Deep Neural Network optimized by Multi-Objective Evolutionary Algorithm for Intrusion Detection of Vehicle CAN Communication

Bin Cao, Tian Shan, IEEE Publication Technology, *Staff, IEEE,*

***Abstract*—With the popularization of vehicle electrification and networking, more and more electronic devices have emerged in the vehicle. It is very necessary to ensure the safety and reliability of the communication between the various modules in the vehicle. Especially as the most commonly used communication protocol on vehicles, Controller Area Network (CAN) communication. Neural architecture search (NAS) is a promising method for automatically design neural architectures. NAS adopts a search strategy to explore the predefined search space to find outstanding performance architecture with the minimum searching costs. In this paper we propose an intrusion detection system (IDS) based deep neural network which is optimized by Multi-Objective Evolutionary Algorithm to promote accuracy and decrease complicity of the neural network. Through graph neural network and convolutional neural network to explore the internal logical feature and spatial feature of CAN message. Because the different CAN length of the different functions of the vehicle CAN message, we take advantage of reinforcement learning to dynamic get a CAN package which as a sample sent to our model. we also propose a method to converting original CAN message to graph data by use of its time sequence feature. We performed an experimental study using the datasets provided and collected while ”Car Hacking: Attack & Defense Challenge” to evaluate our detection system. The experimental results demonstrate that the proposed IDS has significantly low false negative rates and error rates when compared to other deep-learning algorithms.**

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

I. Introduction

IN the last few decades, the implementation of automotive electronics has experienced a rapid growth [1]. This trend has resulted in several changes in the vehicular ecosystem. Drive-by wire (DBW) technology, for example, uses of electronic or electrical systems in the control systems, such as the throttle, brake, and steering, which were traditionally controlled using mechanical linkages. CAN provides a simple and reliable communication protocol as the standard of an in-vehicle network [2], connecting not only sensors and controllers but also the Internet. The adoption of CAN accelerates the applications with the emergence of Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication interfaces [3]. However, the openness of the vehicular system increases the risk of malicious cyberattacks that can severely threat human life.

However, the conventional in-vehicle networks are tremendously vulnerable with the cyberattacks as CAN is developed for an isolated physical system before. For example, every ECU sharing a CAN bus can obtain any ECU-to-ECU message. Furthermore, a CAN packet has no sender's identification as shown in Fig. 1. Recent research works point the weakness of the security.

Koscher et al. [4] conduct 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 are shown in [5], [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 seen during normal driving, and so they are detectable trivially.

To address the aforementioned challenges, In this paper, we propose an IDS using a deep neural network (DNN) to secure the CAN bus from cyber-attacks, 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:

1) First, multi-objective evolutionary algorithm is proposed to optimize the structure and super parameters of CNN and GNN in CAN message intrusion detection.
2) Second, a method of converting CAN into graph data is proposed to convert CANID from time domain to space domain, so as to extract the logical sequence characteristics of CANID using graph neural network.
3) Third, propose reinforcement learning neural network to dynamically obtain the length of CAN message detected each time to adapt to the dynamics and uncertainty of the CAN. The graph neural network and convolution network are used to extract the logical and spatial features of the can message at the same time, and the model is verified by the public CAN dataset. A large number of experiments show that the proposed detection model has better detection performance compared with other deep learning algorithms.

This paper was produced by the IEEE Publication Technology Group. They are in Piscataway, NJ.

Manuscript received April 19, 2021; revised August 16, 2021.

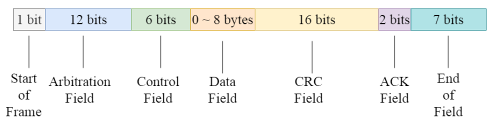

Fig. 1. Format of the CAN data frame.

