings as they do in the GAT. The non-linear activation function is added to (15) to increase the nonlinearity of the model so that the model can be arbitrarily approximated to any non-linear function, and speed up the convergence speed of the model. In the same way, compared with (11), (16) has one more learnable weight vector, and uses an improved calculation method. It improves the model's ability to process non-linear features to improve the model's convergence speed and classification accuracy (see Fig. 16 and Table 9).

4.6 Nodes-Mean

The node features processed by GNN are applied to calculate the average values for anomaly detection, calculated as

$$ave\_h = \frac{\sum_{j=1}^{N} h_j}{N} \tag{19}$$

where N represents the number of nodes in the graph, $h_j \in \mathbb{R}^F$ is j-th node feature, F represents the number of features of the node, $ave\_h \in \mathbb{R}^F$ is the average values of all node features in the graph. Figure 10 shows the process of calculating average values of node features for a graph with three nodes.

4.7 Graph neural network framework for anomaly detection

CAN bus messages are converted into a graph with a topological structure to capture the relationship between adjacent CAN bus messages in time. Furthermore, GNN

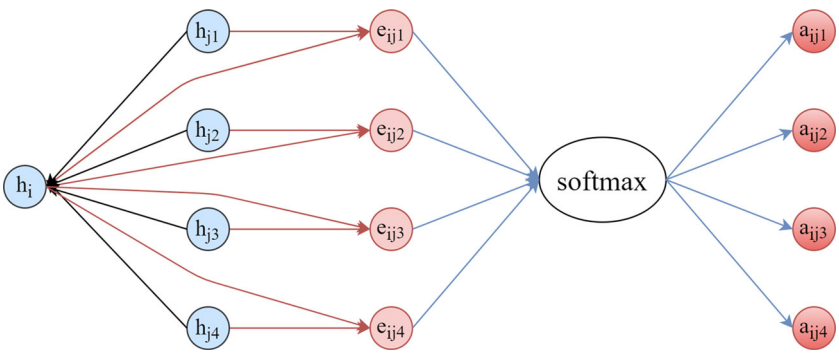


Fig. 9 The calculation process of GAT

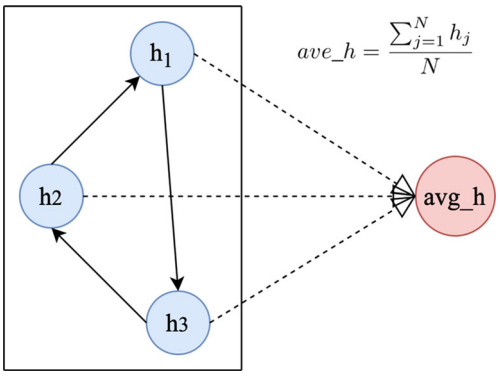


Fig. 10 The process of node-mean calculation

is more promotive to processing these graph data. So the data converted into graphs is input into GNN framework for further feature extraction. The GNN framework is a binary classifier shown in Fig. 7. For the multi-layer GNN layers, the input data ($\hat{G}_j$, $j = (1, 2, ..., \lfloor n/N_v \rfloor)$) is the graph with node features from n CAN bus messages in chronological order. The layers are expressed as:

$$O_a = \{P_1(\hat{G}_j), \forall j = 1, 2, ..., \lfloor n/N_v \rfloor\} \tag{20}$$

where O_a represents the output of the multi-layer GNN layers. The node features of this graph structure are the data field values in the CAN bus messages that construct the graph structure. P_1 represents the process of the multi-layer GNN layers. The output of the multi-layer GNN layers (O_a) is directly used to input subsequent Nodes-Mean layers.

The features processed by the multi-layer GNN layers have a topological structure. To reduce the feature dimension and remove the topological structure, Nodes-Mean is used to calculate the arithmetic average of each node feature based on the topological structure. The features processed by the multi-layer GNN layers are aggregated through the Nodes-Mean calculation (19),

$$O_b = \{P_2(O_a)\} \tag{21}$$

where $O_b = ave\_h$ denotes the output of (19), O_a is the multi-layer GNN layers. P_2 denotes the aggregation process of node features.

The O_b output by Nodes-Mean needs to reduce the feature dimension further to ensure that the subsequent softmax function can correctly identify the abnormal data, so a linear transformation is used to reduce the dimension of each node feature to 2. By using a linear transformation to change the feature dimension of the output O_b, we obtain O_c as:

$$O_c = O_b A^T + b \tag{22}$$

where $O_c \in \mathbb{R}^2$ represents the output of linear transformation, A^T is a learnable weight vector, and b is a learnable bias. Through using the linear transformation of (22), the feature dimension is reduced to 2 (i.e., normal and anomaly) to perform anomaly detection. The O_c is fed into softmax function to achieve the prediction results,

$$y_i' = \frac{\exp(O_{c_i})}{\sum_{j=1}^{k} \exp(O_{c_j})} \tag{23}$$

where O_{c_*} is the output of linear transformation (22), (23) transforms the feature values of O_{c_i} into probability value, y_i' ($\forall i = (1, 2)$) and $k = 2$ represent the probability that the prediction result is normal or abnormal, and the total number of categories, respectively.

The loss function to evaluate the error between the predicted category and the real category is defined as:

$$L_e(w, b) = -\frac{1}{n} \sum_{i=1}^{n} y_i \log(y_i'(w, b)) \tag{24}$$

where y_i represents the real category and y_i' represents the prediction one, w and b respectively represent the weights (i.e., all the learnable parameters) and bias of the GNN framework in hidden layers.

As batches of groups of input train the GNN framework, the actual summation effect is the loss of a batch L_b,

$$L_b(w, b, u) = \sum_{i=1}^{u} L_e^{(i)}(w, b) \tag{25}$$

where u represents the number of a batch of groups of input. In this paper, $u = 100$ represents that a batch is formed with 100 graph structure data composed by CAN bus messages.

The $L_b(w, b, u)$ is the final selected loss function and optimization object in the training phase of the proposed framework. The optimization target is to minimize the error calculated by the loss function. A suitable set of parameters P^* must be found to minimize the batch loss, written as:

$$P^* = \arg \min_{\{w,b\} \in \Omega} L_b(w, b, u) \tag{26}$$

The Adam optimization algorithm [54] is used to search the suitable parameter set P^*. The GNN framework is capable of implementing anomaly detection for CAN bus messages with the obtained parameter set P^*.

4.8 Construction of graph neural network framework and experimental setting

4.8.1 Graph neural network framework construction

To search the GNN framework of good performance, one constructs combined GNN layers and explores their performance via using simulation experiments. The constructed multi-layer GNN layers of Fig. 7 are listed in Table 1. For the convenience of description, we take the models in Fig. 11 as an example to show the architecture of CAN-GAT-2 and GSAGE-CAN-GAT models, respectively. In the

Table 1 Multi-layer GNN layers of Fig.7

Model	Layer 1	Layer 2	Layer 3	Layer 4
CAN-GAT-2	CAN-GAT	CAN-GAT	-	-
GAT-2	GAT	GAT	-	-
GCN-2	GCN	GCN	-	-
GSAGE-2	GSAGE	GSAGE	-	-
GCN-CAN-GAT	GCN	GCN	CAN-GAT	CAN-GAT
GCN-GAT	GCN	GCN	GAT	GAT
GSAGE-CAN-GAT	GSAGE	GSAGE	CAN-GAT	CAN-GAT
GSAGE-GAT	GSAGE	GSAGE	GAT	GAT

GNN framework, the input feature dimension of the multi-layer GNN layers of the first layer is 8, because the data field of the CAN bus message has 8 bytes which are input into the GNN as node features. GCN and GraphSAGE layer update the node features according to the input graph structure. CAN-GAT and GAT layers are used to explore the change of bytes of CAN messages in the data field related to other bytes in adjacent CAN message and update the node features according to the magnitude of the interaction relationship between CAN bus messages. Each GNN layer uses the same two layers to improve the model's ability to capture abstract features [49]. The graph structure needs to be removed to implement linear transformation, so we use Nodes-Mean to aggregate node features (this process does not involve the transformation of feature dimensions). Softmax function is used to calculate and output the category probability (normal or abnormal). The output dimension of the linear transformation is defined as 2. Each GNN layer has 256 neural computing units. The number of training iterations of each model is set as 200, and the training phase is performed using the Adam method with a learning rate of 0.001.

4.8.2 CAN bus message dataset

The applied experimental data is downloaded from the public datasets at Hacking and Countermeasure Research Lab made available for research purposes.[1] The dataset includes a normal CAN bus message dataset and attack dataset comprising fuzzy, RPM, GEAR, DoS, impersonation attacks. These datasets are collected from actual vehicles through the OBD-II interface of the vehicle. In this paper, to ensure the balance of different types of datasets, we intercept the first part of the continuous time period data of the downloaded datasets as training data and the other part as test data. The size of the intercepted dataset is shown in Table 2.

