electronics

MDPI

Article

DRnet: Dynamic Retraining for Malicious Traffic Small-Sample Incremental Learning

Ruonan Wang, Jinlong Fei *, Rongkai Zhang, Maohua Guo, Zan Qi and Xue Li

State Key Laboratory of Mathematical Engineering and Advanced Computing, PLA Strategic Support Force Information Engineering University, Zhengzhou 450001, China
* Correspondence: feijinlong@126.com

Abstract: Deep learning has achieved good classification results in the field of traffic classification in recent years due to its good feature representation ability. However, the existing traffic classification technology cannot meet the requirements for the incremental learning of tasks in online scenarios. In addition, due to the high concealment and fast update speed of malicious traffic, the number of labeled samples that can be captured is scarce, and small samples cannot drive neural network training, resulting in poor performance of the classification model. Therefore, this paper proposes an incremental learning method for small-sample malicious traffic classification. The method uses the pruning strategy to find the redundant network structure and dynamically allocates redundant neurons for training based on the proposed measurement method according to the difficulty of the new class. This enables the network to perform incremental learning without excessively consuming storage and computing resources, and reasonable allocation improves the classification accuracy of new classes. At the same time, through the knowledge transfer method, the model can reduce the catastrophic forgetting of the old class, relieve the pressure of training large parameters with small-sample data, and improve the model classification performance. Experiments involving multiple datasets and settings show that our method is superior to the established baseline in terms of classification accuracy, consuming 50% less memory.

Keywords: malicious traffic classification; small samples; incremental learning; dynamic retraining

Citation: Wang, R.; Fei, J.; Zhang, R.; Guo, M.; Qi, Z.; Li, X. DRnet: Dynamic Retraining for Malicious Traffic Small-Sample Incremental Learning. *Electronics* **2023**, *12*, 2668. https://doi.org/10.3390/electronics12122668

Academic Editors: Alessandra De Benedictis and Salvatore Barone

Received: 8 May 2023
Revised: 5 June 2023
Accepted: 12 June 2023
Published: 14 June 2023

1. Introduction

The arrival of the era of big data has been accompanied by an exponential growth of network traffic, and malicious traffic generated by malicious programs is also endless. Network traffic classification can associate traffic with its generation program. In the field of network security, traffic classification is the first step in the task of network malicious resource detection [1]. Therefore, the accurate classification of network traffic has always been a hot topic in the field. In recent years, deep learning has achieved great success in the field of traffic classification [2–4].

However, deep learning relies on the supervised training of large labeled datasets. Due to the high concealment of malicious traffic and changing attack behavior, the amount of malicious traffic data that can be captured and accurately marked is small, and it is difficult to provide the amount of data that can drive deep learning training. In addition, the current malicious traffic classification model only focuses on the model effect in the offline case and pays less attention to the model performance on the online device. When the deep learning model is deployed on the online device, the constantly updated and changing malicious traffic class requires the classification model to have the ability to add classification tasks. In the classification model based on deep learning, the classifier is often supervised by a set of data. If we want to add classification tasks to the classifier, we must use a large amount of old class data and new class data to jointly retrain the classifier. If the old class data cannot be re-enabled, using only the new class data will cause the

catastrophic forgetting of the classifier, meaning that the classification performance of the classifier for the old class is greatly reduced.

When the network completes training and is deployed in a practical application scenario, it usually cannot carry the training data of the network, which results in the additional consumption of storage resources and is not conducive to data privacy protection. Although, in order to facilitate the test, the small number of old class sample categories used in this paper makes the computing resources occupied by the old class small-sample data seem small. However, in reality, there are many kinds of Trojan horses. The storage resources for the old class small samples and the computing resources required for the retraining of the old class still overwhelm the online equipment. At the same time, malicious traffic is constantly iteratively changing, which is necessary for the update of the classifier. Therefore, an incremental learning method is needed to solve the update of the classifier without combining the old data. In addition, as an important defense line for network security, malicious traffic classification is often deployed on edge computing devices, such as robots, smart devices, and other products. These products are not rich in storage and computing resources due to their light and flexible characteristics. Therefore, the question of how to solve the scalability problem with respect to a small-sample malicious traffic classification model in the case of scarce computing resources is worth exploring in the field of malicious traffic classification.

The ability to allow the model to extend new classification tasks and learn from new class data without forgetting the old class is referred to as incremental learning ability. In recent years, researchers have proposed many incremental learning methods. This includes the method of playback based on old data [5–7], which alleviates the problem of catastrophic forgetting by jointly guiding the classifier with old data and new data. However, this method requires storing old data, and the old data should be representative and equal to the number of new data as, if not, it will affect the classification performance of the classifier for the old class. Therefore, the playback method requires more storage resources and is prone to category imbalance, which makes the classifier overfit the old or new classes and leads to low overall classification performance. Regularization-based methods [8,9] use knowledge distillation as a regularization term to constrain the model, penalizing the model's forgetting of old classes when fitting new data. This process requires storing the weight of the old model to obtain the source of knowledge distillation. However, there is an antagonistic relationship between knowledge distillation and incremental learning. If the model constraint is too strong, the model will seriously fit the current data, resulting in poor robustness. If the model constraint is weak, it cannot effectively alleviate the problem of catastrophic forgetting. Therefore, the current research on regularization methods should also make efforts to design better distillation methods. The method based on model expansion [10,11] aims to adapt to new data by adding a new model structure and keeping the weight of the old model unchanged for the old class so as to avoid forgetting. However, the increase in model structure will inevitably lead to the aggravation of computing resources and computational burden. When the number of new tasks increases gradually, the model will become overwhelmed.

In summary, the incremental learning problem of small-sample malicious traffic classification currently faces the following three major challenges, and no method in the existing literature can fully solve these challenges:

Q1: Learning new tasks brings the catastrophic forgetting of old tasks. Due to the limited computing and storage resources of online devices, when a new task arrives, there is often a performance tradeoff between the old and new classes. The adaptation of new tasks will inevitably lead to a decrease in the accuracy of old tasks. Improving the overall performance of the model is the primary challenge that incremental learning must solve.

Q2: Small-sample data prove difficult in driving neural network training. Due to the hidden and changeable characteristics of malicious traffic, in the real-world, it is often faced with the contradiction between the amount of data that can be captured and the

large parameters of neural networks. Therefore, the performance of the small-sample classification model is the focus of this paper.

Q3: The current incremental learning method has a large demand for computing resources. Mainstream incremental learning methods—playback-based methods, regularization-based methods, and model expansion-based methods—all require computing resources to varying degrees. Therefore, maintaining the performance of the classifier while minimizing the demand for computing resources is one of the challenges that needs to be solved.

Compared with neural networks, humans are often able to remember old tasks in the face of new tasks in the process of learning. Researchers have found [12] that this is because the adult brain contains a large number of 'silent synapses,' and the connections between these neurons remain inactive before forming new memories; therefore, the old memories will not be forgotten when forming new memories. Inspired by the 'silent synapse' in the human brain, as shown in Figure 1, in this paper, we propose a dynamic retraining method (DRnet). The method pruned the trained neural network so that it can form 'silent neurons' for new tasks while reducing the redundant structure of the network. When a new task arrives, we dynamically allocate 'silent neurons' according to the similarity between the old and new tasks, retrain these neurons, and update the classifier so that the network can adapt to the new task. At the same time, the neural network also has some inactive 'silent neurons' waiting for the next new task. For small-sample incremental tasks, we use the idea of transfer learning to freeze the number of network layers of common features when extracting new and old task features, reducing the parameters that the network needs to adjust to adapt to the small-sample data volume.

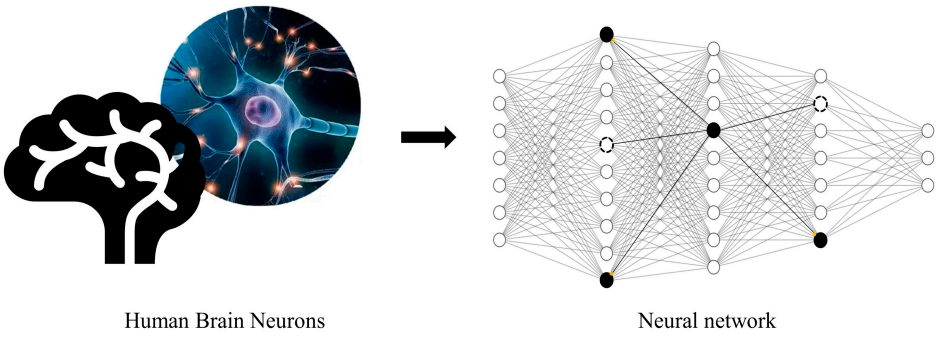

Figure 1. Neural network neurons are analogous to silent synapses.

Our contributions can be summarized as follows:

- We design an incremental learning method for malicious traffic small-sample classification, which alleviates the catastrophic forgetting of old tasks when training new tasks and improves classification accuracy for small-sample incremental tasks.
- An improved transfer learning method is applied to the process of class increment, which alleviates the contradiction between the new tasks of small samples and the large number of parameters to be adjusted in the model.
- The proposed incremental learning method applies a dynamic retraining redundant neuron strategy, which can effectively use redundant neurons to learn new classes while making the model structure more lightweight, saving computing resources.

The remainder of this paper is structured as follows: Section 2 reviews related works in the literature. Section 3 introduces the DRnet method. Section 4 introduces the experimental dataset and data preprocessing method and presents an analysis of the experimental results. Section 5 discusses the effectiveness of the study, its limitations, and future work to be addressed regarding the method. Finally, in Section 6, the work of this paper is summarized, and conclusions are drawn.