II. RELATED WORK

A. Conventional intrusion detection of CAN

The CAN frame is generally unable to support Message Authentication Code (MAC) [7] and other methods of securing communication. Some researchers have attempted to either create new protocols or spread MAC across multiple transmissions. in [8], Tashiro et al. for tampering detection can be conducted for both individual frames and entire sections, they propose a protocol that provides protection against replay, masquerading and injection attacks by sending a partial MAC in each frame. Nowdehi et al. [9] examine many of these altered protocols in light of five criteria for potential CAN message authentication solutions from an industry perspective. They found that no solutions met all the criteria. VatiCAN takes the approach of utilizing maintenance support, "sufficient implementation details" and no excessive overhead, while WooAuth alters the extended CAN protocol to allow more space for authentication codes. Finally, the authors suggest that the CAN bus might "be fundamentally unsuited for secure communication" [9]

B. Deep learning intrusion detection of CAN

Intrusion detection system (IDS) can combine machine learning to train itself to identify abnormal behavior, which can be used as an alternative or supplement to Mac. IDS can prevent spoofing, injection, bus shutdown and denial of service attacks. In [10], Choi et al. Introduced a method called voltage IDS, which uses the inconsistency of ECU signals, first conducts training and testing, identifies the signal characteristics at this stage, and then uses the training data to verify whether the ECU has been damaged. Voltage IDS can detect camouflage attacks by using multi-class classifiers, one of which corresponds to an ECU. It predicts the most likely sender and compares this information with the actual can ID of the message. If they are different, a camouflage attack is detected. In [11], song et al, by converting the ID part of the CAN frame into binary. Used the changed ResNet model to extract the features of the binary text, and learned the features of the intrusion message and the normal message for intrusion detection. Their experimental results show that compared with the traditional machine learning algorithm, this algorithm has lower false positive rate and false positive rate. In [12], Taylor et al. Proposed using deep learning methods for intrusion detection, because they are generated directly from the bit stream on the network, the execution efficiency of these functions is high and the complexity is low. This technology monitors the exchange packets in the vehicle network while training the characteristics offline, and provides a real-time response to the attack with a significantly high detection rate in their experiment.

C. 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 specific search space with constrained resources [13], [14]. Sun et al. [15] used EA with variable coding length to automatically evolve the architecture of CNN. William et al. [16] introduced an evolutionary NAS coding strategy based on directed acyclic graphs (DAG), which has better performance than the randomly generated CNN architecture. Real et al. propose Amoebanet [17], which uses improved tournament selection to evolve network groups, and achieves better results on Imagenet than the handmade model. Wang et al. [18] 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 GAN model. Sun et al. [15] proposed a variable length coding, which can represent different numbers of building blocks and layers to search for the best depth convolutional neural network. Yin [19] uses evolutionary multi-objective method to design CNN architecture, which uses probabilistic SMBO to maximize classification performance and minimize network reasoning time. Elsken et al. [20] described NAS as a bi-objective optimization problem, in which two objectives are to maximize performance and minimize computing resources. Lu et al. [21] proposed nsganet, which can automatically design the network, maximize the model performance and minimize floating point operations (flops).

D. 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 data set and then evaluated on the validation data set 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 methods, they usually need a lot of fitness evaluation, which makes EvoNAS computationally difficult to implement. For example, on CIFAR10 and CIFAR100 datasets, CNN-GA [22] consumes 35 GPU days and 40 GPU days respectively, genetic CNN method [23] consumes 17 GPU days, and large-scale evolutionary algorithm [24] 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 [25] we searched is not only sharing the weight of hypernetwork, but also fine-tuning through training again. MetaQNN [26] 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 [27] uses an alternative model to predict the accuracy of the structure, adding an additional 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 [28] 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 a large number of samples to learn. ChamNet [29] 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.

III. PROPOSED METHOD

The architecture we proposed is shown in the figure 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 message. Intrusion detection part is constructed by graph neural network (GNN) and convolutional neural network (CNN). The network architecture of GNN and CNN is optimized by EA. The parts optimized by EA are marked with dotted lines in the figure2. They are the convolution layer of CNN, the GCN layer of graph neural network and the full connection layer and etc. The specific search space and search process are described below. 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.

