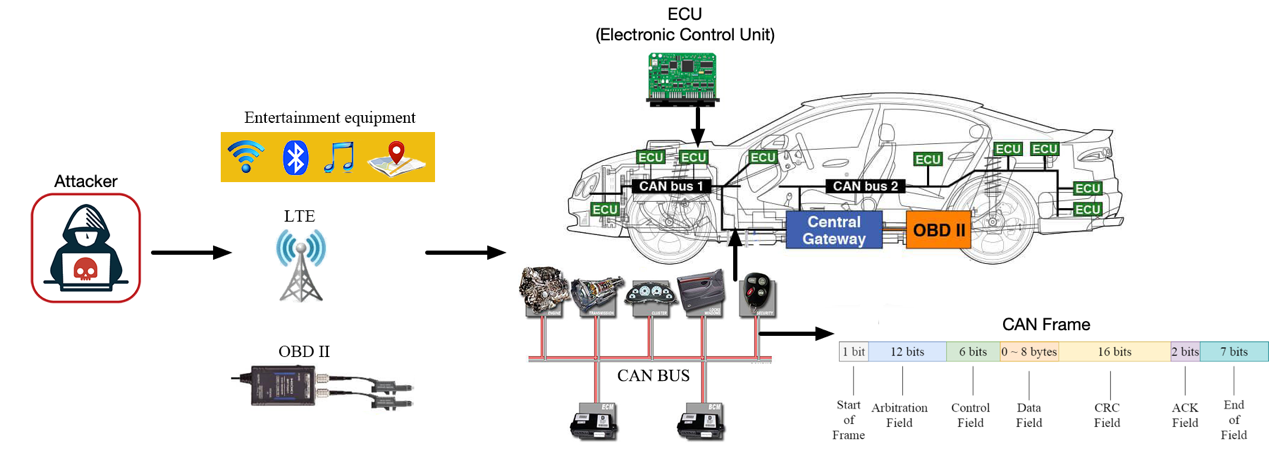
Evolutionary Multiobjective Surrogate-Assisted Neural Networks for Intrusion Detection in Vehicle Control Area Network

***Abstract*—Modern vehicles are increasingly being equipped with sophisticated electronic systems making it necessary to ensure the safety and reliability of the communication between them to ensure driver safety and convenience. The most commonly used communication protocol on vehicles, Controller Area Network (CAN) messages, has no direct support for secure communications, which poses a significant threat to the communication security on the vehicle. Neural architecture search (NAS) adopts a search strategy to explore the search space to automatically design a neural network architecture. In this paper, we propose an intrusion detection model based on a deep neural network of lower complexity than the first generation and increased accuracy designed using a Multi-Objective Evolutionary Algorithm (MOEA) assisted NAS. Graph neural network (GNN) and convolutional neural network (CNN) are utilized to explore the internal logical features and spatial features of CAN, respectively. Due to the different functions associated, vehicle CAN messages have different lengths. Reinforcement learning (RL) is used to dynamically get a CAN package which acts as a sample sent to the proposed model. A method to convert original CAN data to graph data using its time sequence feature is also proposed. The CNN and the GNN of the proposed model are optimized using MOEA. CNN inherits the weights of parents, and GNN inherits the weights of supernet in the evolution process. This weight sharing helps to accelerate the measurement of individual fitness. To assess our intrusion detection model, we conduct an experimental study utilizing real-world datasets. Following network growth, GNN flops reduced to 40% of the supernet, but intrusion detection accuracy increased to more than 95%. The accuracy of CNN was also improved by 5% compared with the best individual of the first generation. The accuracy of the intrusion detection achieved by combining the two models is greater than 99%, which is 4% better than the model without RL.**

***Index Terms*—Intrusion Detection of Controller Area Network (CAN), Multi-Objective Evolutionary Algorithm (MOEA), graph neural network (GNN), convolutional neural network (CNN)**

1. INTRODUCTION

In the last few decades, the implementation of automotive electronics has experienced rapid growth [[1](#_ENREF_1)]. This trend has resulted in several changes in the vehicular ecosystem. Drive-by wire (DBW) technology, for example, uses electronic or electrical systems in the control systems, such as the throttle, brake, and steering, which were traditionally controlled using mechanical linkages. Control Area Networks (CAN) is considered as a simple and reliable communication protocol and a standard for the in-vehicle networks [[2](#_ENREF_2)], connecting multiple **electronic control units** (ECUs). The adoption of CAN accelerates the applications with the emergence of Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication interfaces [[3](#_ENREF_3)]. The openness of the vehicular system increases the risk of malicious cyberattacks that can severely threaten human life. However, conventional in-vehicle networks are tremendously vulnerable to cyberattacks as CAN was originally developed for an isolated physical system. Every ECU on a CAN bus, for example, can receive any ECU-to-ECU communication. Furthermore, a CAN data packet has no sender’s identification as shown in Figure 1. Kwak et al. [[4](#_ENREF_4)] conducted several experiments where a CAN message can be readily fuzzed by a packet injection and modification. Various attack scenarios, e.g., disabling brakes and displaying wrong information on an instrument panel (or dashboard), are shown in [[5](#_ENREF_5)], [[6](#_ENREF_6)]. Attacks on the CAN bus can manifest in several ways. Diagnostic commands deployed during driving can cause malicious effects, e.g., locking the brakes to immobilize the car. However, diagnostic commands should never be displayed when driving normally; thus, they are easily detected. Recent research works [[7](#_ENREF_7), [8](#_ENREF_8)] also point to the weakness of security associated with CAN.



Figure

Typical message injection attack scenarios and format of CAN. An attacker can inject fabricated messages into the CAN bus of the target vehicle via the on-board diagnostic port of the vehicle or a telematics device that provides a wireless communication channel.

Graph data are widely used in multiple areas [[9](#_ENREF_9)], [[10](#_ENREF_10)], for example, used to process Non-Euclidean data features, including in social networks [[11](#_ENREF_11)], text networks [[12](#_ENREF_12)] and biological networks [[13](#_ENREF_13)]. Graph neural networks (GNN) can be used to solve graph-related problem fields in an end-to-end manner [[14](#_ENREF_14)]. Relational modeling is crucial for many networks or graphical data mining tasks, such as link prediction. GNN has recently been successfully applied in the real world, such as image recognition field [[15](#_ENREF_15)], new drug discovery field [[16](#_ENREF_16)] and flow prediction field [[17](#_ENREF_17)]. In this work, the graph data are constructed by the time sequence characteristics of CAN, and the graph neural network extracts the logical characteristics of CAN. In reality, there are examples of using more dynamic graphs, for example in social networks or financial networks. A dynamic graph can change the nodes of previous graph data or the connection between nodes as time progresses. A graph network can use Recurrent Neural Networks (RNNs) [[18](#_ENREF_18)], [[19](#_ENREF_19)] to learn the temporal characteristics of dynamic graphs, and spatiotemporal graph neural network architecture to learn the spatial characteristics of dynamic graphs [[20](#_ENREF_20)]. However, some graph data may be completely different at different times. Our work constructs the graph data of CAN, the nodes or connections of graph data constructed in the two time periods may be completely different. There is very little work to identify the graph network of such graph data. This model uses a graph collapse network [[21](#_ENREF_21)] to detect such graph data.

Developing a custom learning architecture consisting of multiple GNN layers for specific scenarios (e.g., biological and physical network data) is still difficult, even for neural network experts. Falkner et al. [20] pointed out that the deep learning algorithm is very sensitive to many hyperparameters. The work in the literature field [21] shows that the hyperparametric optimization (HPO) of GNNS is essential to achieving satisfactory results in practice. Therefore, the study of effective HPO methods for GNN is of great value for GNN to be applied to various practical problems. In order to automate the model selection process, neural architecture search (NAS) is widely used in fields [22], [23], and has become the focus of deep learning research in recent years. NAS aims to explore the optimal combination of architecture components from the search space to maximize the model performance applicable to the target problem. So far, people have made great efforts to search convolutional neural network (CNN) architecture, which has promoted the latest progress of many important benchmark tasks, such as image classification fields on CIFAR10/100 and ImageNet [24], [25]. In contrast, little work has been done on GNN learning of graphic structure data [26] – [28]. NAS literature has proposed two main types of methods as the most effective problem-solving methods: reinforcement learning (RL) and evolutionary algorithm (EA) [28]. So far, the second technology has been ignored in the GNNs environment. The results show that both RL and EA can find equivalent models in terms of accuracy, and EA is faster in some cases. Two recent related works [28],[29] mainly focus on NAS based on RL and uses RNN as the controller to generate variable-length strings describing GNN architecture to maximize the expected accuracy of the developed network architecture on the verification dataset. Although encouraging results have been achieved, the existing work has always faced the challenge of the following two shortcomings: hyperparametric invariance and high computational intensity. In addition, the controller usually generates candidate GNN architectures and evaluates them in a sequential manner, which is difficult to expand to large search space.

NAS tasks can usually be described as a complex optimization problem [30], [31]. Generally speaking, a neural network with higher complexity with appropriate regularization in place may have higher recognition accuracy for complex problems. Still, it will significantly increase the detection time of intrusion CAN messages. Our algorithm uses two conflicting objectives, namely, the accuracy of intrusion detection and the complexity of the model, for multiobjective optimization. In the field of computational intelligence, evolutionary algorithms (EAS) have been widely used to solve various neural network training problem fields [32], such as weight training field [32], architecture design field [33] and learning rule adaptive field [34]. Recently, evolutionary neural architecture search using EA as a NAS optimizer has attracted more and more attention [35]–[37]. The GNN gene field [26] is proposed through the evolutionary method. It identifies the GNN architecture and trains the hyperparameters in two search spaces, and alternately optimizes the structure and hyperparameters of the graph network (such as learning rate and dropout). Our work puts them into one search space for search. The recognition rate of the graph network architecture found is more than 95%, and the recognition rate of the graph network and convolution network combined is more than 99%.

