Federated Graph Neural Network for Fast Anomaly Detection in Controller Area Networks

Hengrun Zhang, *Member, IEEE*, Kai Zeng, *Member, IEEE*, and Shuai Lin, *Member, IEEE*

***Abstract*— Due to the lack of CAN frame encryption and authentication, CAN bus is vulnerable to various attacks, which can in general be divided into message injection, suspension, and falsification. Existing CAN bus anomaly detection mechanisms either can only detect one or two of these attacks, or require numerous CAN messages during predictions, which can hardly realize real-time performance. In this paper, we propose a CAN bus anomaly detection system that can detect all these attacks simultaneously in as short as 3 milliseconds (ms) based on Graph Neural Network (GNN). This work generates directed attributed graphs based on CAN message streams in given message intervals. Node attributes denote data contents in CAN messages while each edge attribute represents the frequency of a typical CAN ID pair in the given interval. Afterwards, a GNN is trained based on generated CAN message graphs. Considering highly imbalanced training data, a two-stage classifier cascade is developed in this paper, which is composed of a one-class classifier for anomaly detection and a multi-class classifier for attack classification. An *openmax* layer is further introduced to the multi-class classifier to tackle new anomalies from unknown classes. To take advantage of crowdsourcing while protecting user data privacy, we adopt federated learning to train a universal model that covers different driving scenarios and vehicle states. Extensive experiment results show the effectiveness and efficiency of our methodology.**

***Index Terms*— CAN bus intrusion detection, graph neural network, two-stage classifier cascade, federated learning.**

I. Introduction

CONTROLLER Area Network (CAN bus), designed by Robert Bosch GmbH in 1983, is the most popular standard for in-vehicle network communication [1], [2]. The CAN bus is a broadcast medium, and uses a lossless bitwise arbitration method to resolve contentions during data transmissions among different Electronic Control Units (ECUs) in vehicles. In detail, every CAN message is assigned a CAN ID based on its functionality and priority. A message with a lower ID will win the contention when two messages collide. Usually messages containing crucial information, such as those related to powertrain and vehicle safety, will be assigned lower CAN IDs, while infotainment and telematics messages will have higher CAN IDs [3].

Manuscript received 1 February 2022; revised 16 November 2022; accepted 12 January 2023. Date of publication 27 January 2023; date of current version 21 February 2023. This work was supported in part by the Commonwealth Cyber Initiative (CCI) (cyberinitiative.org) and its Northern Virginia (NOVA) Node, an Investment in the Advancement of Cyber Research and Development, Innovation, and Workforce Development. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Rafael F. Schaefer. *(Corresponding author: Kai Zeng.)*

Hengrun Zhang is with the Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China (e-mail: zhanghengrun@ecust.edu.cn).

Kai Zeng is with the Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA 22030 USA (e-mail: kzeng2@gmu.edu).

Shuai Lin is with the School of Economics and Management, Shanghai Institute of Technology, Shanghai 200235, China (e-mail: ls@sit.edu.cn).

Digital Object Identifier 10.1109/TIFS.2023.3240291

A. Background and Motivation

The CAN bus protocol does not provide any authentication or encryption mechanisms. In this case, attackers have the chance to compromise the CAN bus in a vehicle through inserting forged messages. For a typical CAN ID, once forged messages overwhelm normal messages, attackers can take control of relevant operations in the targeted vehicle, which can cause severe consequences if that CAN ID is related to powertrain or vehicle safety. Furthermore, the recent development of Intelligent Transportation Systems (ITS) and Internet of Things (IoT) continuously expand the attack interfaces, which include but are not limited to sensors, Wi-Fi, and On-Board Diagnostics (OBD) [4]. In 2021, Upstream Security's research team provided a global automotive cyber security report based on 633 publicly reported incidents in the last decade, which shows an exponential growth trend in cyber attacks on connected vehicles [5].

Considering this, several Intrusion Detection Systems (IDSs) have been proposed to increase security of CAN bus. In general, if an adversary wants to make forged messages overwhelm normal ones, it can inject forged messages (denial-of-service or fuzzy attack), suspend normal messages (suspension attack), or falsify data contents of normal messages (replay or spoofing attack). **ECUs in the same CAN network usually transmit their messages with a comparatively fixed frequency, which makes the statistics of CAN message sequences comparatively stable.** In this case, if the first two attack strategies happen, message frequencies or message sequences are likely to be changed, which are the motivations for the existing message frequency or sequence-based detection methods [6], [7], [8], [9]. Besides, some existing works [10], [11], [12] consider message falsification attacks. They are based on the observation that **message contents with the same CAN ID will neither vary too much.**

The situation becomes complicated when considering message injection/suspension attacks together with message falsification attacks. Message falsification attacks will not change message frequencies or sequence patterns, which will evade IDSs only considering message injection/suspension. On the other hand, IDSs for message falsification cannot

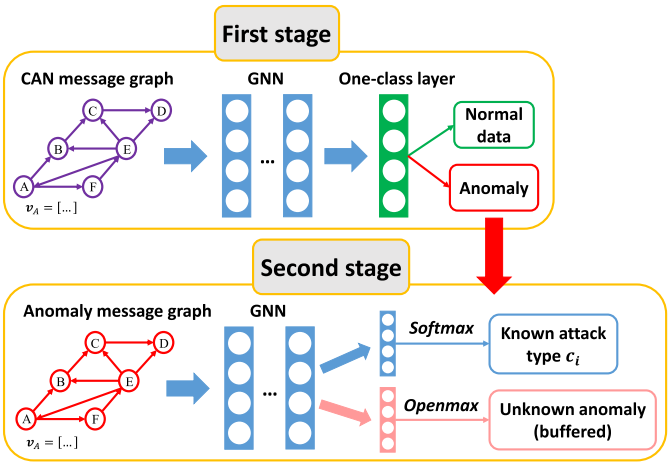


Fig. 1. Architecture of the two-stage classifier cascade.

identify Denial-of-Service (DoS) attacks [13] or bus-off attacks [14], which do not need to change normal message contents. Recently, some related works tried to detect both of the above two kinds of attacks simultaneously based on Long Short-Term Memory (LSTM) autoencoder [10], [15], [16] and bloom filtering [9]. However, these schemes need to generate a separate model and analyze relevant messages for each CAN ID, which can induce very high computation complexity and intrusion detection delay. In order to detect all the attacks, we can also apply ensemble learning [17] to train multiple classifiers, but it will still lead to increased computation complexity and detection delay.

B. Contributions

Based on the above observations, we propose a CAN bus IDS based on GNN in this paper, which can efficiently detect message injection, suspension, and falsification attacks simultaneously. Directed attributed graphs are constructed to include both statistical CAN message sequences and message contents. Therein, message sequences are described by nodes, edges and edge attributes in graphs. Data contents in messages with a typical CAN ID are preprocessed by the **READ** [18] method, and summarized as the corresponding node attributes. With these generated CAN message graphs, a two-stage GNN-based classifier cascade can be trained to build our IDS, which is illustrated in Fig. 1. The first stage has similar functionality to existing IDSs, i.e. anomaly detection. Considering that attacked data are usually hard to acquire in the real world, normal data usually dominate the training set, which will become highly imbalanced. In this case, we replace the traditional *softmax* layer in GNN with a one-class classification layer for the anomaly detection purpose. Once an attacked data sample is captured by the first-stage classifier, this data sample will further go to the second-stage classifier for attack classification. Inspired by [19], besides the *softmax* layer for multi-class classification, the second-stage classifier also has an *openmax* layer to tackle new anomalies from potentially unknown classes, which will be buffered for further investigation and open world recognition [20], [21].

Furthermore, in Section VI, we will show that different vehicle states can lead to variations in message sequences, message contents, and further message graphs. Therefore, the model trained on one vehicle will be constrained by its limited driving scenarios and vehicle states (e.g., a vehicle may mostly drive in local with low speed and a lot of stops), so cannot be applied to other vehicles with different driving scenarios and vehicle states (e.g., a vehicle may mostly drive on highway with high speed and few stops). Similarly, different driving behaviors may also restrict the comprehensiveness of the derived models. For example, an aggressive driver may accelerate a vehicle much more intensively than a conservative driver. To take advantage of crowdsourcing while protecting user data privacy, we further adopt a federated learning framework to train a universal model that covers a wide range of driving scenarios, vehicle states, and driving behaviors. This framework can even include special corner cases, such as accidents or risky situations, which can be contributed by car manufacturers during safety tests. The main contributions of this paper are summarized as follows:

- We propose a CAN bus IDS which can efficiently detect CAN message injection/suspension and message falsification attacks at the same time. Instead of a simple combination of the above two traditional IDSs, we define a CAN message graph to integrate message contents with statistical message sequences in terms of CAN ID pairs.
- We develop a GNN which is fit for directed attributed graphs. Considering that attacked data are hard to acquire in the real world, which may cause highly imbalanced training sets, we develop a two-stage classifier cascade to tackle normal and attacked CAN data respectively. An *openmax* layer is introduced to cope with new anomalies from potentially unknown classes. Furthermore, we also consider federated learning to cover different driving scenarios and vehicle states while protecting data privacy.
- We evaluate the proposed IDS through extensive experiments based on several real-world datasets. Through comparisons with three baselines, we validate that our proposed IDS can achieve similar performance to both IDSs based on statistical CAN message sequences and message contents in much shorter time. Besides, federated learning can effectively combine models derived under different driving scenarios and vehicle states, and greatly improve intrusion detection performance.

II. Related Work

A. Attack Types

According to [4], attacks to in-vehicle networks or CAN buses can be divided into message sniffing, message injection, message suspension and message falsification. Therein, the later three categories of attacks can further impact normal functionalities of vehicles.