A. Reinforcement Learning (RL)

RL network is able to dynamic collect CANs of every detect sample which will send to GNN and CNN. Before each intrusion detection, a certain length of CAN message data is dynamically 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 may also need different frames to complete, so the RL network is used to solve the problem of dynamic acquisition. Before training RL network, a GNN with intrusion detection ability is trained first. When training the GNN, the length of input is generated randomly. There are two convolution layers in our GNN. The output of the convolution layer is converted into one-dimensional vectors, and the two one-dimensional vectors are combined as the state input of RL network. The reward function is designed by using the recognition results of GNN. If the graph network recognition is correct, it will get a positive reward, and if the recognition is wrong, it will get a negative penalty. The RL network architecture design of this work refers to TD3 [30]. To avoid overestimation of value network, 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. As shown in (1).

$$output\_dim_{RL} = MAX(CAN\_LEN) - MIN(CAN\_LEN) \tag{1}$$

B. Graph Neural Network (GNN)

Our work is based on a publicly available datasets which is ATTACK & DEFENSE CHALLENGE 2020 DATASET [31] which is issues on IEEE-DATAPORT. the dataset contains two states of vehicle driving 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 messages. Directly convert these labeled messages into a format that can be input to the network for training and testing. Our task is to quickly and accurately identify the intrusion CAN messages from the dataset. Each data file has millions of CAN data frames, which can better extract various features in the message. First, we set the previous frame of message to point to the next frame of message. Because the message ID is limited, not all messages will point to a new message, it may 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, a large acceleration signal appears when braking, or a vehicle ignition signal appears without a car key signal. These are unreasonable logic. It is 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 messages. Our proposed approach is shown in Figure. 3.

The graph neural network in this work is a graph level classification task, and the network architecture of graph classification task refers to [32]. It 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 part of figure G. N_k represents the number of nodes in the

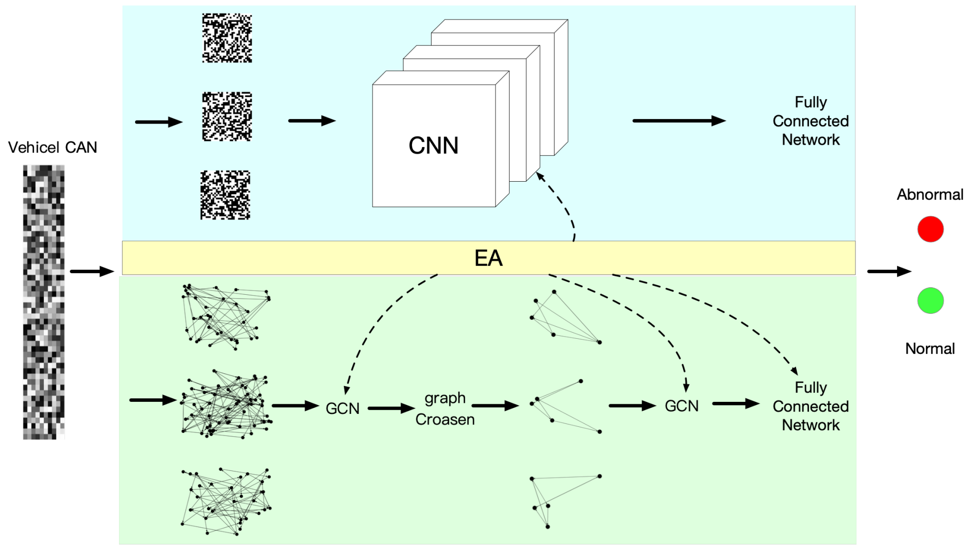

Fig. 2. Overall framework of network architecture.