Neural network architecture search is a process, which generates and evaluates the hyperparameter settings iteratively until the preset stop conditions are met to obtain the best solution. Usually, the objective function of a neural network is selected as the fitness function to evaluate the hyperparameters. However, evaluation is typically costly because training GNN models requires a lot of computing resources. On the other hand, with the development of deep learning, the neural network has more and more layers or various other optional hyperparameters (such as activation function and optimization method), which leads to a large search space and increases the difficulty of the search. With the increase of network components, the consumption of computing resources for searching the network is high, and tedious and arduous efforts are made for GNN architecture adjustment and optimization. Therefore, the existing neural architecture search work is based on finding the best solution with fewer experiments, reducing the evaluation cost, and improving the exploration ability. Although EAs have competitive search performances in various optimization tasks [38], as a group-based optimization method, its calculation cost is usually high. This is especially true for EvoNAS, because EAs typically require a large number of fitness assessments, and each fitness assessment in NAS is computationally expensive because it usually involves training deep neural networks from scratch on a large amount of data. For example, an autoencoder based convolutional neural network (AE-CNN) requires 27 GPUs to obtain an optimized CNN architecture on the cifar10 dataset. Some works search the hyperparameters of graph networks and save computing resources by quickly evaluating individuals. For example, [39] proposes parallel graph architecture search to improve search efficiency. In our work, when CNN evolve, offspring inherit the weights of parent networks and GNN inherits the weight parameters of hypernetworks to achieve rapid individual adaptability evaluation.

Deep reinforcement learning is the combination of reinforcement learning and deep learning. This research field can solve a series of complex decision-making tasks that could not be completed by machines before. Therefore, deep RL has opened up many new medical care applications, robotics, smart grid, finance and other fields [40]. The length and content of the CAN messages are different when the vehicle performs various functions. Based on this inspiration, we use reinforcement learning to obtain the length of the CAN message for each intrusion detection. The experimental results show that the intrusion detection rate of the model increases by 5% after using the variable-length CAN compared with random length. The reinforcement learning network learns the feature vector of the graph convolution output in each graph network as each state input to the reinforcement learning actor network, and the reinforcement learning outputs the next action, that is, the length of the next CAN message. In early reinforcement learning, there was an overestimation of the real action value [41], and [42] established two value networks to solve the overestimation problem. Reinforcement learning can be divided into discrete and continuous types according to the types of output actions. The meaning of action in the model is the length of the output CAN messages. We give the range of message length. The integer value between the minimum and maximum length can be taken. Based on the above requirements, we use the TD3 [43] network as a neural network to dynamically obtain the length of CAN messages which are sent to the intrusion detection model every time.

To address the aforementioned challenges, in this paper, we propose a deep neural network intrusion detection model based on evolutionary multiobjective surrogate-assisted neural architecture search to detect CAN intrusion, such as denial-of-service (DoS) and spoofing attacks, with significantly high accuracy and low complicity. In summary, the main contributions of this paper are as follows:

* A multiobjective evolutionary GNN detection model is proposed, which takes the detection accuracy and the model flops as the optimization objectives. In the evolution process, the node inheritance strategy is adopted to shorten the time of individual evaluation.
* A method of converting CAN messages into graph data is proposed. Two connected nodes of graph data are constructed according to the timing information of two adjacent timestamps. The logic features in CAN are extracted, so that CAN messages can be classified by GNN.
* A model combining GNN and CNN is proposed to detect CAN intrusion messages, and a reinforcement learning network is used to collect intrusion detection samples of CAN dynamically. During the training of the detection model, the logic and spatial characteristics of CAN are learned simultaneously.

The rest of this paper is organized as follows. Section II introduces the related work of our proposed model. Section III describes the encoding of neural network architectures, followed by the details of the proposed intrusion detection model and the conversion method to graph data from CAN in Section IV. Experimental settings and results are presented in Section V, respectively. Finally, Section VI concludes the paper and proposes some points that can be improved of our model in the future.

1. RELATED WORK
2. *Conventional intrusion detection of CAN*

A message authentication code, also known as a MAC, is a cryptographic checksum on data that employs a session key to identify unintentional and intentional data modifications. However, the CAN frame is often incapable of supporting MAC [44] and other communication security techniques. Some researchers [45, 46] have attempted to either create new protocols or spread MAC across multiple transmissions. Tashiro et al. establish a method for detecting tampering with individual frames and entire sections [45]. They developed a system that sends a partial MAC in each frame to mitigate against replay, masquerade, and injection attacks. Nowdehi et al. [46] investigate several of these modified protocols based on five industry-specific criteria for prospective CAN message authentication solutions. They discovered that no solution matched all the criteria. VatiCAN employs maintenance support, "adequate implementation details," and no unnecessary overhead, whereas WooAuth modifies the expanded CAN protocol to allow for more authentication codes. Finally, the authors [46] hypothesize that the CAN bus may be "fundamentally unsuitable for secure communication".

1. *Deep learning intrusion detection of CAN*

Intrusion detection strategy (IDS) can combine machine learning to train itself to identify abnormal behavior, which can be used as an alternative or supplement to MAC. Spoofing, injection, bus shutdown, and denial of service attacks may all be prevented by intrusion detection systems. Choi et al. introduced voltage IDS in [47], which uses ECU signal inconsistencies: first performs training and testing, detects the signal characteristics, and then uses the training data to check if the ECU has been compromised. Using multi-class classifiers, one of which corresponds to an ECU, Voltage IDS may identify camouflage attacks. It estimates the most likely sender and compares it to the message's actual CAN ID. A camouflaged attack is identified if they differ. Song et al. [48] solve this problem by changing the ID portion of the CAN frame to binary. For intrusion detection, the updated ResNet model was used to extract the features of the binary text and learn the characteristics of the intrusion message and the regular message. Their experimental findings suggest that their approach outperforms the classic machine learning technique in terms of false-positive rate. Taylor et al. proposed employing deep learning algorithms for intrusion detection in [49] since they are created directly from the network's bit stream. These functions are simple to implement and have a high execution efficiency. While training the characteristics offline, this approach examines the exchange packets in the vehicle network. It responds to the attack in their experiment with a considerably high detection rate.

1. *Neural architecture search by EA*

Neural architecture search (NAS) aims to automatically design network architecture, which is essentially an optimization problem of finding an architecture with the best performance in a specific search space with constrained resources [50], [51]. Sun et al. [52] used EA with variable coding length to automatically evolve the architecture of CNN. William et al. [53] introduced an evolutionary NAS coding strategy based on directed acyclic graphs (DAG), with better performance than the randomly generated CNN architecture. Real et al. introduced Amoebanet [54], which employs improved tournament selection to evolve network groups and outperforms the homemade model on Imagenet. Wang et al. [55] designed an effective evolutionary algorithm to optimize the generator within the framework of GANs. This method can effectively improve the generation performance and training stability of the GAN model. Yin [56] uses an evolutionary multiobjective method to design CNN architecture, which uses probabilistic SMBO to maximize classification performance and minimize network reasoning time. Elsken et al. [57] described NAS as a bi-objective optimization problem, in which two objectives are to maximize performance and minimize computing resources. Lu et al. [58] proposed NSGANET, which can automatically design the network, maximize the model performance, and minimize floating point operations (flops).

1. *Surrogate*

A major disadvantage of EvoNAS is that in the process of evolutionary optimization, each new candidate neural network needs to be trained on the training dataset and then evaluated on the validation dataset to avoid over-fitting. Therefore, if the network is large and the training dataset is large, the architecture evaluation in EvoNAS may take several hours. Because EAS is a kind of group-based search method, they usually need a lot of fitness evaluation, making EvoNAS computationally challenging to implement. For example, on CI-FAR10 and CIFAR100 datasets, CNN-GA [59] consumes 35 GPU days and 40 GPU days, respectively, the genetic CNN method [60] consumes 17 GPU days, and the large-scale evolutionary algorithm [61] consumes 2750 GPU days. Therefore, in the case of limited computing resources, the agent model can accelerate the fitness evaluation in EvoNAS. Agents are divided into high-level agents and low-level agents. The high-level agent and low-level agent represent the architecture level and the parameter level in the architecture, respectively. High-level agent representation predicts the accuracy of different neural networks by parameterizing the neural network architecture. However, the low-level agent solves the complexity of using SGD optimization from scratch for each architecture after searching multiple architectures. The low-level agent is given a trained hypernetwork and neural network structure, including all sub architectures. The weight of the neural network architecture inherits the weight from the hypernetwork. In the search process, the accuracy of using the weight inherited from the hypernetwork becomes the standard for selecting the architecture. However, the correlation between the accuracy of prediction architecture and the final accuracy of neural architecture through weight sharing is not close. The neural architecture reference MSuNAS [62] we searched is not only sharing the weight of hypernetwork, but also fine-tuning through training again. MetaQNN [63] uses the agent model to predict the final accuracy of candidate architectures (as time series prediction) from the first 25% learning curve of SGD training. PNAs [64] use an alternative model to predict the accuracy of the network architecture, adding a branch to the unit structure, which is repeatedly stacked together. Both methods use the agent method to evaluate the performance of neural architecture. However, the correlation between the prediction accuracy of this method and the actual accuracy of the model is relatively low. OnceForAll [65] also uses an agent model to predict the accuracy of architecture coding. However, the agent model is trained offline for the whole search space, so it needs many samples to learn. ChamNet [66] trains many architectures through complete low-level optimization and selects only 300 high-precision samples with different efficiency (trigger, delay, energy) to train alternative models offline. Our model only conducts online learning on samples close to Pareto frontier, which significantly improves the efficiency of architecture search. Our model evaluation method draws lessons from the idea of MSuNAS.

1. ARCHITECTURE SEARCH SPACE

This section introduces the search space of the model, including the search space of GNN and CNN. The search of GCN includes structure and hyperparameter search. The structure search includes the number of layers of the GCN layer and the width of the prediction layer of the graph network. Hyperparameter search includes learning rate, activation function type, dropout rate, etc. CNN [48] architecture of our model consists of one stem part and four Res CNN block parts. The CNN part of our model retains the stem part and extracts eight positions on the four Res CNN blocks. There are five options for each of these eight locations.