Message injection attack is usually considered together with message suspension attack, since they are both related to message frequency or statistical message sequence in terms of CAN ID. In practice, message falsification attack is realized through a combination of message sniffing, message injection and message suspension. Based on [7], [13], [14], [22], [23], and [24], the specific attack types considered in this paper are summarized as follows:

- **Denial-of-Service (DoS):** This attack can also be called bus-off attack [14], which aims at paralyzing CAN bus systems through continuously injecting legitimate CAN messages with a low CAN ID, e.g. 0×000. Since the CAN bus uses a lossless bitwise arbitration method to tackle data transmission contention, some functionalities assigned with higher CAN IDs will never get the chance to be transmitted.
- **Fuzzy:** This attack will generate and transmit CAN messages with random CAN IDs and data contents. The CAN IDs will range from 0×000 to $0 \times 7FF$ [16], and some of them originally may not be used in the compromised vehicles. Such a type of attack can interfere with some functionalities of victims if extra lower CAN IDs are introduced into the systems. Besides, the randomly generated data contents can mislead vehicles if those CAN IDs are initially used.
- **Suspension:** This is just the message suspension attack. The attacker will try to compromise some ECUs in the CAN bus system, and stop them from sending any CAN messages.
- **Replay:** This attack will try to compromise some ECUs in the CAN bus system, and store some valid CAN messages in a particular time period, which will be transmitted later. Such a type of attack can mislead compromised vehicles since those stored CAN messages are actually outdated. If the attacker can further suspend those sniffed ECUs, it becomes message falsification attack.
- **Spoofing:** Similar to replay attack, spoofing attack will at first try to sniff some ECUs in the CAN bus system. However, spoofing attack will try to impersonate those compromised ECUs by simulating their message transmission frequencies, while the related data contents are usually forged. Message falsification attack can also be realized here by further suspending those compromised ECUs.

B. Intrusion Detection Systems

As has been discussed in Section I, CAN messages with each CAN ID are usually transmitted in a comparatively fixed frequency. Later, researchers notice that fixed frequencies can further infer stable statistical message sequences in terms of CAN ID pairs. In addition, several ECUs may need to collaborate with each other to accomplish a vehicle operation task, which can be realized through transmitting successive CAN messages. For example, after one CAN message reflecting gas increase is transmitted, we will probably see one message denoting an increase in revolutions per minute (rpm) of the vehicle engine. Then, another CAN message representing a vehicle acceleration will also be spotted [25]. Considering these two factors, **the sequences should follow comparatively fixed patterns,** which is the basis of existing CAN bus IDSs.

IDSs can be divided into anomaly detection-based and signature-based methods, which respectively identify intrusions by comparing with normal data and known attacks. Most CAN bus IDSs are anomaly detection-based methods. Message injection or suspension attacks will change CAN message frequencies or statistical message sequences in terms of CAN ID pairs, which is the basis of related works. Early works [8], [26] directly use message frequencies for intrusion detection. Later, some data analysis methods are proposed based on statistical message sequences. Such methods compare two message sequences through statistical metrics, such as cosine similarity, Pearson correlation or chi-squared test [6], [7]. If a significant change in message frequencies or sequences (metric values larger than given thresholds) is detected, they predict that intrusions happen in the second message interval. Recently, some works [6], [27] also consider machine learning based methods, such as LSTM autoencoder, to detect message injection or suspension attacks through statistical CAN message sequence reconstruction. The authors in [28] further explore the possibility to convert CAN message streams to images, and train a Generative Adversarial Network (GAN) for CAN bus intrusion detection.

All the above IDSs cannot properly tackle message falsification attacks. Existing works targeting message falsification attacks are based on the assumption that message contents or some bits therein do not have much variance. Early works consider Hamming distance between the bit representation of each two CAN messages [9] or message entropy [29], [30] for intrusion detection. For machine learning methods [11], [12], [31], they generally adopt techniques originally used for identifying human actions [32], [33], [34].

Recently, the development of natural language processing inspires some CAN bus IDSs based on LSTM [10], [15] to detect message injection/suspension attacks together with message falsification attacks. The authors in [9] and [17] also propose to consider both CAN message frequencies and message contents through bloom filtering and ensemble learning. However, these schemes need to either consider each CAN ID separately or train multiple machine learning models. Considering each CAN ID separately will induce high data collection delay, since sufficient numbers of CAN messages are required for each CAN ID. Our proposed IDS inherits the advantage of those IDSs based on CAN message sequences, which only needs an enough number of overall CAN messages. On the other hand, training multiple machine learning models will bring much higher computation complexity than training a single model. Furthermore, most of existing CAN bus IDSs, especially those based on LSTM, detect attacks on the basis of a trustworthy reference (a normal CAN message sequence in the previous time slot), which usually cannot be guaranteed during intrusion detection.

Finally, open world recognition has also drawn attention in CAN bus intrusion detection. The authors in [21] proposed a transfer learning architecture, which can be trained for unknown attacks. [20] realizes a similar functionality based on a one-class classifier. These two works can be seen as successors of our work. After our IDS buffers enough previously unknown attacks or anomalies, the open world recognition strategies can be introduced to tackle them. A comprehensive comparison between our proposed IDS and existing schemes is summarized in Table I to show the advantages of our method. Note that a comparison of detection functionality is also listed in the table. Existing IDSs for the CAN bus system can only tell whether an attack happens or

TABLE I
COMPARISON BETWEEN OUR PROPOSED IDS AND EXISTING SCHEMES (SA: STATISTICAL ANALYSIS, ML: MACHINE LEARNING)

Schemes	[6]–[8], [26], [27]	[28]	[29], [30]	[11], [12], [31]	[9], [10], [15]	[17]	Our proposed IDS
Categories	SA	ML	SA	ML	ML	ML	ML
Evaluation technology	Cosine similarity, Pearson correlation, and chi-squared test	GAN	Message entropy	Deep learning and bloom filtering	Hamming distance and LSTM	Ensemble learning	GNN
Message injection	✔	✔	✖	✖	✔	✔	✔
Message falsification	✖	✖	✔	✔	✔	✔	✔
Collection delay	Low	Low	Low	Low	High	–	Low
Computation complexity	–	–	–	–	–	High	Low
Reference needed	✔	✖	✔	✖	✔	✖	✖
Detection functionality	Binary	Binary	Binary	Binary	Binary	Binary	Open multi-class

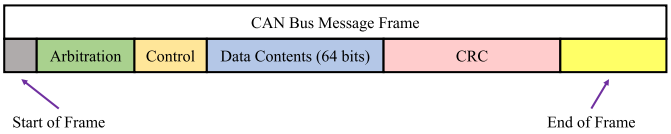


Fig. 2. CAN bus message format.

```
(1615955313.947905) can0 386#FFE000000000000B
(1615955313.948145) can0 405#0000200000000526
(1615955313.948396) can0 428#05010000997E2B
(1615955313.948614) can0 42D#7FF8451E842C24
```

Fig. 3. Streamed CAN bus messages.

not (binary), while our proposed IDS can further figure out the specific attack type, and tackle previously unknown anomalies (open multi-class).

III. PRELIMINARIES AND BASICS

A. CAN Message Description

Fig. 2 illustrates a typical CAN bus message format. Our CAN bus intrusion detection system in this paper is based on CAN ID in the arbitration field and data contents of each CAN message. In the CAN standard, a CAN ID has either 11 or 29 bits, and the version with 11 bits is usually selected for CAN buses in vehicles. Data contents have a length between 0 and 8 bytes (or 64 bits). Recently, Bosch proposed an extended CAN standard, called CAN with Flexible Data-Rate (CAN-FD). Data contents in CAN-FD messages can have a length of up to 64 bytes. In this paper, we are still based on the CAN standard with up to 8 bytes of data.

CAN bus data can be streamed through using the *candump* tool in Linux CAN subsystem, a.k.a. SocketCAN. Fig. 3 shows an example with four streamed CAN bus messages. Therein, the red and blue box respectively include CAN IDs and data contents of those four messages, which are denoted with hexadecimal numerals. The other two components in streamed CAN messages are timestamp and CAN interface name.

B. CAN Message Graph

In this paper, we use message graph to describe CAN message streams. Compared with message sequences, message graphs can further embed message contents. Graph structures have both edge attributes and node attributes, and they provide the possibility to simultaneously detect all the three categories of attacks mentioned in Section I.

Note that in order for real-time analysis, CAN messages are considered in **intervals**, which usually vary from 100 to 200 messages. Streaming such numbers of CAN messages usually costs an order of milliseconds. If the message interval is too short, i.e. less than 100 messages, the message sequence will become unstable, since many messages with higher CAN IDs do not need to be transmitted very frequently, and these CAN IDs may not appear in some of message intervals. We further validate the above analysis based on the dataset in [25]. Fig. 4 respectively shows the cosine similarity under intervals with 50, 100, and 200 CAN messages. Details for cosine similarity derivation will be discussed in Section VI-A. Based on Fig. 4, we can see that message sequences will become much more stable (higher values while lower variations in cosine similarity) when the interval is long enough (100 or 200 CAN messages).

Fig. 5 shows the conversion from a toy CAN message stream with two messages to a CAN message graph. Each node in the graph represents a CAN ID appearing in the given message interval. In this toy CAN message stream, a message with CAN ID `264` is followed by another CAN message `04A`. Therefore, we construct a directed edge from Node `264` to Node `04A` with an attribute or weight of 1. A normal CAN message graph is shown in Fig. 6. In each CAN message sequence, messages are considered in pairs similar to that shown in Fig. 5. Edge attributes or weights refer to the number of corresponding message pairs appearing in the given message interval. The structure of a graph is usually described by an adjacency matrix, denoted by $\boldsymbol{A}_{n\times n}$ in this paper. n here represents the number of nodes in a CAN message graph, which further infers the number of CAN IDs appearing in the corresponding message interval. In addition, since CAN message graphs have edge attributes, we further introduce an edge matrix $\boldsymbol{E}_{n\times n}$ to include this information.

