Advancing Malware Detection in Network Traffic with Self-Paced Class Incremental Learning

Xiaohu Xu, *Graduate Student Member, IEEE*, Xixi Zhang, *Graduate Student Member, IEEE*, Qianyun Zhang, *Senior Member, IEEE*, Yu Wang, *Member, IEEE*, Bamidele Adebisi, *Senior Member, IEEE*, Tomoaki Ohtsuki, *Senior Member, IEEE*, Hikmet Sari, *Life Fellow, IEEE*, and Guan Gui, *Fellow, IEEE*

***Abstract*—Ensuring network security, effective malware detection is of paramount importance. Traditional methods often struggle to accurately learn and process the characteristics of network traffic data, and must balance rapid processing with retaining memory for previously encountered malware categories as new ones emerge. To tackle these challenges, we propose a cutting-edge approach using self-paced class incremental learning (SPCIL). This method harnesses network traffic data for enhanced class incremental learning (CIL). A pivotal technique in deep learning, CIL facilitates the integration of new malware classes while preserving recognition of prior categories. The unique loss function in our SPCIL-driven malware detection combines sparse pairwise loss with sparse loss, striking an optimal balance between model simplicity and accuracy. Experimental results reveal that SPCIL proficiently identifies both existing and emerging malware classes, adeptly addressing catastrophic forgetting. In comparison to other incremental learning approaches, SPCIL stands out in performance and efficiency. It operates with a minimal model parameter count (8.35 million) and in increments of 2, 4, and 5, achieves impressive accuracy rates of 89.61%, 94.74%, and 97.21% respectively, underscoring its effectiveness and operational efficiency.**

***Index Terms*—Malware detection, deep learning, class-incremental learning, sparse pairwise loss, sparse loss.**

I. Introduction

The widespread implementation of fifth-generation (5G) wireless networks marks a major leap forward in high-speed data transmission and broader coverage. This significant progress is primarily attributed to the extensive deployment of numerous base stations, which has notably increased network capacity, enhanced the quality of service (QoS), and minimized latency, as detailed in recent studies [1]. However, with the rapid advancement of 5G technology comes a heightened risk, especially in terms of security concerns. As depicted in Fig. 1, these risks are predominantly associated with the security of communications between Internet of Things (IoT) devices and base stations (BSs), encompassing device-to-device (D2D) interactions. The vulnerability of these communication channels to intrusions presents a significant threat to user privacy and overall network security. For example, denial of service (DoS) attacks, which can gravely compromise the integrity of network servers such as web servers, represent a critical concern. In scenarios of malicious traffic breaches, these attacks could lead to server paralysis and a range of other serious repercussions [2].

Malware detection (MD) is a critical technology designed to identify and neutralize malicious software, thereby safeguarding computer systems and networks [3]. Malware encompasses a range of software types, including viruses, worms, Trojan horses, spyware, and adware, all of which are aimed at damaging, disrupting, or illicitly accessing computer systems. MD employs techniques like static and dynamic analysis for detecting such threats [4]. Static analysis involves a thorough examination of a software's binary or source code to pinpoint potential malicious patterns, while dynamic analysis tracks software behavior in real-time to spot signs of malign activity. Feature-based recognition, another key method in MD, entails scanning and extracting distinctive features from known malware and comparing these to unidentified software [5].

Recently, deep learning has been revolutionarily integrated into MD, enhancing aspects such as feature extraction, classification, execution process analysis, attack detection, and model transfer [6]–[8]. This technological evolution significantly boosts the precision and speed of malware detection, fortifying the security of computer systems and networks. In the realm of network security, particularly for internet-connected systems, the deployment of intrusion detection system (IDS) is essential. IDS plays a pivotal role in identifying malicious network traffic, as cited in recent studies [9]–[11]. Within these systems, MD is vital for accurately categorizing network traffic, which is crucial for not only enhancing QoS but also for detecting malicious exploitation of network resources [12]–[14]. Given the continuous influx of vast amounts of new network traffic, coupled with the need for the precise identification of existing tasks, the adoption of class incremental learning (CIL) [15], [16] has become increasingly important. CIL adeptly meets the challenge of effectively detecting and categorizing this ever-expanding traffic volume.

Neural network-based methodologies frequently encounter a significant obstacle known as catastrophic forgetting, a

This work was supported in part by the Key Project of Natural Science Foundation of the Higher Education Institutions of Jiangsu Province under Grant 22KJA510002. (*Corresponding authors: Yu Wang, Guan Gui*)

Xiaohu Xu, Xixi Zhang, Yu Wang, Hikmet Sari, and Guan Gui are with the College of Telecommunications and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China (e-mail: 1221014133@njupt.edu.cn, 2022010205@njupt.edu.cn, yuwang@njupt.edu.cn, hsari@ieee.org, guiguan@njupt.edu.cn).

Qianyun Zhang is with the School of Cyber Science and Technology, Beihang University, Beijing 100191, China (e-mail: zhangqianyun@buaa.edu.cn)

Bamidele Adebisi is with the Department of Engineering, Faculty of Science and Engineering, Manchester Metropolitan University, Manchester M1 5GD, United Kingdom (e-mail: b.adebisi@mmu.ac.uk).

Tomoaki Ohtsuki is with the Department of Information and Computer Science, Keio University, Yokohama, Japan (e-mail: ohtsuki@keio.jp)

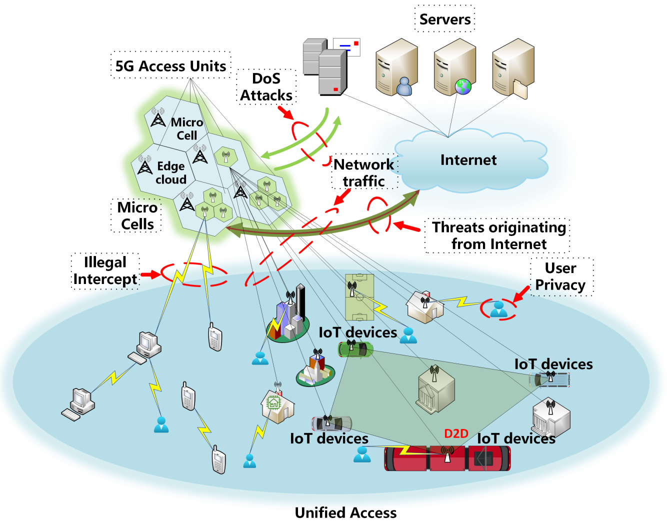


Fig. 1. Illustration of 5G networks and its potential threat landscape.

challenge particularly pronounced in incremental learning scenarios [17]–[19]. These models often suffer a notable decline in performance on tasks they have previously mastered as they acquire new information. This issue underscores a critical dilemma in neural networks known as the stability-plasticity dilemma [16]. Here, plasticity denotes the capacity of the network to assimilate new data, while stability pertains to its ability to retain existing knowledge during the learning process [20].

In this paper, we present a novel approach to resolving the stability-plasticity dilemma by developing the Aggregated Network, an intricate dual-layer architecture. Utilizing ResNet as the foundational model, our design integrates two unique residual blocks at each layer of the network. The first block is devoted to stability, focusing on the preservation of previously acquired knowledge, while the second block, tailored for plasticity, concentrates on assimilating new information. The stability block features a select number of adjustable parameters, specifically designed to bolster knowledge retention. In contrast, the plasticity block uses aggregated weights to alter the outputs of feature maps. These modified outputs are then merged and relayed to the next layer [21]. This architectural design skillfully maintains a dynamic balance between stability and plasticity, by automatically adjusting aggregation weights through the end-to-end optimization of these weights as hyperparameters during the training process [22].