1. *GNN architecture search*

The specific genes and corresponding relationships are shown in Table I. The chromosome is divided into nine parts. The first part of the chromosome is position 0. A gene indicates whether to use the direction information of graph data constructed by CAN IDs. The graph data can determine the direction of the edge of the directed graph according to the sequential relationship between two adjacent frames. Specifically, the CAN IDs of the previous frame points to the next frame. Although using the direction of constructing graph data will make more use of the information of graph data, the experimental results show that using more graph information does not necessarily improve the recognition accuracy. See the next section for the analysis of specific experimental results. According to the formula in the previous section, it is necessary to obtain the Laplace matrix of each subgraph and calculate the eigenvector of each subgraph. Whether to use regularization for the Laplacian matrix before calculating the eigenvector of each subgraph. The regularized Laplacian matrix and the nonregularized Laplacian matrix *L* of the graph network are expressed as

(1)

(2)

where *A* represents the adjacency matrix of graph data and *D* represents the degree matrix of graph data. The third part of the chromosome, positions 2 to 3, is the depth of the two GCN blocks. One GCN block is laid before graph coarsen, and the one is laid after graph coarsen. The fourth part of chromosome, positions 4 to 5, is the number of neurons in each layer of the prediction layers. In the process of evolutionary algorithm search, the neural network architecture refers to [62], [67], are dynamically changed. The appropriate number of layers of graph convolution and the number of neurons in the prediction layer can achieve good detection accuracy while the complexity is not high. Parts 5 to 7 of chromosome, positions 6 to 8, represent the hyperparameters in the network architecture, such as dropout rate, learning rate, etc. Chromosome Part 8, positions 9 to 10, respectively represent how to reduce the dimension into a one-dimensional vector after each convolution block outputs the convolution result. There are three options: sum, average and maximum for each dimension. Part 9 of chromosome represents the activation function after the end of convolution or FCNN.

Table GNN search space

|  |  |  |  |
| --- | --- | --- | --- |
| Part | Position | meaning | Search space |
| 1 | 0 | directed/undirected graph | directed undirected |
| 2 | 1 | Whether Normalization of subgraph | not normalization  normalization |
| 3 | 2-3 | Layers of GCN | one two three |
| 4 | 4-5 | Proportion of neurons in prediction layer | 0.25 0.5 0.75 1.00 |
| 5 | 6 | Dropout | 0.005 0.01 0.02 0.03  0.04 0.05 |
| 6 | 7 | Weight Decay Rate | 5e-4 8e-4 1e-3 4e-3 |
| 7 | 8 | Learning Rate | 5e-4 1e-3 5e-3 1e-2 |
| 8 | 9-10 | way to merge the vectors of every GCN layer | sum average max |
| 9 | 11-18 | Activation function | sigmoid tanh relu  leaky\_relu relu6 |

1. *CNN architecture search*

CNN, such as the variants of Inception and ResNet, are designed to build neural networks by stacking multiple blocks. The neural network architecture design includes the determination of depth (number of layers), width (number of channels) and spatial resolution change (number of pool layers), while the block structure design specifies hierarchical connection and local calculation. Through this block design method, the generated model can not only achieve high performance, but also be extended to different datasets and tasks.

An operation space is defined by a set of possible basic components of network architecture and known successful modules designed by human experts. The five operations used in this work and the corresponding genotype-phenotype mapping are shown in Table II. In the table, spatial separable convolution (SP) and deepwise separable convolution (DW) can reduce network parameters without sacrificing network performance. Here, we use two DW operations and two SP operations, and the kernel size respectively is 3 × 3 and 5 × 5, referred to as SP3, DW3 and SP5, DW5.

Table CNN search space

|  |  |  |  |
| --- | --- | --- | --- |
| Operation type | Kernel size | Short name | Code |
| Spatial Separable Convolutions | 3 | SP3 | 0 |
| Spatial SeparableConvolutions | 5 | SP5 | 1 |
| Depthwise separable convolution | 3 | DW3 | 2 |
| Depthwise separable convolution | 5 | DW5 | 3 |
| Normal convolution | 3 | 3\*3 | 4 |

1. PROPOSED INTRUSION DETECTION MODEL

The architecture we proposed is shown in the Figure2 and can be divided into two parts of requiring CAN part and intrusion detection part. Requiring CAN part is constructed by reinforcement learning (RL) network to dynamically collect CAN messages. Intrusion detection part is constructed by GNN and CNN. The network architecture of GNN and CNN is optimized by EA. The parts optimized by EA are marked with dotted lines in the Figure 2. They are the convolution layer of CNN, the GCN layer of graph neural network and the full connection layer etc. At the same time, using the advantages of two different neural networks, GNN has the advantage of extracting logical features and CNN has the advantage of extracting spatial features. Finally, the output results of the two networks are integrated to obtain the final result. Both GNN and CNN are binary-class networks. Both GNN and CNN are binary networks. Before the final output of the intrusion detection results, it is judged that the output of each network has one row and two columns of vectors, and the result output is the network result with a larger difference between the two columns of the two network output vectors.

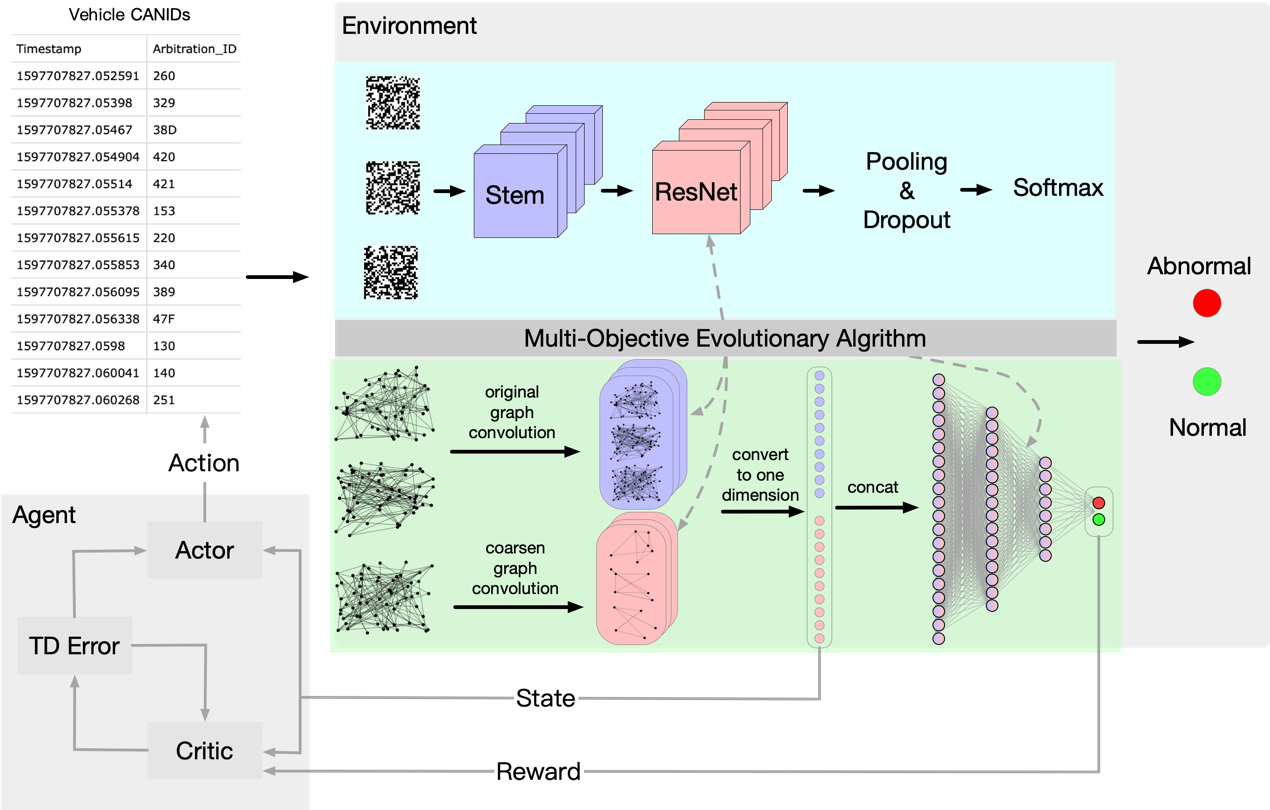


Figure Overall framework of network architecture. The samples of CANID sent to the intrusion detection model are illustrated in the upper right corner of the figure. The CAN intrusion detection part of our model is the environment of RL and it’s in the largest gray box on the right. The RL’s agent is composed of FCNN, which is in the gray box on the lower left. The detection part consists of CNN and GNN. CNN is on the blue background above and GNN is on the green background below. Both CNN and GNN are optimized using the MOEA algorithm. The dotted line in the figure points to the optimized part from the dark gray box of MOEA in the middle. The optimized part in CNN includes the RESNET structure and hyperparameters, such as learning rate, dropout, etc. The optimized part in GNN includes the number of GCN layers, the number of neurons in each layer of the FCNN and hyperparameters, such as learning rate, dropout, etc. The design details of RL, GNN, CNN are described in IV-A, IV-B, IV-C.

1. *Collect CAN by Reinforcement Learning (RL)*

RL network is able to dynamic collect CAN messages of every detected sample, which will send to GNN and CNN. Before each intrusion detection, the length of one of the CAN messages is collected as a sample input to the detection network. In the vehicle CAN message, the message data of different functions or the message of the same function need different frames to complete, so the RL network is used to solve the problem of dynamic acquisition of CAN messages of different lengths. **Before training the RL network, a GNN with intrusion detection ability is trained first. When training the GNN, the length of the input is generated randomly. There are two convolution layers in this GNN. The output of the convolution layer of GNN is converted into one-dimensional vectors, and the two one-dimensional vectors are combined as the state input of the RL network. The reward function is designed by using the recognition results at the out of the flattened layers of GNN. If the GNN graph network recognition is correct, it will get a positive reward, and if the recognition is wrong, it will get a negative reward or penalty. The RL network architecture design of this work refers to Twin Delayed Deep Deterministic Policy Gradients TD3 [43]. To avoid overestimation of value network, the RL network is composed of two value networks and two action networks, and the output is discrete. The dimension of reinforcement learning network output is the difference between the maximum CAN length and the minimum CAN length of intrusion detection is given by**

. （3）

1. *Intrusion detection by Graph Neural Network (GNN)*

*(i) Model Assumptions*

*Here are our design assumptions for the proposed intrusion detection model. We assume:*

1. each ECU also runs intrusion detection software that monitors and identifies intrusions in the in-vehicle network.
2. intruder can gain physical or remote access to the CAN bus via an exposed interface, such as vehicle to everything (V2X), entertainment systems, or advanced driver assistance systems (ADAS), either physically via the OBD-II port or remotely via Bluetooth, WiFi, or telematics services, such as 3G and long-term evolution (LTE).
3. raw payload bit decoding into signal values has already been completed.
4. attacker attempts to manipulate communications in order to affect the vehicle's behavior.
5. attacker can shut off any ECU and/or insert malicious messages arbitrarily.