C. CAN Message Content Preprocessing

The data content in each CAN message can be used for identifying message falsification attacks. As has been discussed in [16], signal semantics are usually described in blocks in message contents, and a proper data division can greatly improve intrusion detection accuracy. The authors in [18] proposed a well-behaved signal boundary extraction

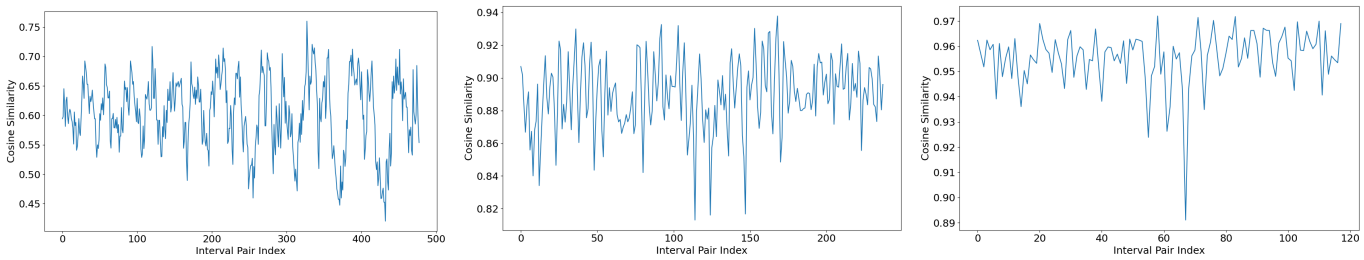


Fig. 4. Cosine similarity under different CAN message intervals. (a) 50 messages; (b) 100 messages; (c) 200 messages.

Fig. 5. A toy example showing conversion from CAN message stream to CAN message graph.

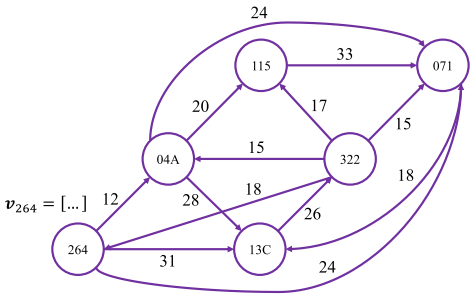


Fig. 6. CAN message graph.

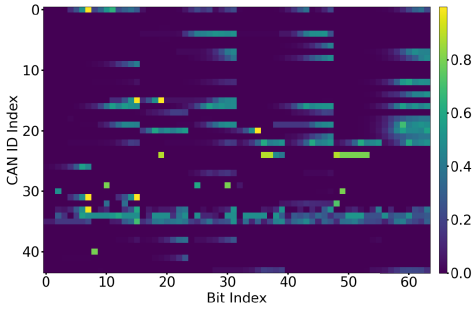


Fig. 7. Bit-flip rate heatmap.

algorithm for CAN message contents, called READ, which divides message contents based on the bit-flip rate. The general idea is that for each signal semantic, the most significant bit in the related data block will vary much slower than the least one. In this case, if a bit with a high bit-flip rate is followed by one with a low rate, these two bits probably belong to two different signal semantics or data blocks.

In order to calculate the bit-flip rate, a certain number of CAN messages need to be collected for each CAN ID. Note that the CAN message content division only needs to be conducted once based on the training set before our IDS model training. In this case, we do not need to worry about any data collection delay during intrusion detection. Besides, we neither need to tackle any CAN message content associated with a new CAN ID during intrusion detection, since it just infers a fuzzy attack according to Section II-A.

A bit-flip rate heatmap based on a dataset in [6] is illustrated in Fig. 7, which is composed of 44 CAN IDs and 23,963 normal CAN messages. Here, CAN IDs are indexed according to their values. From this heatmap, we can intuitively figure out several data blocks in different CAN IDs. A signal boundary will be applied where a light pixel is followed by a dark pixel. Besides, according to [18], we do not need to care about small variations in bit-flip rate. In this case, we further introduce a magnitude array (decimal logarithm of bit-flip rate), and look for drops in this array instead. Finally, for convenience of GNN design in the next section, we need to describe all CAN message contents with a node matrix $\boldsymbol{V}_{n\times n'}$. After applying READ, message contents with different CAN IDs will be separated into different numbers of data blocks, each of which is further converted to a decimal number. n' in the node matrix denotes the maximal number of data blocks across all CAN IDs. We will further align the number of data blocks for all CAN IDs by complementing some 0s in the front.

IV. First-Stage Classifier

A. Graph Neural Network

Graph learning has recently drawn increasing attention. All current graph learning frameworks can be divided into three levels, i.e. node level, edge level and graph level [35]. Therein, node level algorithms can be utilized to determine whether a typical CAN message is forged or not, which is similar to the idea in [9]. Our proposed CAN bus IDS is based on graph level algorithms. All three levels of models start with some graph convolution layers, and graph level schemes realize graph classification tasks via further introducing pooling and readout layers. In this paper, we are based on GNN proposed in [36] for CAN bus intrusion detection.

The convolution layer of GNN in this paper takes the following form:

$$\boldsymbol{Z} = f(\tilde{\boldsymbol{D}}^{-1}\bar{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}) \tag{1}$$

The original GNN in [36] only targets undirected graphs without edge attributes and self-loops. In order to make it fit for CAN message graphs in this paper, we at first concatenate the node matrix $\boldsymbol{V}$ with edge matrix $\boldsymbol{E}$ to generate a descriptor for CAN message graphs, denoted as $\boldsymbol{X}_{n\times(n+n')}$. Such an operation actually embeds edge attributes into node attributes. In other words, attributes of edges starting from a typical node are considered as part of attributes of that node. $\bar{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ represents the adjacency matrix with added self-loops, which are realized through an identity matrix $\boldsymbol{I}$. Since CAN message graphs considered in this paper are directed and allow self-loops, we further tackle the adjacency matrix $\boldsymbol{A}$ to make all diagonal elements 0 before adding $\boldsymbol{I}$. $\tilde{\boldsymbol{D}}$ is a diagonal degree matrix with $\tilde{d}_{ii} = \sum_{j=1}^{n} \bar{a}_{ij}$, where $\tilde{d}_{ii}$ and $\bar{a}_{ij}$ are

respectively elements in $\tilde{\boldsymbol{D}}$ and $\bar{\boldsymbol{A}}$ with the corresponding indices. $\boldsymbol{W} \in \mathbb{R}^{(n+n')\times c}$ denotes model parameters in the convolution layer, where c is the number of feature channels in the convolution layer. $f(\cdot)$ is a pointwise nonlinear activation function, such as Rectified Linear Unit (ReLU).

The whole graph convolution process can be explained as follows. Node and edge attributes are at first fit into the convolution layer through a linear feature transformation $\boldsymbol{XW}$. Afterwards, for each node, its "channel descriptor", denoted by $\boldsymbol{Y} = \boldsymbol{XW}$, will be propagated to its neighborhood including itself through $\bar{\boldsymbol{A}}\boldsymbol{Y}$. Here, we can see that why we need to include self-loops in the adjacency matrix $\boldsymbol{A}$. Then, we normalize the propagation results by multiplying them with the diagonal degree matrix $\tilde{\boldsymbol{D}}$, which aims at keeping a fixed feature scale after graph convolution. Finally, a pointwise nonlinear activation function $f(\cdot)$ is applied before outputting the graph convolution results.

Similar to convolution neural networks applied in image processing, in order to capture graph substructure features in different scales, we need to apply and stack multiple convolution layers. In this case, in the t-th convolution layer, we further have:

$$\boldsymbol{Z}_t = f(\tilde{\boldsymbol{D}}^{-1}\bar{\boldsymbol{A}}\boldsymbol{Z}_{t-1}\boldsymbol{W}_t) \tag{2}$$

Here, $\boldsymbol{Z}_{t-1}$ and $\boldsymbol{Z}_t$ respectively represent the input and output matrix of the t-th convolution layer, and we further have $\boldsymbol{Z}_0 = \boldsymbol{X}$. $\boldsymbol{W}_t \in \mathbb{R}^{c_{t-1}\times c_t}$ includes model parameters in the t-th convolution layer, with $c_0 = n+n'$. After acquiring all $\boldsymbol{Z}_t$ from those convolution layers, the outputs are stacked through concatenation, i.e. $\boldsymbol{Z}_{1:t} = [\boldsymbol{Z}_1, \boldsymbol{Z}_2, \cdots, \boldsymbol{Z}_t]$.

Up to now, the design of graph convolution layers has finished. For graph classification tasks, pooling and readout layers need to be further introduced. Besides pooling layers for downsampling, the authors in [36] further introduce a sorting operation to sort nodes in a graph according to their structural roles, which can make similar graphs fit into readout layers in a comparatively consistent node order. Such an operation is beneficial to tackling CAN message graphs in this paper. As will be discussed in Section VI-A, vehicle states can sometimes cause variations in CAN message graphs, since some ECUs in vehicles are not always activated in order to prolong battery life. Such a situation can further induce node indexing issues, which can be solved by this sorting operation. The readout layers are usually composed of some 1-Dimensional (1-D) convolution layers and dense layers.

B. One-Class Classification

Traditionally, the last layer of a neural network is a *sigmoid* or *softmax* layer for classification tasks. However, to the best of our knowledge, nearly all the current attacked data are simulated for CAN bus IDS evaluation. Intrusions to CAN buses are hard to acquire in real vehicles, which will make the training set highly imbalanced. Considering this, the first-stage classifier will be designed as an anomaly detection-based IDS. The conventional *sigmoid* or *softmax* layer will be replaced with a one-class classification layer, and only normal CAN bus data will be used to train the model.

One-class classification is widely used for anomaly detection, which can train a classifier only by using normal data. One-Class Support Vector Machine (OC-SVM) [37] and Support Vector Data Description (SVDD) [38] are currently two most popular related algorithms. Compared with OC-SVM, SVDD has better computation scalability and better performance in tackling curse of dimensionality [39]. Therefore, we introduce SVDD to our GNN for intrusion detection.