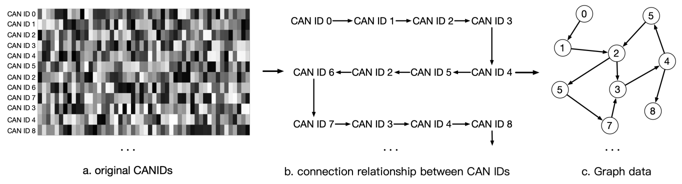
CAN ID 0
CAN ID 1
CAN ID 2
CAN ID 3
CAN ID 4
CAN ID 5
CAN ID 2
CAN ID 6
CAN ID 7
CAN ID 3
CAN ID 4
CAN ID 8
...
a. original CANIDs
CAN ID 0 → CAN ID 1 → CAN ID 2 → CAN ID 3
CAN ID 6 ← CAN ID 2 ← CAN ID 5 ← CAN ID 4
CAN ID 7 → CAN ID 3 → CAN ID 4 → CAN ID 8
...
b. connection relationship between CAN IDs
0
1
2
5
4
3
5
7
8
...
c. Graph data

Fig. 3. Construct graph data from original CAN.

subgraph $G^{(k)}$. $\Gamma^{(k)}$ is the list of nodes in the subgraph $G^{(k)}$. Each subgraph can be regarded as a super node of graph G. Define sampling operator $C^{(k)} \in \mathbb{R}^{N\times N_k}$ as follow:

$$C^{(k)}\left[i,j\right] = 1 \ if \ and \ only \ if \ \Gamma^{(k)}\left(j\right) = v_i \tag{2}$$

$C^{(k)}\left[i,j\right]$ represents the element in i-th row and j-th column of $C^{(k)}$, $\Gamma^{(k)}\left(j\right)$ represents the jth node in the node list $\Gamma^{(k)}\left(j\right)$. This operation indicates the correspondence between the node in the subgraph $G^{(k)}$ 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, we refer to [graph collapse] and use Fourier transform to design pooling operation. The pooling operation pools the constructed graph signal G into G_{coar}. Pooling operation is based on the Fourier transform of each subgraph $G^{k}{}_{k=1}^{K}$. The Laplace matrix of the subgraph $G^{(k)}$ is $L^{(k)}$. $u_1^{(k)}, \ldots, u_{N_k}^{(k)}$ represents the all eigenvectors of the Laplacian matrix $L^{(k)}$ of the k-th subgraph. Then use the upsampling operation $C^{(k)}$ to upsample the eigenvector to the whole graph G. The upsampling equation is represented by (3).

$$\bar{u}_l^{(k)} = \mathbf{C}^{(k)}\mathbf{u}_l^{(k)}, l = 1 \ldots N_k \tag{3}$$

$\Theta_l \in \mathbb{R}^{N\times K}$ represents the pooling matrix containing the L-th eigenvector of all subgraphs.

$$\Theta_l = \left[\bar{u}_l^{(1)}, \ldots, \bar{u}_l^{(k)}\right] \tag{4}$$

Each subgraph does not necessarily have the same number of nodes, that is, the number of eigenvectors of each subgraph is not necessarily equal. $N_{max} = \max_{k=1,\ldots,K} N_k$ represents the maximum number of nodes in all subgraphs. For the subgraph $G^{(k)}$ owns N_k nodes, the lst pooling operation is expressed as:

$$X_l = \Theta_l^T X \tag{5}$$

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

$$X_{coar} = \left[X_0, ..., X_l, \ ..X_{N_{max}}\right] \tag{6}$$

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 do a graph convolution 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 spliced into one input to the full connection layer for classification.

C. optimize GNN by EA

How to find accurate recognition and need less computation is what our EA wants to achieve. The chromosome is divided into nine parts. The first part of chromosome is position 0. There is a gene indicating whether to use the direction information of graph data constructed by CAN IDs. The graph data can determine the direction of the edge of

TABLE I
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 0.75 1.00
5	6	Dropout	0.05 0.1 0.2 0.3 0.4 0.5
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

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 Laplacian matrix before calculating the eigenvector of each subgraph. The regularized Laplacian matrix L_{norm} and the nonregularized Laplacian matrix L of the graph network are expressed as (7)(8).

$$L = D - A \tag{7}$$

$$L_{norm} = I - D^{-1/2} A D^{1/2} \tag{8}$$

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 layer, and the design reference [33]. 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 super parameters in the network structure, 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 fully connection layers. The specific genes and corresponding relationships are shown in Table 1. ed. There are still some deficiencies in this work. Although it can achieve very good recognition accuracy, compared with previous work, the complexity of our network is higher. Our future work can further apply evolutionary algorithm to simplify the structure of the network more carefully.