First, we set the previous frame of CAN message to point to the next frame. Because the message ID has limited length[reference], not all messages will point to a new message, it may also point to a message frame that has appeared before. The direction relationship of the converted graph data is related to the time sequence of the message frame sequence. A sequence of CAN IDs always indicates different functions in the vehicle, and whether the logic of the execution sequence of the functions is reasonable or not can be learned by the GNN. For example, it can learn a large acceleration signal appearing when braking, or a vehicle ignition signal appearing without a car key signal is not reasonable. It is then very likely that an intrusion message has appeared on the CAN bus, which has practical significance. For the GNN to learn more abstract features, we input a partial graph of sequence data to determine whether the message is an intrusion message, instead of inputting one or two frames of message. Our proposed approach is shown in Figure 3.

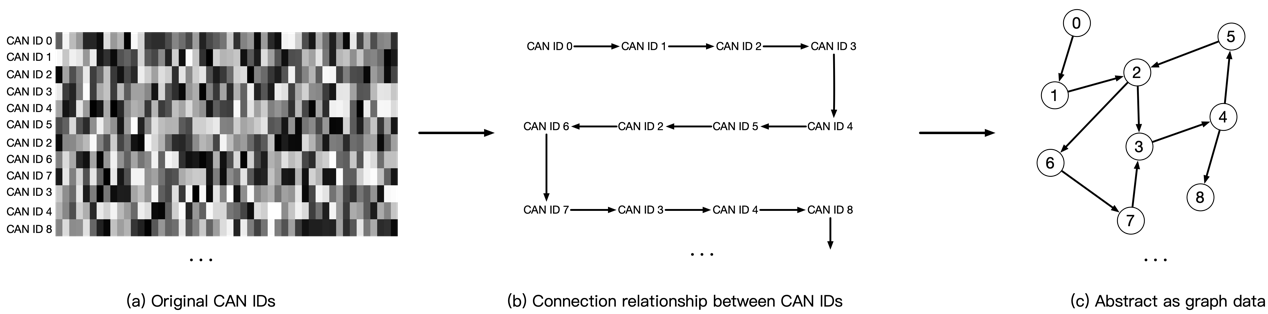


Figure Construct graph data from original CAN.

The GCN performs a graph level classification task, and the network architecture of GCN is inspired by [19]. **The main idea of the GNN of our model is to perform graph convolution on the original graph and the pooled graph respectively through GCN, convert the results of the two GCN into two one-dimensional vectors, combine the two one-dimensional vectors and send them to the fully connected neural network (FCNN) to complete the detection task.** [19] is to divide the constructed graph data into several clusters by clustering method []. Each cluster can be regarded as a subgraph of graph data to obtain the eigen matrix of a series of subgraphs. Using eigenvectors to construct the pooling matrix, each subgraph is pooled into a super node. Graph G connected by given K subgraphs, C is a part of G. represents the number of nodes in the subgraph . is the list of nodes in the subgraph . Each subgraph can be regarded as a super node of graph G. Define sampling operator *C*(*k*) *∈* R*N×Nk* as follow:

if and only if (4)

where represents the element in i-th row and j-th column of , represents the jth node in the node list . This operation indicates the correspondence between the node in the subgraph and the original graph G. Because Fourier transform can convert the graph signal to the frequency domain and take the signal information and the structure information of graph data into account, as in [19], we use Fourier transform to design pooling operation. The pooling operation pools the constructed graph signal G into . Pooling operation is based on the Fourier transform of each subgraph . The Laplace matrix of the subgraph is . , …, represents the all eigenvectors of the Laplacian matrix of the k-th subgraph. Then use the upsampling operation to upsample the eigenvector to the whole graph G. The upsampling equation is represented by

(5)

represents the pooling matrix containing the L-th eigenvector of all subgraphs is given by

(6)

Each subgraph does not necessarily have the same number of nodes, that is, the number of eigenvectors of each subgraph is not necessarily equal. represents the maximum number of nodes in all subgraphs. For the subgraph owns nodes, the *l*st pooling operation is expressed as:

(7)

indicates the result of the L-th pooling operation. The k-th row of contains the information of the k-th subgraph, that is, the k-th super node. Based on the above structure, we can construct times of pooling operation, and combine the results of all pooling operations to form the coarsen matrix shown as:

(8)

In some other graph classification tasks using pooling method, the sum or average method is used to treat each node in the subgraph indiscriminately, and the structure information of nodes in each subgraph cannot be extracted. In our work, we pool the graph data once, and apply a GCN operation to extract features before and after pooling. Finally, the outputs of the two graph convolutions are transformed into one-dimensional vectors by summation or averaging, and the two one-dimensional vectors are merged into a single to the FCNN layers for classification.

The crossover and mutation operations of the GNN are shown in Figure 4 and 5. The crossover operation is to select two parents randomly in the population, and then randomly select the crossover point on each parent, and the separated parts are exchanged to generate two new offspring individuals. Mutation operation is to select a parent in the population with mutation probability, and then randomly select the mutation point on the parent to cause gene mutation.

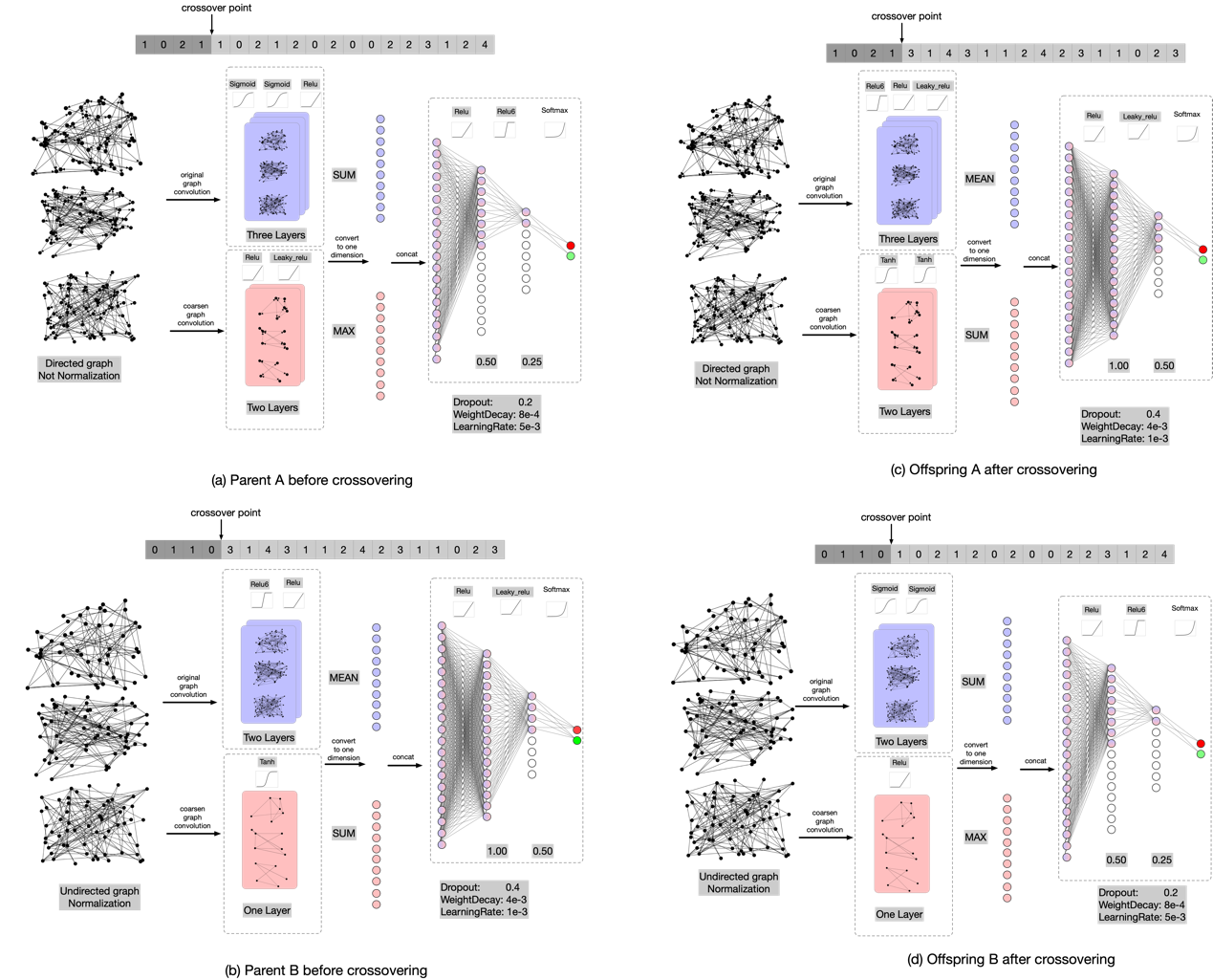


Figure GNN crossover operation.

