IEICE TRANS. INF. & SYST., VOL.E108–D, NO.2 FEBRUARY 2025

PAPER

Incremental Learning for Network Traffic Classification Using Generative Adversarial Networks

Guangjin OUYANG[†,††a)], ***Member***, **Yong GUO**[†,††], **Yu LU**[†], ***and*** **Fang HE**[†,††], ***Nonmembers***

SUMMARY With the rapid development of Internet technology, the type and quantity of network traffic data have increased accordingly, and network traffic classification has become an important research task. In previous research, there are methods based on traditional machine learning and deep learning; compared to machine learning, deep learning can obtain good results by converting network traffic into two-dimensional images and utilizing deep learning classification models. However, all of these methods have some limitations: the trained models cannot learn sustainably, and the generalization ability of the models is limited. In order to solve this problem, we propose a network traffic classification methods based on incremental learning and Mixup, which is based on generative adversarial networks. First, the network traffic is converted into a 2D image, the original database is linearly interpolated using Mixup to reduce the overfitting tendency of the model and improve the generalization ability, and the traffic is classified using the ability of deep learning on the image. Secondly, we improve the traditional incremental learning algorithm. To effectively address the imbalance between old and new categories in incremental learning. The experimental results show that the model performs well in classification experiments, reaching 92.26% and 93.86% accuracy on the ISCXVPN2016 and USTC datasets, respectively, and we can maintain a high accuracy rate with limited storage space in the process of increasing new categories.

key words: *traffic classification, mixup, class incremental learning, deep learning*

1. Introduction

Network traffic classification has received much attention in recent decades [1], [2]. Network traffic classification, specifically, refers to classifying and managing network traffic to identify, classification and process different types of data traffic. The traditional traffic classification methods are port-based and deep packet inspection strategies [3]. In the port-based method, the standard port 21 is applied in FTP protocol, and port 443 is applied in HTTPS protocol; however, random dynamically assigned ports appeared, and more and more applications started to use random ports, and thus, nowadays, few people use ports for traffic classification. On the other hand, deep packet identifies traffic classes by the payload patterns or keywords of the packets in the traffic, but this method cannot handle encrypted traffic. In recent years, with the rise of artificial intelligence, network traffic categorization methods based on machine learning and deep learning [4], [5] have gained increasing attention. Machine learning usually learns based on the features of the traffic. Thus, selecting features can significantly change the accuracy of profound learning results. However, it can guarantee learning without extracting the features of the data but still needs to improve on the problems of costly and time-consuming learning. The models trained by these methods have good accuracy in the training set but poor accuracy in the unseen testing set, which results in poor generalization ability of the model, which cannot adapt to different data and needs better robustness. In addition, the trained models are usually trained on offline datasets, and the models need to be re-trained when encountering data of unknown categories, which does not consider the continuous learning ability of the models and is challenging to apply in real-world application scenarios.

This paper proposes an innovative approach to network traffic classification that skillfully combines the Mixup data generation technique, class incremental learning, and generative adversarial networks. The core idea lies in utilizing the adversarial idea of generative adversarial networks; for the training data, we first use the Mixup method to mix the samples, generate the false samples in the generator, and use the feature learning of the actual samples and false samples in the discriminator, and finally generate the corresponding classes. Specifically, we efficiently characterize network traffic by extracting its Ethernet, TCP/UDP, and IP header information and transforming it into a 2D image. Further, we utilize the Mixup technique in network traffic classification by combining multiple data samples to create new training samples, which significantly improves the model's generalization ability on diverse data. At the same time, we employ a class-incremental learning strategy further to enhance the continuous learning capability of the model. The significant advantage of our approach is that it does not require in-depth analysis of the payload portion of the data packet. Mixup can improve the generalization ability of the model with limited samples. It incorporates incremental learning to allow the model to continuously learn new classes, effectively avoiding the so-called catastrophic forgetting [29] phenomenon. Our experimental results show that in experiments with datasets containing both plain and encrypted traffic, this approach has an accuracy of 92.26% in network traffic categorization. Even in an incremental learning scenario, it can maintain an accuracy of more than 80.55%.

We adopt generative adversarial networks as the core model for network traffic classification. We cleverly uti-

Manuscript received May 30, 2024.
Manuscript revised August 8, 2024.
Manuscript publicized September 13, 2024.

[†]School of Computer and Information Technology, Qiannan Normal University for Nationalities, Duyun 558000, China.
[††]Experimental Training Centre, Qiannan Normal University for Nationalities, Duyun 558000, China.
a) E-mail: ouyangguangjin@sgmtu.edu.cn

DOI: 10.1587/transinf.2024EDP7129

lize the Mixup generative multisample and class incremental learning methods to enhance the model's generalization ability and sustainable learning capability. This combination improves the model's practicality and dramatically extends its application scope.

2. Related Work

2.1 Port-Based Network Traffic Inspection

In the early stages of the Internet, numerous applications were often assigned a fixed port number specified by the Internet Assigned Numbers Authority (IANA). For example, the HTTP protocol commonly uses port number 80, while the HTTPS protocol usually uses port 443, and the standard port number for the e-mail protocol SMTP is 25. A list of publicly available port numbers is also available for querying on the official IANA website. However, relying solely on fixed port numbers to identify and categorize network traffic is unreliable. There are 65,535 ports in the transport layer, of which only the first 1,024 are considered public ports. The remaining port numbers are user-definable and are used by various applications. Especially in developing P2P applications and technologies such as port masquerading [6], [7], these technologies make it more challenging to rely on fixed ports for traffic classification. With the popularization of dynamic and random port technologies, traffic classification for these types of applications becomes more complex and challenging.

2.2 Deep Packet-Based Network Traffic Inspection

Deep packet-based network traffic detection, also known as Deep Packet Inspection (DPI) [8], [9], is mainly achieved by analyzing the contents of the packet's header load. This method works by comparing predefined fixed strings, and if these predefined strings are found in the packet's load, the traffic class can be determined accordingly. For example, Neminath [11] et al. used a bit-level DPI signature generation technique called BitCoding. The BitCoding technique uses only a small number of initial bits in the data stream and uses these invariant bits as signature identifiers. These bit signatures are then encoded and transformed into a newly defined state-transformer transformation constraint counting automaton, which, combined with a variant of the Hemming distance, enables the technique to perform signature similarity detection, which accomplishes the categorization of network traffic. Although deep packet detection excels in accuracy, it suffers from several disadvantages, such as slow processing speed and high resource usage. In addition, this method relies heavily on predefined fixed strings and requires manual labor to add these strings to a library of predefined fingerprints manually.

2.3 Machine Learning and Deep Learning-Based Network Traffic Inspection

Utilizing machine learning methods to classify network traffic is a common practice. The core idea of this approach is based on the observation that different classes of applications generate different traffic characteristics. For example, Web applications tend to generate a large number of traffic byte transfers quickly, while video applications mainly present a large amount of UDP byte traffic. The features of different types of traffic can be effectively captured by extracting these features and selecting appropriate machine learning algorithms (e.g., support vector machines, random forests, etc.) for model training. Once trained, these models can classify unknown traffic [10], [17]. However, machine learning methods rely heavily on the accuracy of feature selection. The quality of model training directly affects the correctness of the classification of new data. Therefore, although machine learning methods are widely used in traffic classification, their classification accuracy may sometimes not be as high as deep packet inspection.

Deep learning, an essential branch of machine learning, has been gaining popularity in recent years for applications in various fields. Within the deep learning framework, the model can learn and extract high-level feature representations of the original data layer by layer by constructing a multi-layered neural network structure. A significant advantage of this process is that it eliminates the need for manual feature extraction, allowing the neural network to automatically learn and output these high-level features [12]–[14], [19]. This property allows deep learning to show significant advantages in accomplishing complex classification tasks. We summarize the details related to the use of deep learning for network traffic classification in several previous research works, including the type of input data they used, the network structure, and the relevant academic papers. This information is organized in Table 1 and is intended to provide the reader with a global view of deep learning in network traffic classification applications.