Moreover, we address two additional challenges intrinsic to our dual-layer architecture: the increase in model parameters and the potential decrease in accuracy due to limited data samples. To counteract the former, we have incorporated a sparse auxiliary loss in the training of the plasticity block, effectively diminishing the overall weight footprint of the network. In scenarios characterized by limited data availability, we introduce an adaptive sparse pairwise loss, a cutting-edge auxiliary approach crafted to optimize model performance under such limitations. Our methodology has been thoroughly vetted using network traffic data. The results, including comprehensive ablation studies, unequivocally demonstrate the superiority of our approach compared to current leading-edge methods in balancing stability and plasticity within the scope of incremental learning. The primary contributions of our study are outlined as follows:

- We derive an innovative dual-branch network model specifically designed to achieve an effective balance between stability and plasticity in class incremental learning.
- Our research introduces a breakthrough loss function that artfully blends adaptive sparse pairwise loss with sparse loss. This innovation is critical for efficiently managing sparse model parameters while skillfully learning new classes. Its effectiveness is particularly pronounced in contexts characterized by limited sample sizes and model pruning, where it demonstrates the ability to distill compact yet powerful features.
- We conduct extensive experimental validation, employing a range of adjustable parameters within our loss function to arrive at optimal solutions.

The paper is structured as follows: Section II introduces related MD methods. Section III details problem formulation. Section IV describes our proposed CIL-based MD methods. Section V presents a range of simulation results that illustrate the performance of our method. Finally, Section VI concludes the paper.

II. Related Work

In this section, we provide an overview of traditional MD methods, deep learning-based MD methods, and the application of incremental learning and CIL approaches for malware traffic classification.

A. *Traditional MD methods*

In earlier studies, various techniques for classifying malware traffic have been introduced. For instance, H. Dreger *et al.* [23] presented a port-based approach, which offers the advantage of simplicity and speed in implementation but falls short in terms of detection accuracy. Other researchers, as seen in [24] and [25], proposed the use of data packet inspection (DPI) methods to enhance detection accuracy, albeit limited to unencrypted data. However, with the increasing prevalence of encrypted network traffic and the growing emphasis on security, DPI methods have become less applicable. V. Paxson *et al.* [26] put forward a solution based on static flow analysis, which is versatile and can be applied to both encrypted and unencrypted data. Nevertheless, a notable limitation of this approach is the need for manual design of data features. The escalating complexity of modern networks and the relatively low accuracy of manually designed features make this method impractical for real-world scenarios.

B. *DL-based MD methods*

In recent years, deep learning has witnessed widespread adoption across diverse domains [27]–[34], and it has also made significant inroads in the realm of network security

[35]. Researchers have delved into the domain of malware traffic classification, exploring methods based on deep learning [36]–[38]. The convolutional neural network (CNN) approach [39]–[41] holds a distinct advantage in its ability to learn traffic characteristics directly from raw data, enabling accurate classification [42]–[44]. Nonetheless, CNN-based methods come with demanding requirements for extensive datasets, necessitating a substantial amount of training data to effectively train the network. Furthermore, CNNs are typically tailored to specific tasks, making them less adaptable to handling both new and existing tasks concurrently. In practice, obtaining large-scale, and up-to-date network traffic data is a challenging task, as network traffic is subject to constant change, with malicious traffic patterns evolving daily.

C. CIL-based MD methods

There exists a significant technology gap in the development of an adaptive traffic detection method. Incremental learning (IL) [45] has emerged as an appealing solution to address this gap. IL can be incorporated into various applications in numerous ways [46]. It encompasses various types of IL methods, categorized into three primary paradigms: regularization, replay, and parameter isolation [15]. The replay method involves the storage of samples in their original format or the generation of pseudo samples using a generative model. During the process of learning new tasks to mitigate forgetfulness, previous task samples are replayed. These samples can be either reused as model input for practice or constrained to optimize the loss of new tasks to prevent interference from past tasks. Regularization-based methods enhance the retention of prior knowledge when learning new data by introducing additional regularization terms in the loss function. This approach can be further subdivided into a data-centric and a priori-centric method. A notable advantage of this approach is the avoidance of storing original input, a focus on privacy, and a reduction in memory requirements. Data-focused methods [45], [47] and prior-focused methods [48] represent further variations within this paradigm. Parameter isolation methods [49] entails the specification of distinct model parameters for each task. The advantage of this approach is its ability to prevent potential forgetting. However, a drawback is the reliance on a task oracle, and it may not be suitable for handling shared tasks.

A conventional benchmark technique known as knowledge distillation using a transfer set [50], initially applied to incremental learning by Li *et al.* [45], has been instrumental. Building upon this foundation, Yan *et al.* [51] incorporated representation learning and employed a limited set of herding exemplars to store and replay prior knowledge. The herding technique involves selecting the nearest neighbors of the average sample per class. Subsequently, using the same herding exemplars, Xu *et al.* [17] explored balanced fine-tuning and temporary distillation to construct an end-to-end framework, and introduced multiple techniques to balance classifiers.

Current CIL methodologies aim to tackle the challenge of retaining knowledge of previous classes while accommodating the assimilation of new classes as the number of categories grows. Nevertheless, these approaches typically necessitate a considerable volume of samples. Moreover, as the number of increments increases, the number of model parameters also increases, which poses a drawback for efficient model deployment. The proposed SPCIL method adeptly meets the dual objectives of acquiring new class knowledge without erasing the memory of the old classes, while simultaneously ensuring that the growth model parameters is not affected by the increasing number of increments. Furthermore, our proposed approach is designed to achieve optimal performance levels.

III. Scenario Description and Problem Formulation

A. Scenario Description

CIL refers to the process of gradually learning new classes or patterns on top of an existing model, rather than retraining the entire model [52]. This learning approach is particularly useful for handling data streams, updating models in dynamic environments, or situations with limited resources. CIL enables the model to acquire knowledge about new classes without forgetting the previously learned knowledge.

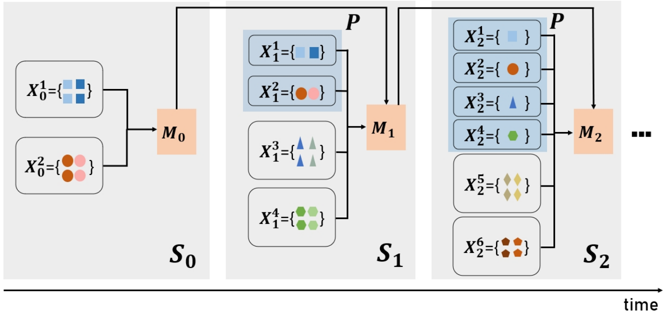


Fig. 2. Scenario description for class incremental learning.

CIL typically involves $n+1$ learning phases in total. This comprises an initial phase and n incremental phases, where the number of classes gradually expands. We present a formalized framework for CIL in Fig. 2. Starting from an initial non-incremental state S_0, a model M_0 is trained from scratch on a dataset $\mathcal{D}_0 = \{(\mathbf{x}_0^i, y_0^i); i = 1, 2, ..., K_0\}$, where $\mathbf{x}_0^i$ and y_0^i represent the set of data and labels for the i-th class in S_0, respectively, and $N_0 = K_0$ is the number of classes in the first non-incremental state. In each incremental state S_t, a new batch of K_t classes is introduced, and the goal is to train a model M_t capable of recognizing $N_t = K_0 + K_1 + \ldots + K_t$ classes. This model is trained using the previous state model M_{t-1} on a dataset $\mathcal{D}_t = \{(\mathbf{x}_t^i, y_t^i); i = 1, 2, ..., K_t\} \cup \mathcal{P}$. It is important to note that while all data from the new K_t classes are available, only a bounded exemplar subset $\mathcal{P}$ of data from the $N_{t-1} = K_0 + K_1 + ... + K_{t-1}$ past classes is utilized. An inherent imbalance favoring new classes emerges and grows across incremental states due to the constrained memory $\mathcal{P}$, which needs to be allocated to an increasing number of past classes with each iteration. Following this inference, the process continues until the final stage of incremental