2. Related Works

Research on incremental learning in the traffic domain: Prasath et al. [13] extended the incremental learning method in the image field to general network security and specific intrusion detection systems for the catastrophic forgetting problem in intrusion detection. This work compares the performance of various methods in simulating real-world distributed alternating batch attacks and concludes that the playback-based method is superior to traditional statistical techniques and the most advanced Boosting model and DNN model in dealing with catastrophic forgetting. Based on the idea of transfer learning, Doshi et al. [8] designed a feature extraction module and a decision module to solve the task of incremental continuous learning. The feature extraction module is used to minimize the complexity of the training and extract motion, position, and appearance features. The decision module is a sequential anomaly detector, which quickly updates the learned model using incremental labels. Through experiments on public datasets, the detector trained by this method is significantly better than the most advanced algorithms. Amalapuram et al. [9] evaluated the challenges and practicality of incremental learning in the design of intrusion detection systems. This work found that class incremental methods have a greater impact on task order sensitivity. At the same time, by applying the current popular incremental learning algorithms—Elastic Weight Combination (EWC) [14] and Gradient Embedding Memory (GEM) [15]—it was found that the performance could be further improved by combining the memory population technology based on the perception of empirical forgetting. Pezze et al. [16] proposed an incremental learning method based on compressed playback. The method uses a super-resolution model to compress and playback the original image and combines the anomaly detection module to enable the model to perform anomaly detection tasks in an environment that constantly learns new tasks. This method represents the first time that the compression module has been applied to incremental learning tasks. A large number of experiments have proved that the method can use fewer computing resources to complete incremental tasks.

Research on small-sample incremental learning: At present, there are few studies on small-sample incremental learning in the field of traffic, so we have some related working methods proposed in other fields to use them as references. Aiming at the demand for fine-grained multi-tasks of classifiers in the field of image classification, Mallya et al. [17] proposed a method that adds multiple tasks to a single deep neural network while avoiding catastrophic forgetting. This work uses redundancy in large deep networks to release parameters and is then used to learn new tasks. By performing iterative pruning and network retraining, this work can sequentially 'package' multiple tasks into a single network while ensuring minimum performance degradation and minimum storage overhead. However, in the process of pruning and retraining, the complexity of the new task is not considered, which greatly affects the classification performance of the model by the order of the arrival of the new task, and there is no effective solution to the small-sample problem. The work of this paper is based on the work of Mallya et al. In the field of object detection, Kang et al. [18] developed a small-sample object detector. This work uses a meta-feature learner, a reweighted module in the first-level detection architecture, and a fully labeled base class and quickly adapts to new classes. Through a large number of experiments, this work proves that the model's few-shot object detection task and class increment task on multiple datasets are much better than the established baseline. Douillard et al. [19] introduced a new distillation loss that constrains the entire convolutional network using the idea of knowledge distillation to solve small-sample incremental learning. This loss balances the forgetting of old classes and the learning of new classes, which is conducive to long-term incremental learning. At the same time, the method proposes a multi-mode similar classifier, which makes the model more robust to the inherent distribution displacement of incremental learning. This method has achieved good experimental results on multiple datasets. Tao et al. [20] used neural gas (NG) networks to represent knowledge for small-sample incremental learning problems and proposed a neural science-inspired topology protection framework for effective class incremental learning. At the same time, a

topology preserving loss (TPL) was designed to maintain the topology of the feature space and reduce forgetting. Through testing on multiple datasets, it was proven that the method could continuously learn new classes and reduce forgetting.

3. Methods

Problem definition: Incremental learning is divided into multi-task incremental learning [21,22] and single-task incremental learning [23,24]. The difference between the two types is that the classifiers assigned to each type of task are different. For multi-task learning, the same model needs to gradually learn many isolated tasks without forgetting how to solve the previous tasks. This means that multi-task learning assigns each type of task to different classifiers. There is no class overlap between different tasks, and the accuracy of each task is calculated separately. Therefore, multi-task learning is very suitable for studying the feasibility of training a single model on a series of disjointed tasks without forgetting how to solve the previous tasks, but it is not suitable for solving class increment problems. For single-task incremental learning, a unified classifier is used and the entire incremental learning process is considered as an overall task, i.e., we add new classes in order, but the classification problem is unique. In the calculation accuracy, we need to distinguish all the classes encountered so far. This article aims to study the incremental problem regarding malicious traffic classification tasks under small-sample conditions. For newly added malicious traffic categories, it is hoped that the classifier can accurately classify them while not forgetting the old ones. The goal of this study is to help solve the single-task incremental learning problem, so this article focuses on single-task learning.

We define the small-sample single-task incremental learning problem as follows: Suppose we have a labeled small-sample target dataset D_T^j, the new small-sample task data are represented by D_E^k, L is the class set of training set, S is the class set of t test set. $\forall_{j,k}$, $L_j \cap L_k = \varnothing$, $S_j \cap S_k = \varnothing$. The model is first trained and tested on D_T^j. After the model can fit the target classification task, D_E^k arrives as a new task, the test set of the model is $S_j + S_k$, and the classifiers need to identify $j + k$ class tasks. At this time, the D_T^j dataset is not allowed for joint training. The process of single-task small-sample incremental learning is shown in Figure 2. In the offline training phase, the base class classifier is trained using the base class data D_T^j. When deployed online, only the new class data D_E^k can be called for network training. At this time, a certain incremental learning method is used to train the classifier to identify all categories $j + k$ of the base class and the new class. Complete the malicious traffic classifier incremental task. The challenges that the task needs to solve are as follows: avoiding the catastrophic forgetting of old tasks caused by new tasks, solving the problem of small-sample task classification performance, and reducing the computing resources required to complete the task.

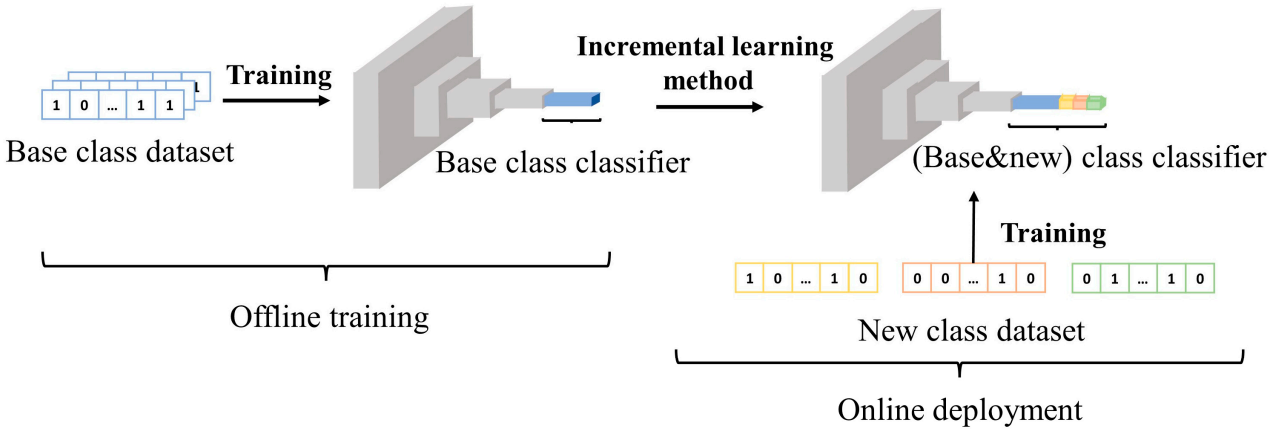

Figure 2. Small-sample single-task incremental learning flow chart.

In order to solve the above challenges, we propose a small-sample malicious traffic classification incremental learning method, the process for which is shown in Figure 3. We designed a fully convolutional network as the source network. In order to maintain the good feature extraction ability of the source network, we used a large dataset to train it and transfer the knowledge learned to the target dataset and new tasks through transfer learning to reduce the number of parameters that needed to be fine-tuned for small-sample data. At this time, the network has a large number of redundant neurons for small-sample classification tasks, so we pruned the network and used the pruned neurons as the reserve network structure of the new task. When performing new tasks, we dynamically allocated new tasks according to the category similarity between new tasks and old tasks and used pruned neurons for training new tasks. We introduce the detailed methods of each stage as follows:

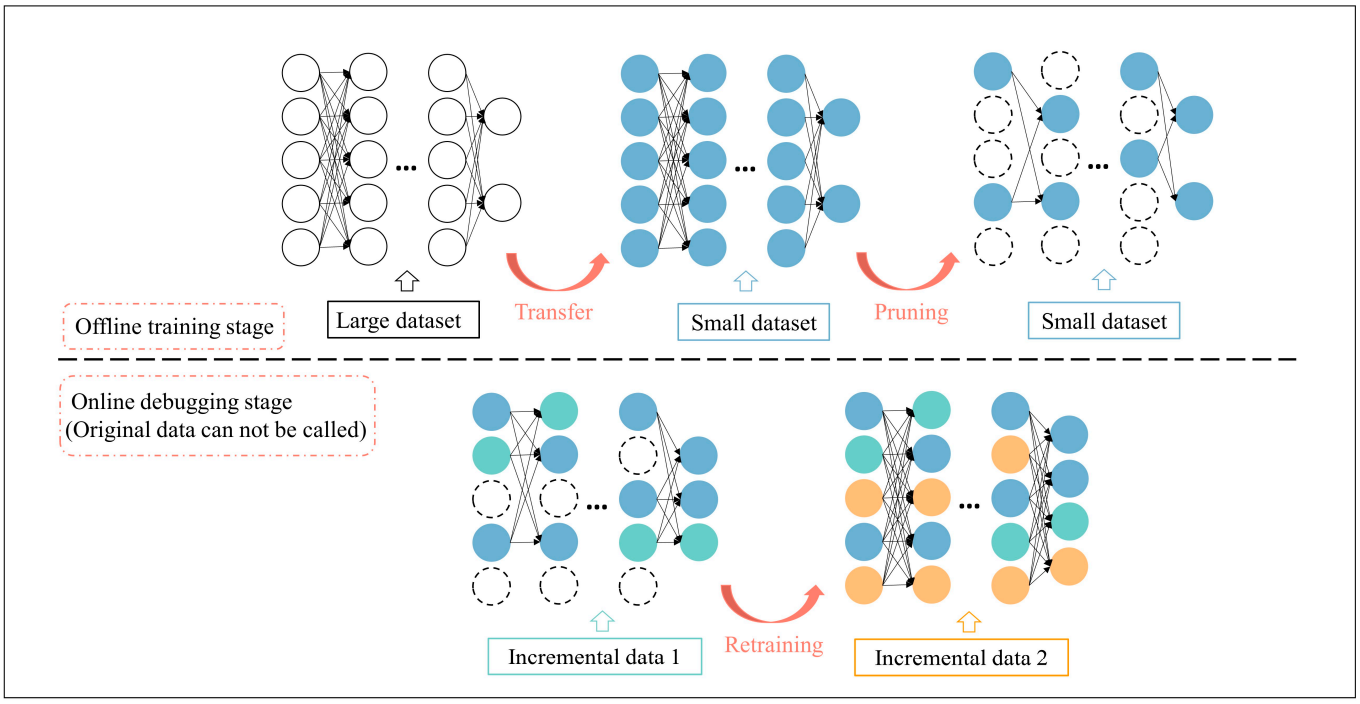

Figure 3. DRnet method basic concept diagram.