Wei Wang [15] was one of the first to use deep learning algorithms to solve the problem of network traffic classification; he designed an end-to-end modeling framework to regenerate the ISCX dataset with 784 bytes of data into IDX3 image format and then combined it with a one-dimensional

Table 1 Related work based on deep learning

Input data	Network Structure	Paper
Payload	1D CNN/2D CNN	Wang et.al(2017)[15]
Five-Tuple	CNN + RNN	Lopez-Martin(2017)[16]
Arrival time, Packet size	SNN	Rasteh A et.al(2022)[14]
Payload[1480B]	1D CNN	Soleymanpour et al. (2021)[24]
Packet direction, length, number of slices, etc.	Graph Structure	Yao Li et al. (2024)[25]
Payload[32x32B]	2D CNN	Ours

CNN for model training with good results. Lopez [16] et al. proposed a network traffic detection method that combines recurrent neural networks (RNN) and Convolutional Neural Networks (CNN) for network traffic detection, using the first 20 packets as input for the image preparation phase and then for each packet, source port, destination port, packet load bytes, TCP window size, arrival interval, and direction were extracted and combined with Recursive Neural Networks (RNN), which gave good results. Rasteh [14] proposed identifying encrypted traffic using packet arrival time and packet size combined with SNN (impulse neural network). Soleymanpour [24] proposed a cost-sensitive CNN (Convolutional et al.) to deal with the problem of class imbalance in encrypted traffic classification. Yao Li [25] proposed using graph structure in the interaction process. An innovative approach to preserve information is to abstract packets as interaction actions, group interaction actions based on consecutive actions in the same direction to form slices, and have different transition states between interaction actions inside and outside the slices. These different transition states are effectively represented by edges with different properties in the graph, and these features are utilized to construct graph structures for traffic classification.

2.4 Mixup Model Generalization

Domain generalization is an essential problem in transfer learning concerning generalization ability. The goal is to construct a model from several different source domains so that the model can effectively predict situations with arbitrary unknown data distributions. Assume that given M source domains data $S_{train} = \{S^i \mid i = 1, \cdots, M\}$, the data distributions in these source domains are different. Thus, the core challenge of domain generalization is how to allow the model to learn a prediction function $h(x)$ with strong generalization ability from the M source domains data so that the error on the unknown test data S_{test} is minimized. Here, the prediction error can be expressed and measured by the expectation and loss functions, and the goal is to minimize the expectation loss. The objective is to minimize the expectation loss, where E, $\ell(\cdot,\cdot)$ are the expectation and loss functions, x is the sample feature, and y is the label of the corresponding sample.

$$\min_h E_{(x,y)\in S_{test}}[\ell(h(x), y)] \tag{1}$$

Domain generalization methods can be divided into three main categories: data manipulation, representation learning, and learning strategies. The category of data manipulation focuses on performing various operations on the input data, and data augmentation and data generation methods are some of the most effective means of improving the generalization ability of models. The classical data augmentation methods include performing operations such as inversion, rotation, and scaling on the input data. These techniques introduce visual changes to increase the diversity of the data by introducing visual variations.

Mixup is a data augmentation technique Zhang et al. [18] proposed in 2017. It generates new training samples by linearly interpolating between them to enhance the generalization ability and robustness of the model. The main effect of Mixup is to reduce the overfitting of the model and to improve the tolerance of the model to noise and variations in the input data. Mathematically, Mixup can be viewed as a smoothing operation on the input space and label space, which encourages the model to learn the linear relationship between inputs and labels, thus making the model more robust in the face of unseen data. Using Mixup, we can augment the model's samples with data, thus improving the model's generalization performance.

2.5 Class Incremental Learning

In applying deep learning models, the so-called catastrophic forgetting problem is often encountered, manifesting in the form of the model forgetting previously learned tasks as it learns new ones. This is usually because the weights of the new tasks in the model are updated to overwrite the weights of the previous tasks, resulting in a degradation of the model's performance on the previously learned tasks. To deal with this challenge, the standard solution strategies can be classified into three categories: regularization-based, replay-based, and parameter isolation-based.

The regularization-based idea is to protect the old knowledge from being overwritten by the new knowledge by imposing a constraint on the loss function of the new task, which usually does not require the use of the old knowledge. A more representative approach is the forget-less learning method proposed by Li and Hoiem [22], where the idea is to optimize the and for the new task under the premise that the and has little effect on predicting the new task samples. The constraint ensures that the model can still remember its old parameters. Saihui Hou [23] and others proposed a new framework for incremental learning of a unified classifier that combines three components: cosine normalization, less forgetting constraints, and interclass separation to mitigate the detrimental effects of imbalance.

Parameter isolation-based approaches segregate the parameters of the old and new tasks by not changing the old task parameters and incrementally expanding the model. PackNet [26] is an approach that hard segregates the parameters of the old and new tasks. Each time a new task arrives, PackNet incrementally uses a portion of the model space, reserving redundant model space by pruning, leaving margin for the next task. J Serrà [27] uses the Hard Attention (HAT) mechanism to mask different parts of the model according to the different tasks, thus assigning different parts of the model to each task. Meanwhile, HAT uses regularization terms to impose sparsity constraints on attention masking to distribute the model space better among tasks.

Replay-based approaches effectively cope with the catastrophic forgetting problem in deep learning by storing some old data in a cache and replaying this old data when learning a new task. A typical example of this approach is

Table 2 Sample size of the datasets

Datasets	Class	Packets	Datasets	Class	Packets	Datasets	Class	Packets
ISCX VPN-NoVPN	Email	12004	WIDE	DNS	73005	USTC-TFC2016	Bittorrent	11000
	VPN-Email	9554					FTP	36000
	Chat	46319		SMTP	350218		Facetime	6000
	VPN-Chat	32451					Gamil	15000
	File Transfer	25718		SSH	248290		MySQL	20000
	VPN-File Transfer	51007					Outlook	15000
	Voip	69023		FTP	40531		SMB	92545
	VPN-Voip	11007					Skype	12000
	Streaming	78736		HTTP	489802		Weibo	12106
	VPN-Streaming	15005					World of Warcraft	14000
	p2p	24091		HTTPS	315408			
	VPN-p2p	36415						

the iCaRL [28] algorithm. iCaRL core idea is to maintain a representative sample set, i.e., an exemplar set, for each observed category. When a new category is introduced, a new exemplar set is created for this new category, while the size of the exemplar set is adjusted for existing categories. The result is that iCaRL can efficiently perform continuous learning and maintain the memory of old knowledge within a limited memory space.

In this area of network traffic classification, combining the approach of class incremental learning and the Mixup method constitutes an innovative attempt. In designing the model, we refer to the algorithmic framework of iCaRL and combine it with a generative adversarial network structure to address the problem of continuous learning of new classes and improve the model generalization. Our goal is to create a model that can adapt to changing types of network traffic while maintaining the memory of past learned knowledge.

3. System Implementation

3.1 Datasets

In our experiments, the selected datasets include three datasets: the ISCX VPN-nonVPN [21] dataset, the WIDE dataset [34], and the USTC-TFC2016 dataset [5]. We put the ISCX VPN-nonVPN dataset. The VPN dataset is categorized into six regular and 6 VPN-encrypted traffic types. The WIDE dataset is a public packet trace from the MAWI working group, which focuses on capturing daily traffic from the USJapan backbone from 14:00 to 14:15 (Japan Standard Time.), publicly available at http://mawi.wide.ad.jp/, where we select traffic from the May 2020 traffic. In the USTC-TFC2016 dataset, we only use benign traffic, which contains only ten categories for subsequent in-depth analyses and processing. In order to provide readers with the details, we exhaustively describe the details of the dataset in Table 2.