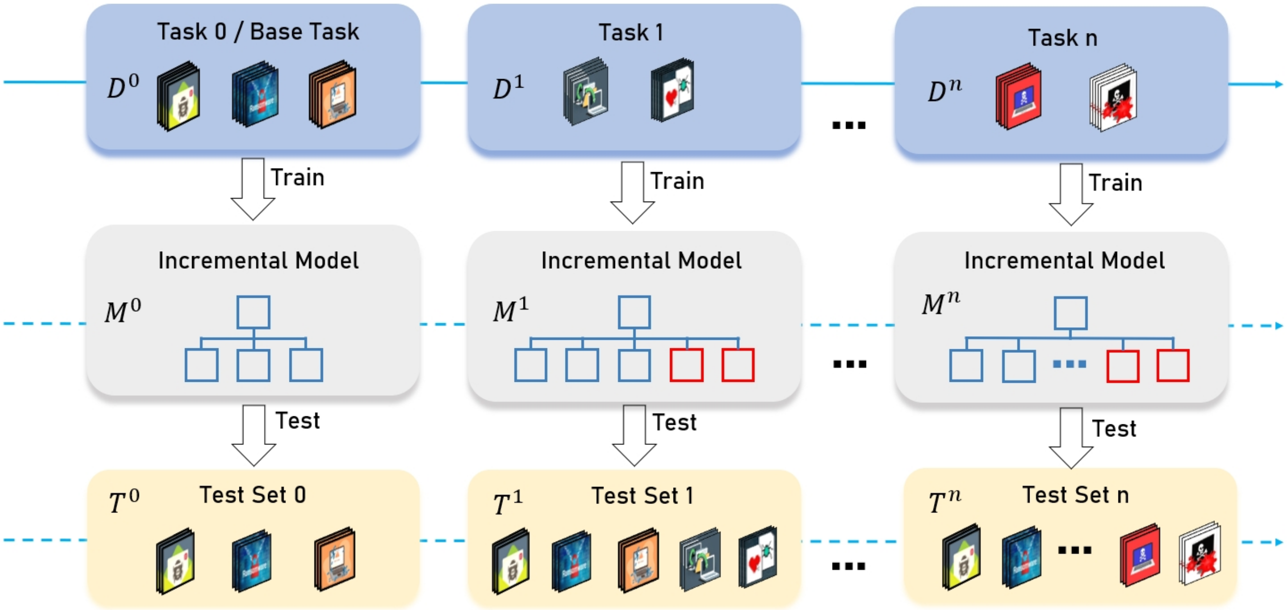


Fig. 3. Problem description for CIL-based MD method.

learning, the n-th stage. At this stage, the trained model M_n becomes the ultimate incremental learning model, capable of recognizing new classes without forgetting the knowledge of the old classes.

B. Problem Formulation

The goal of CIL for MD is to continuously learn new categories from network traffic datasets while avoiding catastrophic forgetting issues. We divide the training set into M classes and K labeled instances, in the form of $\{(\mathbf{x}_i, y_i)\}_{i=1}^{K}$, where $y_i \in \mathcal{Y} = \{1, 2, \ldots, M\}$, represents the label of sample x_i. We define $f_\phi(\mathbf{x}) : \mathcal{X} \rightarrow \mathbb{R}^{M+N}$, a neural network parameterized by ϕ, where $\mathcal{X}$ is the input space of x, and N represents continuously added new classes.

The optimization objective of CIL typically encompasses two main aspects: maintaining stability in knowledge for old classes and achieving accuracy in learning new classes. The problem description for the CIL-based MD method is shown in Fig. 3. The formalization of this objective function $\mathcal{L}$ consists of two components:

1) Old Class Stability: This part aims to ensure that the model does not forget knowledge of old classes while learning new ones. Typically, this can be achieved by minimizing the loss of the model on samples from old classes. Let $\mathcal{L}_{\text{old}}$ represent the loss function for old classes, f^o represents the initially incremented model or the model saved in the previous incremental steps, $\mathcal{D}^o$ represents samples from old classes, and $\boldsymbol{\theta}$ denote the model parameters. The objective for this part can be expressed as:

$$\mathcal{L}_{\text{old}}(\boldsymbol{\theta}) = \min \sum_{(x_i, y_i) \in \mathcal{D}^o} \ell \left(f^o \left(\mathbf{x}_i \right), y_i \right). \tag{1}$$

2) New Class Accuracy: This part aims to ensure that the model effectively learns newly introduced classes. Let $\mathcal{L}_{\text{new}}$ represent the loss function for new classes, ϕ represents the model parameters for incrementally learning new classes, f_ϕ represent the model for learning new classes, and $\mathcal{D}^n$ represents samples from new classes. The objective for this part can be expressed as:

$$\begin{aligned} \mathcal{L}_{\text{new}}(\boldsymbol{\theta}, \phi) &= \min \sum_{(x_i, y_i) \in \mathcal{D}^n} \ell \left(f_\phi^n \left(\mathbf{x}_i \right), y_i \right) \\ &\text{s.t. } \|R(\boldsymbol{\theta})\| \leq \mathcal{C}. \end{aligned} \tag{2}$$

The introduction of new classes results in the dynamic expansion of the network model in both feature extraction and classifier components. Consequently, the model weights increase with the rising number of incremental updates. To address this, we employ a sparse modeling approach to minimize model weights without compromising accuracy, and we constrain the model weights to lie within a constant range $\mathcal{C}$. Combining these two aspects, the complete objective function is a weighted combination of the two components:

$$\mathcal{L} = \lambda \cdot \mathcal{L}_{\text{old}}(\boldsymbol{\theta}) + (1 - \lambda) \cdot \mathcal{L}_{\text{new}}(\boldsymbol{\theta}, \phi), \tag{3}$$

where λ is a hyperparameter that balances the objectives for old and new classes. The optimization of this objective function can be carried out using optimization algorithms such as gradient descent. It is important to note that the specific form may vary depending on the method, and the above representation is one common form. From Eq. (3), it is apparent that when the model is tackling the task of learning new class $\mathcal{D}^n$, it needs to simultaneously consider its loss on past samples from old classes. The optimization of the model, $f(\cdot)$, is geared towards enabling it to possess discriminative capabilities for both old and new classes. If the model is trained incrementally using the entire training dataset $\mathcal{D}^1 \cup \mathcal{D}^2 \cup \cdots \mathcal{D}^M \cup \cdots \mathcal{D}^N$, it allows for a holistic consideration of information from all classes. This approach empowers the model to acquire discriminative capabilities encompassing all categories.

Algorithm 1: Training process of the proposed SPCIL method.

```
[Required hyperparameters]:
Set the init-training epochs $E_{init}$ as 200;
Set the incremental training epochs $E_{train}$ as 200;
Set the $\lambda_a$ as 0.01 for Loss;
Set the $\lambda_b$ as 0.001 for Loss;
Set the $\lambda$ as 0.5 for $\mathcal{L}_{RS}$.
[Init training stage]:
Randomly initialize the parameters $\theta_{init}$;
for $i = 1, 2, \cdots, E_{init}$ do
    Set the init-training class number $N_{init}$;
    $\theta_{init} \leftarrow SPCIL(\mathbf{x}_0, y_0)$;
    $Loss \leftarrow \min(L_{CE}(\mathbf{x}_i, y_i))$;
    Update $\theta_{init}$ with $Loss$
end
Save $\theta_{init}$.
[Incremental training stage]:
Randomly initialize the parameters $\theta_{train}$
$i = N_{init}$;
Set incremental step $t$;
for $(i + t) \leq K$ do
    Load $\theta_{init}$;
    Expand the feature extractor $F_i$ dynamically based on parameter $t$;
    Expand the classifier $y_i$ dynamically based on parameter $t$;
    for $j = 1, 2, \cdots, E_{train}$ do
        Freeze $F_1, F_2, \cdots, F_{i-1}$;
        Feature Fusion $\Phi_i = \Phi_{i-1} \cup F_i$;
        Classifier fusion $C_i = C_{i-1} \cup y_i$;
        $\theta_{train} \leftarrow SPCIL(\mathbf{x}_t, y_t)$;
        $Loss \leftarrow \min(\mathcal{L}_{CE}(\mathbf{x}_i, y_i) + \lambda_a \cdot \mathcal{L}_{SP}(\mathbf{x}_i, y_i) + \lambda_b \cdot \mathcal{L}_{RS}(\mathbf{x}_i, y_i))$;
        Update $\theta_{train}$ with $Loss$;
    end
    $i \leftarrow i + t$;
    $\theta_{init} \leftarrow \theta_{train}$;
end
Save $\theta_{train}$.
```

IV. The Proposed SPCIL MD Method

A. Framework of Proposed SPCIL-Based MD Method

In this section, we propose SPCIL method to address the MD problem using CIL, aiming to achieve a better balance between stability and adaptability. To do so, we introduce a dynamically expandable network that incrementally enhances previously learned representations through new features and a two-stage learning strategy. The proposed SPCIL method consists of stability modules and plasticity modules. The goal of the stability modules is to prevent catastrophic forgetting, while the objective of the plasticity modules is to acquire new knowledge. Additionally, we also incorporate adaptive sparse pairwise (ADASP) loss [62] and sparsity loss to reduce model complexity.