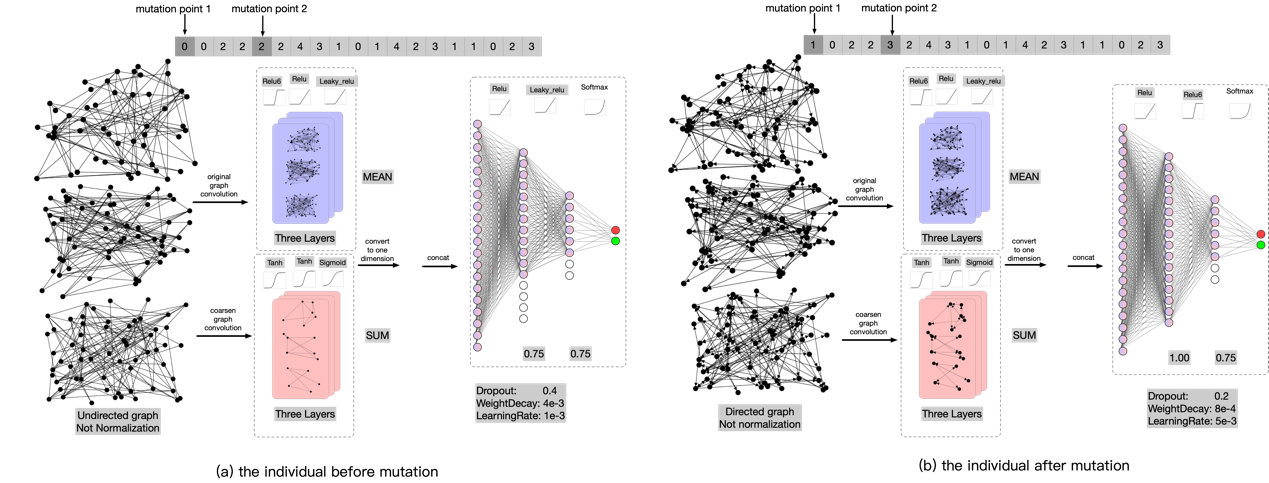


Figure GNN mutation operation

1. *Intrusion detection by convolutional neural network (CNN)*

T**he hexadecimal CAN IDS is transformed into binary to form a sample similar to an image. Each pixel is 0 or 1 respectively. According to the reference paper [48], the number of ID bits of vehicle CAN extended frame is 29 bits, and 29 frames are collected to construct 29 × 29 samples. Through the processing of samples, the detection task is transformed into a task similar to image classification.**  We follow the same block level design method as in [68]– [70]. Block is a small convolution network. To deal with different intermediate information more effectively in forward propagation, four kinds of convolution blocks, shown in Figure 6, are designed according to the different grid sizes of feature mapping. At the same time, the reduction block is designed to increase the deep receptive field, and halve the grid size of the feature map by applying all operations in steps of 2. According to the Convention of modern CNN Architecture [71] – [73], when the grid size of the feature graph is halved, we double the number of channels (filters) of the block to maintain a roughly constant hidden state dimension.

Inception ResNet is a deep convolution model designed to divide images into 1000 categories in the field of image classification and shows promising performance. The overall architecture of the CNN is shown in Figure 6. The input size is 29291, and the input data size is converted to 131328 through the stem module. After the four modules optimized by EA, the data size is 22896. Finally, the data is transformed into a binary classification vector of two dimensions through the softmax module. The whole convolution network architecture is shown in Figure 6.

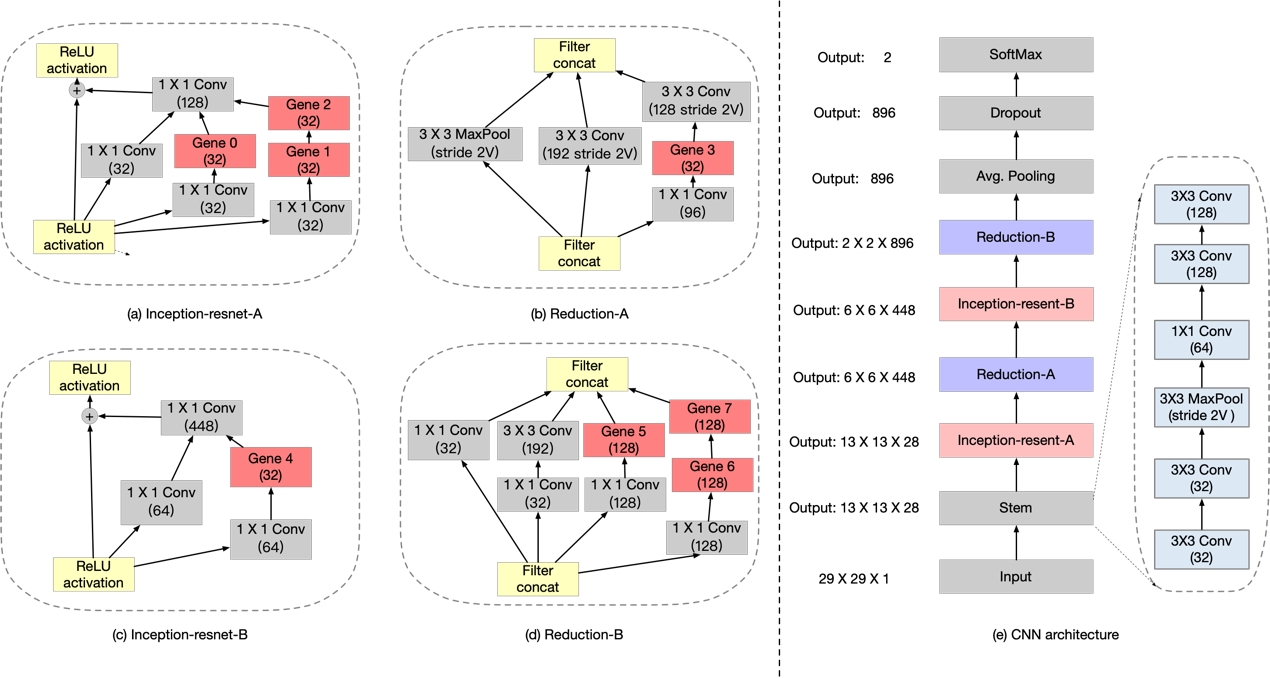


Figure evolutionary part of CNN architecture

Because an individual is retrained every time, the amount of calculation required is normally extensive even for a small population of individuals. To improve the evaluation efficiency of MOEA, an agent is used to exchange the weight parameters of corresponding neural network positions in the process of crossover operation and mutation operation. It can be seen from the schematic diagram that channels in each convolution block are different, and the number of gene channels in each convolution block is the same, which has no impact on the crossover operation, but will affect the mutation operation. At two gene exchange positions on the parent chromosomes, the corresponding weights of the neural network need to be exchanged, so the number of channels of the two genes needs to be the same. Both RedA and ResB convolution blocks have only one gene, and ResA and RedB convolution blocks have three genes respectively. Therefore, mutation can only be performed in ResA block or RedB block. The crossover operation of evolution part first takes out two parent individuals by the tournament selection method with n=2, then randomly selects the intersection on each parent, disconnects at the intersection, and exchanges the separated parts to generate two new offspring individuals. The crossover process is shown in Figure 7. Mutation operation is to mutate each parent with mutation probability p, and then randomly select two points on the parent to exchange. The mutation process is shown in Figure 8.

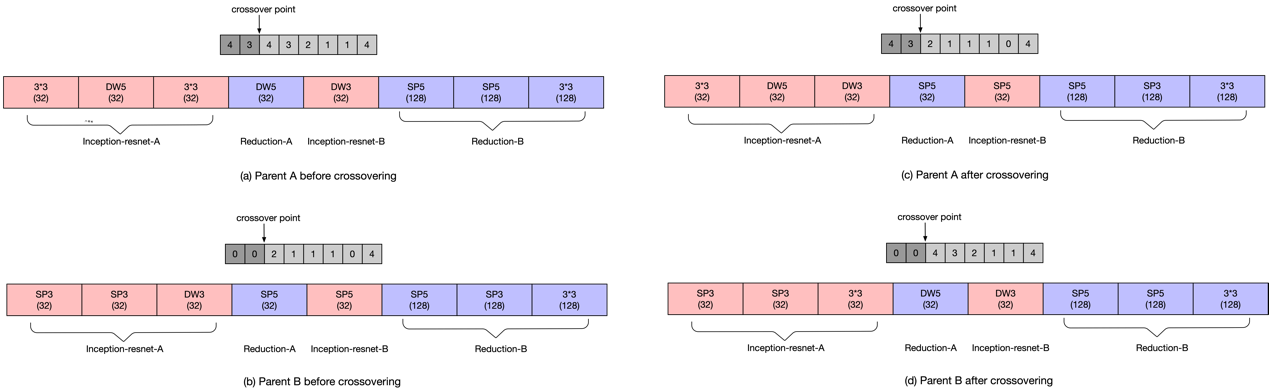


Figure CNN crossover process

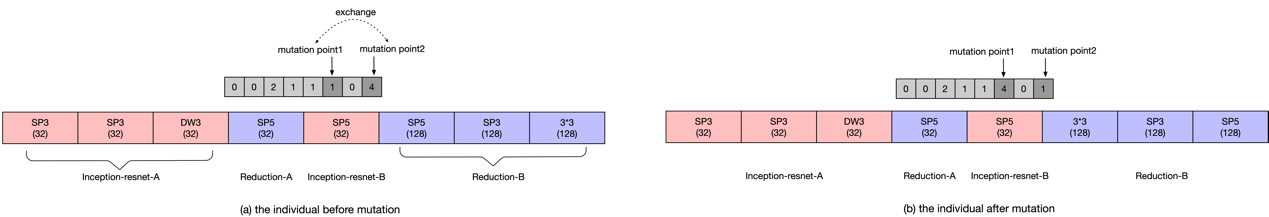


Figure CNN mutation operation

1. EXPERIMENTAL RESULTS
2. *Experiment setup and result*

Our network framework consists of GNN block and CNN block, so we need to convert the data into graph data and binary image data according to the above method for adapting to our proposed architecture. Before the two network blocks, there is a reinforcement learning network, which is responsible for dynamically collecting the samples of the input detection model. Our work is based on a publicly available datasets ATTACK & DEFENSE CHALLENGE 2020 DATASET [74] on IEEE-DATAPORT. The dataset contains two states of vehicle dynamic and stationary, and each part has two types of dataset files: intrusion and normal. In the intrusion file, several intrusion CAN messages are interspersed with normal message. We directly convert these labeled messages into a format that can be the input to the network for training and testing. Our aim is to identify the intrusion CAN messages quickly and accurately from the dataset. Each dataset file has millions of CAN data frames, which can better extract various features in the message. We use the dynamic message part of the vehicle in the dataset, 80% of the data are used for training and 20% of the data are used for testing.

The CNN part trains 26 primary generations in advance, each individual trains 200 epochs, and takes the best accuracy as the adaptability index. Then, after 30 generations of crossover and mutation, the best 26 individuals are selected as the parents of the next generation, and so on. Figure 9 shows the results of 30 generations of precision images. When the evolution process progresses, there are fewer and fewer individuals with low precision, whereas more individuals with better precision have evolved. After evolution the accuracy of every individual reached over 85%. As shown in Table III, although the accuracy of the end of evolution is almost the same as that of the early generation, in the last generation of evolution, the recognition accuracy of excellent individuals selected after training is 5% higher than that of the first generation.