3.2 Data Preprocessing Stage

In the preprocessing stage of network traffic classification, the preprocessing work is divided into three primary levels according to classification tasks: session, flow, and packet.

Fig. 1 The TCP/IP model shows that packets are encapsulated by the data link layer, transport layer, and network layer

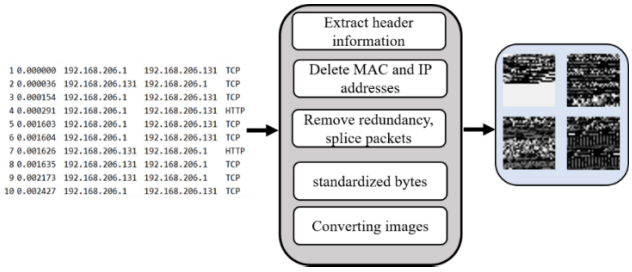

Fig. 2 The workflow of data preprocessing

Considering that we need to capture the fine-grained characteristics of the traffic, we choose to perform preprocessing at the packet level in this phase. As shown in Fig. 2, this preprocessing process mainly includes extracting the Ethernet header, IP header, and TCP header information of each packet, deleting the MAC address and IP address, eliminating redundant packets, normalizing byte conversion, and image generation.

Extracting header information. One of the critical steps in the preprocessing stage of a packet is extracting header information. According to the TCP/IP model in Fig. 1, the packet will be encapsulated in the transmission process through the transport layer, the network layer, and the data link layer, and the encapsulated packet contains the Ethernet header, the Ethernet tail, the IP header, and the TCP/UDP header, which are several important pieces of information. The Ethernet tail does not carry any information about the traffic class, so we chose to remove this section. Ultimately, we focus on extracting and processing the data in the three sections of the Ethernet header, IP header, and TCP header (UDP header) and normalizing this data to ensure consistency and standardization.

Delete MAC and IP addresses. During network communication, the MAC address and IP address are fixed information. To ensure the effectiveness of model training and

reduce the risk of overfitting, we cull out the MAC address and IP address in each packet during the preprocessing stage. This practice helps prevent the model from relying too much on these invariant features, thus improving the model's ability to learn the essential features of traffic data. By removing this static information, we can make the model more focused on recognizing dynamic features that are more meaningful for the classification task.

Redundant and spliced packets are eliminated. In the process of network communication, we find that the pcap file will contain plain and encrypted traffic of different applications; in this case, it is difficult for us to ensure the accuracy of the extracted traffic in order to minimize the interference of irrelevant traffic, we decided to take the following measures to get as much as possible the traffic we need. First of all, for the ISCX dataset, we mainly distinguish between encrypted and non-encrypted traffic classification between each application; take the Email category as an example; it has four files email1a, email1b, email2a, email2b, respectively, we read the contents of these four files by using the Scapy library in python and then analyze each packet sequentially For each packet, if the source/destination port is 53 (DNS protocol, which is more common), or if the packet protocol type is ARP (Address Resolution Protocol), STP (Spanning Tree Protocol), etc., the packet is excluded. Secondly, for the WIDE dataset, we mainly distinguish different types by different port numbers; for example, in the FTP category, we only keep the packets with port number 21. As for the USTC dataset, we adopt a similar approach to the ISCX dataset to ensure the accuracy of the subsequent experiments by eliminating packets with port numbers unrelated to the application or packets with protocol types that do not belong to the application. The problems encountered in real networks are far more complex than those in the experiments. Thus, our approach only improves the accuracy of the data to a certain extent, and it is limited by the fact that irrelevant packet protocols and port numbers must be specified manually, which requires human adjustment of the strategy. Finally, to standardize the length of the packets, we padded all packets to 1024 bytes to facilitate the generation of 2D 32×32 images. In Table 6, we also experimented with different image sizes separately, so if the images to be generated are 64×64 and 128×128, we will fill them to 4096 and 16384 bytes, respectively. We adopt a padding strategy based on the transmission direction for the blank portions of the packets. Specifically, packets from source to target are defined as upstream packets with the blank part padding value set to 0, while packets from target to source are defined as downstream packets with padding of 255. This transmission direction-based padding helps the neural network to learn and differentiate between different upstream and downstream types of traffic data more efficiently, thus improving the learning effect of the model.

Normalized Byte Conversion. In the final preprocessing step, we convert the 1024-byte contents of each packet into an equal-sized array and perform normalization on each element of the array. This step aims to unify the range of data for subsequent processing and analysis. The normalization process is essential to improve the neural network's performance during the training process. Scaling the data to a normalized range helps avoid numerical computational problems such as vanishing or exploding gradients and significantly improves the neural network's convergence speed. Normalization ensures that all features are treated equally during training, resulting in a more efficient and stable training process.

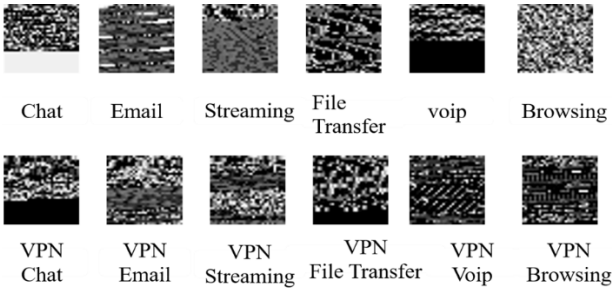

Fig. 3 Visualization of all classes of traffic in ISCX dataset

Image Generation. Based on the previous data processing, a series of 1024-byte arrays are generated into a two-dimensional 32×32 image, as in Fig. 3.

3.3 Semi-Supervised GAN (SGAN)

Generative Adversarial Network (GAN) is a deep learning model proposed by Ian Goodfellow [20] and his colleagues in 2014. GAN consists of two main parts: generator and discriminator. The role of the generator is to take random noise as an input and to learn accurate data by learning its feature distribution. The purpose of the discriminator is to determine whether the data is accurate or generated by the generator. Generator G and Discriminator D play adversarial roles during the training process, respectively, and jointly optimize the whole model through adversarial training. The objective function of GAN can be expressed by Eq. (2).

$$\min_G \max_D V(D,G) = \mathrm{E}_{x\sim P_{data(x)}}[\log D(x)] + \mathrm{E}_{z\sim P_z(z)}[\log(1 - D(G(z)))] \quad (2)$$

Although GAN performs well in the field of image generation, due to its risk of overfitting and the fact that GAN itself only discriminates between true and false samples, it cannot be applied in classification tasks; based on this, we utilize SGAN [33], which is an extension of GAN, to achieve the task of classifying samples by utilizing the generative power of GAN. Semi-supervised GAN, also known as the so-called SGAN, which is an extension of GAN, can generate high-quality samples by combining generative adversarial networks and semi-supervised learning and improve the classification task's performance by utilizing unlabeled data. The basic idea of SGAN is to use the discriminators not only for the actual and generative data but also for classifying the actual data of the known categories. During the training process, the generator G and the discriminator D are trained

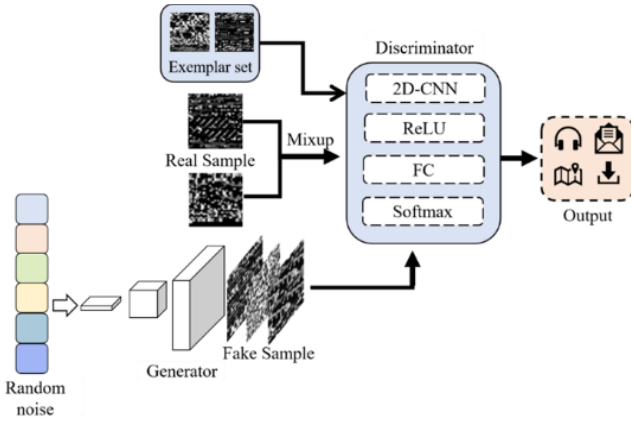

Fig. 4 Overall of framework

on the dataset first; the input data belongs to one of the N categories, and the discriminator D is used to determine which of the N+1 categories the input data belongs to, in addition to this, the output of D is used to determine whether the input data belongs to an additional class of the REAL or FAKE category.