[1]The CAN Bus Message Dataset used herein includes two public datasets. Their URLs are http://ocslab.hksecurity.net/Dataset/CAN-intrusion-dataset, and http://ocslab.hksecurity.net/Datasets/CAN-intrusion-dataset, respectively.

5 Performance evaluation

This section evaluates each GNN framework's performance (shown in Table 1) for anomaly detection on CAN bus messages. And the best one in Table 1 and its internal principles of implementation of anomaly detection is explored, followed by the anlysis of the time cost of anomaly detection systems.

5.1 Effectiveness evaluation of graph neural networks

To evaluate the anomaly detection performance of each GNN model, the classic classification metrics method is applied. The accuracy, recall, precision, F1 score are calculated by collecting the number of true negatives (TN), true positives (TP), false negatives (FN), and false positives (FP).

$$precision = \frac{TP}{TP + FP}, \tag{27}$$

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}, \tag{28}$$

$$recall = \frac{TP}{TP + FN}, \tag{29}$$

$$F1 = \frac{precision \times recall}{precision + recall} \times 2. \tag{30}$$

The receiver operating characteristic (ROC) [55, 56] curve is used to evaluate the ability of each model to detect certain types of attacks. The ROC curve draws the relationship between true positive rate (TPR) and false positive rate (FPR), the area under the ROC curve indicates the performance of the model. An ideal anomaly detection system should have higher TPR, area and lower FPR.

$$TPR = \frac{TP}{TP + FN}, \tag{31}$$

$$FPR = \frac{FP}{FP + TN}. \tag{32}$$

Table 3 shows the performance of the compared GNN models, from which one can see that the performance metrics

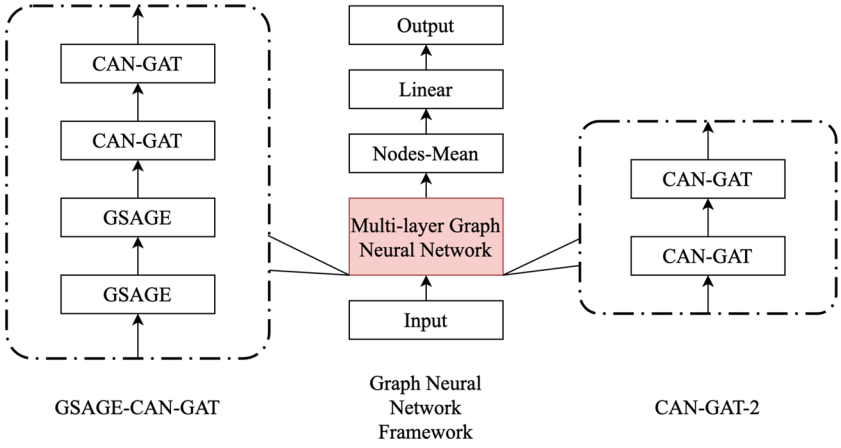


Fig. 11 Architecture of CAN-GAT-2 and GSAGE-CAN-GAT

of each GNN model can reach about 0.86 or higher. CAN-GAT-2 and GAT-2 have excellent performance. The precision of GAT-2 reaches 0.99944. While the evaluation metrics of CAN-GAT-2 can reach 1.0. The overall performance of CAN-GAT-2 is slightly better than GAT-2. The detection capability of GCN-GAT is only inferior to CAN-GAT-2 and GAT-2. GSAGE-GAT and GSAGE-CAN-GAT have the similar accuracy. The performance of GSAGE-2 is slightly higher than that of GCN-2. GCN-CAN-GAT has the lowest performance in the GNN framework we built.

Figure 12(a) shows the performance of each model on the test datasets. It can be found that CAN-GAT-2 and GAT-2 perform best on the test dataset; The area of the models is about 1.0. Other models have similar performance and their area is between 0.95 and 0.97. Figure 12(b)-(f) presents the ability of each model to detect different types of attacks. We can see that CAN-GAT-2 and GAT-2 can effectively detect DoS, fuzzy, impersonation, GEAR, and RPM attacks. Except for CAN-GAT-2 and GAT-2, which can effectively detect impersonation attacks, the detection capabilities of other models have a sharp decline when detecting impersonation attacks. The reason may be that impersonation attacks mimic the behavior of normal CAN bus messages, which reduces the difference between the impersonation attack features and the normal features, resulting in a decrease in the detection effect. GNNs can better detect DoS, fuzzy, GEAR and RPM attacks. In addition, DoS, fuzzy, GEAR and RPM attacks as the injection attacks that add extra CAN bus messages will change the original time features and the interaction relationship between adjacent CAN bus messages. However, impersonation attacks have limited changes in these two features, this is also why impersonation attacks are more difficult to detect.

5.2 Analysis of the anomaly detection process with CAN-GAT-2

From the results shown in the above subsection, CAN-GAT-2 and GAT-2 can detect abnormal CAN bus messages effectively. The GCN and GSAGE network layers aggregate the node features through the edges in the graph, and each edge contributes the same magnitude to update the node features. While the CAN-GAT and GAT network layers can assign weights to edges, so that each edge can update the node features according to different magnitude. In the following, we will explore how the CAN-GAT-2 implements anomaly detection through the magnitude of interaction relationship between adjacent CAN bus messages (attention weights).

The CAN-GAT layer can learn the magnitude of the interaction relationship between adjacent CAN bus messages from the training datasets. Equations 16 and (17) are respectively used to calculate the attention weight and its normalized value. The normalized attention weight is as edge weight and stored on edge. Figure 13 shows how the magnitude of the edge weight changes during the training

Table 2 Datasets used in the experiments

Dataset type	Number of training datasets	Number of test datasets
Normal messages	1410,000	940,000
DoS Attack	282,000	188,000
Fuzzy Attack	282,000	188,000
RPM Attack	282,000	188,000
GEAR Attack	282,000	188,000
Impersonation Attack	282,000	188,000

Table 3 The performance of compared GNN models

Model	Accuracy	Precision	Recall	F1 score
CAN-GAT-2	**1.00000**	**1.00000**	**1.00000**	**1.00000**
GAT-2	0.99944	0.99944	0.99944	0.99944
GCN-2	0.90319	0.90319	0.90752	0.90293
GSAGE-2	0.90665	0.90665	0.91071	0.90642
GCN-CAN-GAT	0.86955	0.86955	0.87134	0.86939
GCN-GAT	0.91907	0.91907	0.92466	0.91880
GSAGE-CAN-GAT	0.90686	0.90686	0.91228	0.90655
GSAGE-GAT	0.90686	0.90686	0.91228	0.90655

process of the CAN-GAT-2. The edge is colored according to the magnitude of the normalized attention weights of the second layer CAN-GAT-2, which can refer to the color bar on the right. From Fig. 13, As the training epoch number increases, the number of edges with a high magnitude gradually gathers. The edge with high magnitude represents a closer relationship between the two nodes connected by the edge. At the early epochs of training, the model cannot effectively aggregate the edge with high magnitude to update the node features. As the performance of the model gets improved, the model effectively finds several critical edges.

To further explore the role of CAN-GAT in anomaly detection, information entropy is used to quantify the degree of concentration in each node, written as:

$$H(a_{ij_{j\in\mathcal{N}(i)}}) = -\sum_{j\in\mathcal{N}(i)} a_{ij}\log a_{ij} \tag{33}$$

where a_{ij} is the normalized attention weight, $\mathcal{N}(i)$ denotes the set of one-hop neighbors of node v_i. A low information entropy represents a high degree of concentration. Information entropy of 0 means that all information comes from one node. The highest information entropy $\log(\mathcal{N}(i))$ is uniformly distributed, which is equivalent to each edge having a similar attention weight. The highest information entropy means that each edge is equally important. This makes the attention mechanism lose its effect and causes the update of node features to be the same as the GCN and GSAGE network layers. Thus, low information entropy can highlight the edge with high magnitude. These edges are the key to distinguishing the normal and abnormal CAN bus messages.

5,000 normal CAN bus messages are used for the CAN-GAT-2 model to test all nodes' aggregated histogram of entropy values. Figure 14 shows the distribution histogram of the information entropy output by the two-layer networks of CAN-GAT-2, where the degree of concentration of the second layer of CAN-GAT-2 is slightly higher than that of the first layer. From Fig. 14, the information entropy of most nodes is close to 0, which indicates that the degree of concentration of information is high, and there are only several critical edges to transmit information. This is consistent with the results shown in Fig. 13(d).