The main flow chart of the method is shown in Figure 4. After the original malicious traffic data packet was preprocessed, the model was trained in the form of data packets, and the format was batchsize $\times$ 1500. In the model training phase, the large dataset trains the original model, and the training dataset size was m $\times$ 1500. Then, the model was migrated, the parameters of the first four convolutional neural networks of the model were frozen, and the small-sample dataset of j $\times$ 1500 was used for fine-tuning. In the model pruning stage, the redundant neuron parameters in the model were set to 0. At this time, the number of convolutional layer neurons is the minimum remaining number x when the accuracy drop is less than 0.1, and the small-sample dataset is used to fine-tune the remaining parameters. In the final class increment stage, the similarity between the incremental data and the base class data was first calculated to determine the number of neurons y used to train the incremental data. After changing the model structure, the original model parameters were frozen, and the new dataset with a size of k $\times$ 1500 was used to train the assigned neurons y, achieving the goal of class increment.

Source model training phase: We designed an end-to-end fully convolutional malicious traffic classification network using a 1D Convolutional Neural Network (1DCNN) as the initial network. The network consists of ten one-dimensional convolutional layers and seven maximum pooling layers. The purpose of using the fully convolutional network is because the fully convolutional network abandons the traditional fully connected layer and

replaces it with a convolutional layer, which saves a lot of network parameters. Secondly, the fully connected layer structure is difficult to change, and this is not conducive to our maximum pruning operation later. The weight sharing between the convolutional layers and the sparse connection provides convenience for subsequent pruning and class increment. After each convolutional layer, we added a Batch Normalization (BN) layer. In addition to the traditional prevention of overfitting and acceleration of model convergence, the BN layer is also used to evaluate the filter contribution of subsequent pruning. We set the number of neurons in the convolutional layer to 300, added the Rectified Linear Unit (RELU) function after the BN layer, and took a 0.05 dropout rate after each fully connected layer to prevent overfitting. The detailed hyperparameters of the network structure settings are shown in Table 1. We used large datasets to train the initial network to train its good feature extraction ability (obtaining a good initial parameter set is crucial for subsequent transfer and pruning of the network). We set 30 epochs for the initial network. The learning rate was 0.01.

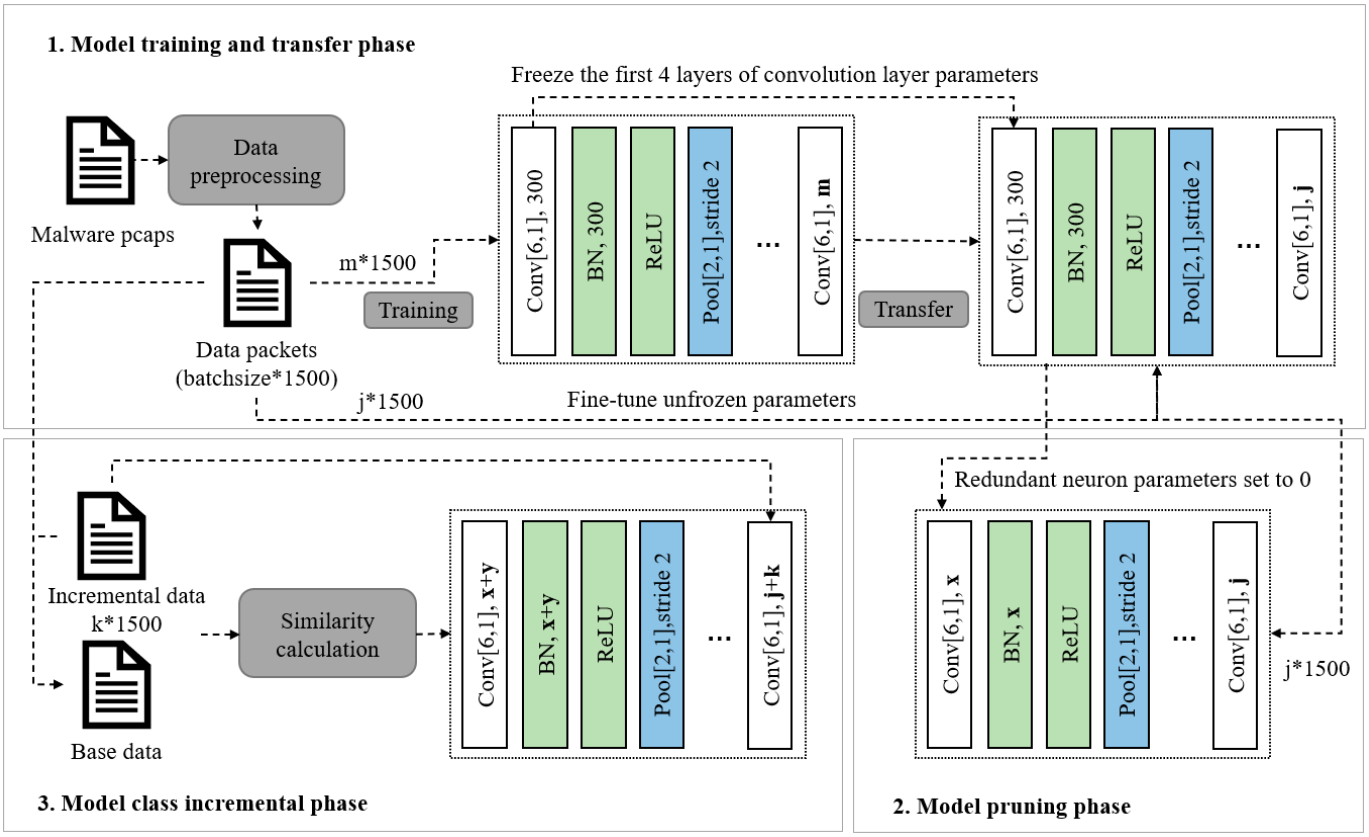

Figure 4. DRnet method detailed flow chart.

The selection of hyperparameters in network construction and training is based on the empirical conclusions of researchers and the actual situation of the problems solved in this paper. Firstly, the network convolution layer was set to 10 layers, and the maximum pooling layer was set to 7 layers. This is because the small sample size cannot support neural network training that is too complex. The number of neurons was set to 300, because the subsequent increment was based on the retraining of redundant neurons in the current network. Therefore, more neurons were set than the traditional classical traffic classification network. In addition, we referred to the empirical parameter settings of the current network, such as the initial learning rate (set to 0.01), the convolution kernel size (set to 6), and the Dropout rate (0.05).

Table 1. Detailed parameters of source network.

Operation	Kernel Size	Strides	Channels	Padding	Dropout	Nonlinearity
Input packet	-	-	-	-	-	-
Convolution	6	1	300	1	0.05	BN+ReLU
Max pooling	2	2	-	0	0.05	-
Convolution	6	1	300	1	0.05	BN+ReLU
Max pooling	2	2	-	0	0.05	-
Convolution	6	1	300	1	0.05	BN+ReLU
Max pooling	2	2	-	0	0.05	-
Convolution	6	1	300	1	0.05	BN+ReLU
Max pooling	2	2	-	0	0.05	-
Convolution	6	1	300	1	0.05	BN+ReLU
Max pooling	2	2	-	0	0.05	-
Convolution	6	1	300	1	0.05	BN+ReLU
Max pooling	2	2	-	0	0.05	-
Convolution	6	1	300	1	0.05	BN+ReLU
Convolution	6	1	300	1	0.05	BN+ReLU
Max pooling	2	2	-	0	0.05	-
Convolution	6	1	100	1	0.05	BN+ReLU
Convolution	6	1	Class number	1	0.05	BN+ReLU

Transfer phase: In order to solve the small sample problem of malicious traffic, we used an improved transfer learning method. In this study, two types of small-sample data were involved: small-sample target dataset and new task small-sample data, so we actually applied transfer learning in two parts. When training the classification network for the target dataset, because the small-sample dataset is not enough to train the large neural network, based on the trained initial network, we froze the first four-layer parameters of the network and fine-tuned the remaining parameters. Because the shallow network of the model often extracts low-level features, this operation did not affect the performance of the fine-tuned classifier. However, this can greatly reduce the number of parameters that need to be adjusted to improve the performance of the classification network. When faced with new tasks, we hoped that the model could train new tasks while maintaining the classification ability of old tasks. Therefore, we maintained the network weights of all old tasks during training and used its feature extraction ability for old tasks to participate in retraining with neurons assigned to new tasks. In this process, the old dataset does not participate in training but transfers its acquired knowledge. At the same time, the weight of the pruned redundant parameters is not set to 0, but the original parameters are maintained. This method ensures that the newly added small-sample data requires less adjustment when training the assigned redundant parameters so that the small-sample data can better fit the network parameters.