Figure 4 shows the whole training process. We realize this process by modifying the discriminator's output network and using the Softmax output layer, corresponding to the discriminator's output with N+1 categories: [CLASS_1, CLASS_2, CLASS_3, ... CLASS_N, FAKE]. We can train the classification task on our input data using this approach.

3.4 Mixup

In order to achieve the risk of preventing overfitting during SGAN training, we consider processing the samples to improve the model's generalization ability. Standard methods include rotating the samples, flipping them, etc. However, the ability to improve is limited. Mixup based on data generation is a method to minimize the empirical risk, which is an excellent way to achieve the model generalization. The core idea of Mixup is that, in a training iteration, it will randomly mix the samples from any two classes of data, and this mixing process is controlled by the value, which determines the weight of these two classes of data in the final mixed sample. This parameter determines the weight of these two data classes in the final mixed sample.

$$\mu(x,y) = \frac{1}{n}\sum_{j}^{n} \underset{\lambda}{E}[\delta(x = \lambda \cdot x_i + (1-\lambda)\cdot x_j, \\ y = \lambda \cdot y_i + (1-\lambda)\cdot y_j)] \tag{3}$$

where $\lambda \sim Beta(a,a)$, $a \in (0,\infty)$, where x and y are two feature-target vectors randomly selected from the training data, $\lambda \in [0,1]$, the occupancy ratio between the hybrid a parameter controls.

$$x = \lambda \cdot x_i + (1-\lambda)\cdot x_j \tag{4}$$

$$y = \lambda \cdot y_i + (1-\lambda)\cdot y_j \tag{5}$$

Mixup implementation is straightforward and introduces a minimal computational overhead; thus, in this paper, we use this method to enhance the model generalization ability; according to the paper [18], we set $\lambda = 0.5$. Thus, in the subsequent experiments, we first go through the Mixup sample mixing method on the input data so that the model's generalization ability can be improved during the training process.

3.5 Improved Class Incremental Learning

Class incremental learning is a dynamic learning method that relies on the initial training data and can progressively acquire new learning capabilities from a continuous data stream. rebuffi [28] et al. proposed an innovative method named iCaRL. The method is unique in maintaining a set of exemplar samples for each observed class. In the iCaRL model, the exemplar set for each class is designed as a subset of all the samples of that class to capture and include the most representative information of that class.

Although iCaRL preserves the classification performance of the old classes by selecting the example set, due to performance constraints (e.g., the example set is limited), it is not possible to fully preserve the classification performance of the old classes and balance the old and the new classes at the same time, in order to solve this problem, Yue Wu [31], [32] proposed to integrate the distillation loss on the old samples and the new training samples by proposing a new loss function and the cross-entropy loss, and in order to balance the relationship between the old and new classes, a scalar is utilized to pair represent the data deviation between the old and new classes. The method mainly draws on the ideas of Lwf [22] and iCarl, where the introduction of old data is improved by enabling similar predictions between the old and new classifiers on the old n classes, thus taking distillation loss on the old classifiers and cross-entropy loss on the old and new classifications. Distillation losses are calculated as follows:

$$L_d = \sum_{x\in X^n \cup X^m} \sum_{k=1}^{n} -\frac{\hat{f}_k(x)}{T} \log\left(\frac{f_k(x)}{T}\right) \tag{6}$$

The cross-entropy loss is calculated as follows:

$$L_c = \sum_{x\in X^n \cup X^m} \sum_{k=1}^{n+m} -\delta_{y=k} \log[f_k(x)] \tag{7}$$

The overall loss function is

$$L = \lambda L_d + (1-\lambda)L_c \tag{8}$$

Where λ is used as a balancing factor to balance these two terms, and for the data imbalance between the old and new classes, factor β is used and applied to the output of the new m class.

$$f^{n+m}(x) = [f_1(x), f_2(x), \ldots f_n(x), \\ \beta f_{n+1}(x), \ldots \beta f_{n+m}(x)] \tag{9}$$

Algorithm 1 Improved class incremental learning algorithm

Input: Training samples $X^s,\cdots,X^t$ of the new class $s,\cdots,t$, current model parameters Θ, Current exemplar set $P=\{P_1,\cdots,P_{s-1}\}$, memory capacity K

Output: The updated model parameters Θ, the set of exemplar P

1. $D^{exemplar} \leftarrow \bigcup_{y=1,\cdots,s-1}\{(x,y): x \in P_y\}$ #Current Exemplar Set
2. $D^{new} \leftarrow \bigcup_{y=s,\cdots,t}\{(x,y): x \in X^y\}$ #New Training Samples
3. for y=1 to s-1 do
4. $q_i^y \leftarrow g_y(x_i)$ for $all(x_i,\square) \in D^{exemplar}$
5. end for
6. $L(\Theta)=\lambda L_d+(1-\lambda)L_c$ #Network Training with Distillation Loss and Cross-entropy Loss Function
7. $\Theta \leftarrow L(\Theta)$ #Update Model Parameters
8. $m \leftarrow K/t$ #Divide the Number of Exemplar Per Class
9. for y=1 to s-1 do
10. $P_y \leftarrow P_y[1:m]$
11. end for
12. for y=s to t do
13. $P_y \leftarrow ConstructExemplarSet(X^y,m,\Theta)$
14. end for
15. $P \leftarrow \{P_1,\cdots P_t\}$ #Update the ExemplarSet

Table 3 Experimental environment

Category	Parameters
System	Centos 7.5 64
CPU	Intel Xeon Gold 5320 2.20GHz
Memory	256GB
Graphics	NVIDIA A100
Deep Learning Framework	Pytorch 2.1
Python version	3.9
Memory budget	2000
Batch size	64

Table 4 SGAN generator architecture

Layer Operation	Kernel	Strides	Dropout	Activation
1x1x100 Input(Latent Space) Dense				
Conv2DTranspose	4x4	2x2		ReLU
Conv2DTranspose	4x4	2x2		ReLU
Conv2DTranspose	4x4	2x2		ReLU

Table 5 SGAN discriminator architecture

Layer Operation	Kernel	Strides	Dropout	Activation
32x32x1 Input(Image)				
Conv2D	5x5	2x2	0.3	ReLU
Conv2D	3x3	2x2	0.3	ReLU
Conv2D	3x3	2x2	0.3	ReLU
Flatten				
Dense				Softmax

4. Experiments and Analysis

4.1 Experimental Environment

The experiments use PyTorch as the deep learning framework, and the specific experimental environment is shown in Table 3. The operating system used for the experiment is Centos7 64-bit, the CPU is Intel Xeon Gold 5320 2.20GHz, the RAM is 256GB, and the development environment is Python 3.9. the experimental batch size is 64, and the memory budget K is set to 2000. at the beginning of training, the learning rate is set to 0.1.

In Table 4, the generator starts with a 100 dimensional random noise by input, then in the Dense layer, it is mapped into $4 \times 4 \times 128 = 2048$ neurons, then Reshape is used to reshape this 2048 dimensional vector into a 4×4 feature map with a depth of 128, Next are three transposed convolutional layers with a convolutional kernel size of 4×4 and a step size of 2, no padding is applied The activation functions are all ReLU functions, and finally the Tanh activation function is applied to limit the output values to between -1 and 1.