To explore the influence of abnormal CAN bus messages on the information entropy of CAN-GAT-2, we use the normal dataset and different attack datasets to obtain the information entropy output by the second layer of CAN-GAT-2. The model outputs the sum of the entropy of 5,000 CAN bus messages each time, and in each dataset, 50,000 CAN bus messages are calculated. Figure 15 shows the information entropy calculated by CAN-GAT-2 for each dataset. Dots represent the outliers in the dataset. The horizontal line in the rectangular box represents the median. From Fig. 15, we can see that the information entropy of normal CAN bus messages is about 200-300. The information entropy of Fuzzy and impersonation attacks is between 340-480 and 340-420, respectively. The information entropy values of these three datasets are pretty similar. This may be because both fuzzy and impersonation attacks partially imitate normal CAN bus messages. GEAR and RPM attacks have a wide variety of entropy values, allowing for identifying most GEAR and RPM attacks. However, DoS attacks only send the same CAN bus messages periodically which make it to be better distinguished.

Through the above experiments, we can see that the information entropy output by CAN-GAT can be used to distinguish different datasets. It is different from the traditional anomaly detection method based on information entropy [37] that directly analyzes the CAN bus messages. The anomaly detection method that directly analyzes the entropy of CAN bus messages can only defend against limited injection attacks and DoS attacks. As shown in Fig. 15, choosing an appropriate threshold for the entropy output by CAN-GAT-2 can effectively detect the DoS, GEAR and RPM attacks, and distinguish most fuzzy and impersonation attacks.

5.3 Comparison experiments with baseline methods

To compare the proposed approach with other baseline methods, we choose other anomaly detection methods to

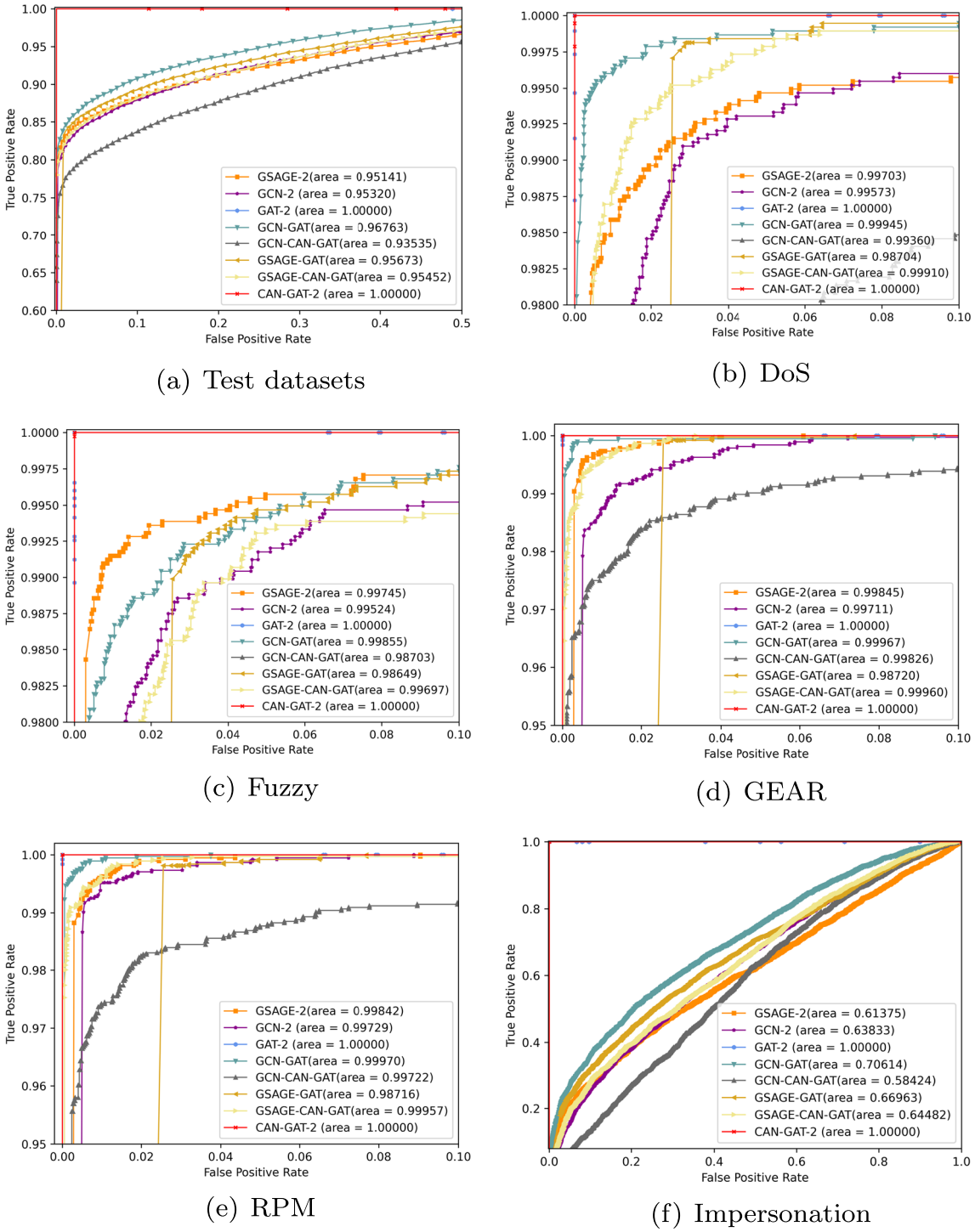
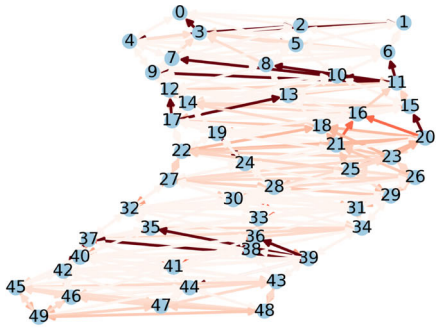
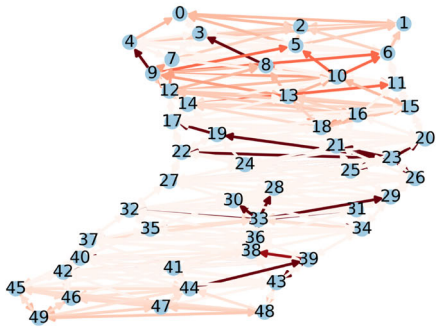


Fig. 12 The performance of each model on the test datasets and attack datasets

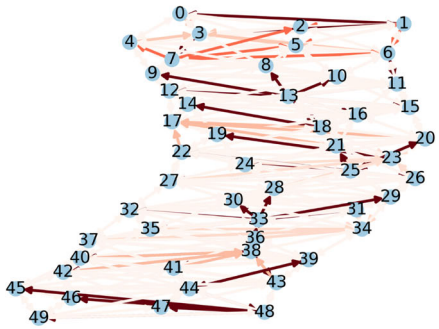
perform detection experiments with the same public datasets. We reimplemented Autoencoder+SVM [24], SIMATT-SECCU [23], MT-LSTM [26], H/R-RNN [43], reduced Inception-ResNet [28] and other methods, and respectively used the datasets in Table 2 and Table 5 to implement the training and testing to obtain evaluation metrics results. Support Vector Machine, k-nearest neighbors, Decision Trees and Multilayer Perceptron are implemented using the default settings of scikit-learn. Autoencoder+SVM, SIMATT-SECCU, MT-LSTM, H/R-RNN use the parameter settings of the corresponding reference paper to reproduce. MT-LSTM's threshold for judging abnormal data has been changed



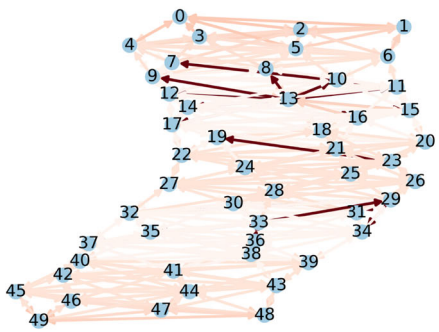
(a) Training 50 epochs



(b) Training 100 epochs



(c) Training 150 epochs



(d) Training 200 epochs

Fig. 13 Changes of magnitude of the edge weight during the training process of the GNN framework

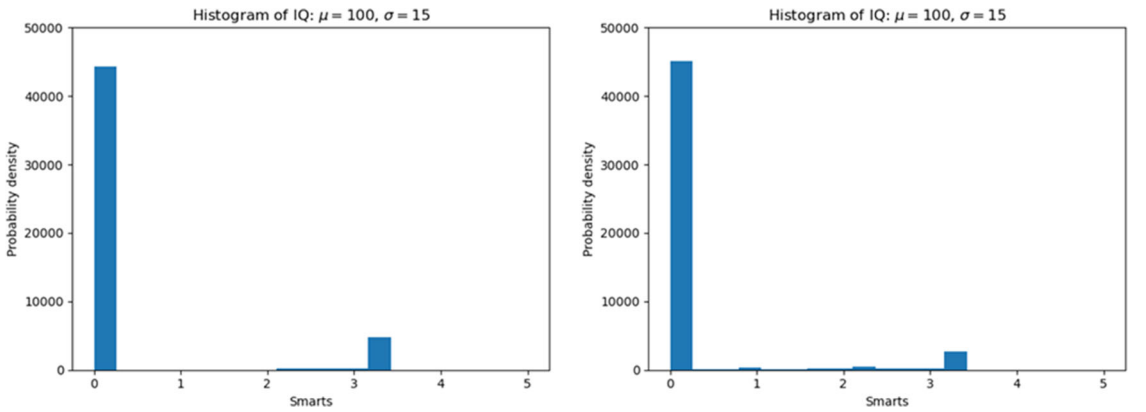
according to the dataset in Table 5. The results of the detection experiments are shown in Table 4 and Table 6 respectively.