Pruning phase: In this study, the purpose of our pruning was not only to lighten the network in the traditional sense but also to screen out the redundant network structure that can be used by new task training. Therefore, we hoped to eliminate as many redundant neurons as possible without affecting the classification accuracy of the current task. Unlike Mallya et al. [17], we only performed network pruning once, which greatly saved the pruning overhead. Our approach used the scaling factor of the BN layer as an indicator to rank neurons to determine their importance. We reduced the accuracy of the classifier by less than 1% as a criterion that does not affect the classification performance of the classifier. We represented this part of the neuron as R. According to experience, more than 80% of the neurons are identified as redundant neurons. We ranked these neurons according to weight and size so that they could be assigned to new tasks. At this time, we ensured that the R neurons did not participate in the back propagation and used a small-sample dataset to retrain the remaining neurons. This is due to a change in the network structure caused by the pruning operation, which needs to reestablish the connection between neurons. After

the pruning phase is complete, the network will be deployed on the online network device, and the old data will not be called again.

Incremental phase: Mallya et al. selected 50% or 75% of the redundant parameters of the current task as the new task training for the class incremental task. Because the redundant parameters are constantly occupied, the first-arriving parameters can be assigned to more neurons for training. Therefore, this method is only effective when the difficulty of the new task is difficult. However, in the real world, the task will not appear in an orderly manner. We propose to dynamically match the number of parameters required for the task according to the difficulty of the task. In order to evaluate the difficulty of the new task, we introduced the Maximum mean discrepancy (MMD) distance [25]. MMD is a nonparametric measure used to measure the distance between distributions based on kernel embedding in the reproducing kernel Hilbert space. Suppose $\varphi(x)$ maps each instance to the kernel $k(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$ related Hilbert space $\wp$, n_s and n_e are the sample size of small-sample datasets and new datasets. Then, the domain samples X_s and Y_E of two distributions, whose MMD distance is defined as follows:

$$MMD(X_S, Y_E) = \left\| \frac{1}{n_s} \sum_{i=1}^{n_s} \varphi(x_i^s) - \frac{1}{n_E} \sum_{i=1}^{n_E} \varphi(y_i^E) \right\|_{\wp} \tag{1}$$

By introducing the MMD distance, we can obtain the position and distance of the two datasets in the Hilbert space. In our opinion, the closer the distance between the two domains is, the higher the similarity is. Therefore, we take the distance from the representative sample to the center position in X_s and the ratio of the distance from the two datasets as the similarity. In order to alleviate the influence of the deviation sample on the similarity, we select the median of the nearest and farthest distance from the center position in X_s as the distance of the representative sample. The similarity measure method we designed is defined as follows:

$$W(X_S, Y_E) = \frac{\| Average[(\max\varphi(x_i^s), \min\varphi(x_i^s)] \|_{\wp}}{MMD(X_S, Y_E)} \tag{2}$$

After obtaining the similarity of the network, we assigned the number of neurons required for the current new task according to the similarity, which is expressed as:

$$R_E = R * [1 - W(X_S, Y_E)] \tag{3}$$

When the network adds new tasks, the neurons in the last convolutional layer will be re-adjusted to a new number of classifications. At the same time, the new tasks need to be trained for the assigned parameters. Therefore, we use the new sample data to retrain the network. For the neurons of the old task, we kept their weights unchanged. At this time, only a small number of neurons participate in training. We set the epoch to 10 and the learning rate to 0.01.

Then, whenever the network has new task requirements, the operation of dynamic allocation and retraining will be re-performed to adapt to the new task. At this point, the network will be able to dynamically allocate and adjust the network neurons without affecting the old task to complete the goal of classifying new tasks to adapt to the changing real world. Because the network does not increase the structure compared with the source network and the old data does not need to be stored, the consumption of computing resources is small.

4. Results

In order to evaluate the effectiveness of our method in small-sample incremental tasks, we simulate the performance of the model facing new tasks after the model is deployed online, that is, when the old data can no longer be called. We test our method on multiple datasets and compare it with other class incremental methods in terms of accuracy and

computational resource consumption. At the same time, we also show our method's dynamic allocation of neurons in the class incremental process. The results and analysis of the method are as follows.

4.1. Datasets and Preprocessing

4.1.1. Datasets

In this study, four datasets were used to train and test the model: USTC-TFC, ISCX VPN-nonVPN, MCFP, and a self-made dataset.

USTC-TFC [26]: This dataset includes pcap packages of malware traffic collected from real network environments by Colorado Technical University (CTU) researchers from 2011 to 2015. We used it as a source dataset to train the source network to obtain good feature extraction ability.

ISCX VPN-nonVPN [27]: This dataset is composed of traffic generated by captured different applications. The captured packets are divided into different pcap files, and we used some classes of data. Different from the other three datasets, the traffic of the dataset is ordinary traffic rather than malicious traffic; therefore, the dataset was used as a small-sample incremental task test method from different domains on inter-domain small-sample tasks.

MCFP: MCFP is a malicious traffic dataset captured by the Czech University of Technology that includes network files, logs, DNS requests, etc., in the pcap format. We used ten types of malicious traffic data as small-sample malicious traffic classification targets and incremental tasks.

Self-made dataset: In order to restore the performance of the model in the real scene, we simulated the malicious behavior of 15 different Trojans or malicious programs and used Wireshark to obtain the traffic data packets in the process and generate pcap packets as a self-made dataset test model. In order to approach the scale of small-sample datasets under realistic conditions, the self-made datasets use part of the captured network malicious traffic data. Table 2 presents the specific information and the number of packets of the fifteen types of Trojan traffic captured. The data preprocessing process of the self-made dataset pcap package is described in Section 4.1.2.

Table 2. Details of self-made dataset.

Self-Made Dataset	Type	Highest Visible Protocol	Size (K)	Quantity
Finalfantasy	Malware	SSH	3180	100
Hackratstyle	Malware	HTTP	955	100
Bagsu	Malware	HTTP	866	100
Conficker	Malware	HTTP	1701	100
MSIL	Malware	HTTP	2118	100
Murlo	Malware	HTTP	1516	100
Wootbot	Malware	HTTP	1523	100
Freerat	Malware	SSH	3114	100
Chongqinghack	Malware	TCP	1360	100
Irat	Malware	SSH	2343	100
Poison-ivy	Malware	SSH	2344	100
Greydove	Malware	HTTP	2257	100
Shangxing2009	Malware	TCP	2344	100
Suncontrol	Malware	TCP	2125	100
Ximencontrol	Malware	TCP	2051	100
Mean	Malware	-	2483	100

In order to approach the size of small-sample datasets under realistic conditions, each type of data in the three small-sample datasets uses only 100 pcap packages for training and testing. Table 3 presents the detailed information and scale of the datasets we use.

Table 3. Dataset Details.

Dataset	Type	Highest Visible Protocol	Class
USTC-TFC	Malware	HTTP&IRC	10
ISCX VPN-nonVPN	Normal	HTTPS&SSL	10
MCFP	Malware	HTTP&IRC&SSH	10
Self-made	Malware	HTTP&SSH&TCP	15

4.1.2. Data Pre-Processing

The preprocessing process realizes the filtering and unified format of network data, including traffic filtering, traffic cleaning, and data unification. The preprocessing process is shown in Figure 5. First, perform traffic filtering to delete packets that do not contain payloads from the captured traffic. In some TCP segments, the SYN, ACK, or FIN flags are set to 1. These flags represent traffic fragments generated during three handshakes. The information they contain is not sufficient to distinguish the information about the program that generated them. These fragments will become invalid samples for the obfuscation classifier, so they will be discarded. The traffic cleanup operation sorts out the information in the sample that is not valid for classification (including deleting the Ethernet header of the physical link information contained in the traffic data) and shields the IP address in a uniform replacement manner to prevent classifiers from attempting to classify data packets by using the IP address. This overfitting is not beneficial for training model capabilities. Since neural networks require the use of fixed size inputs, the preprocessing stage unifies the size of data packets. The process involves filling in zeros for byte vectors smaller than 1500 by injecting zeros at the end of the UDP segment header and making it equal to the length of the TCP header. In order to achieve better performance, a normalization operation is performed to divide all packet bytes by 255, which is the maximum value of a byte. Therefore, all input values are in the range of 0 to 1 to normalize the byte vector.

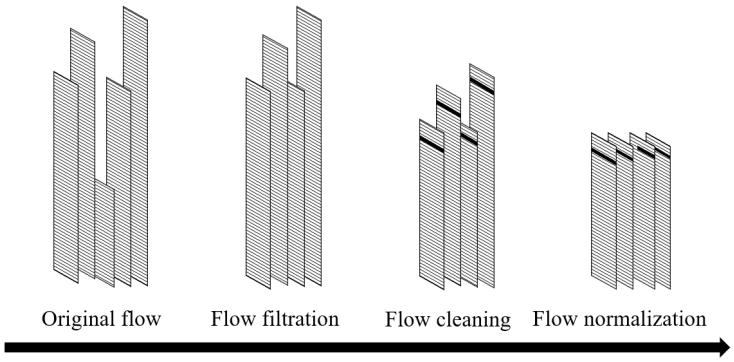

Figure 5. Data pre-processing process.

4.2. Method Performance Analysis

4.2.1. DRnet Method Incremental Learning Visualization

In order to intuitively show the improvement of our method on the catastrophic forgetting of old tasks when fitting new tasks, we introduced the t-SNE method [28] to visualize the distribution of data in the feature space. We used the eight-class data of the self-made dataset as the base class training model. The distribution of the eight-class data in the feature space upon completion of training is shown in Figure 6a. The different colors in Figure 6 represent different types of data in the test set. For cases where the classification boundary is more obvious, we used numbers to distinguish between eight types of data in order to more intuitively display the feature distribution of different types of data. Then, the class incremental task was performed on the trained model (the red stars in Figure 6b are class incremental task distribution). We first used the new class samples directly on the model for training. The distribution of the samples in the feature space when training 1 epoch and 20 epochs is shown in Figure 6b. The method proposed in this paper is used to

assign neurons to new classes and retrain them. The data distribution after training 1 epoch and 20 epochs is shown in Figure 6c.