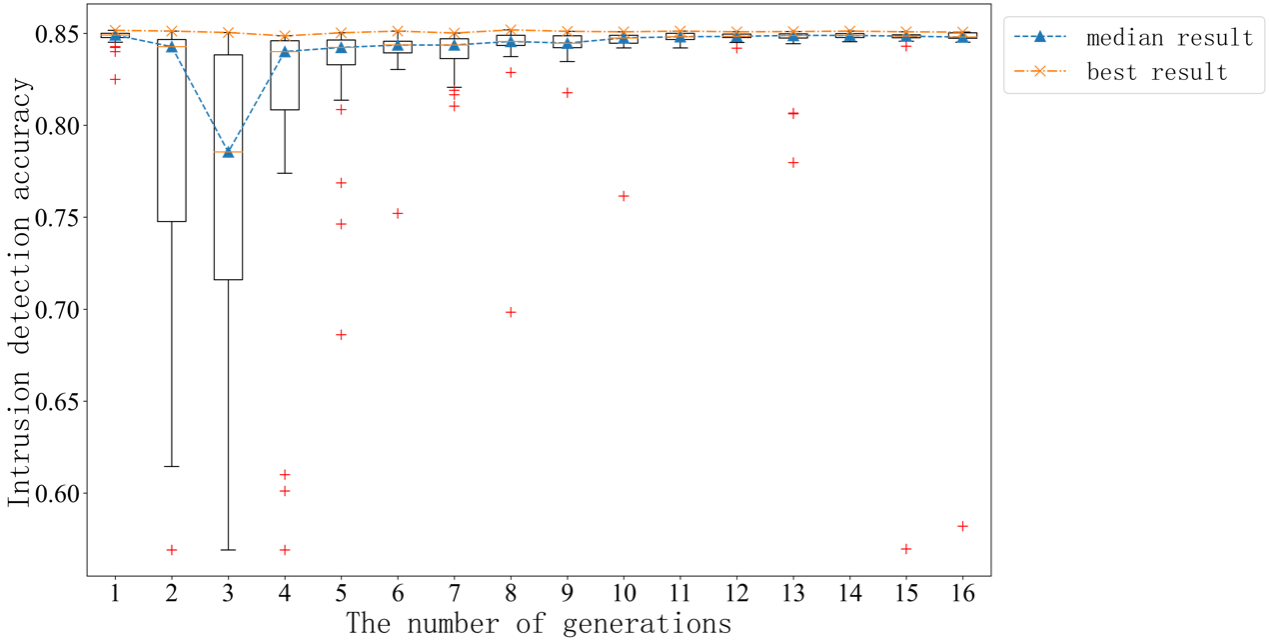


Figure CNN evolution result.

Table Individual comparison between early generation and post evolution

|  |  |  |  |
| --- | --- | --- | --- |
| Stage | Gene Code | Accuracy | Flops |
| First Gen | 2,4,2,0,3,4,0,3 | 85.13% | 2874.5M |
| After evolution without training | 4,4,4,2,3,0,2,3 | 85.15% | 2628.7M |
| After evolution with training | 4,4,4,2,3,0,2,3 | 90.20% | 2628.7M |

In the GNN part, first convert the data into graph data, and obtain the prediction results through graph coarsen, GCN, prediction layer, the number of GCN layers and prediction layers of the GNN etc. The neurons of each prediction layer are optimized by EA. At the beginning, 100 primary generations were randomly generated, and 7 individuals at the Pareto front were selected from the primary generation as the parents of the next generation, evolving for 30 generations. The complexity of 30 generations with flops on the x-axis and the accuracy of an error on the y-axis is shown in Figure 10. Accuracy of error refers to 1 – accuracy (1-acc), so the smaller the accuracy of error, the higher the accuracy. Flops refers to the number of matrix operations in the process of network recognition. They are divided by the number of matrix operations of the hypernetwork to convert the flops index into a range of 0 to 1. Red dots are used to identify the last generations of individuals. The last individuals evolved in the lower left corner of Figure 10 represent higher accuracy and lower complexity.

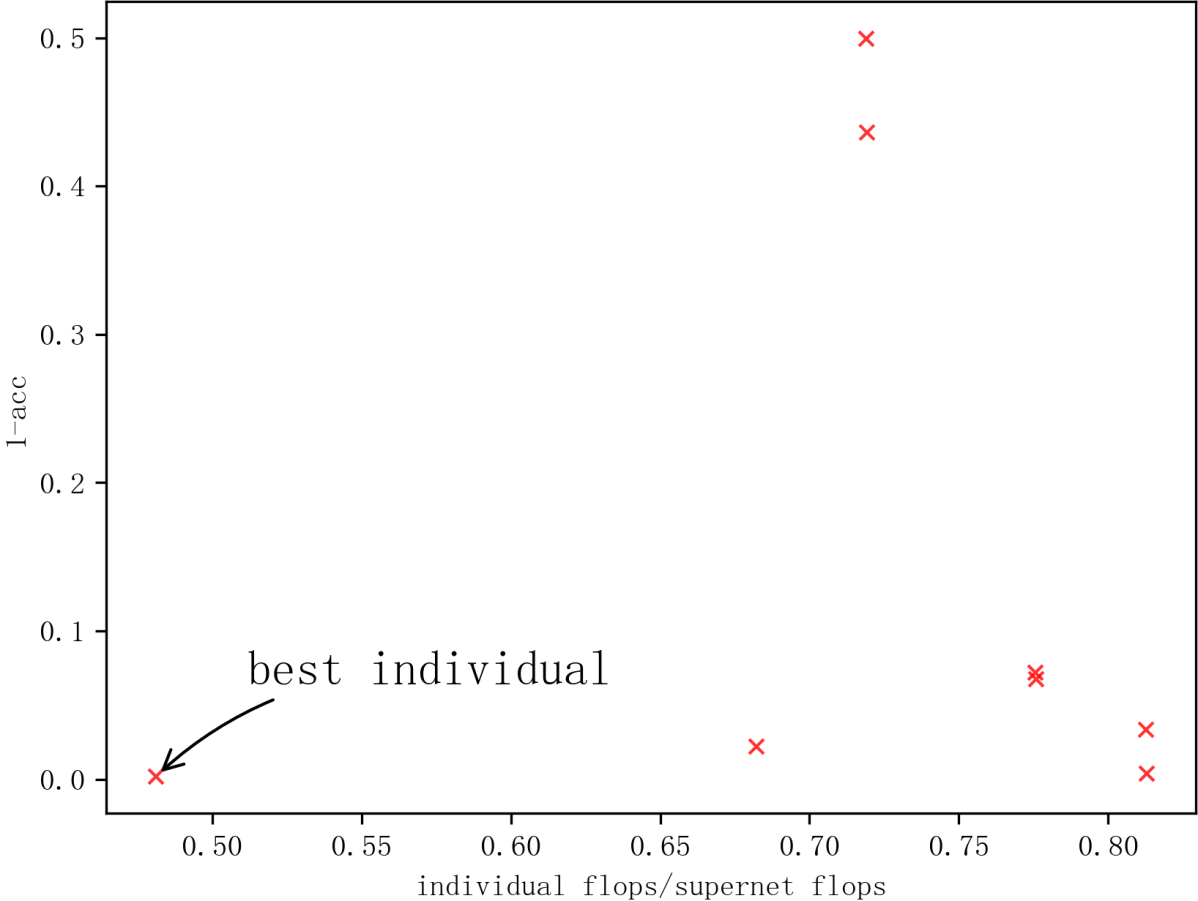
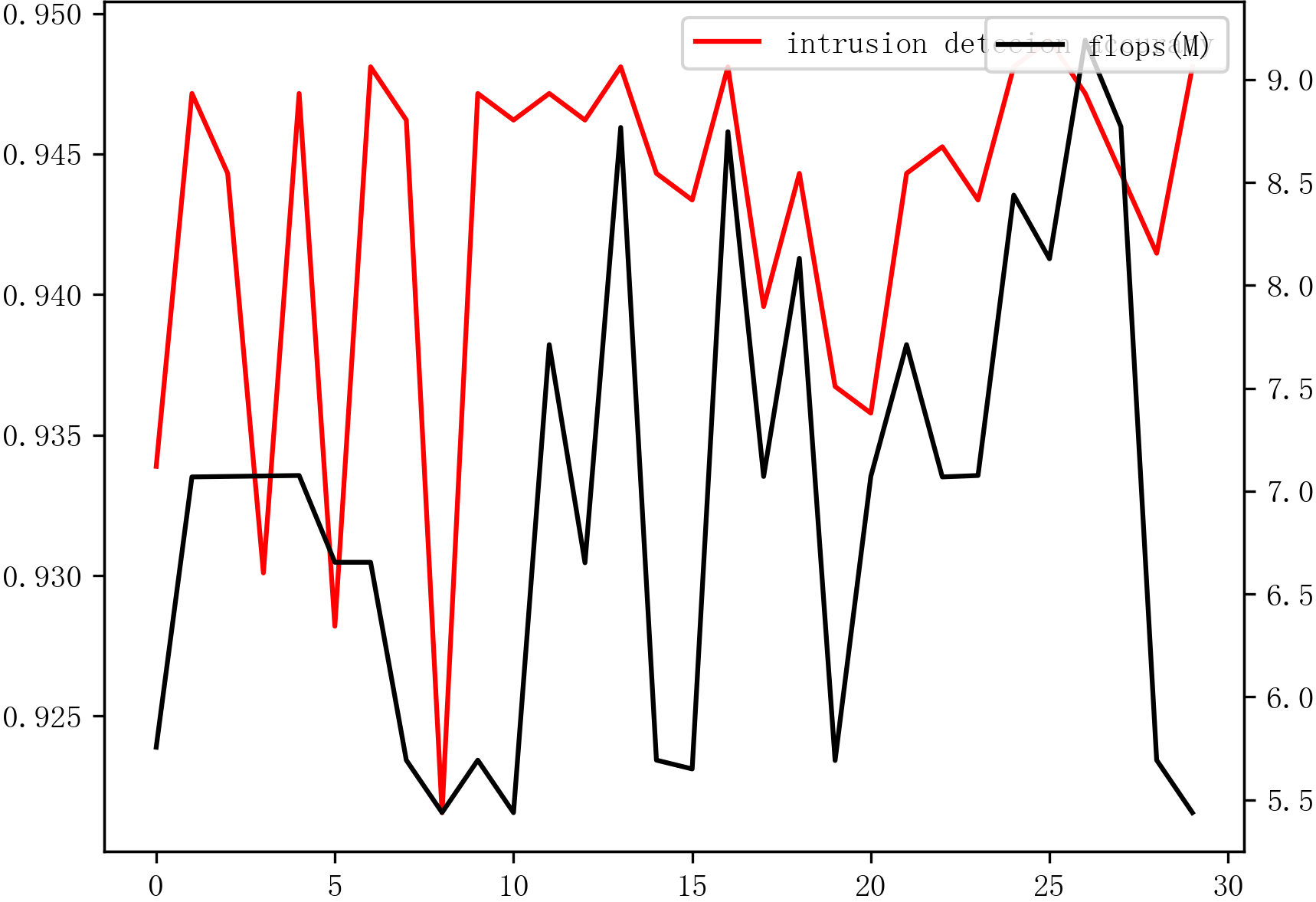
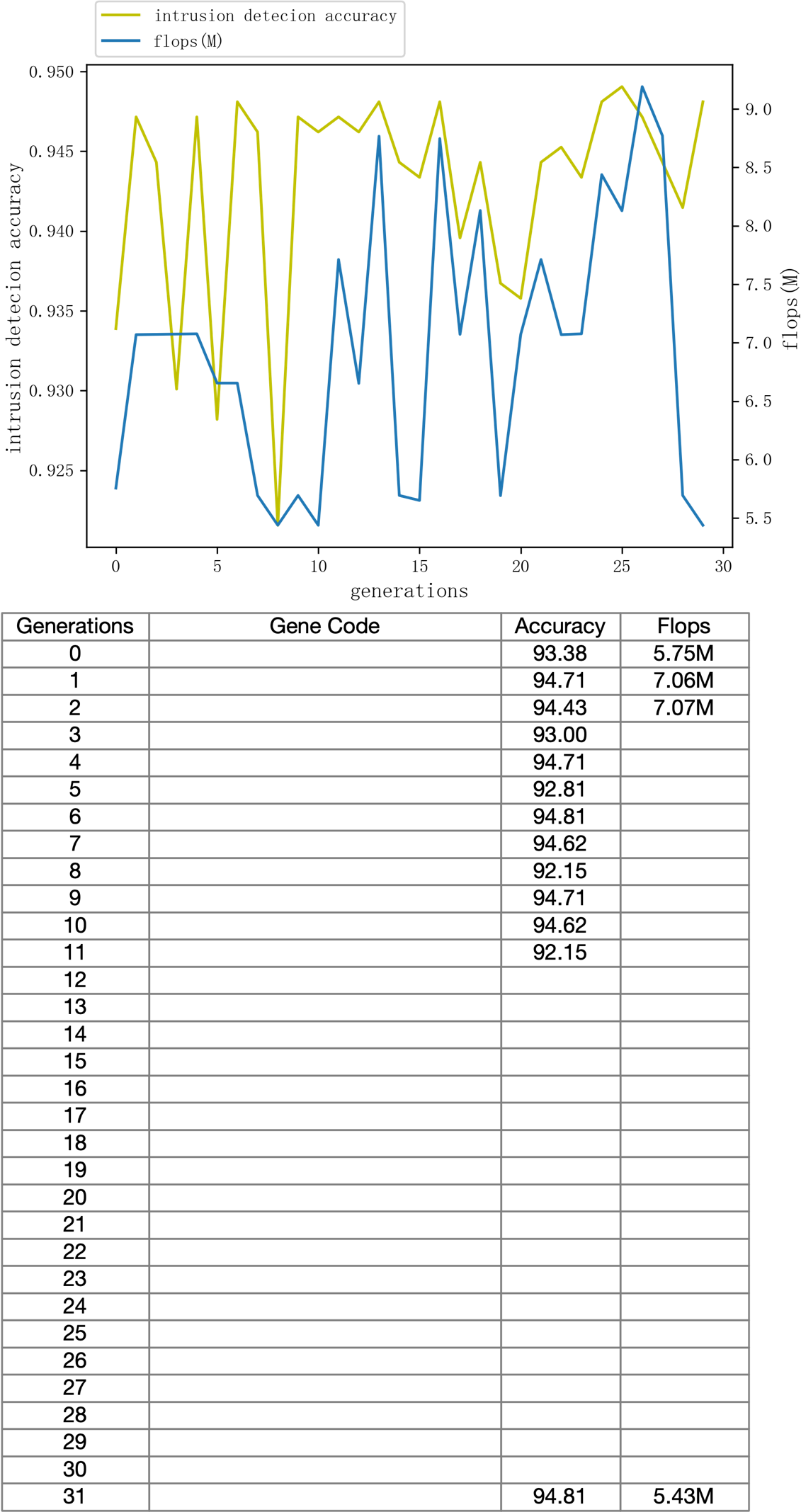