TABLE II
CNN SEARCH SPACE

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

D. convolutional neural network (CNN)

In this paper, the hexadecimal CAN IDS is transformed into binary to form a sample similar to the image. Each pixel is 0 or 1 respectively. According to the reference paper, the number of ID bits of vehicle CAN extended frame is 29 bits, and 29 frames are collected × 29 samples. Deep convolution networks, such as variants of inception and ResNet, are designed and constructed by stacking multiple blocks. The network structure 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 stipulates layered 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 or tasks. Therefore, we follow the same block level design method as in [34]–[36]. 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 3, 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 [37]–[39], 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.

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 study and the corresponding genotype-phenotype mapping are shown in Table 2. 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.

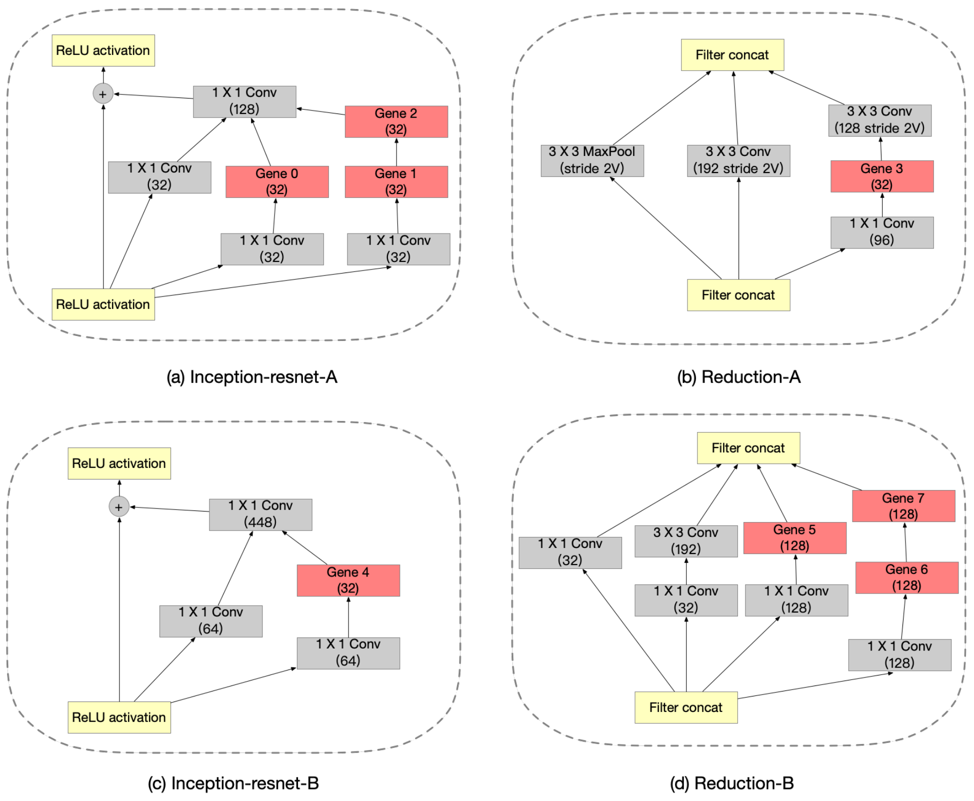

Fig. 4. Res-Inception blocks of CNN.

We define that the crossover operation of genetic algorithm is to randomly select a point on two parental chromosomes to disconnect and exchange chromosome genes with each other. The mutation operation of genetic algorithm is to randomly select two genes on a chromosome for exchange. Because the individual is retrained every time, the amount of calculation required is very huge. To improve the evaluation efficiency of genetic algorithm, agent method 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 of four different convolution blocks that the number of gene channels in each convolution block is 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 a chromosome, and 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.