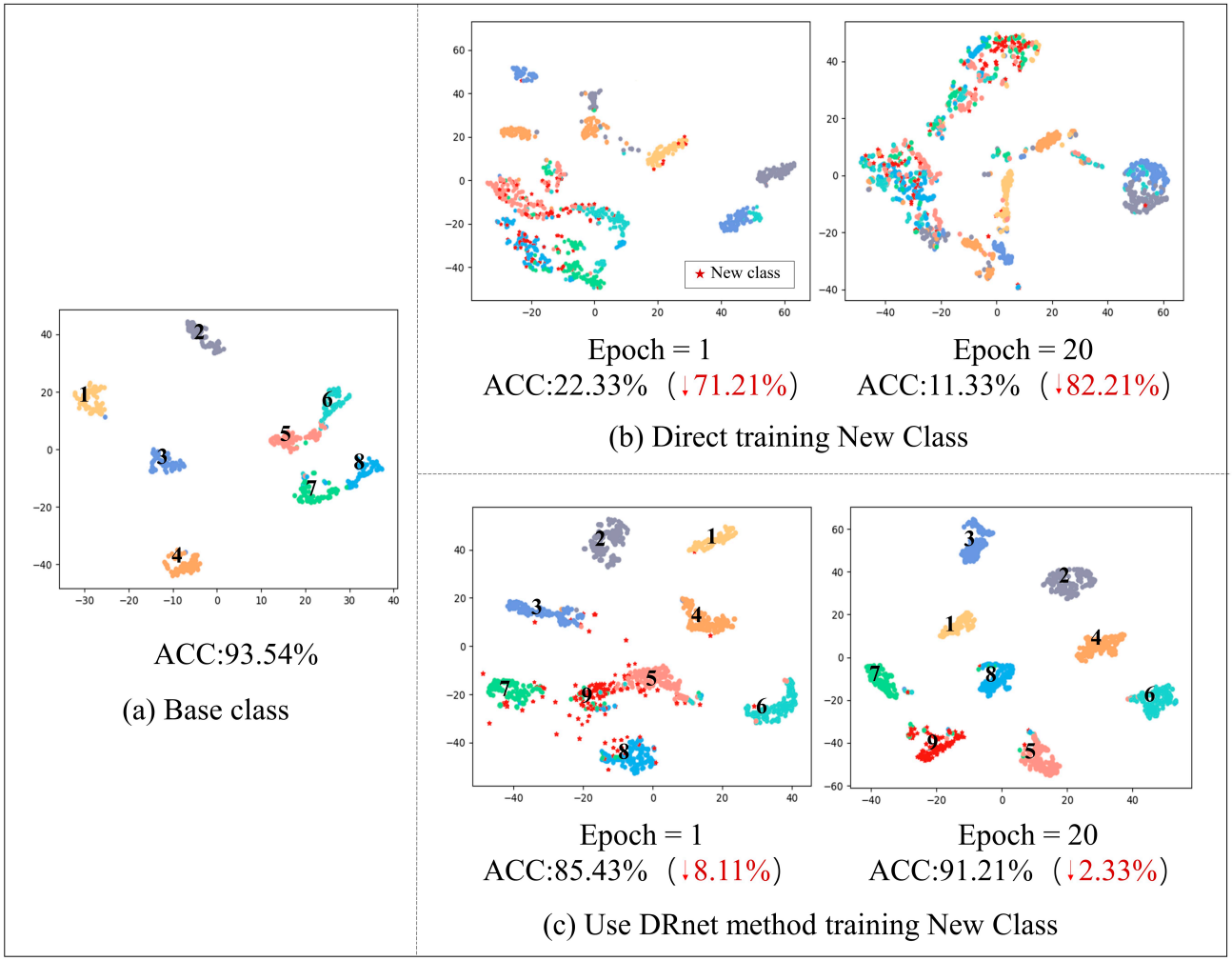

Figure 6. Class incremental task data visualization.

Through Figure 6, we can see that the base class samples can be classified well at first, and the classification accuracy rate reaches 93.54%. Subsequently, since the old data are not stored, we directly trained the new class. At this time, the model will fit the new task data as the target, so the model cannot maintain the feature space topology of the base class sample. We can see from Figure 6b that, at 1 epoch, the distribution of the base class sample in the feature space is confused with the new class. However, there are still a small number of base classes that can maintain their spatial topology, and the accuracy of the model is reduced to 22.33%. When the model was trained for 20 epochs, the base class was catastrophically forgotten, and because the new class was small-sample data, the model could not complete the task of fitting the new class, and the model accuracy was as low as 11.33%. When using the method proposed in this paper to train new class tasks, because we allocate additional neurons to train new tasks, it will not affect the weight of the trained base class. Therefore, at 1 epoch, the topology of the base class sample in the feature space can be maintained, the classification accuracy of the base class barely decreases, and the overall accuracy of the model is 85.43%. After training 20 epochs, the accuracy of the model is 91.21%, which completes the task of class increment well. Experiments show that our method can effectively alleviate the problem of catastrophic forgetting in class incremental tasks.

Answer to Q1: When performing class incremental learning, the DRnet method has an accuracy of 91.21%, which is only 2.33% lower than the base class accuracy of 93.54%. The method effectively alleviates the catastrophic forgetting problem.

4.2.2. Small-Sample Class Incremental Task Classification Performance

The work of this paper is based on the PackNet method. Therefore, in this section, we describe the performance of our proposed method and the PackNet method on different datasets under the same settings and models. At the same time, in order to show the impact of catastrophic forgetting on classification accuracy, we used new tasks to directly train the model for comparison. For the three types of settings, we first used the source dataset to pretrain our proposed model and the model transfer method to fit the target small-sample dataset. At this time, the target small-sample dataset is the base class data of different datasets. Subsequently, for the incoming new incremental tasks, we used three different methods of incremental learning settings for training, using the same training rounds and learning rates. For the PackNet method, in order to improve its accuracy as much as possible, we selected a 50% iterative pruning rate based on the original text to train the new task. After the training, the classification performance of the model for the base class and the new class is tested.

Incremental Task Classification Performance of Small-Sample Class in the Same Domain

We take incremental tasks and base class tasks belonging to the same dataset as incremental tasks in the same domain. Therefore, we randomly used five class data from three datasets, namely the preprocessed ISCX VPN non-VPN dataset, the MCFP dataset, and the self-made dataset, as base class tasks and added incremental tasks to the model one by one. Figure 7 shows the performance of our method, the PackNet method, and the direct training class increment method when the class increment task and the base class task belong to the same domain.

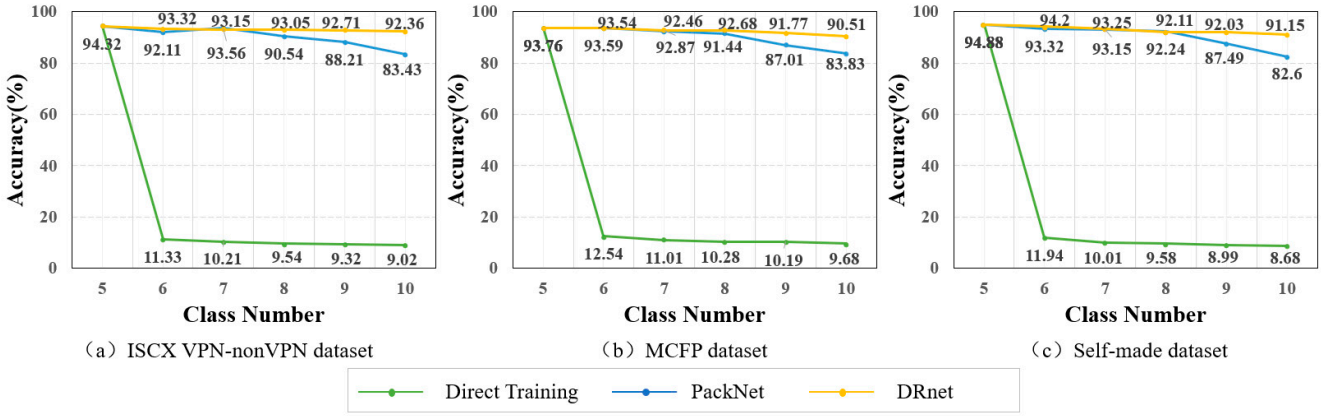

Figure 7. Comparison of accuracy of incremental task classification with small-sample classes in the same domain.

According to the line graphs of Figure 7, we can see that, with the incremental addition of new tasks, the direct training method cannot maintain the knowledge of the model for the old tasks. Therefore, when only one new class is added, serious catastrophic forgetting occurs, with an accuracy rate of only 11.33%. In the subsequent new tasks, catastrophic forgetting is more serious and, as a result, the accuracy rate continues to decline. The PackNet method allocates redundant parameters by iterative pruning, which makes the number of parameters that can be allocated continuously decrease. Therefore, after adding up to nine classes, the accuracy of the model decreases. When our method allocates neurons, it dynamically allocates neurons by similarity calculation, which is not related to the order of task arrival, but only related to the difficulty of the task. Therefore, the accuracy rate is always above 90%. It is worth noting that, in Figure 7a,c, when the number of classes is

six, the accuracy rate decreases. This is worthy of analysis because the small-sample data volume cannot drive the currently allocated parameter quantity. Our method performs two knowledge transfer processes for small-sample incremental tasks, namely, the transfer of large datasets to target datasets and the knowledge transfer of target datasets to new tasks, which alleviates the pressure of small-sample datasets on fine-tuning the number of parameters, further proving the rationality of our method.