Figure GNN revolution result.

|  |  |  |  |
| --- | --- | --- | --- |
| Generation | Gene Code | Accuracy | Flops |
| 0 |  | 93.38% | 5.75 |
| 1 |  | 94.71% | 7.06 |
| 2 |  | 94.43% | 7.07 |
| 3 |  | 93.00 | 7.07 |
| 4 |  | 94.71 | 7.07 |
| 5 |  | 92.81 | 6.65 |
| 6 |  | 94.81 | 6.65 |
| 7 |  | 94.62 |  |
| 8 |  | 92.15 |  |
| 9 |  | 94.71 |  |
| 10 |  | …… | …… |





We compared the impact of using the direction information of graph data on Intrusion Detection. The pooling matrix required for graph collapse classification is composed of eigenvectors of the Laplacian matrix of subgraphs. We also compare the impact of Laplacian matrix normalization on intrusion detection. It can be seen from Figure 11 that the result of using both graph data direction information and Laplace normalization is not the best. Removing the direction information of graph data will improve the effect of intrusion detection, so more use of graph data information will not necessarily improve the results of intrusion detection. Removing the Laplacian matrix normalization will also make the intrusion detection effect better, and reduce the number of calculations needed for data preprocessing. Therefore, adding the direction information of graph data and the normalization information of the Laplace matrix into the search space is reasonable. The final training results of the three networks can be seen in Table IV.

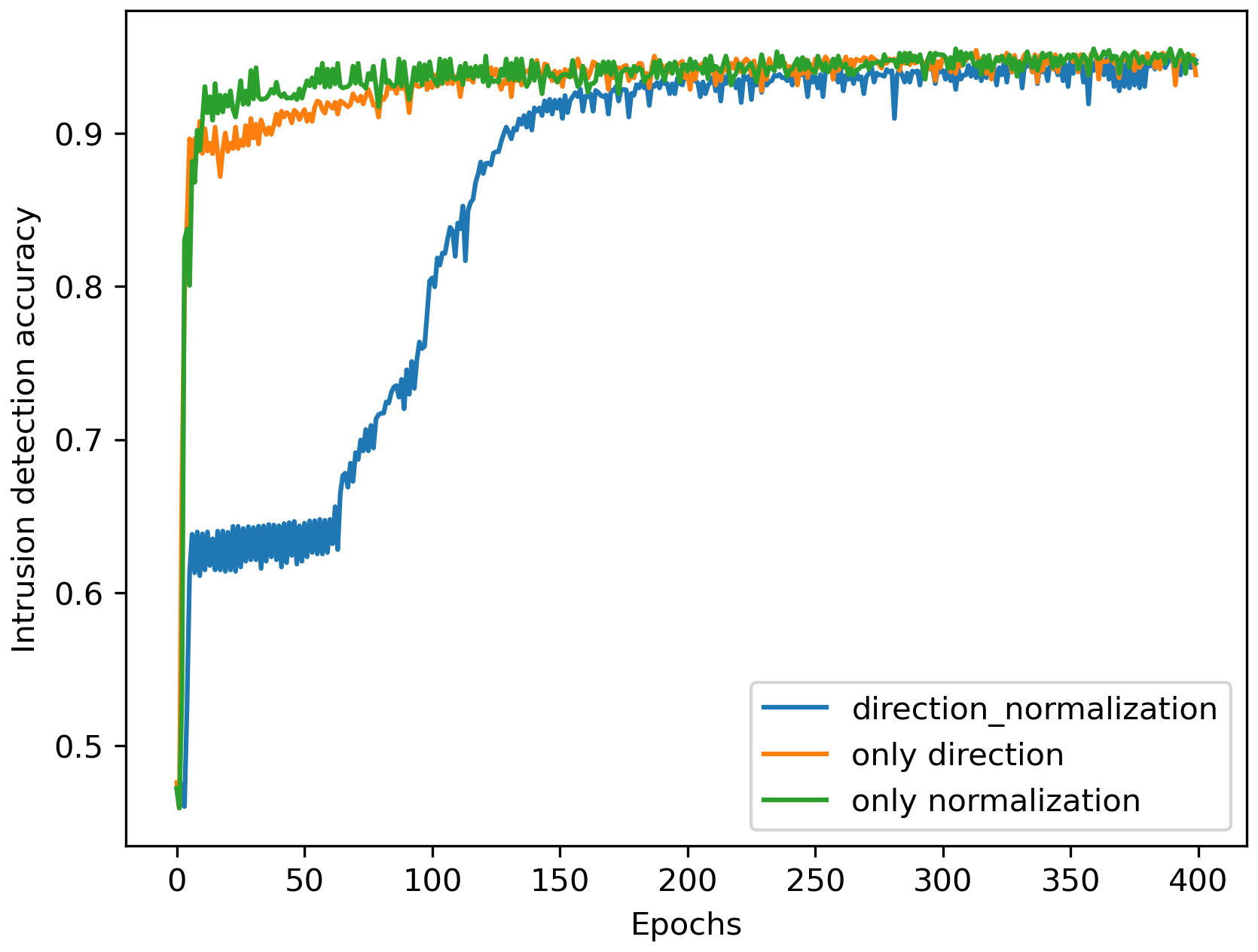


Figure The impact of graph data processing on Intrusion Detection. the three lines show the impact of directed/undirected graph and processing normalization/no normalization for subgraph’s laplacian matrices to learning logical feature of CAN. The final results of three individuals described on this figure shown in Table IV.