Inception ResNet is a kind of deep convolution model. It is designed to divide images into 1000 categories in the field of image classification, and shows very excellent performance. The overall architecture of the CNN is shown in Figure 5. The input size is $29 \times 29 \times 1$, and the input data size is converted to $13 \times 13 \times 28$ through the stem module. After the four modules optimized by EA, the data size is $2 \times 2 \times 896$. Finally, the data is transformed into a binary classification vector with dimension 2 through the softmax module. The whole convolution network architecture is shown in Figure 5.

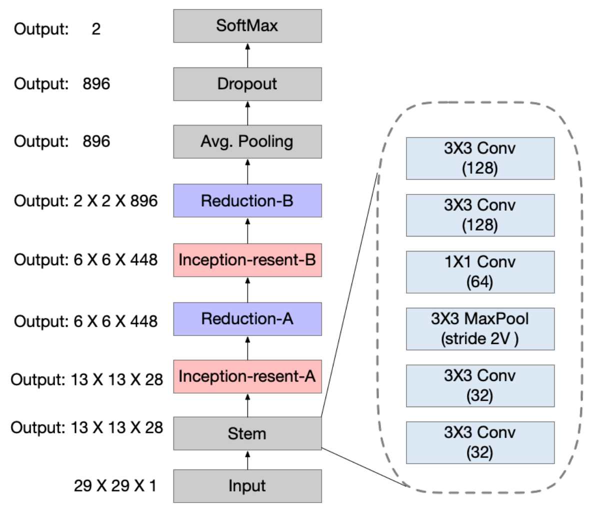

Fig. 5. Res-Inception blocks of CNN.

IV. Experiment and Result

A. experiment setup and result

Use common dataset, Car_Hacking_Challenge_Dataset_rev 20Mar2021, tests our proposed algorithm framework and compares it with some other deep learning methods. The dataset is divided into dynamic and static parts. We use the

TABLE III
CNN SEARCH SPACE

Stage	Gene code	accuracy	Complexity
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

dynamic message part of the vehicle in the dataset, 80% of the data is used for training and 20% of the data is used for testing. Our network framework consists of GNN block and CNN block, so we need to convert the data into graph data and binary 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 system. 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. 30 generations of precision images are obtained as shown in figure4, and each generation is marked with different colors. It can be seen from the figure that with the evolution process, there are fewer and fewer individuals with low precision, and individuals with higher precision have evolved. After evolution the accuracy of every individual reached over 85%. As shown in Table3, 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. In the GNN part, first convert the data into graph data, and obtain the prediction results through graph coarsen, GCN, prediction layer, etc. The number of convolution layers and prediction layers of the graph network. 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, flops, is x-axis, accuracy error, acc_ error is the image of the y-axis. Red dots are used to mark the dots of the last three generations of individuals. Flops refers to the number of matrix operations in the process of network recognition. Divide by the number of matrix operations of hypernetwork, for converting the flops index into a range of 0 to 1. Acc_ error refers to 1-acc, so the smaller the acc_error, the higher the accuracy. It can be seen from the figure that the last individual evolved to the position in the lower left corner of the figure, representing higher accuracy and lower complexity.

B. compare with other deep learning method

This experiment compares the experimental results of our method with LSTM, DNN, CNN and GNN. [12] 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

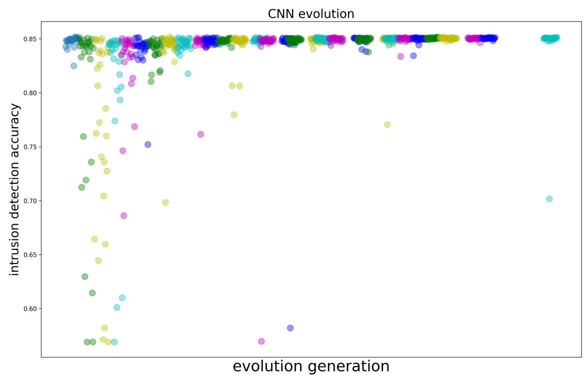

Fig. 6. CNN evolution result.