In Table 4 and Table 6, the obtained results clearly demonstrate that the comprehensive performance of the CAN-GAT-2 model outperforms those of other anomaly detection methods. Moreover, the detection capability of GAT-2 is only slightly inferior to CAN-GAT-2. In Table 4, CAN-GAT-2 and GAT-2 perform equally well when it comes to attack detection. However, according to the complete display of Table 3, the GAT-2 mistakenly categorizes normal CAN messages into abnormal CAN messages, resulting in false alarm. In Table 6, the comprehensive detection capability of CAN-GAT-2 against various attacks is slightly higher than that of GAT-2.

To further explore the performance of CAN-GAT-2 and GAT-2. We respectively use the datasets in Table 2 to train these two models, and use the datasets in Table 7 as the test datasets. We found that CAN-GAT-2 and GAT-2 can accurately detect the attack data in the test datasets. However, 21 false alarms occurred when using the GAT-2 model to detect the tested data. However, the proposed CAN-GAT-2 did not issue such a false alarm, which demonstrates its good performance. Figure 16 shows the training loss values of CAN-GAT-2 and GAT-2 when using the training dataset in Table 5. It can be seen that the convergence speed of CAN-GAT-2 is faster than that of GAT-2.

One can see that the SIMATT-SECCU model performs well in large datasets, but its ability to detect DoS, gear and RPM attacks on small datasets declines. SIMATT-SECCU methods were developed based on recurrent neural networks. It is generally considered that recurrent neural networks are more appropriate for processing time sequential data such as CAN bus messages. The experimental results show that, with the small training dataset, the SIMATT-SECCU model also uses an attention mechanism, but cannot capture sufficient characteristics to perform anomaly detection with appropriate precision. CAN-GAT-2 preserved time features when transforming CAN bus messages into graphs, and applied CAN-GAT-2 and GAT-2 to capture its time features during the training epochs. CAN-GAT-2 and GAT-2 can still be well understood of the interaction relationship between adjacent CAN bus messages and capture the changes of this relationship even under small data sets to ensure the generalization ability of the anomaly detection model.

The Autoencoder+SVM model performs well on large datasets. However, the ability to detect DoS and impersonation attacks on small datasets is decreased. In general, CAN-GAT-2, GAT-2, SIMATT-SECCU and Autoencoder+SVM models have outstanding performance under large datasets. The SIMATT-SECCU model as a lightweight



(a) The first layer of CAN-GAT-2 (b) The second layer of CAN-GAT-2

Fig. 14 The distribution histogram of the information entropy output by the CAN-GAT-2 two-layer network

anomaly detection method simplifies the feature extraction process, resulting in insufficient feature capture on small datasets. Similarly, the autoencoder of the Autoencoder+SVM model failed to capture enough features of normal CAN bus messages on a small dataset, resulting in unsatisfactory attack detection. However, the CAN-GAT-2 and GAT-2 models fully exploit the characteristics of CAN bus messages, even with small datasets. It can learn enough features to implement anomaly detection.

When compared to other machine learning algorithms, the Support Vector Machine (SVM) and reduced Inception-ResNet model also presented decent detection performance against DoS, fuzzy, GEAR, and RPM attacks. However, the SVM and reduced Inception-ResNet model's ability to recognize impersonation attacks is weak. This may be because the characteristics of the impersonation attack are similar to normal CAN bus messages, making it difficult for the model to find a suitable pattern to match the impersonation attack. It can also be observed from Fig. 15 that the range of information entropy of impersonation attacks is close to that of the normal datasets. In comparing Autoencoder+SVM and SVM, Autoencoder+SVM adds autoencoder to make a further abstract representation of the original data. This makes Autoencoder+SVM effective in detecting impersonation attacks.

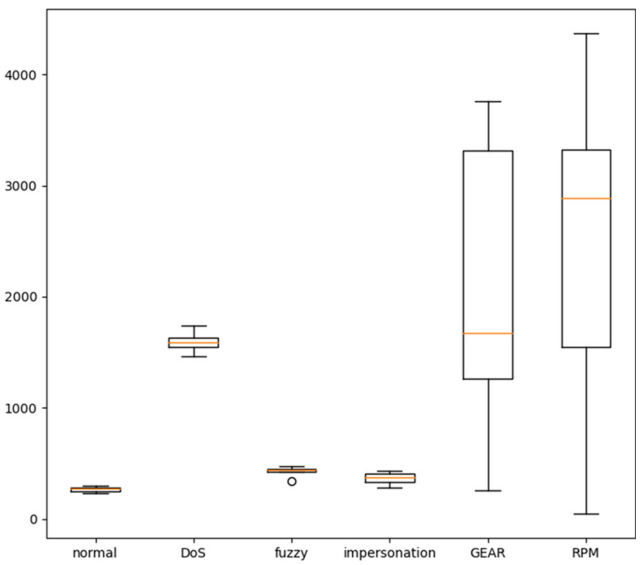


Fig. 15 The information entropy calculated by CAN-GAT-2

Traditional methods such as k-nearest neighbors, Decision Trees and Multilayer Perceptron (2 hidden layers, 112 hidden units) have insufficient detection capabilities for impersonation attacks. They cannot capture suitable features on small datasets.

According to the results shown in Table 4 and Table 6, CAN-GAT-2 performs better among the compared methods on the whole. H/R-RNN is a detection method integrating rules and heuristics, in which the rule-based method (H_{DoS}, H_{Fuzz} and H_{Re}) is derived by analyzing the signature of dynamic CAN messages, and the heuristic method (RNN-based) is used to train large-scale CAN messages. Table 4 and Table 6 show that the performance difference between CAN-GAT-2 and H/R-RNN models is relatively small. The H/R-RNN model can also receive excellent detection results on small datasets, but its detection capability to fuzzy and impersonation attacks is slightly weakened on large datasets. For the larger datasets shown in Table 7, the detection results in Table 8 indicates H/R-RNN's detection performance for fuzzy, impersonation and GEAR attacks is slightly weakened, and the detection capability of RPM attacks dropped obviously. While the CAN-GAT-2 still maintained a relatively ideal detection performance. The above simulation results show that under the same training datasets, CAN-GAT-2 can accurately detect more attacks.

To test the robustness of each detection method, we repeatedly sample different data from Table 5 as training