As shown in Table 5, for the discriminator, assuming that the size of the input image is $32 \times 32 \times 1$, a single channel image, it first passes through three convolutional layers with convolutional kernel sizes of 5×5, 3×3, and 3×3, all with a step size of 2×2, and at the same time, it applies Dropout to randomly discard a certain proportion of the input units, and the proportion of the Dropout is 0.3, and then it utilizes the Flatten layer, which converts the data into a one-dimensional variables, and passed to the fully connected layer, according to the different experiments, the output nodes connected by the fully connected layer are also different, for example, in the ISCX classification experiment, there are a total of 12 categories, but we also need to classify the true and false samples, so the final output of the category is 13.

4.2 Results

The experiment is a multi-classification problem, and in order to measure the performance of the experiment, the accuracy, precision, and recall criteria are used as evaluation metrics. In the validation process, we designed three experiments to evaluate the accuracy of the 2D image models obtained after data preprocessing. These experiments focus on the effect of image size, image composition, and different network structures in the traffic model on the model performance, respectively.

The first is the effect of image size on model evaluation, as shown in Table 6; we investigate the different classification effects of the model when different sizes of image size in ISCX, WIDE, and USTC datasets, respectively. For the

Table 6 Comparison of different image sizes

	ISCX VPN-NoVPN			WIDE			USTC-TFC2016		
Image Size	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
32x32	92.26	91.35	89.02	90.78	85.90	84.12	93.86	89.46	90.07
64x64	91.68	90.22	87.35	84.39	79.67	75.19	87.99	85.65	84.15
128x128	86.01	88.54	85.37	76.88	78.65	74.03	85.81	82.51	84.11

Table 7 Comparison of different image compositions

	ISCX VPN-NoVPN			WIDE			USTC-TFC2016		
Image composition	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
Ethernet header + IP header + TCP/UDP header	90.77	91.48	87.64	90.86	91.93	87.92	94.82	92.93	90.41
IP header + TCP/UDP header	87.89	84.11	81.50	87.97	86.37	85.81	95.96	92.42	89.04
Ethernet header + TCP/UDP header	83.25	81.04	80.66	84.51	83.21	83.09	89.82	85.91	87.77

Table 8 Comparison of different network structures

	ISCX VPN-NoVPN			WIDE			USTC-TFC2016		
Network Structures	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
Ours	91.46	90.10	88.32	89.69	88.21	85.25	93.86	91.27	90.01
2D CNN	86.55	82.18	81.97	85.57	83.33	81.47	83.95	80.50	82.23
LeNet-5	83.67	82.99	81.01	82.14	80.81	78.34	86.72	86.09	84.80
GAN	86.07	82.57	80.29	86.65	85.90	84.89	85.68	83.01	82.46
RNN	80.79	83.65	85.51	79.47	75.83	76.24	76.92	71.44	70.47
Random Forest	87.91	86.12	81.80	86.02	84.16	81.59	86.64	83.09	84.98
KNN	73.26	70.29	68.01	69.87	68.95	68.66	74.54	72.29	70.35

discriminator, it is sufficient to modify the corresponding input size. In contrast, in the generator, if the image size is 64×64, we only need to add a transpose convolution layer after the last transpose convolution to enlarge the size of the feature map further. Similarly, for the size of 128×128, two transpose convolution layers need to be added. When the image size is set to 32×32 size, in ISCX dataset, the model's accuracy rate reaches the highest 92.26%, while the precision rate reaches 91.35%. Similarly, in the WIDE dataset, the accuracy and precision of the model are 90.78% and 85.90% respectively, and in the USTC dataset, the accuracy and precision are 93.86% and 89.46% respectively. Thus, we analyze that it is not true that the bigger the image is, the higher the accuracy and precision, and the size 32×32 is the most suitable. In the subsequent experiments, we decided to adopt this image size as the standard for model training.

Second, we investigate the effect of different header information on the model performance. Specifically, as shown in Fig. 1, a packet is composed of an Ethernet header, IP header, TCP/UDP header, and packet load. In actual network structure, there is a large amount of unencrypted and encrypted traffic, and for the consideration of privacy and other issues, we eliminate the extraction of payload. The main fields of Ethernet header are Ethernet type, frame checksum sequence, IP header has header length, service type, time to live, header checksum, TCP/UDP header has sequence number, acknowledgement number, control flag, window size and other fields. The roles of different headers are different, and thus the impact on them in model training should also be different. From some previous studies, more than just a single header to characterize the features obtained from model training is needed. We want to combine more than two headers to verify the impact of different headers. In Table 7, different header information has a significant effect on the accuracy and precision of the model. In the ISCX dataset, the accuracy rate obtained by using Ethernet header, IP header, and TCP/UDP header reaches 90.77%, and this result indicates that the header information of packets, including Ethernet header, IP header, and TCP/UDP header, is crucial for traffic classification because it contains key information such as the packet length, protocol type, and version number, whereas the other types are missing some important key information, resulting in a decrease in accuracy, and thus, in the subsequent experiments, we choose Ethernet header, IP header, and TCP/UDP header as the important components that make up the 2D image.

The model performance of the three datasets with different network structures is shown in Table 8. Firstly, we find that our proposed method has the highest performance in terms of accuracy, precision, and recall compared to the

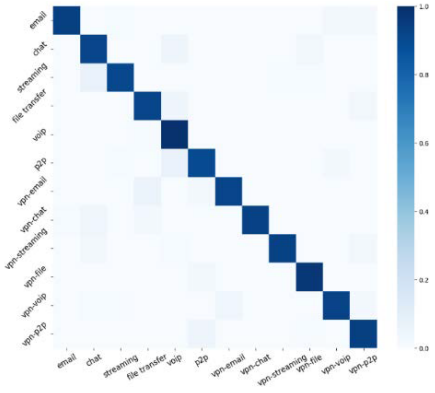

Fig. 5 ISCX VPN-nonVPN confusion matrix

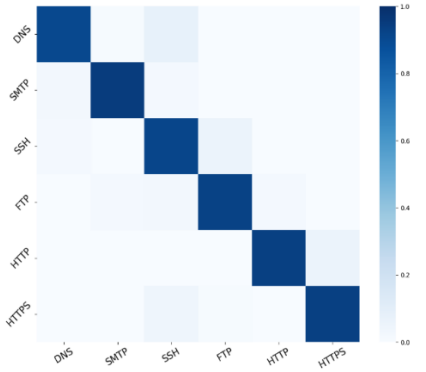

Fig. 6 WIDE confusion matrix

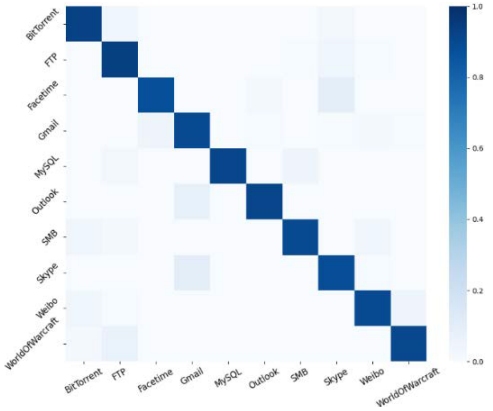

Fig. 7 USTC-TF2016 confusion matrix

Table 9 Effect of λ and β on the incremental learning performance

	λ	β
range	Accuracy	Accuracy
1.0	71.40	74.85
0.7	77.68	81.06
0.5	80.55	78.41
0.3	76.31	73.33
0.1	73.89	68.08
0.0	71.81	61.47

other methods, and the performance slightly decreases in the WIDE dataset, which may be due to the fact that the differences among the categories in the WIDE dataset are small, and thus the classification performance is a bit poorer than that of the other datasets. In the USTC dataset, our accuracy and precision rates reach 93.86% and 91.27%, which are 7.14% higher than the other methods, thus also verifying our assumption that the data enhancement of samples using the Mixup technique is obviously able to improve the model performance to a certain extent than without it. Similarly, the accuracy of the model in the WIDE dataset and the precision rate are also the highest, reaching 89.69% and 88.21%.