Similar to OC-SVM, SVDD comes from the traditional Support Vector Machine (SVM). However, instead of using a hyperplane to define the soft classification border, SVDD tries to find a hypersphere with a center $\boldsymbol{o} \in \mathbb{R}^{c_g}$ and a radius $r > 0$ as the border. Here, c_g represents the output dimensionality of the last dense layer in our GNN. In general, the related optimization problem of SVDD can be defined as follows:

$$\arg\min_{r,\boldsymbol{o},\boldsymbol{\xi}} r^2 + \frac{1}{\nu m}\sum_{k=1}^{m}\xi_k \tag{3a}$$

subject to,

$$\forall k,\ ||\phi_{c_g}(\boldsymbol{X}_k) - \boldsymbol{o}||^2 \le r^2 + \xi_k,\ \xi_k \ge 0 \tag{3b}$$

In Eq. (3a), $\boldsymbol{\xi}$ is called a slack vector, in which all slack variables $\xi_k \ge 0$ altogether construct a soft classification border or hypersphere. The soft classification border allows some of data samples or support vectors originally from one class to cross the borderline and fall in another class, which helps to avoid model overfitting via reducing the radius of the hypersphere to a certain extent. On the other hand, the hypersphere cannot be shrunk without any limitation, since this can potentially induce model underfitting. In this case, a weight $\nu \in (0, 1]$ is introduced here to balance model overfitting and underfitting. In Eq. (3b), the function $\phi_{c_g}(\boldsymbol{x}_k)$ is the kernel function usually used in SVM for space mapping. Here, we can consider the whole architecture of GNN except the last one-class classification layer as the kernel function. $\boldsymbol{X}_k$ refers to the k-th CAN message graph discussed in Section IV-A in the training set. $||\cdot||^2$ is the $\mathcal{L}^2$ norm.

With a small trick, we can embed the constraint of the above optimization problem into its objective function. At first, we can set:

$$\xi_k = \max\{0, ||\phi_{c_g}(\boldsymbol{X}_k) - \boldsymbol{o}||^2 - r^2\} \tag{4}$$

Then, we replace ξ_k in Eq. (3a) with Eq. (4). Besides, we also consider a model parameter regularization term to further dodge model overfitting. At last, we can get the following optimization problem without any constraint, which can be seen as the loss function of our whole GNN model.

$$\arg\min_{r,\boldsymbol{W}_{all}} r^2 + \frac{1}{\nu m}\sum_{k=1}^{m}\max\{0, ||\phi_{c_g}(\boldsymbol{X}_k) - \boldsymbol{o}||^2 - r^2\} + \frac{\lambda}{2}||\boldsymbol{W}_{all}||^2 \tag{5}$$

Here, λ is the weight decay to balance the penalty for large model parameters. $\boldsymbol{W}_{all}$ denotes a vector which includes all the parameters in our GNN model except the last one-class layer. We no longer consider the center of the hypersphere $\boldsymbol{o}$ as a coefficient to be tuned here, and follow the strategy in [39] to fix $\boldsymbol{o}$. Some data samples are at first chosen from the training

set. Afterwards, we let these data samples pass through the initialized whole GNN except the last one-class classification layer, and record the outputs from the last dense layer. Finally, the center $\boldsymbol{o}$ is set to the mean of these outputs. Note that the above loss function needs to investigate all the CAN message graphs in the training set at the same time. If gradient descent is used for model parameter update, CAN message graphs for training need to be considered in batches. With the concern of computation complexity, the training process in this paper is based on mini-batch Stochastic Gradient Descent (SGD).

V. Second-Stage Classifier

After the first-stage classifier filters out all the anomalies, they will further be sent to the second-stage classifier to detect the specific attack type. The second-stage classifier can be seen as a signature-based IDS. In order to tackle the potential new anomalies from unknown classes, such as zero-day attacks, the second-stage classifier further introduces an *openmax* layer.

During the training process, the second-stage still utilizes the traditional *softmax* function in the output layer. The *openmax* layer is deployed during intrusion detection to replace the *softmax* layer, which revises the output of the penultimate layer in the original classifier to reject data samples not close enough to any known class. In this paper, the output of the penultimate layer can be denoted as an activation vector $\mathcal{A}_{C\times 1}$, where C represents the number of known types of attacks. The design of the *openmax* layer is based on meta-recognition algorithms [40], [41], in which the distribution of the activation vector is analyzed through Extreme Value Theory (EVT), and the Weibull distribution is figured out. In detail, the EVT fitting in the *openmax* layer design adapts the concepts of Nearest Class Mean [42], [43] or Nearest Non-Outlier [44] per attack type to the activation vector. Each attack type c_i is represented with a mean activation vector (MAV), which is derived through averaging all the activation vectors of the training samples belong to this attack type, denoted as $\bar{\mathcal{A}}_{c_i}$. Then, within each attack type, we calculate the Euclidean distance between the activation vector of each training sample $\mathcal{A}_{c_i,j}$ and the MAV $\bar{\mathcal{A}}_{c_i}$. Afterwards, we conduct the Weibull fitting for each attack type based on EVT:

$$||\mathcal{A}_{c_i,j} - \bar{\mathcal{A}}_{c_i}||_2 \sim Weibull(\alpha_i, \beta_i, \boldsymbol{\gamma}_i) \tag{6}$$

where α_i, β_i and $\boldsymbol{\gamma}_i$ respectively denote the shape, scale and position parameter to describe a Weibull distribution. $\boldsymbol{\gamma}_i$ has the same dimensionality as $\bar{\mathcal{A}}_{c_i}$ for shifting the data. All the Weibull fitting is pre-computed at the end of the training process.

When a new anomaly CAN message graph comes to the second-stage classifier, the *openmax* layer will follow Algorithm 1 for attack type classification with potential unknown anomaly rejection. From Algorithm 1, we can see that the *openmax* layer in general adapts the *softmax* function to open world recognition, which is realized by introducing an extra class c_0 (line 8 of Algorithm 1). Such a class is used to include all the anomaly CAN message graphs that are not quite similar to any existing attack type, and further infers a potential unknown class. Line 4 of Algorithm 1 generates probabilities in which the new anomaly CAN message graph belongs to a selected number of top-ranked classes or attack types. Such probabilities are derived from the corresponding fitted Weibull distributions, and used to revise the activation vector (line 5 of Algorithm 1).

Algorithm 1 *Openmax* Attack Type Classification With Potential Unknown Anomaly Rejection

Input : $\mathcal{A}(\boldsymbol{X}) = [a_1, a_2, \cdots, a_C]^T$: activation vector of the new anomaly CAN message graph $\boldsymbol{X}$. $\bar{\mathcal{A}}_{c_i}$: MAV representing the known attack type c_i. $\{\alpha_i, \beta_i, \boldsymbol{\gamma}_i\}$: parameter set describing the fitted Weibull distribution related to the known attack type c_i. κ: number of "top" classes to revise.

Output: c^*: attack type, unknown class if $c^* = c_0$ or probability $P(c = c^*|\boldsymbol{X}) < \epsilon$

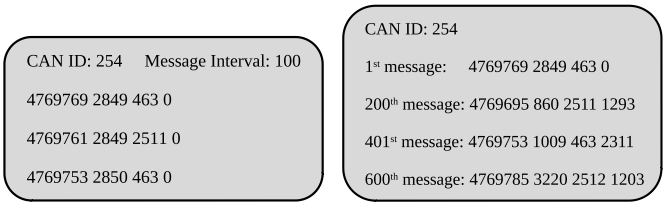
1 Initialize $\boldsymbol{S} = [s_1, s_2, \cdots, s_C] \leftarrow \boldsymbol{0}$, $\boldsymbol{\omega} = [\omega_1, \omega_2, \cdots, \omega_C] \leftarrow \boldsymbol{1}$;
2 $\boldsymbol{S} \leftarrow argsort(\mathcal{A}(\boldsymbol{X}))$;
3 **for** $i = 1, 2, \cdots, \kappa$ **do**
4 $\quad \omega_{s_i} \leftarrow 1 - \frac{\kappa - i}{\kappa} e^{-(\frac{||\mathcal{A}(X) - \bar{\mathcal{A}}_{s_i} - \boldsymbol{\gamma}_{s_i}||_2}{\beta_{s_i}})^{\alpha_{s_i}}}$;
5 $\mathcal{A}(\boldsymbol{X}) \leftarrow \mathcal{A}(\boldsymbol{X}) \circ \boldsymbol{\omega}$; `//Element-wise product.`
6 $a_0 \leftarrow \sum_{i=1}^{C} a_i(1 - \omega_i)$;
7 **for** $i = 0, 1, \cdots, C$ **do**
8 $\quad P(c = c_i|\boldsymbol{X}) = \frac{e^{a_i}}{\sum_{j=0}^{C} e^{a_j}}$;
9 $c^* = \arg\max_c P(c = c_i|\boldsymbol{X})$;

VI. Federated Graph Neural Network Learning

A. Vehicle State and Can Message Relationship

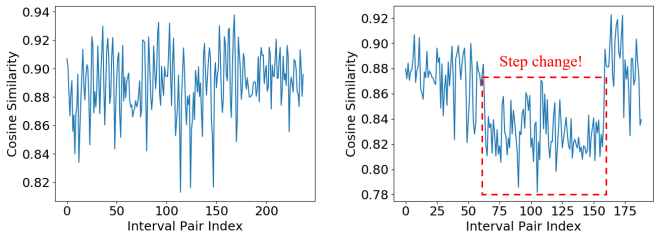
Currently, existing works are based on the assumption that CAN bus intrusions can change statistical message sequences and message contents. In fact, changes of vehicle states can also cause CAN message variations but in a *reasonable* way. Intuitively, CAN message contents will change to reflect different vehicle states. On the other hand, vehicle states can also affect CAN message sequences, since some ECUs may not get activated all the time to prolong battery life in vehicles. For example, tire pressure sensors will sleep most of the time and wake up only when vehicles start to travel at high speeds (over 40 km/h), or during diagnosis and the initial CAN ID binding phases [45].