Table 4 Comparison results with other baseline methods on large datasets

DoS	Accuracy	Precision	Recall	F1 score
CAN-GAT-2	**1.00000**	**1.00000**	**1.00000**	**1.00000**
GAT-2	**1.00000**	**1.00000**	**1.00000**	**1.00000**
Support Vector Machine	0.99495	0.99497	0.99495	0.99495
k-nearest neighbors (k = 2)	0.92846	0.93741	0.92846	0.92809
Decision Trees	0.93138	0.93139	0.93138	0.93138
Multilayer Perceptron	0.90918	0.91351	0.90918	0.90894
Autoencoder+SVM [24]	0.98829	**1.00000**	0.98830	0.99411
SIMATT-SECCU [23]	**1.00000**	**1.00000**	**1.00000**	**1.00000**
MT-LSTM [26]	0.57139	0.57000	0.58131	0.57560
H/R-RNN [43]	**1.00000**	**1.00000**	**1.00000**	**1.00000**
reduced Inception-ResNet [28]	0.99923	0.99846	**1.00000**	0.99923
Fuzzy	Accuracy	Precision	Recall	F1 score
CAN-GAT-2	**1.00000**	**1.00000**	**1.00000**	**1.00000**
GAT-2	**1.00000**	**1.00000**	**1.00000**	**1.00000**
Support Vector Machine	0.99920	0.99920	0.99920	0.99920
k-nearest neighbors (k = 2)	0.82939	0.83039	0.82939	0.82926
Decision Trees	0.93404	0.93406	0.93404	0.93404
Multilayer Perceptron	0.92673	0.93495	0.92673	0.92638
Autoencoder+SVM [24]	0.97500	**1.00000**	0.97500	0.98734
SIMATT-SECCU [23]	**1.00000**	**1.00000**	**1.00000**	**1.00000**
MT-LSTM [26]	0.70639	0.66001	0.85132	0.74356
H/R-RNN [43]	0.99979	**1.00000**	0.99957	0.99979
reduced Inception-ResNet [28]	0.99514	0.99845	0.99182	0.99512
Impersonation	Accuracy	Precision	Recall	F1 score
CAN-GAT-2	**1.00000**	**1.00000**	**1.00000**	**1.00000**
GAT-2	**1.00000**	**1.00000**	**1.00000**	**1.00000**
Support Vector Machine	0.59827	0.77330	0.59827	0.52169
k-nearest neighbors (k = 2)	0.59867	0.63457	0.59867	0.56999
Decision Trees	0.83883	0.85165	0.83883	0.83735
Multilayer Perceptron	0.56702	0.60134	0.56702	0.52698
Autoencoder+SVM [24]	0.92500	**1.00000**	0.92500	0.96104
SIMATT-SECCU [23]	**1.00000**	**1.00000**	**1.00000**	**1.00000**
MT-LSTM [26]	0.59016	0.58527	0.61886	0.60160
H/R-RNN [43]	0.99862	**1.00000**	0.99723	0.99861
reduced Inception-ResNet [28]	0.83925	0.99774	0.68004	0.80881
GEAR	Accuracy	Precision	Recall	F1 score
CAN-GAT-2	**1.00000**	**1.00000**	**1.00000**	**1.00000**
GAT-2	**1.00000**	**1.00000**	**1.00000**	**1.00000**
Support Vector Machine	0.99920	0.99920	0.99920	0.99920
k-nearest neighbors (k = 2)	0.92846	0.93741	0.92846	0.92809
Decision Trees	0.96011	0.96114	0.96011	0.96008
Multilayer Perceptron	0.92473	0.93244	0.92473	0.92440
Autoencoder+SVM [24]	**1.00000**	**1.00000**	**1.00000**	**1.00000**
SIMATT-SECCU [23]	**1.00000**	**1.00000**	**1.00000**	**1.00000**
MT-LSTM [26]	0.71115	0.66250	0.86084	0.74876
H/R-RNN [43]	**1.00000**	**1.00000**	**1.00000**	**1.00000**
reduced Inception-ResNet [28]	0.99923	0.99846	**1.00000**	0.99923

Table 4 (continued)

RPM	Accuracy	Precision	Recall	F1 score
CAN-GAT-2	**1.00000**	**1.00000**	**1.00000**	**1.00000**
GAT-2	**1.00000**	**1.00000**	**1.00000**	**1.00000**
Support Vector Machine	0.99907	0.99907	0.99907	0.99907
k-nearest neighbors (k = 2)	0.92846	0.93741	0.92846	0.92809
Decision Trees	0.95918	0.96060	0.95918	0.95914
Multilayer Perceptron	0.91489	0.92034	0.91489	0.91462
Autoencoder+SVM [24]	0.99946	**1.00000**	0.99947	0.99973
SIMATT-SECCU [23]	**1.00000**	**1.00000**	**1.00000**	**1.00000**
MT-LSTM [26]	0.70270	0.65805	0.84393	0.73949
H/R-RNN [43]	**1.00000**	**1.00000**	**1.00000**	**1.00000**
reduced Inception-ResNet [28]	0.99923	0.99846	**1.00000**	0.99923

Table 5 The small datasets used in the experiments

Dataset type	Number of training datasets	Number of test datasets
Normal messages	250,000	125,000
DoS Attack	50,000	25,000
Fuzzy Attack	50,000	25,000
RPM Attack	50,000	25,000
GEAR Attack	50,000	25,000
Impersonation Attack	50,000	25,000

Table 6 Comparison results with other baseline methods on small datasets

DoS	Accuracy	Precision	Recall	F1 score
CAN-GAT-2	**1.0000**	**1.0000**	**1.0000**	**1.0000**
GAT-2	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Support Vector Machine	0.9940	0.9940	0.9940	0.9940
k-nearest neighbors (k = 2)	0.8860	0.9071	0.8860	0.8845
Decision Trees	0.8300	0.8319	0.8300	0.8297
Multilayer Perceptron	0.7630	0.7640	0.7630	0.7627
Autoencoder+SVM [24]	0.9000	**1.0000**	0.9000	0.9473
SIMATT-SECCU [23]	0.9090	0.9591	0.8545	0.9038
MT-LSTM [26]	0.6387	0.5925	0.8880	0.7108
H/R-RNN [43]	**1.0000**	**1.0000**	**1.0000**	**1.0000**
reduced Inception-ResNet [28]	0.98260	0.96742	0.99884	0.98288
Fuzzy	Accuracy	Precision	Recall	F1 score
CAN-GAT-2	0.9980	**1.0000**	0.9980	0.9990
GAT-2	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Support Vector Machine	**1.0000**	**1.0000**	**1.0000**	**1.0000**
k-nearest neighbors (k = 2)	0.6890	0.6943	0.6890	0.6868
Decision Trees	0.8630	0.8630	0.8630	0.8629
Multilayer Perceptron	0.8760	0.8863	0.8760	0.8751
Autoencoder+SVM [24]	**1.0000**	**1.0000**	**1.0000**	**1.0000**
SIMATT-SECCU [23]	**1.0000**	**1.0000**	**1.0000**	**1.0000**
MT-LSTM [26]	0.6545	0.6009	0.9196	0.7269
H/R-RNN [43]	**1.0000**	**1.0000**	**1.0000**	**1.0000**
reduced Inception-ResNet [28]	0.97158	0.96670	0.97680	0.97173

Table 6 (continued)

Impersonation	Accuracy	Precision	Recall	F1 score
CAN-GAT-2	**1.0000**	**1.0000**	**1.0000**	**1.0000**
GAT-2	0.9760	**1.0000**	0.9760	0.9878
Support Vector Machine	0.5260	0.7566	0.5260	0.3886
k-nearest neighbors (k = 2)	0.6520	0.6612	0.6520	0.6469
Decision Trees	0.8150	0.8165	0.8150	0.8147
Multilayer Perceptron	0.6250	0.6411	0.6250	0.6139
Autoencoder+SVM [24]	0.8040	**1.0000**	0.8040	0.8913
SIMATT-SECCU [23]	**1.0000**	**1.0000**	**1.0000**	**1.0000**
MT-LSTM [26]	0.5950	0.5673	0.8007	0.6641
H/R-RNN [43]	**1.0000**	**1.0000**	**1.0000**	**1.0000**
reduced Inception-ResNet [28]	0.97158	0.96670	0.97680	0.97173
GEAR	Accuracy	Precision	Recall	F1 score
CAN-GAT-2	**1.0000**	**1.0000**	**1.0000**	**1.0000**
GAT-2	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Support Vector Machine	**1.0000**	**1.0000**	**1.0000**	**1.0000**
k-nearest neighbors (k = 2)	0.8860	0.9071	0.8860	0.8845
Decision Trees	0.8960	0.8983	0.8960	0.8958
Multilayer Perceptron	0.8910	0.9062	0.8910	0.8899
Autoencoder+SVM [24]	**1.0000**	**1.0000**	**1.0000**	**1.0000**
SIMATT-SECCU [23]	0.4545	**1.0000**	0.4545	0.6250
MT-LSTM [26]	0.6709	0.6093	0.9524	0.7431
H/R-RNN [43]	**1.0000**	**1.0000**	**1.0000**	**1.0000**
reduced Inception-ResNet [28]	0.98318	0.98373	0.98318	0.98317
RPM	Accuracy	Precision	Recall	F1 score
CAN-GAT-2	**1.0000**	**1.0000**	**1.0000**	**1.0000**
GAT-2	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Support Vector Machine	**1.0000**	**1.0000**	**1.0000**	**1.0000**
k-nearest neighbors (k = 2)	0.8860	0.9071	0.8860	0.8845
Decision Trees	0.9110	0.9146	0.9110	0.9108
Multilayer Perceptron	0.8690	0.8774	0.8690	0.8682
Autoencoder+SVM [24]	0.9960	**1.0000**	0.9960	0.9980
SIMATT-SECCU [23]	0.8181	**1.0000**	0.8181	0.9000
MT-LSTM [26]	0.6689	0.6083	0.9484	0.7412
H/R-RNN [43]	**1.0000**	**1.0000**	**1.0000**	**1.0000**
reduced Inception-ResNet [28]	0.97912	0.96719	0.99188	0.97938