We are aware that, in contrast to task-incremental learning, CIL involves the addition of new classes during the inference process. Specifically, in this model, it observes a sequence of class groups Y_n along with their corresponding training data X_n. At the n-th step, the dataset X_n takes the form $(\mathbf{x}_n^i, y_n^i)$, where $\mathbf{x}_n^i$ represents the input image, and $y_n^i \in Y_n$ corresponds to labels from the label set Y_n. As new classes are introduced with each incremental step, the prediction at the n-th step should encompass the summation of both new and old class labels, and **Algorithm 1** describes the training process of the SPCIL method.

Our approach employs a pre-storage strategy, where a portion of the data is saved as memory, denoted as M_n, for future training use. At the n-th step, for the training, we decouple the training process into the following two consecutive stages:

1) Feature Extraction Stage: To strike a better balance between stability and adaptability, we maintain the previous feature representations while training the incoming and memory data using two separate feature extractors [61]. Specifically, we train the data from the old class, denoted as X_{n-1}, with the feature extractor F_1, and the data from the new class, denoted as X_n, is trained using the feature extractor F_2. Subsequently, we fuse the features from both the new and old classes. Additionally, we introduce an auxiliary loss on the new extractor to encourage the learning of diverse and discriminative features. To enhance the model efficiency, we introduce a pruning method based on channel-level masks, which dynamically expands the representation of the model according to the complexity of the new classes. An overview of this process is depicted in Fig. 4.

2) Classifier Fusion Stage: Following the feature extraction stage, we combine the feature results obtained from the feature extractor F_1 and the feature extractor F_2. At step n, we retrain the classifier using all the current data ($X_n = X_{n-1} \cup M_n$) and employ fine-tuning techniques to address class imbalance issues. This results in a classifier that encompasses all the classes, including both old and new classes.

B. Expandable Feature Extraction

From Fig. 4, it is evident that our feature extractor is composed of both the feature extractor F_1 and the feature extractor F_2. Our SPCIL model is structured as illustrated in Fig. 4. The feature extractors F_1 and F_2 are both implemented using ResNet18 [43]. At the n-th step, our model consists of the feature extractor Φ_n and the classifier C_n. The feature extractor Φ_n is constructed by extending the feature extractor Φ_{n-1} obtained from the previous $n-1$ steps using the new feature extractor F_t. More specifically, for a given data $\mathbf{x} \in X_n$, the features F_e extracted by Φ_n are obtained using the following formula:

$$F_e = \Phi_n = [\Phi_{n-1}, F_n], \tag{4}$$

where Φ_{n-1} is obtained through training up to the previous $n-1$ steps and can be represented by the following formula:

$$\Phi_{n-1} = [F_1, F_2, \cdots, F_{n-1}]. \tag{5}$$

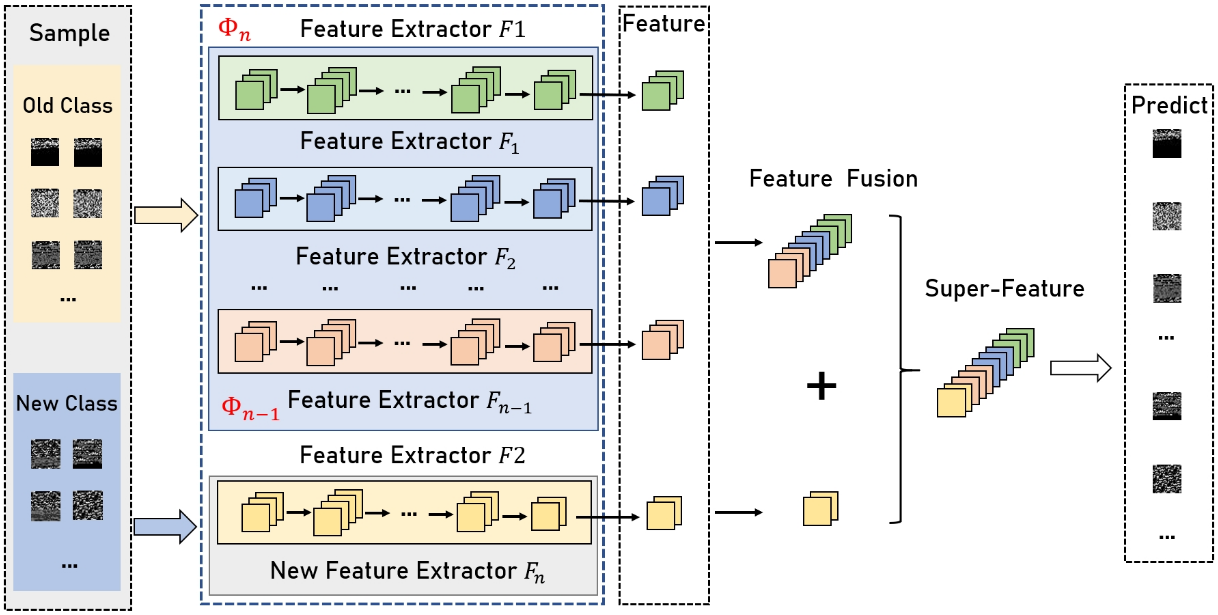


Fig. 4. Framework of the proposed SPCIL MD method.

From the above formulas, it is evident that each training iteration expands the feature extractor based on the increment size. The newly obtained feature extractor, F_n, subsequently becomes the old feature extractor Φ_{n-1} for the next increment, and a new feature extractor is introduced to extend this old feature extractor, and so on. This iterative process allows us to develop feature extractors for all classes.

Furthermore, to mitigate catastrophic forgetting during the training process, we freeze the learned function Φ_{n-1} at the n-th step since it captures the intrinsic structure of the previous data. This means that the parameters of the previous super feature extractor Φ_{n-1} and the statistics of batch normalization are not updated. Additionally, we instantiate F_n with F_{n-1} to initialize it, allowing us to rapidly adapt and perform forward transfer by reusing prior knowledge.

C. Expandable Classifier Fusion

After feature extraction, it becomes necessary to expand the classifier to enhance its classification capacity, as the introduction of new classes demands a corresponding augmentation. Furthermore, following the feature extraction phase, there is a need to merge the features extracted by the feature extractor Φ_{n-1} and the feature extractor F_n to create a super-feature with dynamic expansion capabilities. Following the super feature creation, we retrain the classifier to accommodate the addition of new classes. At the nth increment, the training dataset expands from $\mathcal{D}_{old} = (X^1, X^2, \cdots, X^{n-1})$, where $X^r = \{(\mathbf{x}_1{}^r, y_1{}^r), (\mathbf{x}_2{}^r, y_2{}^r), \cdots, (\mathbf{x}_{n-1}{}^r, y_{n-1}{}^r)|1 \leq n \leq 20\}$ to $\mathcal{D}_{new} = (X^1, X^2, \cdots, X^n)$, where $X^r = \{(\mathbf{x}_1{}^r, y_1{}^r), (\mathbf{x}_2{}^r, y_2{}^r), \cdots, (\mathbf{x}_n{}^r, y_n{}^r)|1 \leq n \leq 20\}$, and the labels transition from $Y_{old} = (y_1, y_2, \cdots, y_{n-1})$ to $Y_{new} = (y_1, y_2, \cdots, y_n)$. This enables the classifier to perform classification and recognition for both new and old classes, ultimately achieving a dynamically expandable classifier. The classifier is now a fusion of both new and old classes.

D. Loss Function

To enhance accuracy while reducing network model complexity, we propose a novel loss function comprising three components: the cross-entropy (CE) loss, the sparse pairwise (SP) loss, and the L1 regularized sparse (RS) loss. Its formulation is as follows:

$$Loss = \mathcal{L}_{CE} + \lambda_a \cdot \mathcal{L}_{SP} + \lambda_b \cdot \mathcal{L}_{RS}, \tag{6}$$

where λ_a and λ_b are loss parameters. In our experiments, we set λ_a to 0.01 and λ_b to 0.01 [63].