Table comparison results of three GNNs

|  |  |  |
| --- | --- | --- |
| Contrast Factor | GNN gene | Accuracy |
| Direction normalization | 1100 0000 000 0021 0124 | 94.86% |
| Only direction | 1000 0000 000 0021 0124 | 95.43% |
| Only normalization | 0100 0000 000 0021 0124 | 95.52% |

1. *Compare with other deep learning method*

This experiment compares the experimental results of our method with LSTM, DNN, CNN and GNN. Taylor in [49] uses LSTM and Fully Connected Neural Network to extract the timing features of the data part of CAN. However, each CANID needs to train a complete neural network separately, so it is computationally expensive. The loss function of the network is defined as the CrossEntropy loss of each bit corresponding to two adjacent frames of CAN messages. In the network verification stage, the maximum loss value of all bits is used as the basis for whether it is an intrusion message. The publicly available Car Hacking Challenge Dataset rev20Mar2021 [74], is used for training and verifying the neural network. They use the normal message part to train the LSTM network, and then use the data mixed with the abnormal messages to test the network. We find that good results can be obtained after a small number of epochs. It is mentioned in the paper that some bits with low change frequency can be ignored by analyzing the data segment of CAN ID. Kang et al. [75] additionally employ the 64-bit data segment of CAN messages to feed the bit stream into the DNN network directly. The advantage of this network, not discussed in detail in this paper, is that it is more efficient and can be directly input the network without data processing. We adopt five layers of fully connected layers. The first layer has 64 neurons to receive 64 bits CAN messages bitstream, the second layer has 128 neurons, the third layer has 512 neurons, the fourth layer has 256 neurons, the fifth layer has 32 neurons, and the sixth layer outputs the second classification. Through the verification of public dataset, the accuracy is low, but the neural network is simple and does not need data preprocessing. Song et al. [48] uses a similar convolutional network component to ours. We utilize our network framework directly. The identical design as described in the study may be directly constructed using the gene "44440444". The dataset in [48] was gathered by the author themselves. The classification accuracy of the public dataset does not match that of the author's own experiment, which is lower than that of the MOEA-evolved convolution network. Our algorithm combines the two advantages of the spatial feature extraction of CNN and the logical feature of GNN, compares the difference of two-dimensional variables output by the two networks when outputting the results, and takes the network result with large difference as the final result, which is the result of taking a network that is more confident in the intrusion detection. Using this method, the two forms of networks can complement each other. Table V shows the comparison result of these network architectures. It can be seen that our network can achieve higher recognition accuracy, which is very important for network security. From the results, it can be seen that the accuracy of a single GNN network is lower than that of the combination of two networks.

1. *Limitations of the Proposed Model*

*1.* *Although we use MOEA to optimize GNN of our model, the computational complexity of the proposed model combining RL, CNN and GNN, is higher than that of the previous work. The CNN has the highest computational complexity among the three neural networks. However, our model does indeed obtain the best accuracy of intrusion detection of CAN.*

*2.* *Converting the original CAN message into binary image and graph data is needed for intrusion detection of CNN and GNN. Therefore, compared with the LSTM [75] which directly input the CAN messages, our method needs additional data preprocessing.*

*3.* *CNN and GNN should detect the same CAN message at the same time. However, their detection system is slightly asynchronous by nature. For example, CNN can only recognize multiple frames of 29 for each packet, whereas the GNN accepts the entire packet as input. This leads to a minor error in CNN intrusion detection.*

Table Compare with other algorithms

|  |  |
| --- | --- |
| Deep Learning Method | Accuracy |
| LSTM | 97.02% |
| DNN | 95.36% |
| CNN | 85.06% |
| GNN | 96.43% |
| Our method without RL | 95.17% |
| Our method with RL | **99.87%** |

1. CONCLUSION AND FUTURE WORK

We propose an intrusion detection model in the vehicle control area network based on a lower complexity deep neural network and increased accuracy using a multi-objective evolutionary algorithm assisted neural architecture search. As far as we know, GNN is used for CAN intrusion detection for the first time in this work, and detection results of GNN alone can reach more than 95%. We also combine the dual advantages of GNN and CNN in this work. The detection results of the two networks are complementarily used. In addition, CAN's logical and spatial properties are also employed. Although it can achieve pretty good recognition accuracy, compared with the previously published work, the computational complexity of our network is slightly higher. Evolutionary algorithms (EAs) are critical in the future and exhibit great promise in asynchronous computing systems in terms of cost and reliability. We may expand our proposed EAs to significantly simplify network design by developing a more extensive search space. A combination model of GNN and LSTM (replacing computationally expensive CNN) can be considered to reduce model’s complexity. It can be considered to search the overall architecture of RL, CNN and GNN concurrently, so as to improve the detection accuracy and reduce the complexity. Future research may also attempt to increase the number of MOEA objective functions, e.g., false-positive and true positive rates, to produce a more robust intrusion detection model.

[1] A. Prasad and P. Shanthi, "Automotive Electronic Control Unit Reprogramming Using Delta Method-A Review," in *2021 Asian Conference on Innovation in Technology (ASIANCON)*, 27-29 Aug. 2021 2021, pp. 1-6, doi: 10.1109/ASIANCON51346.2021.9544595.

[2] H. Zhang, X. Meng, X. Zhang, and Z. Liu, "CANsec: A Practical in-Vehicle Controller Area Network Security Evaluation Tool," *Sensors,* vol. 20, no. 17, p. 4900, 2020. [Online]. Available: <https://www.mdpi.com/1424-8220/20/17/4900>.

[3] M. M. K. Tareq, O. Semiari, M. A. Salehi, and W. Saad, "Ultra Reliable, Low Latency Vehicle-to-Infrastructure Wireless Communications with Edge Computing," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 9-13 Dec. 2018 2018, pp. 1-7, doi: 10.1109/GLOCOM.2018.8647367.

[4] B. I. Kwak, M. L. Han, and H. K. Kim, "Cosine similarity based anomaly detection methodology for the CAN bus," *Expert Systems with Applications,* vol. 166, p. 114066, 2021/03/15/ 2021, doi: <https://doi.org/10.1016/j.eswa.2020.114066>.

[5] 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, doi: 10.1109/ACCESS.2020.2982544.

[6] G. Karopoulos, G. Kambourakis, E. Chatzoglou, J. L. Hernández-Ramos, and V. Kouliaridis, "Demystifying In-Vehicle Intrusion Detection Systems: A Survey of Surveys and a Meta-Taxonomy," *Electronics,* vol. 11, no. 7, p. 1072, 2022. [Online]. Available: <https://www.mdpi.com/2079-9292/11/7/1072>.

[7] H. J. Jo and W. Choi, "A Survey of Attacks on Controller Area Networks and Corresponding Countermeasures," *IEEE Transactions on Intelligent Transportation Systems,* pp. 1-19, 2021, doi: 10.1109/TITS.2021.3078740.

[8] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "An Effective In-Vehicle CAN Bus Intrusion Detection System Using CNN Deep Learning Approach," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 7-11 Dec. 2020 2020, pp. 1-6, doi: 10.1109/GLOBECOM42002.2020.9322395.

[9] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network Representation Learning: A Survey," *IEEE Transactions on Big Data,* vol. 6, no. 1, pp. 3-28, 2020, doi: 10.1109/TBDATA.2018.2850013.

[10] P. Cui, X. Wang, J. Pei, and W. Zhu, "A Survey on Network Embedding," *IEEE Transactions on Knowledge and Data Engineering,* vol. 31, no. 5, pp. 833-852, 2019, doi: 10.1109/TKDE.2018.2849727.

[11] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph Neural Networks in Recommender Systems: A Survey," *ACM Comput. Surv.,* 2022, doi: 10.1145/3535101.

[12] Q. Xie, J. Huang, P. Du, M. Peng, and J.-Y. Nie, "Graph Topic Neural Network for Document Representation," presented at the Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 2021. [Online]. Available: <https://doi.org/10.1145/3442381.3450045>.

[13] S. Attarilar *et al.*, "The Toxicity Phenomenon and the Related Occurrence in Metal and Metal Oxide Nanoparticles: A Brief Review From the Biomedical Perspective," (in English), *Frontiers in Bioengineering and Biotechnology,* Review vol. 8, 2020-July-17 2020, doi: 10.3389/fbioe.2020.00822.

[14] J. Zhou *et al.*, "Graph neural networks: A review of methods and applications," *AI Open,* vol. 1, pp. 57-81, 2020/01/01/ 2020, doi: <https://doi.org/10.1016/j.aiopen.2021.01.001>.

[15] Q. Li, X. Peng, Y. Qiao, and Q. Peng, "Learning label correlations for multi-label image recognition with graph networks," *Pattern Recognition Letters,* vol. 138, pp. 378-384, 2020/10/01/ 2020, doi: <https://doi.org/10.1016/j.patrec.2020.07.040>.

[16] M. Sun, S. Zhao, C. Gilvary, O. Elemento, J. Zhou, and F. Wang, "Graph convolutional networks for computational drug development and discovery," *Briefings in Bioinformatics,* vol. 21, no. 3, pp. 919-935, 2019, doi: 10.1093/bib/bbz042.

[17] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence,* vol. 33, no. 01, pp. 922-929, 07/17 2019, doi: 10.1609/aaai.v33i01.3301922.

[18] J. Skarding, B. Gabrys, and K. Musial, "Foundations and Modeling of Dynamic Networks Using Dynamic Graph Neural Networks: A Survey," *IEEE Access,* vol. 9, pp. 79143-79168, 2021, doi: 10.1109/ACCESS.2021.3082932.

[19] A. Taheri, K. Gimpel, and T. Berger-Wolf, "Learning to Represent the Evolution of Dynamic Graphs with Recurrent Models," presented at the Companion Proceedings of The 2019 World Wide Web Conference, San Francisco, USA, 2019. [Online]. Available: <https://doi.org/10.1145/3308560.3316581>.

[20] X. Wang *et al.*, "Traffic Flow Prediction via Spatial Temporal Graph Neural Network," in *Proceedings of The Web Conference 2020*: Association for Computing Machinery, 2020, pp. 1082–1092.

[21] Y. Ma, S. Wang, C. C. Aggarwal, and J. Tang, "Graph Convolutional Networks with EigenPooling," presented at the Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery &amp; Data Mining, Anchorage, AK, USA, 2019. [Online]. Available: <https://doi.org/10.1145/3292500.3330982>.