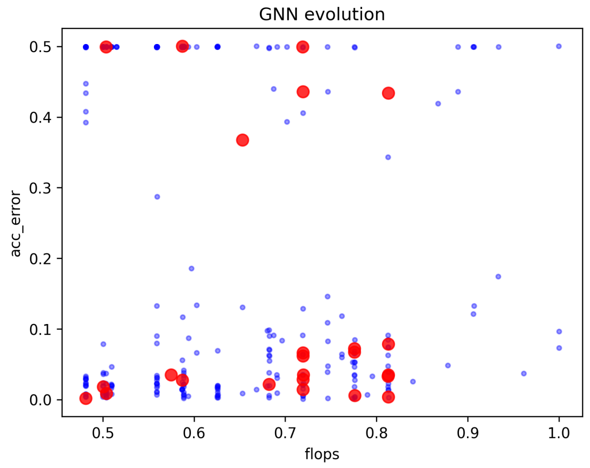

Fig. 7. GNN revolution result.

is cumbersome. The loss function of the network is defined as the CrossEntropy loss of each bit corresponding to two adjacent frames of CAN message. In the network verification stage, the maximum loss value of all bits is used as the basis for whether it is an intrusion message. Public dataset, Car_Hacking_Challenge_Dataset_rev20Mar2021, is used to training and verifying the neural network. Use the normal message part to train the LSTM network, and use the data mixed with 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. [40] also uses the 64 bits data segment of CAN message to directly input the bit stream to DNN network. The advantage of this network is that it is more efficient and can be directly input to the network without data processing. The specific structure of the network is not pointed out in the paper. We adopt five layers of fully connection. The first layer has 64 neurons to receive 64 bits CAN message 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 data sets, the accuracy is low, but the neural network is simple and does not need special construction data.1 [11] are the same

TABLE IV
CNN SEARCH SPACE

Deep learning method	accuracy	complexity
LSTM	97.02%	0.057M
DNN	95.36%	6.8M
CNN	85.06%	2628.7M
GNN	96.43%	54.3M
our method without RL	95.17%	2673.1M
our method with RL	99.87%	2673.1M

with our network convolution part. I directly use our network framework. Using gene "44440444" can directly construct the same architecture as in the paper. The data set in is the data collected by the author himself. The classification accuracy of the public data set does not reach the accuracy of the author's own experiment, which is lower than that of the convolution network we evolved by MOEA. 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. As shown in the table, it is 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. GNN module is also a part of the neural network architecture proposed by us. 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.

V. Conclusion

We use multiple network composition to ensure the accuracy of intrusion detection and minimize the complexity of the network. As far as we know, graph network is used for can intrusion detection for the first time in this work, and only graph network detection results can reach more than 95%. At the same time, we combine the dual advantages of graph network and convolution network. Through the output results of the two networks, the detection results of the two networks are complementary. At the same time, the logical and spatial characteristics of can data are us

References

[1] G. Leen and D. Heffernan, "Expanding automotive electronic systems," *Computer*, vol. 35, no. 1, pp. 88–93, 2002.

[2] S. Woo, H. J. Jo, and D. H. Lee, "A practical wireless attack on the connected car and security protocol for in-vehicle can," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 993–1006, 2015.

[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)*, 2018, pp. 1–7.

[4] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *2010 IEEE Symposium on Security and Privacy*, 2010, pp. 447–462.

[5] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *20th USENIX Security Symposium (USENIX Security 11)*. San Francisco, CA: USENIX Association, Aug. 2011. [Online]. Available: https://www.usenix.org/conference/usenix-security-11/comprehensive-experimental-analyses-automotive-attack-surfaces

[6] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *Def Con*, vol. 21, no. 260-264, pp. 15–31, 2013.

[7] O. Avatefipour and H. Malik, "State-of-the-art survey on in-vehicle network communication (can-bus) security and vulnerabilities," 2018. [Online]. Available: https://arxiv.org/abs/1802.01725

[8] A. Tashiro, H. Muraoka, S. Araki, K. Kakizaki, and S. Uehara, "A secure protocol consisting of two different security-level message authentications over can," in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, 2017, pp. 1520–1524.