Inter-Domain Small-Sample Class Incremental Task Classification Performance

In the real world, new tasks and base class tasks do not always come from the same domain. We used malicious traffic and normal traffic as different domains to test the classification accuracy of our method and the other two methods when facing incremental tasks from different domains.

We used the normal traffic ISCX VPN-nonVPN dataset as the base class, and the malicious traffic MCFP dataset and the Self-made dataset were used as incremental tasks, respectively, to test the classification accuracy of the three methods when five new classes and ten new classes are added. Figure 8 shows the accuracy of the three methods and the downward trend line. When the incremental learning method is not used, the accuracy of the model shows a significant downward trend. After using the incremental method, it can be seen that the PackNet method has a certain competitiveness when five new classes are added. When ten classes are added, the accuracy of the PackNet method decreases significantly, and the accuracy of the incremental task between domains decreases slowly by using our method. This is because the PackNet method has a fixed number of allocated neurons. When incremental tasks are difficult and numerous, unreasonable allocation methods make it difficult for allocated neurons to train new tasks. Our method can dynamically allocate more neurons to fit incremental samples for dissimilar tasks and has more competitive advantages in difficult tasks.

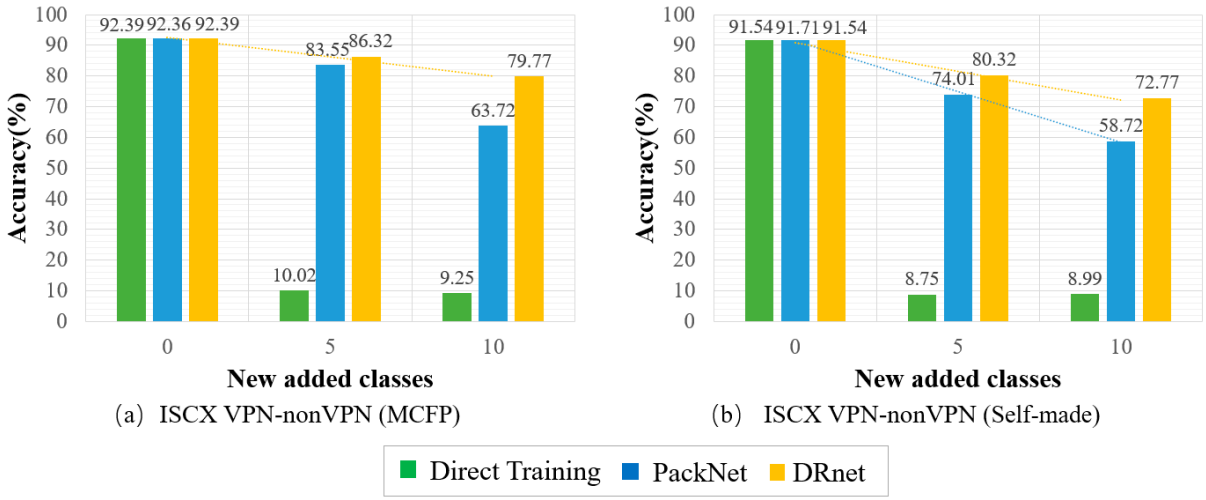

Figure 8. Inter-domain small-sample class incremental task performance.

Answer to Q2: The method can achieve more than 93% accuracy on malicious traffic small-sample classification base class data. In the same domain small-sample incremental tasks, the average accuracy can reach more than 90% when adding five types of data. For the inter-domain small-sample incremental task, the average accuracy of the new five categories can reach more than 80%, and the average accuracy of the new ten categories can reach more than 70%—better than the established baseline.

4.3. Comparison with Other Class Incremental Methods

We used the self-made small-sample dataset as the base class data and tested the performance of the classical incremental method Icarl method [29], EWC method [14],

PackNet method, and our proposed method in adding five tasks and ten incremental tasks. For the new classes, we adopted a sequential incremental approach to observe the influence of the order of task arrival on the method. The new classes were all small-sample tasks. For the Icarl method, we replayed the base class data and trained it with the new class samples. For the EWC method, we followed the regularization term added to the loss function by the author in the original text and rewrote the loss function for training new classes. For the PackNet method, in order to add many new tasks as possible, we chose the 75% pruning rate given by the original author to match the incremental tasks. Tables 4 and 5 show the performance comparison of our proposed method with other class incremental methods.

Table 4. Performance of different methods when adding five classes of tasks.

Method	Accuracy	Memory Occupancy (K)	FLOPS	Prediction Time (s)
Icarl	90.43 (±0.52)	52,156	5.52×10^{11}	2.34
EWC	80.77 (±1.68)	28,754	2.08×10^{11}	1.62
PackNet	85.60 (±0.94)	28,696	4.36×10^{11}	2.02
DRnet	91.15 (±0.88)	12,878	3.78×10^{11}	1.89

Table 5. Performance of different methods when adding ten classes of tasks.

Method	Accuracy	Memory Occupancy (K)	FLOPS	Prediction Time (s)
Icarl	88.96 (±1.01)	66,001	11.62×10^{11}	2.58
EWC	71.28 (±1.72)	30,599	2.38×10^{11}	1.99
PackNet	79.94 (±1.28)	30,541	5.22×10^{11}	2.47
DRnet	87.79 (±1.15)	17,475	4.96×10^{11}	2.46

In the field of malicious traffic classification, due to the timeliness and fast attack speed of some malicious programs, it is also important to evaluate the time complexity of traffic classification methods. We referred to several public papers in the field of traffic classification that discuss time complexity [30,31] and used the prediction time of the model on the test set as an indicator to evaluate the real-time response of the method. The results are shown in Tables 4 and 5. The results indicate that the method proposed in this paper yields no significant differences in prediction time compared to other methods. For the prediction times of 10 and 15 types of data, all four methods are roughly within 1–2 s.

From Table 4, we can see that, when a small number of new tasks arrive, among the three methods, the proposed method has a reasonable allocation of redundant network structure, and after the model transfer, the parameter range that the small-sample dataset needs to adjust becomes smaller, so the accuracy rate reaches a maximum of 91.15%. The EWC method only relies on the correction of the parameters to achieve the purpose of not forgetting the old tasks, and the accuracy is low. However, the EWC method does not need to store the old data and only needs to train the new tasks once, which has an advantage in the consumption of computing resources. Although Icarl has shown good classification accuracy, due to the small number of new task samples, its classification accuracy has not been optimal. At the same time, due to the need to store old data, it consumes a lot of computing resources, and its memory consumption is four times that of our proposed method. PackNet needs to reprune and retrain the parameters every time it adds a new task, and the computational overhead is large. The proposed method performs sorting and pruning operations before online deployment. This approach not only saves memory usage because it does not require multiple parameter sorting but also the number of calculations is reduced.

Table 5 shows the performance of each method as more class incremental tasks (ten new types of tasks) arrive as the model remains online. Due to the need to store all the data required for the current classification task in the Icarl method, both memory requirements and computing resources have increased dramatically. However, the classification accuracy of this method is the best of the four methods, reaching 88.96%. This is because, after the task volume gradually increases, whether it is a regularization method or a model-based method, the current model weights cannot take into account more classifications. At this time, the memory-based method occupies a certain advantage. After the class incremental tasks gradually increased, the EWC method could not correct the model parameters to meet many new tasks due to the regularization term, and the accuracy rate decreased to 71.28%. In the PackNet method, due to the gradual decrease in task allocation parameters, the classification accuracy began to decline, and its current accuracy was 79.94%. When the proposed method adds ten new tasks, the model redundancy parameters are based on a reasonable allocation strategy, which can maintain an accuracy of 87.79%. However, as the tasks continue to increase, the model structure becomes huge, and the number of calculations and memory usage gradually increases.

Answer Q3: When adding five tasks and ten tasks, our method has the lowest memory requirement while maintaining good accuracy, and the number of calculations also has certain advantages.

In order to visually show the classification performance of the four methods when adding five types of tasks, we drew the Receiver Operating Characteristic (ROC) curve and calculated the Area Under the Curve (AUC) value for the classification performance of different methods in the base class and the new class task, and also drew the average ROC curve (black in the graph) to judge the overall classification performance of the method, as shown in Figure 9. The x-axis of the ROC curve represents the false positive rate (FPR), and the y-axis represents the true positive rate (TPR). Therefore, the closer the curve is to the upper left corner, the better the classification performance of the current task classifier. It can be clearly seen from Figure 9 that the ROC curve of our proposed method is closer to the upper left corner. At the same time, we tested the effect of task arrival order on the classification accuracy of the method by adding incremental classes and incremental tasks. It can be seen that, for the new ninth class, the classification performance of our method is better than other methods, which is due to our task difficulty measurement strategy. At the same time, we found that the classification accuracy of the PackNet method was greatly affected by the task order, and the accuracy of the new task gradually decreased with an increase in the task arrival time.

In addition, when the positive and negative distribution of the test samples is uneven, the PR curve can more effectively reflect the quality of the classifier than the ROC curve. In order to comprehensively evaluate the model, we evaluated the Precision-Recall (PR) indicators of the four methods for the classification tasks of the new ten classes. In the PR curve, P represents the precision, and R represents the recall. The PR plot can intuitively show the recall and precision of the learner in the sample as a whole. When the precision and recall are high, the model performance is better, so the curve is expected to be close to the upper right corner. At the same time, if the PR curve of one method is completely 'wrapped' by the curve of another method, it can be asserted that the performance of the latter is better than the former. If the PR curves of the two methods intersect, it is difficult to generally determine which is better. We drew PR curves for the performance of the four methods when the model adds ten classes and calculated the mean Average Precision (mAP), as shown in Figure 10. From Figure 10, we can see that, among the four methods, the Icarl method performs as well as our proposed method on ten new tasks but is significantly better than the EWC and PackNet methods.