The confusion matrix is an important method for evaluating the performance of a model, which provides a detailed demonstration of the model's ability to classify different categories. In Figs. 5-7, we show the results of the confusion matrix visualization on the ISCX, WIDE, and USTC datasets. From this confusion matrix, we can observe that the classification accuracy of most of the categories is more than 90%, the classification of the categories of VPN and no-VPN in the ISCX dataset is also very accurate, and a small number of category samples, such as the samples of files transfer and VPN-voip, are lower than the others, which may be due to the fact that the number of samples is too small, and the model is able to learn to the ability to be worse This may be due to the small number of samples and the poorer ability of the model to learn them. Nevertheless, the overall results of the confusion matrix demonstrate the effectiveness of our model in classifying encrypted traffic.

In the class incremental learning process, parameter λ controls the balance between the distillation loss and the cross-entropy loss, and the value ranges from 0-1. In the following experiments, we carry out incremental learning in the ISCX dataset with a 2-category classification and choose the final accuracy of the model as the reference value, and at the same time, we change the value of the parameter A with an incremental increase of 0.2 in order to observe the model's accuracy in the different values of the scenarios.

In Table 9, the left side is the incremental learning process on the ISCX dataset in the case of 2 categorizations, and we adjust different values of λ based on the fixed β value of 0.7, and the final incremental learning accuracy of the model is adjusted with the change of the value. Since λ is an important parameter for adjusting distillation loss and cross-entropy loss, the final accuracy of the model changes as the value of λ changes, and when λ is set to 1 and 0, the model is in distillation loss and cross-entropy loss, respectively, and at this time the accuracy of the model is at its lowest, and thus these two values are not an appropriate choice, and based on the results of the experiments, we finally set its value to 0.5.

On the other hand, on the right side of Table 9, we adjust the value of β while fixing the value of λ as 0.5, and the model will gradually show different situations because β is used to balance the old and new data and thus the adjustment of the balance number is beneficial for the model to improve the accuracy in the case of relatively few samples of the old

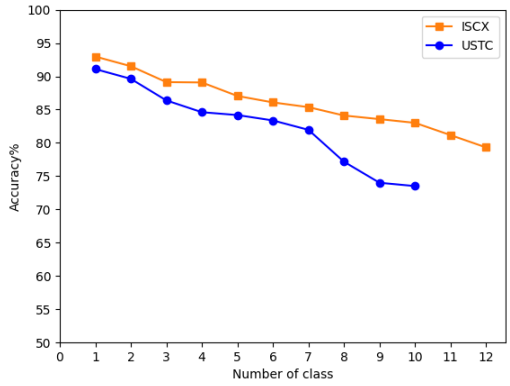

Fig. 8 Incremental learning with step size 1

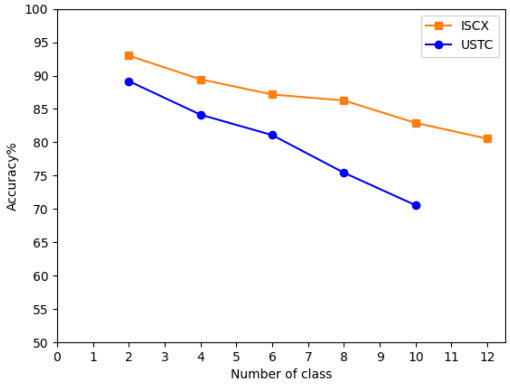

Fig. 9 Incremental learning with step size 2

data, and from the experiment, we can also see that, through the adjustment of the different, the model's accuracy varies greatly, and after the experimental comparison, we found that the accuracy of the model is the highest when set at 0.7, and thus finally set the β value to 0.7.

4.3 Class Incremental Experimental Results

There are 12 different classes in the ISCX VPN-nonVPN dataset and ten different classes in the USTC dataset. This provides a test scenario for us to conduct class incremental learning experiments. In the following experiments, we used different step sizes (1, 2, 4, 6) to incrementally increase the number of classes that the model needs to learn, where step size means the number of classes incremented in each round. With this approach, we were able to observe how the accuracy of the training changed as the number of categories increased. At the beginning of our experiments, we first perform data augmentation on the ISCX and USTC datasets using the Mixup method, a step designed to enhance the model's generalization ability. Subsequently, during class incremental learning, we noticed that even though new classes were continuously added to the model, old knowledge was still retained, effectively avoiding the problem of catastrophic forgetting. The experimental results show that in the ISCX dataset, the accuracy decreases at a slower rate when the step size is set to 1. This may be due to the fact that only a small amount of changes are made to the old knowledge, thus better-preserving memorability. Surprisingly, the final accuracy of the model is the highest when the step size is increased to 2. In the USTC dataset, we found that the final accuracy of the model is not as good as the ISCX dataset when the step size is 1 and 2. This may be due to the fact that the sample size of the USTC dataset is too small, which leads to underfitting during the final model training process, and the experimental results are shown in Figs. 8-11.

In order to better evaluate the model against other different methods, in Figs. 12 and 14, we have done comparative experiments in the ISCX and USTC datasets with the traditional iCaRL method, Lwf method [22], and fine-tuning method, respectively, and the experiments have proved that our method also outperforms other traditional methods. The

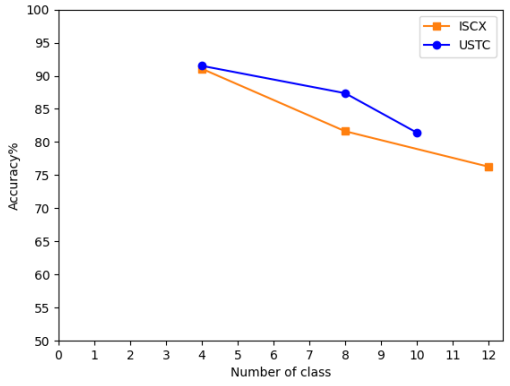

Fig. 10 Incremental learning with step size 4

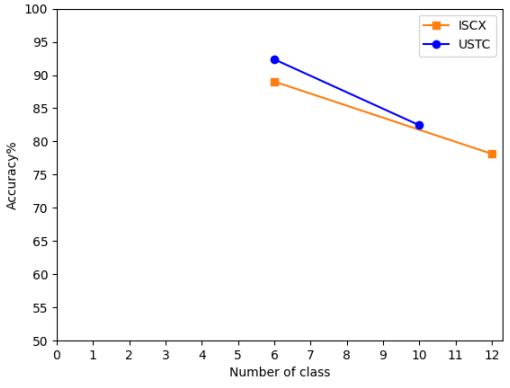

Fig. 11 Incremental learning with step size 6

final results of the experiments also verify the excellence of our proposed method.

However, since the ISCX dataset has only 12 categories and the USTC dataset has only 10 categories (20 categories even when malicious traffic is added), which do not give a good indication of testing in incremental scenarios with more categories, we use the set of web site fingerprints provided by [30], which contains 95 sites and thus can be interpreted, from a categorization point of view, as 95 categories. There are 1000 tracks for each site, and we perform incremental learning according to 10 categories per batch, adding five categories the last time. Figure 13 shows the experimental results. Our proposed method is able to get the highest

Table 10 Experimental results with ISCX VPN-No VPN, USTC-TFC2016, ISCX Tor-Non Tor