The CE loss function is a commonly used loss function for classification problems, widely employed in training neural networks. It is typically utilized to measure the disparity between the predicted output of the model and the actual target values. This loss function effectively quantifies the performance of the model in classification tasks and provides clear gradient information for the backpropagation algorithm, enabling parameter updates to enhance accuracy. Its formulation is

$$\mathcal{L}_{CE} = H(y_i, p_i) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(p_i), \tag{7}$$

where y_i represents the actual multi-class labels. p_i is the model's predicted probability vector, and N represents the total number of categories.

The SP loss is an objective function designed for classification problems, aiming to minimize classification error rates by comparing the scores of two instances. In contrast to the traditional CE loss, SP loss exhibits excellent performance and greater robustness when dealing with issues such as sample imbalance and noisy data. The SP loss is utilized to measure the similarity or dissimilarity between sample pairs. It aids the model in determining which features or dimensions are most critical for the task during the learning process. The SP loss extracts a positive pair and a negative pair for each class individually, and its formulation is:

$$\mathcal{L}_{SP} = \frac{1}{N}\sum_{i}^{N} \log\left(1 + e^{\frac{s_i^- + s_i^+}{T}}\right), \tag{8}$$

where N is the total number of classes, s_i^+ denotes the similarity of the selected positive pairs for the i-th class from all its positive pairs, s_i^- represents the similarity of the selected negative pairs for the i-th class from all its negative pairs, and T is the temperature coefficient.

The addition of new classes leads to a gradual increase in model parameters, and with the growing number of incremental learning iterations, the parameter size of the model continues to expand. To address this issue, we introduce a sparse loss function to induce sparsity in the model parameters. ℓ_1 regularization sparse loss function is commonly used in sparse learning or feature selection. Its objective is to minimize the loss term while encouraging the parameters of the model to have more zero values through the regularization penalty term $\lambda * \sum \theta_i$, thus achieving sparsity, which retains only the most important features or model parameters. Its general form is as follows:

$$\mathcal{L}_{RS}(\theta) = \mathcal{L}(\theta) + \lambda \sum_{i=1}^{n} |\theta_i|, \tag{9}$$

where θ represents the model that needs to be made sparse, θ_i refers to the model's parameters, λ is the weight of $L1$ regularization. In our experiments, λ is set to 0.5. $\mathcal{L}(\theta)$ is the loss function specific to the problem, often utilizing cross-entropy loss. But for the sake of clarity, we omit the loss function in this context.

V. Experimental Results and Discussions

A. Dataset and Experiment Setup

The dataset utilized is the USTC-TFC2016 [36], which comprises a total of 142,700 samples across 20 categories. This dataset undergoes a four-stage processing pipeline, including traffic segmentation, data cleaning, image conversion, and ultimately, the transformation of results into the IDX format suitable for network training. During the image conversion phase, each unit of traffic data is transformed into a grayscale image. The resulting grayscale images have dimensions of 784 bytes (28×28×1).

The dataset is partitioned into training, validation, and test sets in a 7:2:1 ratio. As our focus is on class incremental, we initialize training based on 10 classes (an increment of 2), followed by 8 classes (an increment of 4), and 5 classes (an increment of 5). The initial training epoch is set to 200, and the incremental training epoch is also set to 200. Detailed configurations of the experimental parameters are provided in Table I.

TABLE I
Experimental parameters.

Parameter	Value
Dataset	USTC-TFC2016
The number of train samples	102,746
The number of val samples	25,687
The number of test samples	14,267
Device	Geforce GTX 3090 Ti
Environment	PyTorch
Optimizer	SGD
Batch size	32
Learning rate	0.001
Init epoch	200
Incremental epoch	200

B. Performance of SPCIL MD Method

In the context of multi-class classification tasks, overall accuracy (OA) and average accuracy (AA) are two commonly used performance metrics. OA refers to the proportion of correctly classified samples in the overall dataset, calculated by considering True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) across all classes. On the other hand, AA represents the average classification accuracy for each class, focusing on the performance of individual categories. These metrics provide a comprehensive evaluation of the performance of the model across the entire dataset and individual classes. In this paper, we utilize both of these metrics to assess the performance of the model. Their formulas are as follows:

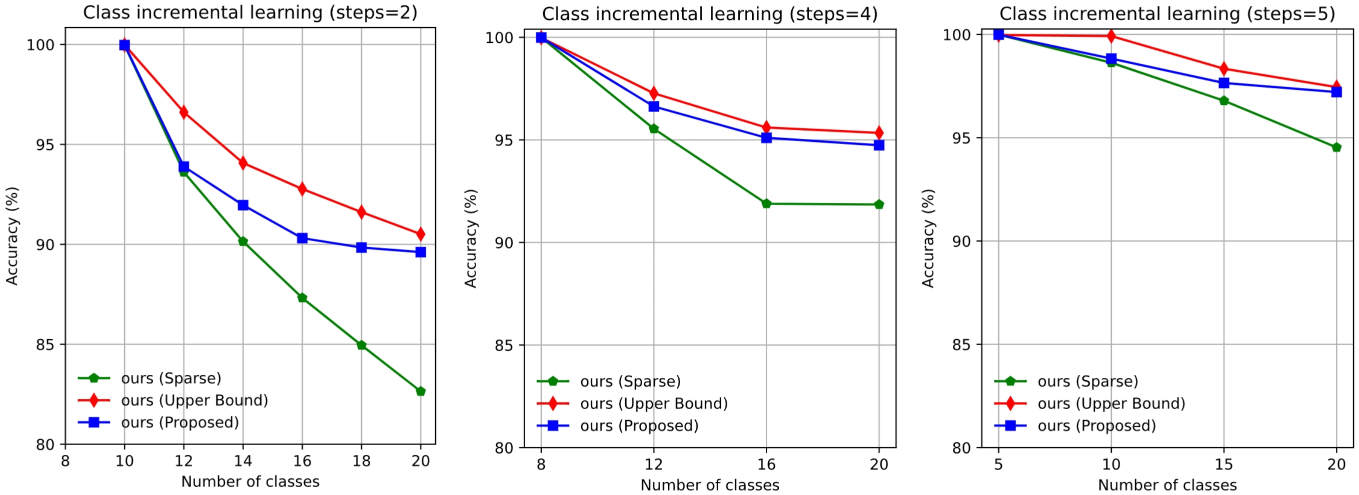
$$\text{OA} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{10a}$$

$$\text{AA} = \frac{1}{N}\sum_{i=1}^{N} \frac{TP_i}{TP_i + FN_i}, \tag{10b}$$

where N is the total number of classes.

We conduct experiments to investigate the performance of the SPCIL network. We test the network under the incremental steps of 2, 4, and 5, selecting different numbers of categories to incrementally add to the network at each step. Specifically, we selected 10 categories as the base category and added 2 new categories for each step for the incremental step of 2. For the incremental step of 4, we chose 8 categories as the base category and added 4 new categories at each step. For the incremental step of 5, we select 5 categories as the base category and add 5 new categories in each step. We also explore the model performance without sparsity, with L1 regularization sparse loss, and with sparse pairwise loss added to the L1 regularization sparse loss.

The experimental results are shown in Fig. 5. As can be seen from the figure, without any sparsity constraints, our network performed the best, reaching the upper limit of its potential performance. However, when we added the L1 regularization sparse loss function, the model's sparsity increased, leading to a decrease in performance. Nevertheless, further introduction of sparse pairwise loss to the loss function improved the performance and approached the upper limit. Upon examining Table II, it is evident that the model parameters of our SPCIL network are 8.35 million, significantly lower than



(a) Incremental steps = 2

(b) Incremental steps = 4

(c) Incremental steps = 5

Fig. 5. The AA of different incremental steps.

the parameters of the network without employing sparse strategies (67.08 million). More importantly, our network achieves superior model parameter efficiency, maintaining accuracy while also surpassing the parameter count of other minimal models (11.18 million).