Table 7 Test datasets of CAN-GAT-2, GAT-2 and H/R-RNN models

Dataset type	Number of test datasets
Normal messages	940,000
DoS Attack	374,000
Fuzzy Attack	309,000
Impersonation Attack	713,000
GEAR Attack	4161,000
RPM Attack	4339,000

and test datasets to evaluate each method. In the training and test datasets in Table 5, the normal data and abnormal data of the same ratio are sampled as each model's training and test datasets (the size of the test dataset is one-third of the training dataset). The sampling was repeated 15 times. Table 9 shows the detection performance of each model. H/R-RNN is a multi-classification method, and the evaluation method is different from other binary classification methods. For the consistency of the evaluation criteria, it is not listed in

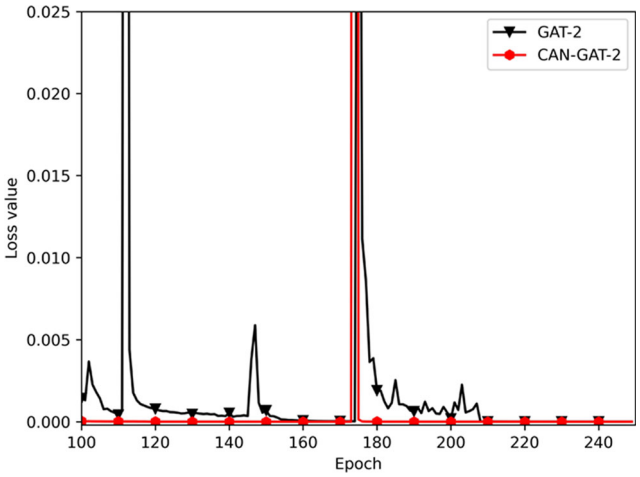


Fig. 16 CAN-GAT-2 and GAT-2 training loss

Table 9. Table 9 presents the minimum, maximum, arithmetic mean and variance values of each model's performance indicators. One can see that CAN-GAT-2 obtains the best performance and the smallest variance. The performance of GAT-2 is slightly lower than CAN-GAT-2, and the performance of reduced Inception-ResNet is also relatively stable. It is worth noting that MT-LSTM, as an unsupervised anomaly detection method, needs to set a threshold as the basis for anomaly judgment. The test dataset changed in the repeated sampling of 15 times, and the threshold was not automatically appropriately adjusted, resulting in unsatisfactory experiment results.

Table 8 Comparison results of CAN-GAT-2 and H/R-RNN models on the datasets in Table 7

DoS	Accuracy	Precision	Recall	F1 score
CAN-GAT-2	**1.00000**	**1.00000**	**1.00000**	**1.00000**
H/R-RNN [43]	**1.00000**	**1.00000**	**1.00000**	**1.00000**
Fuzzy	Accuracy	Precision	Recall	F1 score
CAN-GAT-2	**1.00000**	**1.00000**	**1.00000**	**1.00000**
H/R-RNN [43]	0.99974	**1.00000**	0.99974	0.99987
Impersonation	Accuracy	Precision	Recall	F1 score
CAN-GAT-2	**1.00000**	**1.00000**	**1.00000**	**1.00000**
H/R-RNN [43]	0.99865	**1.00000**	0.99865	0.99933
GEAR	Accuracy	Precision	Recall	F1 score
CAN-GAT-2	**1.00000**	**1.00000**	**1.00000**	**1.00000**
H/R-RNN [43]	0.99998	1.00000	0.99998	0.99999
RPM	Accuracy	Precision	Recall	F1 score
CAN-GAT-2	**1.00000**	**1.00000**	**1.00000**	**1.00000**
H/R-RNN [43]	0.79480	**1.00000**	0.79480	0.88567

Table 9 Comparison results with other baseline methods on small datasets randomly sample data as training and test datasets

CAN-GAT-2	Accuracy	Precision	Recall	F1 score
min	0.99899	0.99898	0.99900	0.99899
max	1.00000	1.00000	1.00000	1.00000
mean	0.99983	0.99983	0.99983	0.99983
var	8.86×10^{-8}	8.99×10^{-8}	8.71×10^{-8}	8.86×10^{-8}
GAT-2	Accuracy	Precision	Recall	F1 score
min	0.99838	0.99837	0.99839	0.99838
max	1.00000	1.00000	1.00000	1.00000
mean	0.99969	0.99969	0.99969	0.99969
var	1.91×10^{-7}	1.93×10^{-7}	1.89×10^{-7}	1.91×10^{-7}
Support Vector Machine	Accuracy	Precision	Recall	F1 score
min	0.90323	0.91944	0.90248	0.90217
max	0.91919	0.93009	0.91965	0.91873
mean	0.91017	0.92400	0.90997	0.90939
var	1.88×10^{-5}	1.10×10^{-5}	1.81×10^{-5}	1.94×10^{-5}
k-nearest neighbors (k = 2)	Accuracy	Precision	Recall	F1 score
min	0.86949	0.86991	0.86913	0.86932
max	0.88262	0.88291	0.88254	0.88258
mean	0.87646	0.87691	0.87643	0.87640
var	1.74×10^{-5}	1.66×10^{-5}	1.70×10^{-5}	1.74×10^{-5}
Decision Trees	Accuracy	Precision	Recall	F1 score
min	0.88383	0.88385	0.88371	0.88377
max	0.90990	0.90991	0.90992	0.90990
mean	0.89424	0.89431	0.89423	0.89423
var	4.83×10^{-5}	4.80×10^{-5}	4.87×10^{-5}	4.85×10^{-5}
Multilayer Perceptron	Accuracy	Precision	Recall	F1 score
min	0.84727	0.84754	0.84736	0.84726
max	0.87253	0.87297	0.87263	0.87250
mean	0.86066	0.86113	0.86066	0.86060
var	4.23×10^{-5}	4.38×10^{-5}	4.29×10^{-5}	4.23×10^{-5}
Autoencoder+ SVM [24]	Accuracy	Precision	Recall	F1 score
min	0.95515	0.95754	0.95491	0.95504
max	0.97292	0.97487	0.97229	0.97286
mean	0.96202	0.96469	0.96204	0.96196
var	2.18×10^{-5}	2.02×10^{-5}	1.78×10^{-5}	2.17×10^{-5}
SIMATT-SECCU [23]	Accuracy	Precision	Recall	F1 score
min	0.95960	0.96154	0.96078	0.95959
max	1.00000	1.00000	1.00000	1.00000
mean	0.98653	0.98714	0.98643	0.98649
var	1.26×10^{-4}	1.13×10^{-4}	1.24×10^{-4}	1.27×10^{-5}

Table 9 (continued)

MT-LSTM [26]	Accuracy	Precision	Recall	F1 score
min	0.49327	0.48999	0.49521	0.41391
max	0.50386	0.50499	0.50200	0.43050
mean	0.49907	0.49792	0.49897	0.42347
var	7.57×10^{-6}	2.17×10^{-5}	4.87×10^{-6}	2.01×10^{-5}
reduced Inception-ResNet [28]	Accuracy	Precision	Recall	F1 score
min	0.99520	0.99520	0.99520	0.99520
max	1.00000	1.00000	1.00000	1.00000
mean	0.99898	0.99898	0.99899	0.99898
var	1.88×10^{-6}	1.89×10^{-6}	1.88×10^{-6}	1.88×10^{-6}

5.4 Detection time analysis

In an anomaly intrusion detection system, it is generally necessary to identify anomalies real time. Therefore, the detection time is an important metric for evaluating anomaly detection systems. The detection time [28] is calculated with

$$t_d = t_f - t_o,\ (t_{start} \le t_o \le t_{end}) \tag{34}$$

where t_f is the moment when the attack is detected, and t_o is the moment when the attack enters vehicle networks. The detection time t_d represents the length of time between the attack enters vehicle networks and the attack is detected. t_{start} and t_{end} are the start time and end time of collecting 50 CAN bus messages.

t_f can be divided into two parts: data collection t_c and model detection t_m. The detection time is equivalent to

$$t_d = t_c - t_o + t_m \tag{35}$$

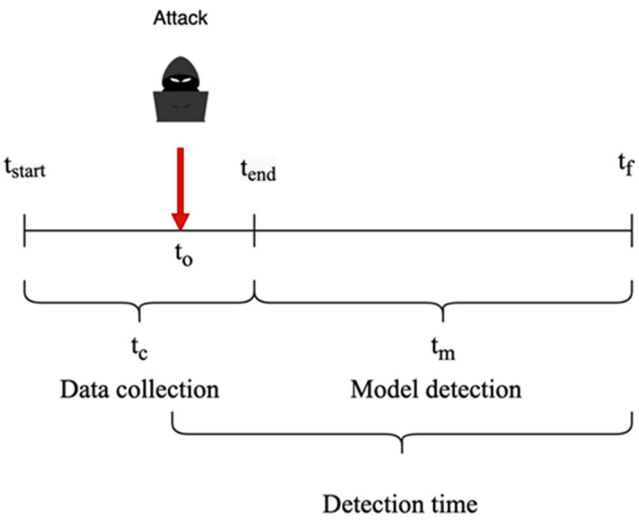


Fig. 17 The time course of model detection

Table 10 the time cost of the proposed model and other published models to detect 2000 CAN bus messages

Model	CAN-GAT-2	GAT-2	H/R-RNN	reduced Inception-ResNet	MT-LSTM
Time	16.939	18.932	4785.987	263.274	81.000

Figure 17 describes the time course of model detection. we can see that during the data collection, the attack may occur at any time between t_{start} and t_{end}. When $t_o = t_{start}$, the detection time is maximum ($t_d = t_c + t_m$), and when $t_o = t_{end}$, the detection time is minimum ($t_d = t_m$).