We further validate the above claims based on the dataset in [25] without any attacks, which is collected when the vehicle has an acceleration process from 0 to 30 mph (about 48 km/h) followed by a deceleration process back to 0 mph. Therein, the targeted CAN ID is `254`, which is related to vehicle speed information, and the dataset has 707 messages with this CAN ID. Message contents are preprocessed based on READ. Fig. 8(a) shows all relevant CAN messages within a 100 message long interval, which includes the 1st CAN message, while Fig. 8(b) lists the data content of the 1st, 200th, 401st and 600th message. In a single message interval, we can consider that the vehicle is in the same state, while two CAN messages from two different intervals may correspond to



(a) (b)

Fig. 8. CAN message contents with CAN ID 254. (a) In the same message interval; (b) Several message intervals apart.



(a) (b)

Fig. 9. Cosine similarity comparison between 2 message sequences with (a) 1 message interval apart, and (b) 50 message intervals apart.

two different vehicle states. From Fig. 8(b), we can observe obvious value changes in the second and fourth data block, which are not reflected in Fig. 8(a). Besides, the fourth data block seems to be related to the vehicle speed, which makes its value vary within a certain range considering the vehicle speed limit. In the other words, message content changes with CAN ID 254 should have the above two features, or be *reasonable*. Note that the value of the third data block actually jumps between around 463 (odd indices) and around 2511 (even indices), which is not considered as a change here.

Next, we show that vehicle states can also affect CAN message sequences. We split the whole dataset with 23,963 CAN messages to 239 message intervals with each having 100 messages. For each message interval, we count the number of appearing times for each possible CAN ID pair, and generate the corresponding statistical message sequence. Note that in order to make a proper comparison, we at first go through all CAN message pairs in the dataset to record all possible CAN ID pairs. Afterwards, we compare two message sequences based on cosine similarity. Fig. 9 shows comparison results under two situations. In the first scenario, we compare any two consecutive CAN message sequences. In detail, cosine similarity between 1st and 2nd message sequence is indexed in 1st interval pair, cosine similarity between 2nd and 3rd message sequence is indexed in 2nd interval pair, and so on. In the second scenario, we compare any two CAN message sequences with 50 message intervals apart. In detail, cosine similarity between 1st and 51st message sequence is indexed in 1st interval pair, cosine similarity between 2nd and 52nd message sequence is indexed in 2nd interval pair, and so on.

Based on the above two comparisons, we can notice a step change when two message sequences are 50 intervals apart, which cannot be observed in the fist situation. In a short time, a.k.a when we compare two consecutive message sequences, we can consider that the vehicle state does not change. In this case, message sequences vary within a certain range, which is reflected in Fig. 9(a). On the other hand, two message sequences with 50 intervals apart will correspond to two different vehicle states. Since some ECUs, such as tire pressure sensors, will only get activated in some vehicle states, noticeable variations, such as the step change in Fig. 9(b), can happen. In addition, message sequence changes may further have intrinsic connections with some message content variations, such as the above CAN ID 254 related to vehicle speed. In order to differentiate CAN message variations caused by vehicle state changes, we need to collect data from as many driving scenarios and vehicle states as possible, which may have to come from different vehicles considering the limitations discussed in Section I. Federated learning can then be applied for model training while protecting data privacy.

B. Federated Learning

Federated learning can be seen as a type of distributed machine learning, in which a cloud server collaborates with several local users for model training. The whole training set in federated learning is actually composed of those local datasets owned by each local device, which are non-Independent and Identically Distributed (non-IID). During model training, each local device will train a local model based on its own dataset, and only upload model parameters to the cloud server for aggregation, through which data privacy can get protected. In this paper, the non-IID property can be caused by different driving scenarios or vehicle states. In order to improve intrusion detection performance of our IDS, we consider a federated learning environment. Vehicles in different status can train our GNN model based on their own CAN bus data, and then upload model parameters to the cloud server for model aggregation.

In this paper, we evaluate two federated learning schemes, i.e. FedAvg [46] and FedProx [47], which are both based on mini-batch SGD to update local parameters. The whole training process can be divided into a certain number of communication rounds between the cloud server and vehicles. In each communication round, part of vehicles are selected to upload their model parameters to the cloud server for aggregation after local iterations. In the l-th selected vehicle, some generated CAN message graphs are selected at each local iteration. During the training of the first-stage classifier, for each selected message graph $\boldsymbol{X}_k$, we will derive the corresponding loss based on the loss function defined in Eq. (5) and current local model parameters in the l-th vehicle, denoted as $f(r_l, \boldsymbol{W}_{all,l}, \boldsymbol{X}_k)$. Under mini-batch SGD, the mini-batch loss of the l-th selected vehicle can be calculated by averaging losses across all selected message graphs:

$$F(r_l, \boldsymbol{W}_{all,l}) = \frac{1}{m_l} \sum_{k \in P_l} f(r_l, \boldsymbol{W}_{all,l}, \boldsymbol{X}_k) \tag{7}$$

where P_l is the set of selected message graphs in the l-th vehicle at each local iteration, and m_l denotes the total number of elements in P_l.

When it comes to local model parameter update in the l-th vehicle, the mini-batch gradient $\boldsymbol{G}_\tau(r_l, \boldsymbol{W}_{all,l})$ at each local iteration τ can be derived by computing partial derivatives of $F(r_l, \boldsymbol{W}_{all,l})$ on each model parameter. Then, model parameters in the l-th vehicle will be updated as follows:

$$[r_l, \boldsymbol{W}_{all,l}]_{\tau+1} \leftarrow [r_l, \boldsymbol{W}_{all,l}]_\tau - \eta \boldsymbol{G}_\tau(r_l, \boldsymbol{W}_{all,l}) \tag{8}$$

Here, we use $[r_l, \boldsymbol{W}_{all,l}]_\tau$ to denote a vector which includes all parameters in our one-class GNN model. η represents the learning rate. After a given number of iterations, all the L selected vehicles will upload their learned model parameters to the cloud server for aggregation:

$$[r, \boldsymbol{W}_{all}] = \frac{1}{m} \sum_{l=1}^{L} m_l [r_l, \boldsymbol{W}_{all,l}] \tag{9}$$

The result is considered as the globally learned model parameters after each communication round. Finally, $[r, \boldsymbol{W}_{all}]$ will be broadcast to all vehicles for the following possible local iterations.

The above model parameter update is generally based on FedAvg, which will not necessarily provide convergence guarantee. FedProx improves the convergence performance by further introducing a proximal term. Such a term can ensure that updated local model parameters will not drift away from the global model parameters derived in the last communication round. Mathematically, the mini-batch loss function of the i-th vehicle will be modified to:

$$F(r_l, \boldsymbol{W}_{all,l}) \leftarrow F(r_l, \boldsymbol{W}_{all,l}) + \frac{\mu}{2} ||[r_l, \boldsymbol{W}_{all,l}] - [r, \boldsymbol{W}_{all}]_{pre}||^2 \tag{10}$$

Here, we use $[r, \boldsymbol{W}_{all}]_{pre}$ to represent the global model parameters derived in the previous communication round. μ is a hyperparameter to balance the effect of the proximal term.

The training of the second-stage classifier follows a similar process except that the traditional *softmax* and cross-entropy function are respectively used here as the output layer and loss function. In addition, the deployment of the *openmax* layer needs the Weibull fitting, which is conducted at the end of model training. In the federated learning scenario, the cloud server will at first request vehicles to upload the activation vectors of anomaly CAN message graphs and the corresponding labels in their training sets. Afterwards, the Weibull fitting will be conducted in the cloud server. Since only the activation vectors instead of raw data need to be uploaded, the privacy guarantee of federated learning is not violated. Besides, the Weibull fitting only needs one single communication round between the cloud server and vehicles, which will not induce much extra communication overhead.

VII. Experimental Results

All the experiments in this paper are conducted in a simulated platform based on an Intel(R) Core(TM) 3.4 GHz 4-core processor, 16 GB RAM, and an NVIDIA GeForce GTX 1080Ti GPU. The used datasets are from the real world.

A. Datasets and Experiment Setup

In this section, we will conduct extensive experiments to evaluate our GNN-based IDS under CAN message injection attacks, message suspension attacks and message falsification attacks. In detail, for the first-stage classifier, we compare our IDS with three baselines on anomaly detection, which respectively represent IDSs considering statistical CAN message sequences and message contents. When it comes to the second-stage classifier, we evaluate its performance in classifying specific attack types and identifying new anomalies from unknown classes.

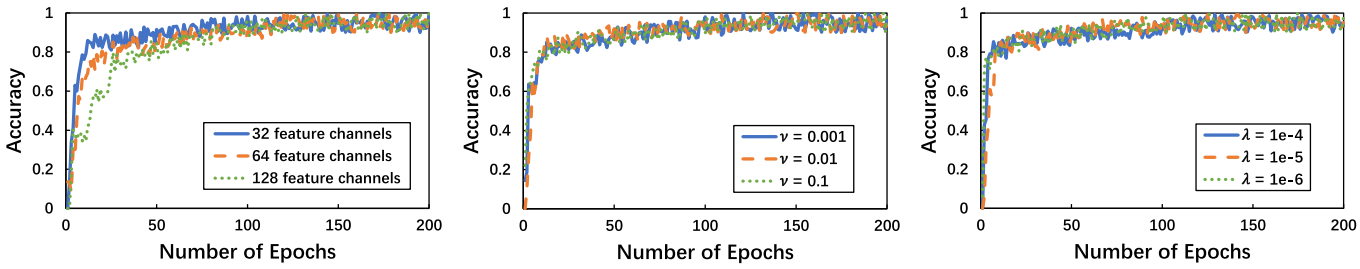
In the first baseline [6], the authors developed three kinds of strategies to detect CAN bus intrusions via message sequences, i.e. thresholds for cosine similarity, Pearson correlation and LSTM. Data for evaluation were collected on a Ford Transit 500 [25], which were separated into three sets. The first dataset is composed of 23,963 normal CAN messages, while message injection attacks are considered in the remaining two datasets. In the second and third dataset, CAN IDs related to vehicle speed and rpm (`254` and `115`) are respectively targeted, and compromised CAN messages are randomly injected into the these two datasets after given time spots. In total, the second and third dataset each have 88,492 and 30,308 CAN messages.