[9] N. Nowdehi, A. Lautenbach, and T. Olovsson, "In-vehicle can message authentication: An evaluation based on industrial criteria," in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, 2017, pp. 1–7.

[10] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "Voltageids: Low-level communication characteristics for automotive intrusion detection system," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 2114–2129, 2018.

[11] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, p. 100198, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209619302451

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

[13] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," 2020. [Online]. Available: https://arxiv.org/abs/2006.02903

[14] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-bench-101: Towards reproducible neural architecture search," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 7105–7114. [Online]. Available: https://proceedings.mlr.press/v97/ying19a.html

[15] S. Yanan, X. Bing, and Z. Mengjie, "Yen gary g," in *An experimental study on hyper-parameter optimization for stacked auto-encoders. In 2018 IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 1–8.

[16] F. Ye, C. Doerr, and T. Bäck, "Interpolating local and global search by controlling the variance of standard bit mutation," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 2292–2299.

[17] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 4780–4789, Jul. 2019. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4405

[18] C. Wang, C. Xu, X. Yao, and D. Tao, "Evolutionary generative adversarial networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 6, pp. 921–934, 2019.

[19] S. Hamdioui, S. Kvatinsky, G. Cauwenberghs, L. Xie, N. Wald, S. Joshi, H. M. Elsayed, H. Corporaal, and K. Bertels, "Memristor for computing: Myth or reality?" in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017, pp. 722–731.

[20] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," 2018. [Online]. Available: https://arxiv.org/abs/1804.09081

[21] C. Tang, Y. Wang, Z. Dong, G. Hu, Z. Wang, M. Wang, and H. Chen, *XIndex: A Scalable Learned Index for Multicore Data Storage*. New York, NY, USA: Association for Computing Machinery, 2020, p. 308–320. [Online]. Available: https://doi.org/10.1145/3332466.3374547

[22] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing cnn architectures using the genetic algorithm for image classification," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3840–3854, 2020.

[23] L. Xie and A. Yuille, "Genetic cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[24] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W.

Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 2902–2911. [Online]. Available: https://proceedings.mlr.press/v70/real17a.html

[25] Z. Lu, K. Deb, E. Goodman, W. Banzhaf, and V. N. Boddeti, "Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 35–51.

[26] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," 2017. [Online]. Available: https://arxiv.org/abs/1705.10823

[27] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[28] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," 2019. [Online]. Available: https://arxiv.org/abs/1908.09791

[29] X. Dai, P. Zhang, B. Wu, H. Yin, F. Sun, Y. Wang, M. Dukhan, Y. Hu, Y. Wu, Y. Jia, P. Vajda, M. Uyttendaele, and N. K. Jha, "Chamnet: Towards efficient network design through platform-aware model adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[30] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 1587–1596. [Online]. Available: https://proceedings.mlr.press/v80/fujimoto18a.html

[31] H. Kang, B. I. Kwak, Y. H. Lee, H. Lee, H. Lee, and H. K. Kim, "Car hacking: Attack & defense challenge 2020 dataset," 2021. [Online]. Available: https://dx.doi.org/10.21227/qvr7-n418

[32] Y. Ma, S. Wang, C. C. Aggarwal, and J. Tang, "Graph convolutional networks with eigenpooling," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery &amp; Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 723–731. [Online]. Available: https://doi.org/10.1145/3292500.3330982

[33] J. Yu and T. S. Huang, "Universally slimmable networks and improved training techniques," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[34] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016. [Online]. Available: https://arxiv.org/abs/1611.01578

[35] X. Jin, J. Wang, J. Slocum, M.-H. Yang, S. Dai, S. Yan, and J. Feng, "Rc-darts: Resource constrained differentiable architecture search," 2019. [Online]. Available: https://arxiv.org/abs/1912.12814

[36] X. Zheng, R. Ji, L. Tang, B. Zhang, J. Liu, and Q. Tian, "Multinomial distribution learning for effective neural architecture search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[38] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[39] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[40] M.-J. Kang and J.-W. Kang, "A novel intrusion detection method using deep neural network for in-vehicle network security," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, 2016, pp. 1–5.