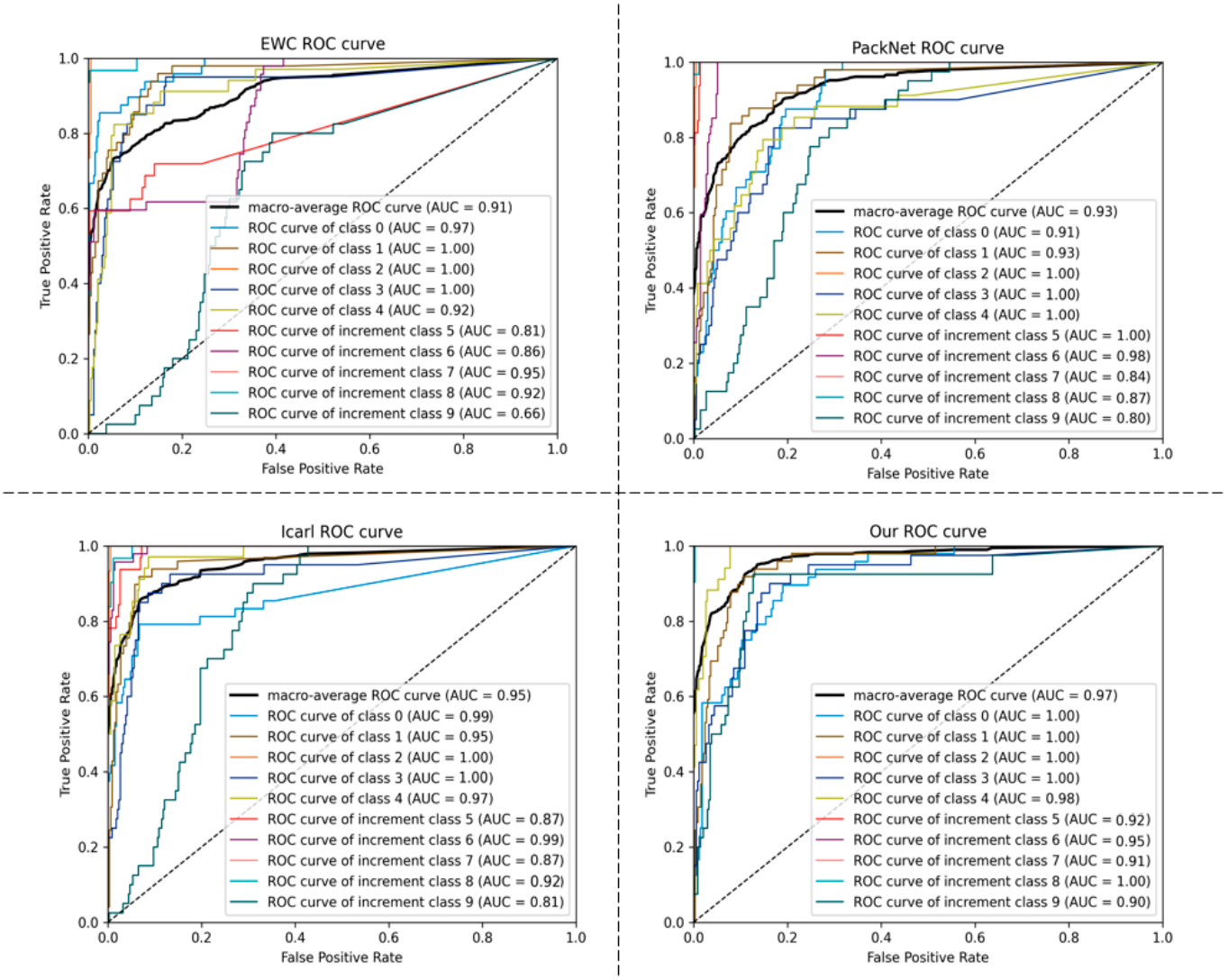

Figure 9. Multi-classification ROC curves of different incremental learning methods.

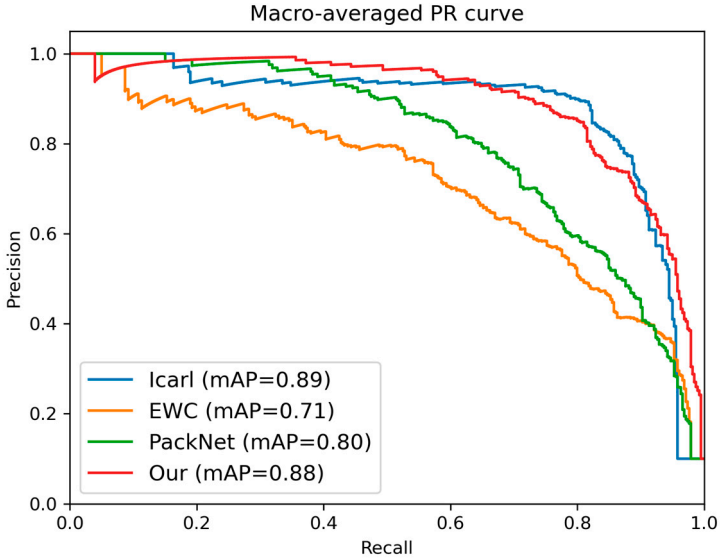

Figure 10. Multi-classification PR curves of different incremental learning methods.

4.4. Analysis of Incremental Task Neuron Allocation Strategy

Figure 11 shows the allocation strategy diagram for tasks with different similarities when using our method for class increment. The ordinate represents the number of neurons assigned to different tasks by the current layer, and the abscissa represents the different layers of the network.

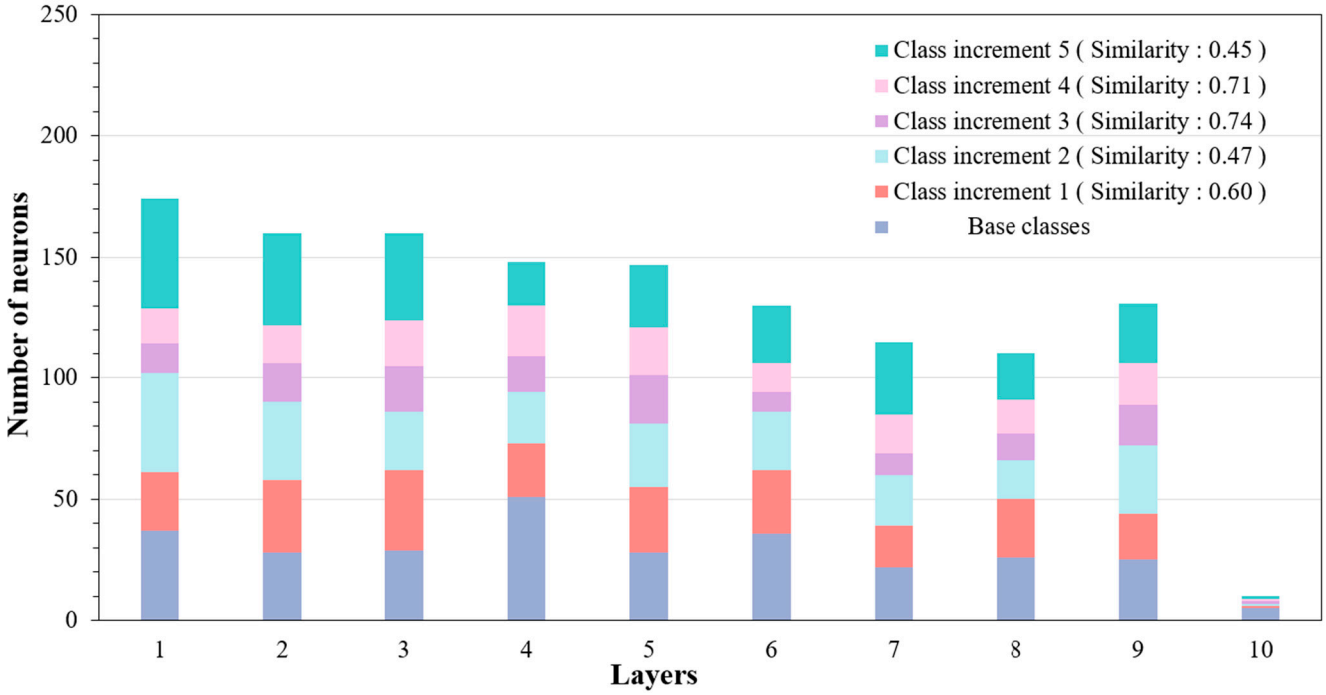

Figure 11. Visualization of neuron allocation strategy.

It can be seen that our proposed method allocates neurons for different similarities between new classes and base classes. The purple part of the graph is the number of neurons required for the base class task. When a class incremental task comes, we judge the number of neurons required for the task according to the similarity measurement method between the class incremental task and the base class task and allocate redundant neurons while maintaining the number and weight of neurons in the old task. Therefore, in this process, the number of neurons in each layer of the network is gradually increasing, which also means that the training of new tasks will not affect the classification accuracy of old tasks. For example, class incremental task five has a similarity of less than 50% with the base class. We believe that the features extracted by the current network structure cannot effectively classify the task. Therefore, on the basis of the features extracted by the current network, sufficient neurons are assigned to the task for training. The third class incremental task has a high similarity with the current base class domain, so it is assigned a smaller number of neurons. Since we use the convolutional layer instead of the fully connected layer as the classification layer, the tenth layer is the classification layer of the network. We set the base class task as five classifications. As the task increases, the number of classification layers increases one by one. The number of neurons in this layer represents the number of classification tasks in the current network.

5. Discussion

5.1. Method Effectiveness Analysis

In order to address the need of solving the incremental tasks of small-sample classes when the model is deployed online, we studied the scalability of the malicious traffic small-sample classification model. Due to the conservation of computing resources and the protection of data privacy, we propose a class increment method without storing old

classes. Firstly, based on the model transfer method, the small sample problem regarding malicious traffic is solved. Secondly, we pruned the redundant neurons of the model for small-sample tasks and imitated the 'silent synapse' structure of the human brain. The pruned neurons are used as a network structure that can be called at any time for the training of new categories. When training new classes, the weight of old classes is fixed. The method alleviates the catastrophic forgetting of old classes, solves the problem of data privacy, and does not increase computing resources.