The second and third baseline [10], [16] both designed IDSs based on message contents and LSTM. The evaluation dataset is *CAN Signal Extraction and Translation Dataset* provided by Hacking and Countermeasure Research Lab [48]. In order to make reasonable comparisons, we follow the strategies in [16] and [49] to generate anomaly data with the five attack types discussed in Section II-A, i.e. DoS, fuzzy, suspension, replay, and spoofing attack.

We generally follow the architecture of GNN model in [36]. Such GNN model at first has four graph convolution layers, with the last layer having only one feature channel for the convenience of node sorting, which is realized in a *SortPooling* layer. The *SortPooling* layer is followed by two 1-D convolution layers, one *MaxPooling* layer, one dense layer and one dropout layer. The original *softmax* layer is replaced with SVDD for the first-stage classifier, and the *openmax* layer is considered in the second-stage classifier. During the GNN model training, tons of hyperparamters can be tuned. In this paper, we focus on the number of feature channels in the first three graph convolution layers. We also test different settings of two hyperparameters in SVDD, i.e. ν and λ. In addition, the hyperparameter μ in FedProx will also be explored. We follow the settings in [36] for the remaining hyperparameters. In detail, the output of the *SortPooling* layer is set to a $k_s \times \sum_{t=1}^{4} c_t$ tensor, where k_s is set to a value which is not larger than the number of nodes in 60% of CAN message graphs in the training set. The first 1-D convolution layer has 16 feature channels followed by a *MaxPooling* layer with a filter size 2 and step size 2. The second 1-D convolution layer has 32 feature channels, a filter size 5 and step size 1. The dense layer has 128 hidden neurons, followed by a dropout layer with a 0.5 drop rate. For activation functions, hyperbolic tangent function (tanh) is selected in graph convolution layers, and ReLU in all the other necessary layers. Mini-batch SGD is optimized with Adam [50].

B. Statistical Can Message Sequences

In this subsection, we at first evaluate training performance under different GNN model settings, and select the best setting based on experimental results. In detail, we explore the number of feature channels in graph convolution layers with $\{32, 64, 128\}$, ν with $\{0.001, 0.01, 0.1\}$ and λ with $\{1e{-}4, 1e{-}5, 1e-6\}$. We use the first dataset in the first baseline as the training set, which only contains normal CAN messages. For message intervals, we test the case with a length of 100 CAN



(a) (b) (c)

Fig. 10. GNN model hyperparameter setting comparison related to the first baseline. (a) Feature channel; (b) ν; (c) λ.

TABLE II
COMPARISON RESULTS WITH THE FIRST BASELINE (IN %), ANOMALY THRESHOLD IN THE FIRST BASELINE: 0.87

Attack	IDS	Accuracy	Prediction	Recall	F1
Rpm	CS	96.65	91.40	96.59	93.92
	PC	97.32	93.48	97.73	95.56
	LSTM-CS	97.32	92.47	97.73	95.03
	LSTM-PC	96.80	91.49	97.73	94.51
	GNN	**100**	**100**	**100**	**100**
Speed	CS	89.20	68.29	88.89	77.24
	PC	90.57	71.60	92.06	80.55
	LSTM-CS	96.93	89.71	95.31	92.43
	LSTM-PC	96.93	88.57	96.88	92.54
	GNN	**99.41**	**96.92**	**100**	**98.44**

TABLE III
SCALABILITY EVALUATION (IN %)

Attack	Interval Length	Accuracy	Prediction	Recall	F1
Rpm	150	97.65	93.48	97.73	95.56
	200	97.06	92.39	96.59	94.44
Speed	150	94.71	80.60	85.71	83.08
	200	96.43	87.14	96.83	91.73

messages. The size of mini-batches is 10. Each setting is based on a 10-fold cross validation, which is further run for 10 times. The averaged accuracy curves under different hyperparameter settings are shown in Fig. 10. Based on those comparisons, we can see that the number of feature channels in graph convolution layers can affect convergence performance, while the other two hyperparameters do not have too much impact. In this experiment, we choose 32 feature channels, $\nu = 0.01$ and $\lambda = 1e-6$.

Afterwards, we conduct comparisons with the first baseline to show the effectiveness of our proposed IDS in detecting CAN message injection attacks. Note that in the original second and third dataset of the first baseline, data contents of injected CAN messages also get changed. In detail, all injected rpm and speed messages respectively end with "FFF" and "FFFF". In order to reflect the ability of our IDS in detecting intrusions only changing statistical message sequences, we randomly select one of normal messages with the same CAN ID in the same message interval, and use its data content in the relevant injected CAN messages. We make these modifications to simulate DoS attacks or bus-off attacks, which cannot be identified by IDSs considering CAN message contents. The comparison with the first baseline based on a 100 CAN message long interval is shown in Table II, in which the considered metrics include accuracy, precision, recall and F1-score. In this table, rpm and speed respectively represent those two corresponding attacks. CS, PC, LSTM-CS and LSTM-PC respectively represent the three kinds of intrusion detection strategies in the first baseline, i.e. thresholds for Cosine Similarity, thresholds for Pearson Correlation and statistical CAN message sequence reconstruction based on LSTM for intrusion detection. GNN corresponds to our IDS. Based on the comparison, we can see that our IDS can achieve even better performance than the first baseline in detecting CAN message injection attacks.

Besides, in order to evaluate the scalability of our IDS, we further consider the situation that message intervals have different lengths during intrusion detection, i.e. 150 and 200. The results are shown in Table III. From this table, we can see that our IDS can still achieve fairly high performance even when the length of message intervals during intrusion detection is different from the length during model training.

At last, we consider in this paper that message intervals longer than 200 will impact real-time performance of IDSs, since collecting 200 CAN messages will cost more than 100 ms. The average prediction time of our IDS for each CAN message graph is 3 ms.

C. CAN Message Contents

The related dataset used in this subsection is at first separated into a training set (the first 80%) and a test set (the last 20%). We assume here that the first 80% dataset should be able to cover most of vehicle states appearing in the last 20% of the whole data, which can be validated by the following experimental results. Similar to Section VII-B, the training set here also only contains normal CAN messages. The message interval length is set to 100 for both the training and test set. Although the second and third baseline consider CAN message contents for intrusion detection, they can both detect message injection (DoS and fuzzy), message suspension, and message falsification (replay and spoofing) attack at the same time. Therefore, we also apply all the five attack types in the test set for comprehensive comparisons. Those attacks are randomly deployed in different message intervals of the test set. GNN model hyperparameters and the mini-batch size are the same as those selected in the last subsection. Table IV shows related comparison results, where LSTM-P and CLAM respectively denote the second and third baseline.

TABLE IV
COMPARISON RESULTS WITH THE SECOND AND THIRD BASELINE (IN %), ANOMALY THRESHOLD IN THE SECOND AND THIRD BASELINE: 0.83

Attack	IDS	Accuracy	Prediction	Recall	F1
DoS	LSTM-P	97.26	94.20	91.60	92.90
	CLAM	98.75	95.70	97.10	96.40
	GNN	**100**	**100**	**100**	**100**
Fuzzy	LSTM-P	97.10	93.30	93.30	92.80
	CLAM	97.80	96.20	95.20	96.70
	GNN	**98.14**	**97.56**	**97.01**	**97.28**
Suspension	LSTM-P	97.38	91.90	93.80	92.90
	CLAM	97.90	93.90	94.90	94.40
	GNN	**99.72**	**98.88**	**99.73**	**99.30**
Replay	LSTM-P	97.25	92.90	95.80	94.30
	CLAM	**97.75**	**94.90**	**95.80**	**95.30**
	GNN	97.73	94.71	95.66	95.18
Spoofing	LSTM-P	95.75	91.80	90.80	91.30
	CLAM	97.00	94.80	92.90	93.80
	GNN	**97.91**	**96.10**	**94.30**	**95.19**

TABLE V
REAL-TIME PERFORMANCE COMPARISON

IDS	NoO	DCT (ms)	DT (ms)	TT (ms)
LSTM-P	100 per CAN ID	$\geq$ 991.20	7.60	$\geq$ 998.80
CLAM	12 per CAN ID	$\geq$ 110.84	1.35	$\geq$ 112.19
GNN	100 in total	48.87	3.20	**52.07**

From the results, we can see that in general, our IDS can achieve comparable performance to the second and third baseline. After taking a deeper insight, we find that message falsification attacks, i.e. replay and spoofing attacks, are hard to detect if related data contents only have slight changes compared with normal CAN message contents, which explains why our IDS performs a little less effective towards message falsification attacks. Considering the robustness of CAN bus, IDSs targeting statistical message sequences usually can achieve high performance, since injected CAN messages should overwhelm normal messages to take control of CAN buses, especially those DoS attacks, which will usually make significant changes to statistical message sequences. However, message falsification attacks can easily control CAN buses via slightly modifying normal CAN messages. For example, a speed reading change from 35 km/h to 40 km/h can trigger the activity of some ECUs, such as tire pressure sensors mentioned in Section VI-A. It is tricky to judge those slightly changed data contents, which sometimes can just be noise. How to balance the sensitivity of related IDSs is worth considering in practical applications.

Although LSTM-P and CLAM can detect all the three kinds of attack at the same time, they both need a certain number of observations for each CAN ID to detect intrusions based on LSTM, which can induce high data collection delay and impact real-time performance. Table V shows a real-time performance comparison with them. In this table, NoO, DCT, DT, and TT respectively represent Number of Observations, Data Collection Time, Detection Time, and Total Time for each intrusion detection. The NoO of LSTM-P and CLAM are the number for each CAN ID while our GNN corresponds to the total number of CAN messages. Based on our observations, even for CAN IDs with the highest message frequency, only 4 to 6 messages can be found in each 100 CAN messages. In this case, the total number of required CAN messages before each intrusion detection have to be much larger than 200 in LSTM-P and CLAM, or over 2 times larger than the necessary number in our proposed IDS. Such a number difference can further cause DCT difference, and seriously impact the real-time performance, which is also reflected in Table V. The DT of our proposed IDS is 3.20 ms in average. Although it is larger than the DT of CLAM, the gap is almost negligible compared with the difference of DCT, which is further reflected in TT. Last but not least, the memory consumption of our first-stage classifier for intrusion detection is 354.6 KB, which is also less than that of LSTM-P (13,417 KB) and CLAM (682 KB).