It can also be observed from the figure that as the incremental step size decreases, the performance decreases when the SPCIL increment reaches 20 classes. This is because, in our incremental training approach, we freeze the results of previous steps of incremental training. When the incremental step size is smaller, there are more frozen layers, resulting in lower performance. Therefore, the performance for the incremental step of 2 (89.61%) is lower than that for the incremental step of 4 (94.74%), which is also lower than that for the incremental step of 5 (97.21%).

TABLE II
PERFORMANCE OF DIFFERENT NETWORKS ON OA.

Methods	#Param (million)	OA (%)		
		CIL=2	CIL=4	CIL=5
Upper Bound	67.08	91.51	97.45	95.34
BiC	11.18	51.52	75.3	89.77
EWC	11.18	28.26	44.36	51.95
Foster	22.38	85.15	90.53	93.85
Finetune	11.18	27.49	44.14	51.42
LwF	11.18	29.41	54.36	61.65
iCaRL	11.18	88.69	93.05	95.94
SimpleCIL	11.69	81.21	80.14	78.63
ours (SPCIL)	**8.35**	**91.18**	**97.36**	**95.06**

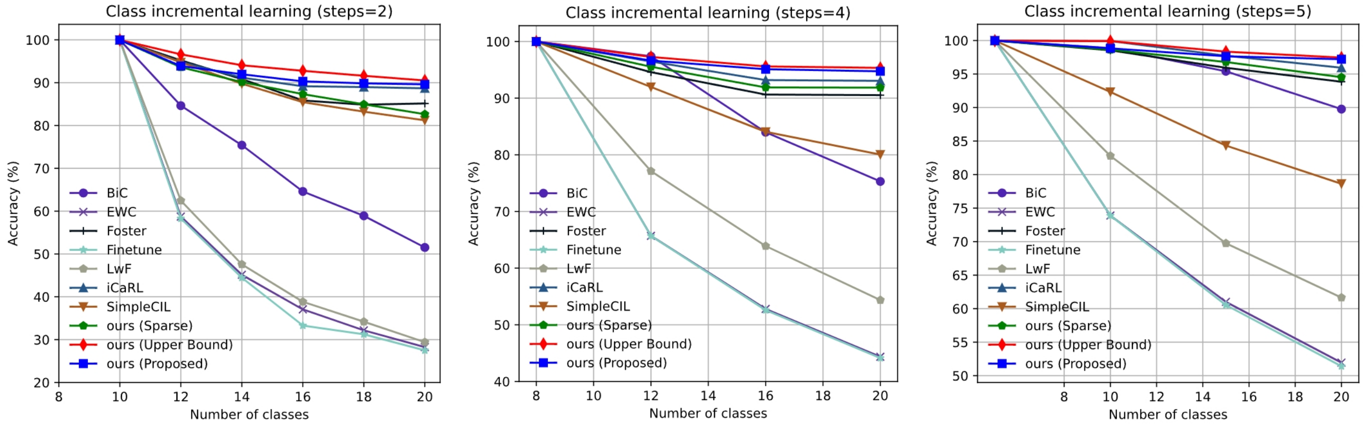
C. Performance of Different Network

To validate the performance of the SPCIL model, we compared it with existing state-of-the-art incremental learning methods. The baseline approach (Fine-tune) which involves simply updating parameters on new tasks and is susceptible to severe catastrophic forgetting. BiC [64] trains an additional adaptation layer based on iCaRL, adjusting the logits for new classes. EWC [66] utilizes the Fisher information matrix to weigh the importance of each parameter and regularizes them to overcome forgetting. Foster [67] dynamically expands new modules to fit the residuals between the target and the output of the original model. LwF [45] utilizes knowledge distillation to align the output probability between the old and new models. iCaRL [65], an extension of LwF, introduces exemplar sets for rehearsal and utilizes the nearest center mean classifier for classification. SimpleCIL [68] configures the classifiers of pre-trained models to use prototype features. The experimental results are shown in Fig. 6.

To present a clearer comparison of the performance of different network models in incremental learning, we have included tables displaying the accuracy rates for increments of 2, 4, and 5 in Tables III, IV, and V, respectively. These tables provide a detailed comparison of the performance of different models in various incremental learning scenarios, enabling a more comprehensive evaluation of their abilities.

We compare the performance of different models using OA curves. The OA curve is presented in Fig. 7. From Fig. 7, it is evident that our proposed SPCIL model consistently outperforms existing models. Furthermore, we summarize the OA value for different models in Table II, where #Paras represents the average number of parameters during inference over steps, measured in millions. The table indicates that our model achieves high accuracy with fewer parameters than other models.

Based on Table III, we can draw the following conclusions that our network SPCIL (Upper Bound) performs the best, only slightly lower than EWC by 0.02% at CIL = 10, and superior to the other networks in other incremental stages. From Table IV, our network SPCIL (Upper Bound) still maintains the best but not optimal performance, only slightly lower than BiC by 0.05% at CIL = 10, and higher than the other networks in other incremental stages. Similarly, from Table V, our network SPCIL (Upper Bound) still maintains the optimal performance, only slightly lower than BiC by 0.04% at CIL = 10, and higher than the other networks in other incremental stages.

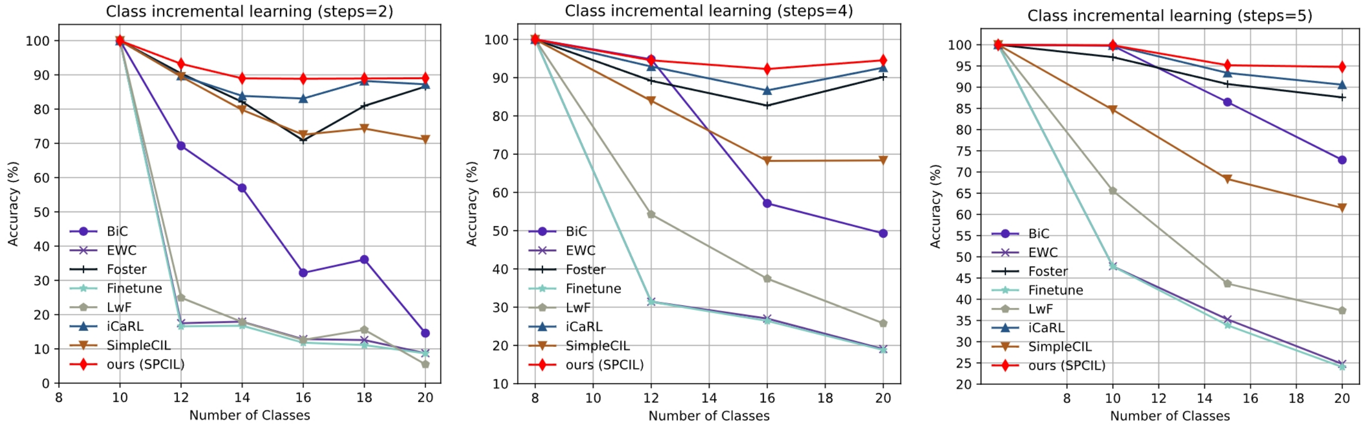


(a) Incremental steps = 2

(b) Incremental steps = 4

(c) Incremental steps = 5

Fig. 6. The AA of different networks.



(a) Incremental steps = 2

(b) Incremental steps = 4

(c) Incremental steps = 5

Fig. 7. The performance of OA with different networks.

TABLE III
OPERATION TIME OF DIFFERENT SECONDARY CLASSIFIER.

Methods	CIL = 10	CIL = 12	CIL = 14	CIL = 16	CIL = 18	CIL = 20
Upper Bound	99.99	96.61	94.07	92.77	91.61	90.51
BiC	99.98	84.63	75.41	64.60	58.91	51.52
EWC	**99.99**	58.74	45.14	37.06	32.16	28.26
Foster	99.98	95.21	90.84	85.85	84.86	85.15
Finetune	99.93	58.28	44.45	33.29	31.25	27.49
LwF	99.98	62.46	47.58	38.84	34.19	29.41
iCaRL	99.96	94.85	91.18	89.16	88.97	88.69
SimpleCIL	99.97	94.74	89.76	85.45	83.23	81.21
ours (Sparse)	99.96	93.61	90.14	87.32	84.95	82.64
ours (SPCIL)	99.97	**93.89**	**91.96**	**90.31**	**89.84**	**89.61**

TABLE IV
OPERATION TIME OF DIFFERENT SECONDARY CLASSIFIER.