We used a laptop of Intel core i7-11800H, 2.30GHz CPUs, 32G RAM and Windows 10 operating system, to implement each model detection time analysis experiment. In the CAN-GAT-2 model, since each CAN bus message converted into a graph has the same topology, it can be pre-loaded in the RAM in advance, so the time for converting CAN bus messages into a graph can be ignored. The model detection time t_m is about 8.995 ms. Moreover, the CAN bus transmits approximately 2,000 CAN bus messages per second. It takes about 25 ms for the CAN bus to generate 50 CAN bus messages. If $t_o = t_{start}$, the model needs 33.995 ms to detect anomalies and if $t_o = t_{end}$, the model only needs 8.995 ms to identify anomalies. The CAN-GAT-2 model can implement real-time detection. Time is mainly spent on model detection. To compare the detection time difference of each model, Table 10 shows the time cost of the proposed model and other published models to detect 2000 CAN bus messages. CAN-GAT-2 has the least time cost, and GAT-2 is second. HR/-RNN and reduced Inception-ResNet use too many recurrent neural network layers and convolutional neural network layers, respectively, resulting in excessive time cost. MT-LSTM employs a few LSTM layers, the detection time cost is moderate.

6 Conclusion

The robust anomaly-based detection system for in-vehicle network by GNN framework has been presented in this paper. A method that transforms CAN bus messages into graph structure was proposed, and the GNNs were applied to detect anomalies in in-vehicle networks. The transformed graph structure can retain the time features of CAN bus messages and extract the relationship between adjacent CAN bus messages. Then, a CAN-GAT GNN model is proposed and a variety of GNN models were studied, especially, the comprehensive performance of the CAN-GAT-2 model is excellent among the compared GNNs.

Furthermore, the influence of the attention mechanism of the GNN in detecting CAN bus messages was analyzed. It is found that the attention weight will be concentrated on several edges. The information entropy was used to quantify the attention weight. By calculating the sum of the information entropy of multiple nodes, one finds that each attack type has a fixed range of information entropy. This may be a factor for the attention mechanism to improve the accuracy of the model. Multiple experiments were implemented to analyze the proposed method, and the evaluation results show that the CAN-GAT-2 mechanism achieves excellent detection capabilities. CAN-GAT-2 has good detection speed performance to detect an attack, and can implement the needs of real-time detection. However, like other machine learning-based methods, the time spent in the training phase of the proposed model requires further optimization.

For the future work, we will focus on further optimizing the accuracy of GNN anomaly detection on in-vehicle networks. In this paper, only the relationship between adjacent CAN bus messages is considered in each graph, the interaction of CAN bus messages between the graphs is one of our researches in the future.

References

1. Tuohy S, Glavin M, Hughes C, Jones E, Trivedi M, Kilmartin L (2015) Intra-vehicle networks: A review. IEEE Trans Intell Transp Syst 16(2):534–545. https://doi.org/10.1109/TITS.2014.2320605
2. Fröschle S, Stühring A (2017) Analyzing the capabilities of the CAN attacker. In: Simon N, Foley DG, Snekkenes E (eds) Computer Security – ESORICS 2017, pp. 464–482. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-66402-6n_27
3. Marchetti M, Stabili D (2019) READ: Reverse engineering of automotive data frames. IEEE Transactions on Information Forensics and Security 14(4):1083–1097. https://doi.org/10.1109/TIFS.2018.2870826
4. Woo S, Jo HJ, Lee DH (2015) A practical wireless attack on the connected car and security protocol for in-vehicle CAN. IEEE Trans Intell Transp Syst 16(2):993–1006. https://doi.org/10.1109/TITS.2014.2351612
5. Wu W, Kurachi R, Zeng G, Matsubara Y, Takada H, Li R, Li K (2018) IDH-CAN: A hardware-based ID hopping CAN mechanism with enhanced security for automotive real-time applications. IEEE Access 6:54607–54623. https://doi.org/10.1109/ACCESS.2018.2870695
6. Lin C, Sangiovanni-Vincentelli A (2012) Cyber-security for the controller area network (CAN) communication protocol. In: International Conference on Cyber Security, Washington, DC, USA, pp 1–7
7. Nilsson DK, Larson UE, Jonsson E (2008) Efficient in-vehicle delayed data authentication based on compound message authentication codes. In: IEEE 68th Vehicular Technology Conference, Calgary, BC, Canada, pp 1–5
8. Wang E, Xu W, Sastry S, Liu S, Zeng K (2017) Hardware module-based message authentication in intra-vehicle networks. In: ACM/IEEE International Conference on Cyber-Physical Systems, Pittsburgh, PA, USA, pp 207–216
9. Bulck JV, Mühlberg JT, Piessens F (2017) VulCAN: Efficient component authentication and software isolation for automotive control networks. In: Annual Computer Security Applications Conference, Orlando FL USA, pp 225–237
10. Lu Z, Wang Q, Chen X, Qu G, Lyu Y, Liu Z (2019) LEAP: A lightweight encryption and authentication protocol for in-vehicle communications. In: IEEE Intelligent Transportation Systems Conference, Auckland, New Zealand, pp 1158–1164
11. Macher G, Sporer H, Brenner E, Kreiner C (2017) An automotive signal-layer security and trust-boundary identification approach. Procedia Computer Science 109C:490–497. https://doi.org/10.1016/j.procs.2017.05.317
12. Macher G, Sporer H, Brenner E, Kreiner C (2018) Signal-layer security and trust-boundary identification based on hardware-software interface definition. Journal of Ubiquitous Systems and Pervasive Networks 10(1):1–9. https://doi.org/10.5383/JUSPN.10.01.001
13. Wu W, Li R, Xie G, An J, Bai Y, Zhou J, Li K (2020) A survey of intrusion detection for in-vehicle networks. IEEE Trans Intell Transp Syst 21(3):919–933. https://doi.org/10.1109/tits.2019.2908074
14. Chakraborty S, Al Faruque MA, Chang W, Goswami D, Wolf M, Zhu Q (2016) Automotive cyber-physical systems: A tutorial introduction. IEEE Design & Test 33(4):92–108. https://doi.org/10.1109/MDAT.2016.2573598
15. Wasicek A, Derler P, Lee EA (2014) Aspect-oriented modeling of attacks in automotive cyber-physical systems. In: ACM/EDAC/IEEE Design Automation Conference, San Francisco, CA, USA, pp 1–6
16. Abbott-McCune S, Shay LA (2016) Intrusion prevention system of automotive network CAN bus. In: IEEE International Carnahan Conference on Security Technology, Orlando, FL, USA, pp 1–8
17. Malhotra P, Vig L, Shroff G, Agarwal P (2015) Long short term memory networks for anomaly detection in time series. In: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, Belgium, pp 89–94
18. Protogerou A, Papadopoulos S, Drosou A, Tzovaras D, Refanidis I (2021) A graph neural network method for distributed anomaly detection in IoT. EVOLVING SYSTEMS 12(1, SI):19–36. https://doi.org/10.1007/s12530-020-09347-0
19. Yu T, Wang X (2020) Topology verification enabled intrusion detection for in-vehicle CAN-FD networks. IEEE Commun Lett 24(1):227–230. https://doi.org/10.1109/LCOMM.2019.2953722
20. Qin H, Yan M, Ji H (2021) Application of controller area network (CAN) bus anomaly detection based on time series prediction. Vehicular Communications 27:100291. https://doi.org/10.1016/j.vehcom.2020.100291
21. Ji H, Wang Y, Qin H, Wang Y, Li H (2018) Comparative performance evaluation of intrusion detection methods for in-vehicle networks. IEEE Access 6:37523–37532. https://doi.org/10.1109/ACCESS.2018.2848106
22. Li X, Yu Y, Sun G, Chen K (2018) Connected vehicles' security from the perspective of the in-vehicle network. IEEE Netw 32(3):58–63. https://doi.org/10.1109/MNET.2018.1700319
23. Xiao J, Wu H, Li X (2019) Internet of things meets vehicles: Sheltering in-vehicle network through lightweight machine learning. Symmetry 11:1388:1–21. https://doi.org/10.3390/sym11111388
24. Xiao J, Wu H, Li X, Yuan L (2019) Practical IDS on in-vehicle network against diversified attack models. In: International