D. Attack Type Classification

In this subsection, we will evaluate the performance of our second-stage classifier. We still use the dataset in the last subsection, follow the strategies in [16], [49] to generate anomaly data with the five attack types, and consider 100 CAN message long intervals for message graph generation. Different from the last subsection, we will apply those five attack types uniformly to the whole dataset before training (80%) and test (20%) separation. The model training of the second-stage classifier is quite similar to that of the first-stage classifier except that the traditional *softmax* and cross-entropy function are respectively used here as the output layer and loss function. GNN has the same structure and hyperpatameters as those in the first-stage classifier.

When it comes to specific attack type classification, the *openmax* layer will be used to replace the *softmax* layer. At first, let us not consider open world recognition, in which c_0 is not introduced and ϵ is set to 0. Fig. 11 shows a confusion matrix based on 4,740 attacked data samples, where each specific attack type owns 948 data samples. Based on the confusion matrix, we can see that our second-stage classifier works pretty well in identifying DoS and suspension attack, which can only cause difference in statistical CAN message sequences. On the other hand, replay and spoofing attack can be misclassified to each other because of CAN message content changes. It is sometimes very tricky to tell whether an attacked data sample belongs to replay or spoofing attack. Spoofing and replay attack respectively use randomly generated and previously seen CAN messages to replace normal ones. Chances are that randomly generated CAN messages can sometimes be similar to or even the same as those previously seen messages. The situation of fuzzy attack is also complicated. Different from DoS and suspension attack, which usually focus on a single CAN ID, fuzzy attack randomly generates several CAN IDs. If a specific fuzzy attack does not cause an obvious change on the statistical CAN message sequence, it will be quite similar to a fuzzy or replay attack, which causes misclassifications shown in Fig. 11.

Next, let us evaluate the ability of our second-stage classifier in identifying new anomalies from unknown classes. Here, we respectively leave one attack type as the "unknown" class and consider the other four attack types during model training.

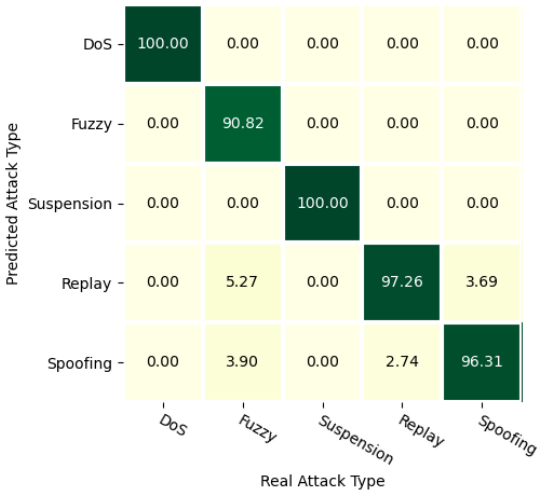


Fig. 11. Specific attack type classification confusion matrix (in %).

TABLE VI
OPEN WORLD RECOGNITION PERFORMANCE EVALUATION (IN %)

Attack	Accuracy	Prediction	Recall	F1
DoS	100	100	100	100
Fuzzy	87.13	62.86	87.13	73.03
Suspension	100	100	100	100
Replay	95.45	84.01	95.36	89.33
Spoofing	93.38	77.90	93.35	84.93

For each targeted attack type, we determine the most proper probability threshold ϵ in Algorithm 1 based on the optimal cut-point in the corresponding receiver operating characteristic (ROC) curve. Table VI shows the evaluation results. Note that here, we combine the four attack types used in model training to a "general" class, through which the open world recognition problem can be seen as a two-class classification problem. In general, we can see that our second-stage classifier can effectively identify new anomalies not quite similar to any existing attack type, which usually infers unknown classes. Besides, fuzzy, replay and spoofing attack sometimes cannot be identified when they are considered as the "unknown" class. The reason behind this is actually quite similar to the above.

E. Effect of Federated Learning

As has been discussed previously, vehicle states can affect statistical CAN message sequences and message contents. In this subsection, we evaluate the impact of federated learning on intrusion detection performance improvement based on the first-stage classifier. Related experiments are conducted based on datasets in the first baseline. We split the first dataset to 10 portions for local training to simulate an environment with 10 vehicles. The second and third dataset are still used as test sets. GNN model hyperparameters and the mini-batch batch size are the same as before in each simulated vehicle. The center of the hypersphere $\boldsymbol{o}$ in the one-class classification layer is set to the average of local centers derived from all simulated vehicle. We at first conduct experiments to explore the hyperparameter settings in FedProx, which are illustrated in Fig. 12. 10-fold cross validation is still utilized

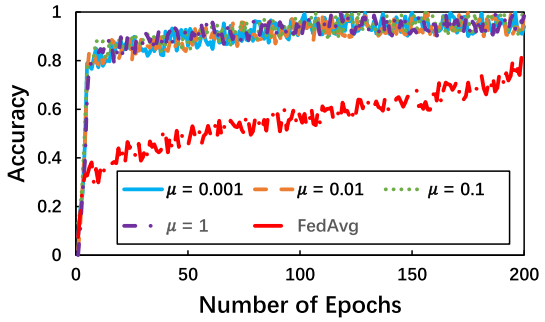


Fig. 12. Hyperparameter setting comparison related to federated learning.

TABLE VII
ACCURACY COMPARISON OF FEDERATED LEARNING (IN %)

		Without FL	FedAvg	FedProx	Centralized
Rpm	100	32.01	80.03	99.04	100
	150	34.32	81.35	94.65	97.65
	200	36.30	88.61	97.65	97.06
Speed	100	33.33	78.82	99.41	99.41
	150	29.04	74.71	97.62	94.71
	200	37.29	84.98	98.82	96.43

here. Considering that federated learning has model parameter aggregation after local training, accuracies are indexed by the number of communication rounds instead of epochs. Based on comparison results, we can see that μ does not have too much impact on convergence performance, and we select $\mu = 0.01$ in the following experiments. We also include the convergence performance of FedAvg to show the improvement of FedProx.

Table VII shows the accuracy comparisons based on test sets and GNN model parameters after 120 communication rounds. For the situation without considering federated learning (Without FL), we only use the data in the first simulated vehicle for training. Based on the comparisons, we can see that intrusion detection performance can get seriously degraded if training sets only cover limited vehicle states. In the real-world, such situations can happen if we only track one vehicle having limited driving scenarios. However, this issue can be solved by collecting and integrating multiple local models derived from different vehicles. For more comprehensive comparison, we also include the traditional centralized learning scenario. We can see that the convergence performance of FedProx is comparable to that of centralized learning. We also did related evaluation for the second-stage classifier, and figured out similar trends.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we propose a CAN bus IDS based on GNN, which can efficiently detect CAN message injection, suspension, and falsification attacks at the same time. A CAN message graph is developed here to integrate statistical message sequences with message contents. A GNN fit for directed attributed graphs is constructed and trained to predict intrusions. Considering that attack data are hard to acquire in the real world, which may cause highly imbalanced training sets, we develop a two-stage classifier cascade to tackle normal and attacked CAN data respectively. In the first-stage classifier,

we replace the traditional *softmax* layer in GNN with a one-class classification layer for anomaly detection and train the GNN only with normal CAN messages. Once anomaly CAN data are spotted, they will be further passed to the second-stage classifier to determine the specific attack type, in which an *openmax* layer is introduced to tackle anomalies from potential unknown classes. We also notice that changes of vehicle states can affect CAN message graph patterns in a reasonable way. In this case, federated learning is considered to cover a wide range of driving scenarios and vehicle states while protecting user data privacy. We validate our IDS based on several real-world datasets and comparisons with three baselines. Experimental results show that our IDS can achieve similar performance to those IDSs only based on statistical CAN message sequences and message contents.

As is shown in Fig. 1, once anomalies are rejected as from unknown classes, they will be buffered for further investigation. If necessary, new anomalies can be further tackled by transfer learning [21] or extra one-class classifier training [20]. In this paper, we assume that all vehicles participating in federated learning are from the same model. The lack of relevant public specifications can cause large divergence on CAN message streams from different vehicle models, which further brings challenges to the related machine learning model design. Separating different vehicle models through hierarchical structures may be a potential path to explore. In addition, we also assume that both vehicles and cloud servers are honest. Privacy and security in federated learning have recently drawn increasing attention [51], [52], [53]. Malicious participants can try to impact the aggregated model performance through injecting poisoning data or uploading poisoning local models during model training. In this case, how to filter out these attacks can be a new challenge. Currently, Byzantine-tolerant distributed learning [54], [55], [56] is widely considered as one potential solution. Besides, those cryptographic algorithms [57], [58], [59] can also be considered to prevent attackers' reverse-engineering. These two strategies can be further applied in our intrusion detection system construction, which is usually conducted offline and does not have a very high requirement for real-time performance.

REFERENCES

[1] K. Koscher et al., "Experimental security analysis of a modern automobile," in *Proc. IEEE Symp. Secur. Privacy*, May 2010, pp. 447–462.

[2] F. Martinelli, F. Mercaldo, V. Nardone, A. Orlando, and A. Santone, "Who's driving my car? A machine learning based approach to driver identification," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 367–372.

[3] D. K. Nilsson, P. H. Phung, and U. E. Larson, "Vehicle ECU classification based on safety-security characteristics," in *Proc. IET Road Transp. Inf. Control RTIC United Kingdom Members' Conf.*, May 2008, pp. 1–7.

[4] W. Wu et al., "A survey of intrusion detection for in-vehicle networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 919–933, Mar. 2020.

[5] Upstream Auto. (2021). *Upstream Security's 2021 Global Automotive Cybersecurity Report*. [Online]. Available: https://upstream.auto/2021report/

[6] M. Jedh, L. B. Othmane, N. Ahmed, and B. Bhargava, "Detection of message injection attacks onto the CAN bus using similarity of successive messages-sequence graphs," 2021, *arXiv:2104.03763*.