Methods	CIL = 8	CIL = 12	CIL = 16	CIL = 20
Upper Bound	99.99	97.27	95.60	95.34
BiC	99.99	**96.68**	83.98	75.30
EWC	99.99	65.70	52.80	44.36
Foster	99.99	94.59	90.63	90.53
Finetune	99.99	65.69	52.59	44.14
LwF	99.99	77.12	63.90	54.36
iCaRL	99.99	96.46	93.19	93.05
SimpleCIL	99.99	91.97	84.06	80.04
ours (Sparse)	99.99	95.54	91.89	91.85
ours (SPCIL)	**99.99**	96.63	**95.10**	**94.74**

TABLE V
OPERATION TIME OF DIFFERENT SECONDARY CLASSIFIER.

Methods	**CIL = 5**	**CIL = 10**	**CIL = 15**	**CIL = 20**
Upper Bound	99.99	99.92	98.34	97.45
BiC	99.99	**98.87**	95.41	89.77
EWC	99.99	73.89	61.01	51.95
Foster	99.99	98.54	95.93	93.85
Finetune	99.99	73.88	60.54	51.42
LwF	99.99	82.79	69.75	61.65
iCaRL	99.99	99.92	97.73	95.94
SimpleCIL	99.99	92.33	84.33	78.63
ours (Sparse)	99.99	98.63	96.79	94.53
ours (SPCIL)	**99.99**	98.83	**97.65**	**97.21**

In addition, it can be observed from the data that the stability of our network is generally superior to the other networks.

Therefore, our proposed SPCIL model demonstrates excellent performance and stability in incremental learning tasks compared to other network models, and thus has significant advantages.

VI. Conclusion

In this paper, we propose an SPCIL method based on CIL for MD in IDS. Compared with traditional methods and other CIL-based techniques, our method employs a dual-branch network model and introduces adaptive sparse pairwise loss and sparse loss in the loss function, enabling SPCIL to achieve superior performance with fewer model parameters, and it can learn new tasks without forgetting old tasks. Experimental results demonstrate that the performance of the SPCIL method outperforms the other CIL methods and approaches its upper limit performance. Therefore, our proposed SPCIL method can be used for IDS to detect malware identification in the field of network security. The principles and framework of our SPCIL method can provide valuable references for class-incremental algorithms across various domains. Our research holds significance not only in the field of malicious software detection but also offers valuable insights and methods for addressing similar challenges in other domains. This cross-domain expansion opens up new possibilities for different application scenarios, prompting class-incremental methods to play a more widespread role in addressing practical challenges. Furthermore, our model is sparse, resulting in a lightweight approach that is efficient and easy to deploy. In future work, we will consider more practical scenarios for MD problems under few-shot sample conditions, which is a promising direction.

References

[1] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 76–81, May 2019.

[2] Y. Dhote, S. Agrawal and A. J. Deen, "A survey on feature selection techniques for Internet traffic classification," in *Proc. CICN*, Jabalpur, India, 12-14 Dec. 2015, pp. 1375–1380.

[3] J. Ning, G. Gui, *et al.*, "Malware traffic classification using domain adaptation and ladder network for secure industrial Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17058–17069, Sept. 2022.

[4] R. Zhao, G. Gui, *et al.*, "A novel intrusion detection method based on lightweight neural network for Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9960–9972, June 2022.

[5] R. Zhao, L. Yang, Y. Wang, Z. Xue, G. Gui and T. Ohtsuki, "A semi-supervised federated learning scheme via knowledge distillation for intrusion detection," in *Proc. ICC*, Seoul, Korea, 16-20 May 2022, pp. 2688–2693.

[6] X. Zhang, L. Hao, G. Gui, Y. Wang, B. Adebisi, and H. Sari, "An automatic and efficient malware traffic classification method for secure Internet of Things" *IEEE Internet Things J.*, vol. 11, no. 5, pp. 8448–8458, Mar. 2024.

[7] Y. Lin, Y. Tu, Z. Dou, "An improved neural network pruning technology for automatic modulation classification in edge devices," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5703–5706, May 2020.

[8] Y. Tu, Y. Lin, C. Hou, S. Mao, "Complex-valued networks for automatic modulation classification," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 10085–10089, Sept. 2020.

[9] X. Fu, G. Gui, *et al.*, "Lightweight automatic modulation classification based on decentralized learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 1, pp. 57–70, Mar. 2022.

[10] C. Wang, X. Fu, *et al.*, "Interpolative metric learning for few-shot specific emitter identification," *IEEE Trans. Veh. Technol.*, vol. 72, no. 12, pp. 16851–16855, Dec. 2023.

[11] Y. Lin, M. Wang, X. Zhou, G. Ding, S. Mao "Dynamic spectrum interaction of UAV flight formation communication with priority: A deep reinforcement learning approach," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 3, pp. 892–903, Sept. 2020.

[12] Y. Lin, H. Zhao, X. Ma, Y. Tu, M. Wang, "Adversarial attacks in modulation recognition with convolutional neural networks," *IEEE Trans. Reliab.*, vol. 70, no. 1, pp. 389–401, Jan. 2020

[13] R. Zhao, J. Yin, *et al.*, "An efficient intrusion detection method based on dynamic autoencoder," *IEEE Wireless Commu. Lett.*, vol. 10, no. 8, pp. 1707–1711, Aug. 2021.

[14] Y. Wang, G. Gui, *et al.*, "Federated learning for automatic modulation classification under class imbalance and varying noise condition," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 1, pp. 86–96, Mar. 2022.

[15] M. De Lange, R. Aljundi, *et al.*, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3366–3385, July 2021.

[16] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov and J. van de Weijer, "Class-incremental learning: Survey and performance evaluation on image classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5513–5533, 1 May 2023.

[17] X. Xu, Y. Liang, *et al.*, "Self-evolving malware detection for cyber security using network traffic and incremental learning," in *Proc. DSA*, Wulumuqi, China, 4-5 Aug. 2022, pp. 454–463.

[18] J. Zhou, Y. Peng, *et al.*, "A novel radio frequency fingerprint identification method using incremental learning," in *Proc. IEEE VTC*, London, United Kingdom, 19-22 June 2022, pp. 1–5.

[19] Y. Wang, W. Chen, *et al.*, "Task offloading for post-disaster rescue in unmanned aerial vehicles networks," *IEEE/ACM Trans. Netw.*, vol. 30, no. 4, pp. 1525–1539, Aug. 2022.

[20] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, May 2019.

[21] Y. Liu, B. Schiele and Q. Sun, "Adaptive aggregation networks for class-incremental learning," in *Proc. IEEE CVPR*, Nashville, TN, USA, 19-25 June 2021, pp. 2544–2553.

[22] X. Tao, X. Hong, X. Chang, S. Dong, X. Wei and Y. Gong, "Few-shot class-incremental learning," in *Proc. IEEE CVPR*, Seattle, WA, USA, 13-19 June 2020, pp. 12180–12189.

[23] H. Dreger, A. Feldmann, M. Mai, V. Paxson, and R. R. Sommer, "Dynamic application layer protocol analysis for network intrusion detection," in *USENIX Security Symposium*, Vancouver, B.C. Canada, July 31-Aug. 4, 2006, pp. 1–6.

[24] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in network identification of P2P traffic using application signatures," in *Proc. IWWWC*, New York, NY, USA, May 2004, pp. 512–521.

[25] M. Finsterbusch, C. Richter, E. Rocha, J. A. Muller and K. Hanssgen, "A survey of payload-based traffic classification approaches," *IEEE Commun. Surv. Tutor.*, vol. 16, no. 2, pp. 1135–1156, Feb. 2014.

[26] V. Paxson, "Empirically derived analytic models of wide-area TCP connections," *IEEE/ACM Trans. Netw.*, vol. 2, no. 4, pp. 316–336, Aug. 1994.

[27] B. Mao, F. Tang, Z. Md. Fadlullah, and N. Kato, "An intelligent route computation approach based on real-time deep learning strategy for software defined communication systems," *IEEE Trans. Emerg. Top. Comput.*, vol. 9, no. 3, pp. 1554–1565, July 2021.