Scenario	Datasets	Classes	Rate	Accuracy
A	ISCX VPN-No VPN	Email,facebook_video,Facebookchat,Gmailchat,hangouts_audio,Netflix,scpdown,scpup,skype_audio,skype_file,skype_video,voipbuster,youtube,vpn_aim_chat,vpn_email,vpn_facebook_chat,vpn_ftps,vpn_hangouts_audio,vpn_sftp,vpn_skype_audio,vpn_skype_files,vpn_vimeo,vpn_voipbuster,vpn_youtube	25%	93.45%
			30%	94.39%
			35%	89.81%
B	USTC-TFC2016	BitTorrent,Facetime,Ftp,Gmail,Mysql,Outlook,Skype,SMB,Weibo,WorldOfWarcraft,Cridex,Geodo,Htbot,Miuref,Neris,Nsis-ay,Shifu,Tinba,Virut,Zeus	25%	90.24%
			30%	88.09%
			35%	87.05%
C	ISCX Tor-Non Tor	Audio_tor_spotify,Browsing_tor,Chat_tor_aim,Chat_tor_facebook,Chat_tor_skype,File_tor_skype,Tor_vimeo,Video_Tor_youtube,Aim_Chat,Browsing,Facebook_Chat,Ftp_filetransfer,Icq_Chat,P2P,Spotify,SSL,Workstation,Youtube	25%	92.77%
			30%	95.36%
			35%	92.41%

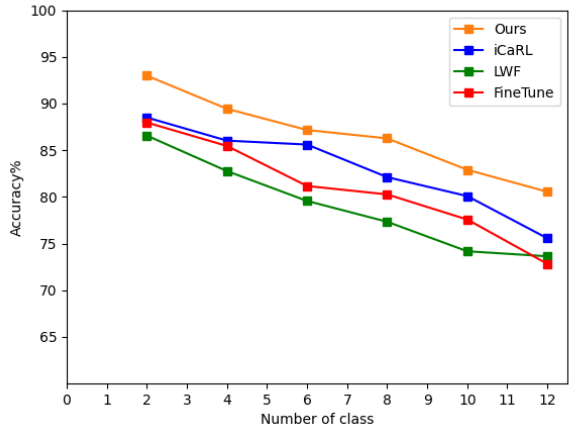

Fig. 12 Comparison of the accuracy of different classes of incremental learning methods on the ISCX dataset with a learning step of 2

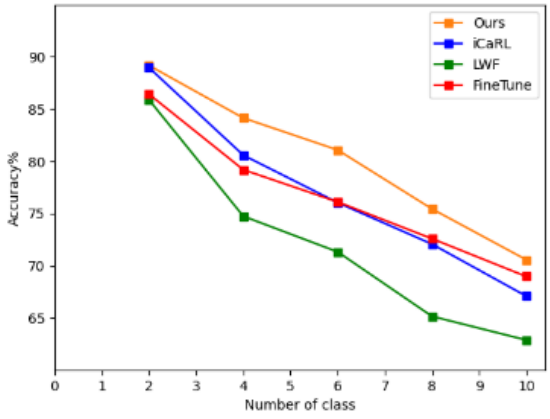

Fig. 14 Comparison of the accuracy of different classes of incremental learning methods on the USTC dataset with a learning step of 2

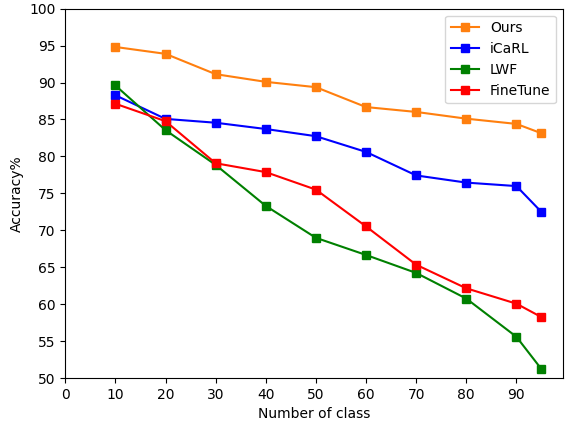

Fig. 13 Comparison of the accuracy of different classes of incremental learning methods on the fingerprint dataset with a learning step of 10

results compared to other methods, and the accuracy of the final model reaches 83.13%.

In order to validate the incremental experiments further, we conducted three scenarios to simulate experiments in natural environments based on ISCX VPN-No VPN, USTC-TFC2016, and ISCX Tor-NonTor [35], respectively. Taking Scenario A as an example, we extracted the data from the ISCX VPN-No VPN dataset for a more segmented category categorization and took the Email category, for example, it consists of four pcap files email1a, email1b, email2a, email2b, with a total of 24 categories, in which a portion of the categories are randomly selected as the training set for the model. The rest of the categories are not involved in the training and are used as the categories for the incremental experiments. We set up three comparison experiments, where the Rate represents the ratio of the remaining categories to all the categories at the beginning of training. At the same time, the step size of each category growth is 2, and the default memory budget K is modified to be set to 5000 to increase the number of stored exemplars.

In the USTC-TFC2016 dataset, we merge the datasets of benign and malicious categories for training, and there are 20 categories. Similarly, we randomly select a portion of the categories as the model training set and set the Rate to represent the ratio of the remaining categories to all the categories. We also utilize the ISCX Tor-NonTor dataset, which collects both Tor and NonTor data, and we extract part of the content for categorization. Moreover, there are a total of 18 classes. Similarly, we randomly select a portion of the classes as the model training set and set the Rate to represent the ratio of the remaining classes to all the classes.

In Table 10, we show the model's final accuracy. We can see that the accuracy of the model significantly improved when the exemplars was increased. In the case of ISCX VPN-No VPN, in which the rate reaches 30%, the final accuracy of the model reaches 94.39%, and we think that, in the case of increasing the exemplars. In contrast, the model can

retain more knowledge of the old categories; the accuracy will improve; in addition, if we use more detailed category increments this time, the model can still maintain a high accuracy. For the USTC-TFC2016 dataset, we combine the benign and malicious categories, and the model improves compared to the previous experiments in the case of enlarging the exemplars, and the final accuracy of the model reaches 90.24% in the case of a ratio of 25%. In the ISCX Tor-Non Tor dataset, our accuracy reaches up to 95.36%.

5. Conclusion

In this paper, we propose a method for network traffic classification. Firstly, we use Mixup to enhance the data of the samples so as to extend the model generalization ability, and then we improve the class incremental learning method, which solves the problem of the imbalance between the old and the new class data in the process of incremental learning, and we use three datasets, namely, ISCX, WIDE, and USTC, to test the performance of the model, and the experiments prove that our proposed method is able to reach up to 83.13% under incremental learning due to other methods in the market. To further simulate the incremental scenarios in the natural environment, we also designed three scenario experiments, and finally, we obtained 94.39%, 90.24%, and 95.36% accuracies in three dataset scenarios of ISCX VPN-NoVPN, USTC-TFC2016, and ISCX Tor-NoTor respectively. In our future work, we plan to further extend the dataset to be able to perform continuous learning in more complex network environments, and we will also tune and optimize the generative adversarial network to obtain better performance.

Acknowledgments

This research was supported in by the Guizhou Provincial Department of Education 2024 Natural Science Research Program for Youth Science and Technology (Qian Education Technology 2024 No. 237, Qian Education Technology 2022 No. 385), the Natural Science Foundation of Guizhou Provincial Department of Education (No.2018.439), Research on Malware Detection Technology under Cloud Collaboration by the Natural Science Foundation of Qiannan Normal University for Nationalities (qnsy2018021), Education Quality Enhancement Project Fund of Qiannan Normal University for Nationalities (2019xjg0203), Research Project on Political and Ideological Education at the School Level of Qiannan Normal University for Nationalities (qnsysz202302).

Author Contributions: Guangjin Ouyang: Data curation, Methodology, Software, Funding acquisition, Writing – original draft. Yong Guo: Conceptualization, Formal analysis, Funding acquisition. Yu Lu: Investigation, Methodology, Writing – review & editing. Fang He: Formal analysis, review & editing.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Q. Wang, W. Li, H. Bao, Z. Tang, W. Wang, F. Liu, and L. Ying, "High-Efficient and Few-shot Adaptive Encrypted Traffic Classification with Deep Tree," IEEE Military Communications Conference, Rockville, MD, USA, pp.458–463, 2022.

[2] A.R. Khesal and M. Teimouri, "The Effect of Network Environment on Traffic Classification," International Conference on Computer and Knowledge Engineering, Wuhan, China, pp.059–064, 2022.