[7] R. Islam, R. U. D. Refat, S. M. Yerram, and H. Malik, "Graph-based intrusion detection system for controller area networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 1727–1736, Mar. 2020.

[8] H. Min Song, H. Rang Kim, and H. Kang Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2016, pp. 63–68.

[9] B. Groza and P.-S. Murvay, "Efficient intrusion detection with Bloom filtering in controller area networks," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 1037–1051, Apr. 2019.

[10] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *Proc. IEEE Int. Conf. Data Sci. Adv. Analytics (DSAA)*, Oct. 2016, pp. 130–139.

[11] D. Kosmano et al., "A novel intrusion detection system against spoofing attacks in connected electric vehicles," *Array*, vol. 5, Mar. 2020, Art. no. 100013.

[12] F. Amato et al., "CAN-bus attack detection with deep learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5081–5090, Aug. 2021.

[13] C.-W. Lin and A. Sangiovanni-Vincentelli, "Cyber-security for the controller area network (CAN) communication protocol," in *Proc. Int. Conf. Cyber Secur.*, Dec. 2012, pp. 1–7.

[14] K. Iehira, H. Inoue, and K. Ishida, "Spoofing attack using bus-off attacks against a specific ECU of the CAN bus," in *Proc. 15th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2018, pp. 1–4.

[15] S. Longari, D. Humberto Nova Valcarcel, M. Zago, M. Carminati, and S. Zanero, "CANnolo: An anomaly detection system based on LSTM autoencoders for controller area network," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1913–1924, Jun. 2020.

[16] H. Sun, M. Chen, J. Weng, Z. Liu, and G. Geng, "Anomaly detection for in-vehicle network using CNN-LSTM with attention mechanism," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 10880–10893, Oct. 2021.

[17] L. Yang, A. Moubayed, I. Hamieh, and A. Shami, "Tree-based intelligent intrusion detection system in Internet of Vehicles," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[18] M. Marchetti and D. Stabili, "READ: Reverse engineering of automotive data frames," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 1083–1097, Apr. 2019.

[19] A. Bendale and T. E. Boult, "Towards open set deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1563–1572.

[20] X. Guo et al., "Multi-stage deep classifier cascades for open world recognition," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 179–188.

[21] S. Tariq, S. Lee, and S. S. Woo, "CANTransfer: Transfer learning based intrusion detection on a controller area network using convolutional LSTM network," in *Proc. 35th Annu. ACM Symp. Appl. Comput.*, Mar. 2020, pp. 1048–1055.

[22] H. Lee, S. H. Jeong, and H. K. Kim, "OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame," in *Proc. 15th Annu. Conf. Privacy, Secur. Trust (PST)*, Aug. 2017, pp. 57–66.

[23] A. Tomlinson, J. Bryans, and S. A. Shaikh, "Towards viable intrusion detection methods for the automotive controller area network," in *Proc. ACM Comput. Sci. Cars Symp.*, 2018, pp. 1–9.

[24] W. Choi, H. J. Jo, S. Woo, J. Y. Chun, J. Park, and D. H. Lee, "Identifying ECUs using inimitable characteristics of signals in controller area networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 4757–4770, Jun. 2018.

[25] L. B. Othmane, L. Dhulipala, M. Abdelkhalek, N. Multari, and M. Govindarasu, "On the performance of detecting injection of fabricated messages into the CAN bus," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 1, pp. 468–481, Feb. 2022.

[26] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive CAN bus," in *Proc. World Congr. Ind. Control Syst. Secur. (WCICSS)*, Dec. 2015, pp. 45–49.

[27] M. Marchetti and D. Stabili, "Anomaly detection of CAN bus messages through analysis of ID sequences," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 1577–1583.

[28] E. Seo, H. Min Song, and H. Kang Kim, "GIDS: GAN based intrusion detection system for in-vehicle network," in *Proc. 16th Annu. Conf. Privacy, Secur. Trust (PST)*, Aug. 2018, pp. 1–6.

[29] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2011, pp. 1110–1115.

[30] M. Marchetti, D. Stabili, A. Guido, and M. Colajanni, "Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms," in *Proc. IEEE 2nd Int. Forum Res. Technol. Soc. Ind. Leveraging Better Tomorrow (RTSI)*, Sep. 2016, pp. 1–6.

[31] M.-J. Kang and J.-W. Kang, "A novel intrusion detection method using deep neural network for in-vehicle network security," in *Proc. IEEE 83rd Veh. Technol. Conf. (VTC Spring)*, May 2016, pp. 1–5.

[32] M. Baccouche et al., "Sequential deep learning for human action recognition," in *Proc. Int. Workshop Human Behav. Understand.*, 2011, pp. 29–39.

[33] L. Ding, W. Fang, H. Luo, L. Peter, B. Zhong, and X. Ouyang, "A deep hybrid learning model to detect unsafe behavior: Integrating convolution neural networks and long short-term memory," *Autom. Construct.*, vol. 86, pp. 118–124, Feb. 2018.

[34] H. Rahmani, A. Mian, and M. Shah, "Learning a deep model for human action recognition from novel viewpoints," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 667–681, Mar. 2018.

[35] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2020.

[36] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.

[37] B. Schölkopf, J. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, Jul. 2001.

[38] D. M. J. Tax and R. P. W. Duin, "Support vector data description," *Mach. Learn.*, vol. 54, no. 1, pp. 45–66, Jan. 2004.

[39] L. Ruff et al., "Deep one-class classification," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4393–4402.

[40] W. J. Scheirer, A. Rocha, R. J. Micheals, and T. E. Boult, "Meta-recognition: The theory and practice of recognition score analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1689–1695, Aug. 2011.

[41] P. Zhang, J. Wang, A. Farhadi, M. Hebert, and D. Parikh, "Predicting failures of vision systems," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 3566–3573.

[42] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu, "Diagnosis of multiple cancer types by shrunken centroids of gene expression," *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 10, pp. 6567–6572, 2002.

[43] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[44] A. Bendale and T. Boult, "Towards open world recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1893–1902.

[45] I. Rouf et al., "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study," in *Proc. USENIX Conf. Secur.*, 2010, pp. 1–16.

[46] H. B. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[47] T. Li, A. Kumar Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2018, *arXiv:1812.06127*.

[48] H. M. Song and H. K. Kim. (2020). *Can Signal Extraction and Translation Dataset*. [Online]. Available: https://ocslab.hksecurity.net/Datasets/can-signal-extraction-and-translation-dataset

[49] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "CANet: An unsupervised intrusion detection system for high dimensional CAN bus data," *IEEE Access*, vol. 8, pp. 58194–58205, 2020.

[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[51] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1175–1191.

[52] K. Wei et al., "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3454–3469, 2020.

[53] E. Bagdasaryan et al., "How to backdoor federated learning," in *Proc. 23rd Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.

[54] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[55] G. Damaskinos, E.-M. El-Mhamdi, R. Guerraoui, A. Guirguis, and S. Rouault, "AGGREGATHOR: Byzantine machine learning via robust gradient aggregation," in *Proc. Mach. Learn. Syst.*, vol. 1, 2019, pp. 81–106.

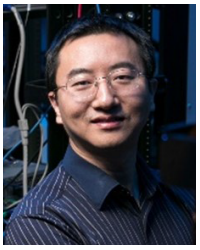
[56] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2168–2181, Jul. 2020.

[57] S. Nürnberger and C. Rossow, "vatiCAN—Vetted, authenticated CAN bus," in *Proc. Int. Conf. Cryptograph. Hardw. Embedded Syst.*, Santa Barbara, CA, USA. Berlin, Germany: Springer, 2016, pp. 106–124.

[58] T. Sugashima, D. K. Oka, and C. Vuillaume, "Approaches for secure and efficient in-vehicle key management," *SAE Int. J. Passenger Cars-Electron. Elect. Syst.*, vol. 9, no. 1, pp. 100–107, 2016.

[59] M. D. Pesé, J. W. Schauer, J. Li, and K. G. Shin, "S2-CAN: Sufficiently secure controller area network," in *Proc. Annu. Comput. Secur. Appl. Conf.*, Dec. 2021, pp. 425–438.

Hengrun Zhang (Member, IEEE) received the Ph.D. degree from the Computer Science Department, George Mason University, Fairfax, VA, USA, in 2022. He is currently an Assistant Professor with the Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, China. His research interests include machine learning, network privacy and security, federated learning applications in the Internet of Things (IoT), and performance optimization in federated learning.



Kai Zeng (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the Worcester Polytechnic Institute (WPI) in 2008. He was a Post-Doctoral Scholar with the Department of Computer Science, University of California at Davis (UCD), from 2008 to 2011. He was with the Department of Computer and Information Science, University of Michigan-Dearborn, as an Assistant Professor, from 2011 to 2014. He is currently an Associate Professor with the Department of Electrical and Computer Engineering and the Department of Computer Science, George Mason University. His current research interests include cyber-physical systems/the IoT security and privacy, 5G and beyond wireless network security, machine learning, spectrum sharing, and edge computing. He was a recipient of the U.S. National Science Foundation Faculty Early Career Development (CAREER) Award in 2012, the Excellence in Post-Doctoral Research Award from UCD in 2011, and the Sigma Xi Outstanding Ph.D. Dissertation Award from WPI in 2008. He is also an Associate Editor of the IEEE TRANSACTIONS ON MACHINE LEARNING IN COMMUNICATIONS AND NETWORKING and IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING.

Shuai Lin (Member, IEEE) received the Ph.D. degree in safety science and engineering from Beijing Jiaotong University, China, in 2018. She was a Post-Doctoral Researcher at the Antai College of Economics & Management, Shanghai Jiao Tong University, in 2022. She is currently an Assistant Professor with the School of Economics and Management, Shanghai Institute of Technology, Shanghai, China. Her research interests are in the areas of failure data analysis, system reliability evaluation, and system safety assessment.