In addition, when allocating redundant neurons, we abandoned the proportional allocation strategy in the traditional method and designed a measurement method to measure the similarity between classes. The dynamic allocation of redundant neurons according to task difficulty not only makes the allocation reasonable but also ensures that the classification performance of the new task model will not vary greatly with the order of task arrival or task difficulty.

Experiments show that our method has better classification accuracy than the traditional class increment method, whether it is the same domain small-sample malicious traffic increment task with high similarity or the difficult inter-domain small-sample malicious traffic increment task. Additionally, our method also has certain advantages that become clear when comparing memory usage and calculation times.

5.2. Limitations and Future Work

This paper explores the scalability of the malicious traffic classification model. The proposed dynamic parameter allocation method based on interclass similarity shows good performance in both intra-domain small-sample class incremental tasks and inter-domain small-sample classification tasks. However, the method still has some limitations, which we plan to explore further in future work.

Firstly, the method trains new classes based on neurons with current task redundancy. The advantage of this method is that it can complete the class incremental task without affecting the classification performance of the old classes. However, the redundant parameters of the model are limited. The focus of this paper is to ensure the classification accuracy of new classes. Therefore, the standard for allocating redundant neurons is only the number of neurons required by the current task compared with the old class task; the limitation of the number of class increments is not considered. When new tasks continue to arrive, there arises a situation wherein the model cannot add new tasks due to the exhaustion of a redundant parameter. At the same time, as the difficulty of new tasks increases, the redundant parameters will be exhausted earlier. For this problem, future research could consider model expansion, which could help further meet the needs of class increment by expanding the model. However, this will inevitably lead to an increase in computing resources. Balancing the consumption of computing resources and the performance of new class classification is also worthy of further exploration among researchers.

Secondly, in the real world, many practical applications cannot adapt to experimental settings with nonoverlapping tasks. Furthermore, due to the continuous updating of malicious traffic and changes associated with it, the distribution of classes will gradually change, which is often called concept drift. When deployed online, defining whether the class with concept drift belongs to the new class has become a topic of increasing concern. The problem of concept drift in the field of malicious traffic needs to be further defined. At the same time, the model also needs some robustness to resist the changes in category over time.

6. Conclusions

In this paper, we proposed a small-sample incremental learning method for malicious traffic. On the fully convolutional network we designed, based on the designed inter-class similarity measurement method, the redundant neurons of the classification model are dynamically reused to adapt to incremental tasks. At the same time, we also applied the idea of transfer learning to transfer the knowledge learned from the large dataset to the target small sample and small-sample incremental tasks, which improves the mismatch between

the small-sample dataset and the model parameters. Through extensive experiments, we have demonstrated that our method outperforms other incremental methods in terms of accuracy and computational resource consumption in the small-sample incremental problem of malicious traffic. The work of this paper can provide new ideas for the small-sample incremental learning problem of malicious traffic.

Author Contributions: R.W. was mainly responsible for designing the research, data analysis, and writing the manuscript for this study. J.F. was mainly responsible for data collection and manuscript modification. R.Z. was mainly responsible for design the research. M.G. was mainly responsible for data collection and the production of charts. X.L. and Z.Q. were mainly designing the experiment and preparation. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by National Key Research and Development Project of China (2019QY1302).

Data Availability Statement: This study analyzed three publicly available datasets. These data can be found at the following websites: https://www.unb.ca/cic/datasets/vpn.html (accessed 5 June 2020), https://github.com/yungshenglu/USTC-TFC2016 (accessed 21 June 2019), https://mcfp.fel-k.cvut.cz/publicDatasets/datasets.html (accessed 14 June 2020).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Haque, M.; Palit, D. A review on deep neural network for computer network traffic classification. *arXiv* **2022**, arXiv:2205.10830.
2. Hameed, A.; Violos, J.; Leivadeas, A. A deep learning approach for IoT traffic multi-classification in a smart-city scenario. *IEEE Access* **2022**, *10*, 21193–21210. [CrossRef]
3. Wang, H.; Xu, T.; Yang, J.; Wu, L.; Yang, L. Sessionvideo: A Novel Approach for Encrypted Traffic Classification via 3D-CNN Model. In Proceedings of the 2022 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS), Takamatsu, Japan, 28–30 September 2022; pp. 1–6.
4. Kim, H.; Kim, M.; Ha, J.; Roh, H. Revisiting TLS-Encrypted Traffic Fingerprinting Methods for Malware Family Classification. In Proceedings of the 2022 13th International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 19–21 October 2022; pp. 1273–1278.
5. Doshi, K.; Yilmaz, Y. Rethinking video anomaly detection—A continual learning approach. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 3–8 January 2022; pp. 3961–3970.
6. Cao, Y.; Gan, H. CLAD: A Deep Learning Framework for Continually Learning in Anomaly Detection. In Proceedings of the 2022 5th International Conference on Software Engineering and Information Management (ICSIM), Yokohama Japan, 21–23 January 2022; pp. 158–163.
7. García González, G.; Casas, P.; Fernández, A.; Gómez, G. Steps towards continual learning in multivariate time-series anomaly detection using variational autoencoders. In Proceedings of the IMC 22—22nd ACM Internet Measurement Conference, Nice, France, 25–27 October 2022; pp. 774–775.
8. Doshi, K.; Yilmaz, Y. Continual learning for anomaly detection in surveillance videos. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020; pp. 254–255.
9. Amalapuram, S.K.; Tadwai, A.; Vinta, R.; Channappayya, S.S.; Tamma, B.R. Continual Learning for Anomaly based Network Intrusion Detection. In Proceedings of the 2022 14th International Conference on COMmunication Systems & NETworks (COMSNETS), Bangalore, India, 4–8 January 2022; pp. 497–505.
10. Alam, M.S.; Yakopcic, C.; Subramanyam, G.; Taha, T.M. Memristor Based Neuromorphic Network Security System Capable of Online Incremental Learning and Anomaly Detection. In Proceedings of the 2020 11th International Green and Sustainable Computing Workshops (IGSC), Pullman, WA, USA, USA, 19–22 October 2020; pp. 1–8.
11. Kwon, B.; Kim, T. Toward an Online Continual Learning Architecture for Intrusion Detection of Video Surveillance. *IEEE Access* **2022**, *10*, 89732–89744. [CrossRef]
12. Vardalaki, D.; Chung, K.; Harnett, M.T. Filopodia are a structural substrate for silent synapses in adult neocortex. *Nature* **2022**, *612*, 323–327. [CrossRef] [PubMed]
13. Prasath, S.; Sethi, K.; Mohanty, D.; Bera, P.; Samantaray, S.R. Analysis of Continual Learning Models for Intrusion Detection System. *IEEE Access* **2022**, *10*, 121444–121464. [CrossRef]
14. Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A.A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. USA* **2017**, *114*, 3521–3526. [CrossRef] [PubMed]
15. Lopez-Paz, D.; Ranzato, M.A. Gradient episodic memory for continual learning. In Proceedings of the Advances in Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017.

16. Pezze, D.D.; Anello, E.; Masiero, C.; Susto, G.A. Continual Learning Approaches for Anomaly Detection. *arXiv* **2022**, arXiv:2212.11192.
17. Mallya, A.; Lazebnik, S. Packnet: Adding multiple tasks to a single network by iterative pruning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7765–7773.
18. Kang, B.; Liu, Z.; Wang, X.; Yu, F.; Feng, J.; Darrell, T. Few-shot object detection via feature reweighting. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 8420–8429.
19. Douillard, A.; Cord, M.; Ollion, C.; Robert, T.; Valle, E. Podnet: Pooled outputs distillation for small-tasks incremental learning. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 86–102.
20. Tao, X.; Hong, X.; Chang, X.; Dong, S.; Wei, X.; Gong, Y. Few-shot class-incremental learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 12183–12192.
21. Li, Z.; Hoiem, D. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 2935–2947. [CrossRef] [PubMed]
22. Parisi, G.I.; Kemker, R.; Part, J.L.; Kanan, C.; Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Netw.* **2019**, *113*, 54–71. [CrossRef] [PubMed]
23. Maltoni, D.; Lomonaco, V. Continuous learning in single-incremental-task scenarios. *Neural Netw.* **2019**, *116*, 56–73. [CrossRef] [PubMed]
24. Tao, X.; Hong, X.; Chang, X.; Gong, Y. Bi-objective continual learning: Learning 'new'while consolidating 'known'. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 5989–5996.
25. Gretton, A.; Borgwardt, K.M.; Rasch, M.J.; Schölkopf, B.; Smola, A. A kernel two-sample test. *J. Mach. Learn. Res.* **2012**, *13*, 723–773.
26. Wang, W.; Zhu, M.; Zeng, X.; Ye, X.; Sheng, Y. Malware traffic classification using convolutional neural network for representation learning. In Proceedings of the 2017 International Conference on Information Networking (ICOIN), Da Nang, Vietnam, 11–13 January 2017; pp. 712–717.
27. Lotfollahi, M.; Jafari Siavoshani, M.; Shirali Hossein Zade, R.; Saberian, M. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Comput.* **2020**, *24*, 1999–2012. [CrossRef]
28. Laurens VD, M.; Hinton, G. Visualizing Data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
29. Rebuffi, S.A.; Kolesnikov, A.; Sperl, G.; Lampert, C.H. iCaRL: Incremental classifier and representation learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2001–2010.
30. Di Mauro, M.; Galatro, G.; Liotta, A. Experimental Review of Neural-based approaches for Network Intrusion Management. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 2480–2495. [CrossRef]
31. Dong, S.; Xia, Y.; Peng, T. Network Abnormal Traffic Detection Model Based on Semi-Supervised Deep Reinforcement Learning. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 4197–4212. [CrossRef]