[3] Y. Dhote, S. Agrawal, and A.J. Deen, "A survey on feature selection techniques for internet traffic classification," International Conference on Computational Intelligence and Communication Networks, Jabalpur, India, pp.1375–1380, 2015.

[4] Z. Chen, K. He, J. Li, and Y. Geng, "Seq2img: A sequence-to-image based approach towards ip traffic classification using convolutional neural networks," IEEE International conference on big data, Boston, Ma, USA, pp.1271–1276, 2017.

[5] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," International conference on information networking, Da Nang, Vietnam, pp.712–717, 2017.

[6] H. Dahmouni, S. Vaton, and D. Rossé, "A markovian signature-based approach to IP traffic classification," Proc. 3rd annual ACM workshop on Mining network data, San Diego, California, USA, pp.29–34, 2007.

[7] C. Thay, V. Visoottiviseth, and S. Mongkolluksamee, "P2P traffic classification for residential network," International Computer Science and Engineering Conference, Chiang Mai, Thailand, pp.1–6, 2015.

[8] T. Bujlow, V. Carela-Español, and P. Barlet-Ros, "Independent comparison of popular DPI tools for traffic classification," Computer Networks, vol.76, pp.75–89, Jan. 2015. doi: 10.1016/j.comnet.2014.11.001.

[9] P. Khandait, N. Hubballi, and B. Mazumdar, "Efficient keyword matching for deep packet inspection based network traffic classification," International Conference on COMmunication Systems & NETworkS, Online, pp.567–570, 2020.

[10] V.F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic," IEEE European Symposium on Security and Privacy, Saarbrucken, Germany, pp.439–454, 2016.

[11] N. Hubballi and M. Swarnkar, "*Bitcoding*: Network traffic classification through encoded bit level signatures," IEEE/ACM Trans. Netw., vol.26, no.5, pp.2334–2346, Oct. 2018. doi: 10.1109/TNET.2018.2868816.

[12] Z. Wu, Y.-N. Dong, X. Qiu, and J. Jin, "Online multimedia traffic classification from the QoS perspective using deep learning," Computer Networks, vol.204, p.108716, Feb. 2022. doi: 10.1016/j.comnet.2021.108716.

[13] G. Aceto, D. Ciuonzo, A. Montieri, A. Nascita, and A. Pescapé, "Encrypted multitask traffic classification via multimodal deep learning," IEEE International Conference on Communications, Online, pp.1–6, 2021.

[14] A. Rasteh, F. Delpech, C. Aguilar-Melchor, R. Zimmer, S.B. Shouraki, and T. Masquelier, "Encrypted internet traffic classification using a supervised spiking neural network," Neurocomputing, vol.503, pp.272–282, 2022. doi: 10.1016/j.neucom.2022.06.055

[15] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," IEEE international conference on intelligence and security informatics, Beijing, China, pp.43–48, 2017.

[16] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for Internet of Things," IEEE access, vol.5, pp.18042–18050, 2017. doi: 10.1109/ACCESS.2017.2747560.

[17] H. He, Y. Lai, Y. Wang, S. Le, and Z. Zhao, "A data skew-based unknown traffic classification approach for TLS applications," Future Generation Computer Systems, vol.138, pp.1–12, 2023. doi: 10.1016/j.future.2022.08.003.
[18] H. Zhang, M. Cisse, Y.N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," arxiv preprint arXiv.1710.09412, 2017. doi: 10.48550/arXiv.1710.09412.
[19] W. Wei, H. Gu, W. Deng, Z. Xiao, and X. Ren, "ABL-TC: A lightweight design for network traffic classification empowered by deep learning," Neurocomputing, vol.489, pp.333–344, 2022. doi: 10.1016/j.neucom.2022.03.007.
[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, and Y. Bengio, "Generative adversarial nets," Advances in Neural Information Processing Systems, Montréal, Canada, 2014, vol.27.
[21] G. Draper-Gil, A.H. Lashkari, M.S.I. Mamun, and A.A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," Proc. 2nd international conference on information systems security and privacy, Rome, Italy, vol.1, pp.407–414, 2016.
[22] Z. Li and D. Hoiem, "Learning without forgetting," IEEE Trans. Pattern Anal. Mach. Intell., vol.40, no.12, pp.2935–2947, 2018. doi: 10.1109/TPAMI.2017.2773081.
[23] S. Hou, X. Pan, C.C. Loy, Z. Wang, and D. Lin, "Lifelong learning via progressive distillation and retrospection," Proc. European Conference on Computer Vision, Munich, Germany, pp.437–452, 2018.
[24] S. Soleymanpour, H. Sadr, and M. Nazari Soleimandarabi, "CSCNN: cost-sensitive convolutional neural network for encrypted traffic classification," Neural Processing Letters, vol.53, no.5, pp.3497–3523, 2021. doi: 10.1007/s11063-021-10534-6.
[25] Y. Li, X. Chen, W. Tang, Y. Zhu, Z. Han, and Y. Yue, "Interaction matters: Encrypted traffic classification via status-based interactive behavior graph," Applied Soft Computing, vol.155, p.111423, 2024. doi: 10.1016/j.asoc.2024.111423
[26] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," IEEE conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, pp.7765–7773, 2018.
[27] J. Serra, D. Suris, M. Miron, and Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," International Conference on Machine Learning, New York, USA, pp.4548–4557, 2018.
[28] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C.H. Lampert, "icarl: Incremental classifier and representation learning," IEEE conference on Computer Vision and Pattern Recognition, Hawaii, USA, pp.2001–2010, 2017.
[29] M. McCloskey and N.J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," Psychology of learning and motivation, vol.24, pp.109–165, 1989. doi: 10.1016/S0079-7421(08)60536-8.
[30] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," ACM SIGSAC Conference on Computer and Communications Security, Toronto, Canada, pp.1928–1943, 2018.
[31] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Z. Zhang, and Y. Fu, "Incremental classifier learning with generative adversarial networks," arxiv preprint arXiv.1802.00853, 2018. doi: 10.48550/arXiv.1802.00853.
[32] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," IEEE/CVF conference on computer vision and pattern recognition, Long Beach, USA, pp.374–382, 2019.
[33] A. Odena, "Semi-supervised learning with generative adversarial networks," arxiv preprint arXiv.1606.0158, 2016. doi: 10.48550/arXiv.1606.0158.
[34] K. Cho, K. Mitsuya, and A. Kato, "Traffic data repository at the WIDE project," USENIX 2000 Annual Technical Conference: FREENIX Track, pp.263–270, June 2000.
[35] A. Habibi Lashkari, G. Draper Gil, M.S.I. Mamun, and A.A. Ghorbani, "Characterization of Tor Traffic Using Time Based Features," Proc. 3rd International Conference on Information System Security and Privacy, Porto, Portugal, vol.1, pp.253–262, 2017.

Guangjin Ouyang received the master's degree from Guizhou University, Guiyang, China, in 2018. He is currently an Associate Professorship with the School of Computer and Information, Qiannan Normal University for Nationalities. His research interests include network communications and deep learning.

Yong Guo received the bachelor's degree from Guizhou Normal University, Guiyang, China, in 1991. He is currently an professor with the School of Computer and Information, Qiannan Normal University for Nationalities. His research interests include Computer education, software theory, enterprise informatization.

Yu Lu received the bachelor's degree from Guizhou Normal University, Guiyang, China, in 2004. He is currently an Associate Professorship with the School of Computer and Information, Qiannan Normal University for Nationalities. His research interests include Cloud computing key technology, national culture digitization technology, network information system.

Fang He received the master's degree from Guizhou University, Guiyang, China, in 2019. He is currently an lecturer with the School of Computer and Information, Qiannan Normal University for Nationalities. His research interests include Computer technology applications, computer network technology, data mining analysis.