Conference, Algorithms and Architectures for Parallel Processing, Melbourne, VIC, Australia, pp 456–466
25. Taylor A, Leblanc S, Japkowicz N (2016) Anomaly detection in automobile control network data with long short-term memory networks. In: IEEE International Conference on Data Science and Advanced Analytics, Montreal, QC, Canada, pp 130–139
26. Zhu K, Chen Z, Peng Y, Zhang L (2019) Mobile edge assisted literal multi-dimensional anomaly detection of in-vehicle network using LSTM. IEEE Trans Veh Technol 68(5):4275–4284. https://doi.org/10.1109/TVT.2019.2907269
27. Xiao J, Wu H, Li X (2019) Robust and self-evolving IDS for in-vehicle network by enabling spatiotemporal information. In: IEEE International Conference on High Performance Computing and Communications; IEEE International Conference on Smart City; IEEE International Conference on Data Science and Systems, Zhangjiajie, China, pp 1390–1397
28. Song HM, Woo J, Kim HK (2020) In-vehicle network intrusion detection using deep convolutional neural network. Vehicular Communications 21:100198:1–13. https://doi.org/10.1016/j.vehcom.2019.100198
29. Kang M, Kang J (2016) A novel intrusion detection method using deep neural network for in-vehicle network security. In: IEEE Vehicular Technology Conference (VTC Spring), Nanjing, China, pp 1–5
30. Park S, Choi J.-Y. (2020) Hierarchical anomaly detection model for in-vehicle networks using machine learning algorithms. Sensors 20:3934:1–21. https://doi.org/10.3390/s20143934
31. Marchetti M, Stabili D (2017) Anomaly detection of CAN bus messages through analysis of ID sequences. In: IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, pp 1577–1583
32. Taylor A, Japkowicz N, Leblanc S (2015) Frequency-based anomaly detection for the automotive CAN bus. In: 2015 World Congress on Industrial Control Systems Security (WCICSS)
33. Hoppe T, Kiltz S, Dittmann J (2008) Security threats to automotive CAN networks– practical examples and selected short-term countermeasures. In: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinform.), in: LNCS, vol 5219, pp 235–248. https://doi.org/10.1007/978-3-540-87698-4_21
34. Valasek CMC (2013) Adventures in automotive networks and control units. Tech. White Pap, 99
35. Olufowobi H, Young C, Zambreno J, Bloom G (2020) SAIDuCANT: Specification-based automotive intrusion detection using controller area network (CAN) timing. IEEE Trans Veh Technol 69(2):1484–1494. https://doi.org/10.1109/TVT.2019.2961344
36. Zhou J, Joshi P, Zeng H, Li R (2019) BTMonitor: Bit-time-based intrusion detection and attacker identification in controller area network. ACM Trans Embed Comput Syst 18(6):1–23. https://doi.org/10.1145/3362034
37. Ohira S, Desta AK, Arai I, Inoue H, Fujikawa K (2020) Normal and malicious sliding windows similarity analysis method for fast and accurate IDS against DoS attacks on in-vehicle networks. IEEE Access 8:42422–42435. https://doi.org/10.1109/access.2020.2975893
38. Shin KG, Cho KT (2017) Fingerprinting electronic control units for vehicle intrusion detection
39. Choi W, Joo K, Jo HJ, Park MC, Lee DH (2018) VoltageIDS: Low-level communication characteristics for automotive intrusion detection system. IEEE Transactions on Information Forensics and Security 13(8):2114–2129. https://doi.org/10.1109/TIFS.2018.2812149
40. Katragadda S, Darby PJ, Roche A, Gottumukkala R (2020) Detecting low-rate replay-based injection attacks on in-vehicle networks. IEEE Access 8:54979–54993. https://doi.org/10.1109/ACCESS.2020.2980523
41. Song HM, Kim HR, Kim HK (2016) Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In: International Conference on Information Networking, Kota Kinabalu, Malaysia, pp 63–68
42. Cho KT, Kang GS (2017) Viden: Attacker identification on in-vehicle networks. In: ACM SIGSAC Conference on Computer and Communications Security, Dallas Texas USA, pp 1109–1123
43. Tariq S, Lee S, Kim HK, Woo SS (2020) CAN-ADF: The controller area network attack detection framework. Computers & Security 94:101857:1–12. https://doi.org/10.1016/j.cose.2020.101857
44. Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS (2021) A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems 32(1):4–24. https://doi.org/10.1109/TNNLS.2020.2978386
45. Nathani D, Chauhan J, Sharma C, Kaul M (2019) Learning attention-based embeddings for relation prediction in knowledge graphs. arXiv:1906.01195, https://doi.org/10.18653/v1/P19-1466
46. Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS (2019) A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, pp 1–21, https://doi.org/10.1109/TNNLS.2020.2978386
47. Lee H, Jeong SH, Kim HK (2017) OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame. In: Annual Conference on Privacy, Security and Trust, Calgary, AB, Canada, pp 57–66
48. Xie L, Pi D, Zhang X, Chen J, Luo Y, Yu W (2021) Graph neural network approach for anomaly detection. MEASUREMENT, 180, https://doi.org/10.1016/j.measurement.2021.109546
49. Kipf T, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv:1609.02907
50. Veličkovič P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2017) Graph attention networks. arXiv:1710.10903
51. Linghu Y, Li X (2021) Wsg-inv: Weighted state graph model for intrusion detection on in-vehicle network. In: 2021 IEEE Wireless Communications and Networking Conference (WCNC), pp 1–7
52. Hamilton LW, Ying R, Leskovec J. (2017) Inductive representation learning on large graphs. In: International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, pp 1025–1035
53. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. In: International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, pp 6000–6010
54. Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv:1412.6980
55. Bewick V, Cheek L, Ball J (2004) Statistics review 13: Receiver operating characteristic curves. Critical Care 8(6):508–512. https://doi.org/10.1186/cc3000
56. Pundir S, Amala R (2014) Parametric receiver operating characteristic modeling for continuous data: A glance. Model Assist Stat Appl 9(2):121–135. https://doi.org/10.3233/MAS-130284

Junchao Xiao received the M.S. degree from Shanghai Maritime University, China in 2018. He is currently pursuing the Ph.D. degree with the School of Systems Science and Engineering, Sun Yat-sen University. His research interests include deep learning, network security, and intrusion detection system, etc.

Hongbo Chen received the Ph.D. degree from the Harbin Institute of Technology, in 2007. He is currently a Professor with the School of Systems Science and Engineering, Sun Yat-sen University (SYSU). He is the Dean with the School of Systems Science and Engineering, SYSU. His main research interests include modeling, simulation and analysis of complex systems, intelligent unmanned systems, design of aerospace vehicle, etc.

Lin Yang received the M.S. degree in automation and the Ph.D. degree in communication and electronic system from the National University of Defense Technology, Changsha, Hunan, in 1995 and 1999, respectively. From 2005, he was a Senior Engineer with the China Electronic Equipment and System Engineering Corporation. His research interests include computer security, information system security, network security, trusted computing, security protocol analysis, and big-data security.

Xiangxue Li received the Ph.D. degree from Shanghai Jiao Tong University, in 2006. He was with the School of Information Security Engineering, Shanghai Jiao Tong University. He is currently a Professor with the School of Software Engineering, East China Normal University. He has authored three books and has authored or coauthored more than 70 articles. His research interests include lightweight protocol design, anonymity, and pseudorandom sequence.

Fuli Zhong received the B.S. degree in electronic information engineering and the Ph.D. degree in navigation, guidance and control from the University of Electronic Science and Technology of China, Chengdu, China, in 2011 and 2019, respectively. He is currently a Postdoctoral Fellow with the School of Systems Science and Engineering, Sun Yat-Sen University. His research interests include Cybersecurity and resilience, machine learning, Cyber-physical system, prognostics and health management, full-duplex wireless communications, signal processing, optimization, complex nonlinear dynamic systems, state estimation, identification and control of dynamic systems.