[28] B. Mao, F. Tang, Y. Kawamoto, and N. Kato, "Optimizing computation offloading in satellite-UAV-served 6G IoT: A deep learning approach," *IEEE Netw.*, vol. 35, no. 4, pp. 102–108, July 2021.

[29] Y. Peng, C. Hou, Y. Zhang, Y. Lin, G. Gui, H. Gacanin, S. Mao, and A. Adachi, "Supervised contrastive learning for RFF identification with limited samples," *IEEE Internet Things J.*, vol. 10, no. 19, pp. 17293–17306, Oct. 2023.

[30] X. Zhang, H. Zhao, H. Zhu, B. Adebisi, G. Gui, H. Gacanin, and F. Adachi, "NAS-AMR: Neural architecture search-based automatic modulation recognition for integrated sensing and communication systems," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 3, pp. 1374–1386, Sept. 2022.

[31] Y. Lin, Y. Tu, Z. Dou, L. Chen, and S. Mao, "Contour stella image and deep learning for signal recognition in the physical layer," *IEEE Trans. Cogn. Commun. Netw*., vol. 7, no. 1, pp. 34–46, Mar. 2021.

[32] Y. Lin, H. Zha, Y. Tu, S. Zhang, W. Yan and C. Xu, "GLR-SEI: Green and low resource specific emitter identification based on complex networks and Fisher pruning," *IEEE Trans. Emerg. Top. Comput. Intell.*, early access, doi: 10.1109/TETCI.2023.3303092.

[33] T. Ohtsuki, "Machine learning in 6G wireless communications," *IEICE Trans. Commun.*, vol. 106, no. 2, pp. 75–83, Feb. 2023.

[34] S. I. Popoola, B. Adebisi, *et al*., "Hybrid deep learning for botnet attack detection in the Internet of Things networks," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4944–4956, June 2021.

[35] I. Ahmad, T. Kumar, *et al*., "5G security: Analysis of threats and solutions," in *Proc. IEEE CSCN*, Helsinki, Finland, 18-20 Sept. 2017, pp. 1–5.

[36] W. Wang, M. Zhu, X. Zeng, X. Ye and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. ICOIN*, Da Nang, Vietnam, 11-13 Jan. 2017, pp. 712–717.

[37] N. Gao, L. Gao, Q. Gao and H. Wang, "An intrusion detection model Based on deep belief networks," in *Proc. CBD*, Huangshan, China, 20-22 Nov. 2014, pp. 247–252.

[38] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, Dec. 2019.

[39] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[40] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, July 1997.

[41] X. Zhang, X. Chen, Y. Wang, G. Gui, B. Adebisi, H. Sari, and F. Adachi, "Lightweight automatic modulation classification via progressive differentiable architecture search," *IEEE Trans. Cogn. Commun. Netw.*, vol. 9, no. 6, pp. 1519–1530, Dec. 2023.

[42] Y. Wang, G. Gui, *et al*., "An efficient specific emitter identification method based on complex-valued neural networks and network compression," *IEEE J. Sel. Areas. Commun.*, vol. 39, no. 8, pp. 2305–2317, Aug. 2021.

[43] K. He, X. Zhang, S. Ran and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, Las Vegas, NV, USA, 27-30 June 2016, pp. 770–778.

[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, San Diego, CA, USA, 7-9 May 2015, pp. 1–14.

[45] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018.

[46] T. Wang, Q. Lv, B. Hu and D. Sun, "A few-shot class-incremental learning approach for intrusion detection," in *Proc. ICCCN*, Athens, Greece, 19-22 July 2021, pp. 1–8

[47] H. Jung, J. Ju, M. Jung, and J. Kim, "Less-forgetting learning in deep neural networks," available [online] https://arxiv.org/abs/1607.00122, 2016.

[48] X. Liu, M. Masana, L. Herranz, J. Van de Weijer, A. M. Lopez, and A. D. Bagdanov, "Rotate your networks: Better weight consolidation and less catastrophic forgetting," in *Proc. IEEE ICPR*, Beijing, China, 20-24 Aug. 2018, pp. 1–5.

[49] A. Mallya and S. Lazebnik, "PackNet: Adding multiple tasks to a single network by iterative pruning," in *Proc. IEEE CVPR*, Salt Lake City, UT, USA, 18-23 June 2018, pp. 7765–7773.

[50] X. Fu, S. Shi, *et al*., "Semi-supervised specific emitter identification via dual consistency regularization," *IEEE Internet Things J.*, vol. 10, no. 21, pp. 19257–19269, Nov. 2023.

[51] S. Yan, J. Xie and X. He, "DER: Dynamically expandable representation for class incremental learning," in *Proc. IEEE CVPR*, Nashville, TN, USA, 2021, pp. 3013–3022.

[52] E. Belouadah, A. Popescu and I. Kanellos, "A comprehensive study of class incremental learning algorithms for visual tasks," *Neural Networks*, vol. 135, pp. 38–54, Mar. 2021.

[53] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE CISDA*, Ottawa, ON, Canada, 08-10 July 2009, pp. 1–6.

[54] G. Creech and J. Hu, "Generation of a new IDS test dataset: Time to retire the KDD collection," in *Proc. IEEE WCNC*, Shanghai, China, 7-10 Apr. 2013, pp. 4487–4492.

[55] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *Proc. IEEE CVPRW*, Columbus, OH, USA, 23-28 June 2014, pp. 806–813.

[56] F. Tang, B. Mao, *et al*., "Comprehensive survey on machine learning in vehicular network: Technology, applications and challenges," *IEEE Commun. Surv. Tutor.*, vol. 23, no. 3, pp. 2027–2057, thirdquarter 2021.

[57] S. I. Popoola, R. Ande, *et al*., "Federated deep learning for zero-day botnet attack detection in IoT edge devices," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3930–3944, Mar. 2022.

[58] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," in *Proc. ICLR*, Banff, Canada, 14-16 Apr. 2014, pp. 1–9.

[59] S. Naval, V. Laxmi, M. Rajarajan, M. S. Gaur and M. Conti, "Employing program semantics for malware detection," *IEEE Trans. Inf. Foren. Sec.*, vol. 10, no. 12, pp. 2591–2604, Dec. 2015.

[60] W. Ma, C. Qi, Z. Zhang and J. Cheng, "Sparse channel estimation and hybrid precoding using deep learning for millimeter wave massive MIMO," *IEEE Trans. Commun.*, vol. 68, no. 5, pp. 2838–2849, May 2020.

[61] J. Donahue, Y. Jia, *et al*., "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. ICML*, Beijing, China, 21-26 June 2014, pp. 647–655.

[62] X. Zhou, Y. Zhong, Z. Cheng, F. Liang and L. Ma, "Adaptive sparse pairwise loss for object re-identification," in *Proc. IEEE CVPR*, Vancouver, BC, Canada, 18-23 June 2023, pp. 19691–19701.

[63] S. Yan, J. Xie and X. He, "DER: Dynamically expandable representation for class incremental learning," in *Proc. IEEE CVPR*, Nashville, TN, USA, 20-25 June 2021, pp. 3013–3022.

[64] S. Hou, X. Pan, C. C. Loy, Z. Wang and D. Lin, "Learning a unified classifier incrementally via rebalancing," in *Proc. IEEE CVPR*, Long Beach, CA, USA, 16-20 June 2019, pp. 831–839.

[65] S. -A. Rebuffi, A. Kolesnikov, G. Sperl and C. H. Lampert, "iCaRL: Incremental classifier and representation learning," in *Proc. IEEE CVPR*, Honolulu, HI, USA, 21-26 July 2017, pp. 5533–5542.

[66] J. Kirkpatrick, *et al*., "Overcoming catastrophic forgetting in neural networks," *PNAS*, vol. 114, no. 13, pp. 3521–3526, 2017.

[67] F. Wang, D. Zhou, H. Ye, D. Zhan, "FOSTER: Feature boosting and compression for class-incremental learning," in *Proc. ECCV*, Tel Aviv, Israel, 23-27 Oct. 2022, pp. 1–17.

[68] D. Zhou, H. Ye, D. Zhan, Z. Liu, "Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need," available [online] https://arxiv.org/abs/2303.07338, 2023.