


# Multi-ARCL: Multimodal adaptive relay-based distributed continual learning for encrypted traffic classification <sup>☆</sup>

Zeyi Li <sup>a</sup>, Minyao Liu <sup>b</sup>, Pan Wang <sup>b, \*, Wangyu Su <sup>c</sup>, Tianshui Chang <sup>d</sup>, Xuejiao Chen <sup>e</sup>, Xiaokang Zhou <sup>f</sup></sup>

<sup>a</sup> School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210003, China

<sup>b</sup> School of Modern Posts, Nanjing University of Posts and Telecommunications, Nanjing 210003, China

<sup>c</sup> College of Letters and Science, University of Wisconsin Madison, Madison 53706-1380, USA

<sup>d</sup> School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing 210003, China

<sup>e</sup> School of Communications, Nanjing Vocational College of Information Technology, Nanjing, 210046, China

<sup>f</sup> Faculty of Business Data Science, Kansai University, Osaka 565-0823, Japan

## ARTICLE INFO

### Keywords:

Encrypted traffic classification  
Continual learning  
Machine unlearning  
Distributed learning  
Multimodal learning

## ABSTRACT

Encrypted Traffic Classification (ETC) using Deep Learning (DL) faces two bottlenecks: homogeneous network traffic representation and ineffective model updates. Currently, multimodal-based DL combined with the Continual Learning (CL) approaches mitigate the above problems but overlook silent applications, whose traffic is absent due to guideline violations leading developers to cease their operation and maintenance. Specifically, silent applications accelerate the decay of model stability, while new and active applications challenge model plasticity. This paper presents Multi-ARCL, a multimodal adaptive replay-based distributed CL framework for ETC. The framework prioritizes using crypto-semantic information from flows' payload and flows' statistical features to represent. Additionally, the framework proposes an adaptive relay-based continual learning method that effectively eliminates silent neurons and retrains new samples and a limited subset of old ones. Exemplars of silent applications are selectively removed during new task training. To enhance training efficiency, the framework uses distributed learning to quickly address the stability-plasticity dilemma and reduce the cost of storing silent applications. Experiments show that ARCL outperforms state-of-the-art methods, with an accuracy improvement of over 8.64% on the NJUPT2023 dataset.

## 1. Introduction

Over the past decades, cryptography has posed considerable challenges to ETC [1,2], making the conventional method of relying on port numbers and packet payloads unsuitable [3–7]. In recent years, the emergence of artificial intelligence and high-performance computing technologies [8] has opened up new opportunities for ETC through the ability of DL to automatically extract features [9–12]. Nevertheless, ETC based on DL (DL-ETC) overlooks the real-world challenges.

**The simple representation of network traffic is incomplete.** Although the modality of the flow-based FE method can provide better results in the laboratory. This FE method makes it difficult to represent

all the information and characteristics of the flow in a single modality. If the network context is changed [13], i.e. the method of collecting the traffic is changed, the classifier's performance will crack.

**DL-ETC lacks adaptivity.** The classifier trains a fixed number of applications in the laboratory and tests their accuracy [14,15]. However, it must be noted that the classifier's accuracy decreases significantly due to an increasing number of new applications in an open environment. Researchers usually collect new traffic and combine it with traffic from old applications for retraining. It is undeniable that this approach has significant disadvantages including catastrophic forgetting and high storage costs.

<sup>☆</sup> This document is the results of the research project funded by National Natural Science Foundation (General Program) Grant No. 61972211, China, Suzhou High Performance Programmable Switching Chip Innovation Consortium No. LHT202326 and Development of an Ultra-large-scale Ubiquitous Network Quality Monitoring System Based on Trusted Edge Intelligence under Grant SYG202311.

\* Corresponding author.

E-mail addresses: [2022040506@njupt.edu.cn](mailto:2022040506@njupt.edu.cn) (Z. Li), [1222097606@njupt.edu.cn](mailto:1222097606@njupt.edu.cn) (M. Liu), [wangpan@njupt.edu.cn](mailto:wangpan@njupt.edu.cn) (P. Wang), [wsu46@wisc.edu](mailto:wsu46@wisc.edu) (W. Su), [b23100227@njupt.edu.cn](mailto:b23100227@njupt.edu.cn) (T. Chang), [chenxj@njcit.cn](mailto:chenxj@njcit.cn) (X. Chen), [zhou@kansai-u.ac.jp](mailto:zhou@kansai-u.ac.jp) (X. Zhou).

<https://doi.org/10.1016/j.jpdc.2025.105083>

Received 5 March 2024; Received in revised form 7 January 2025; Accepted 1 April 2025

Available online 3 April 2025

0743-7315/© 2025 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

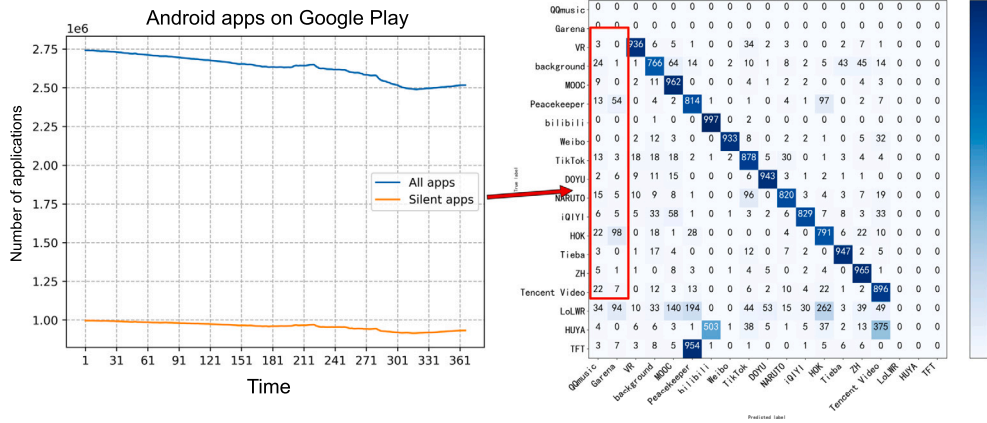


Fig. 1. The number of silent applications and the effect of silent applications on the previous model.

To address the above challenges, the multimodal-based Continual Learning (CL) approach is the most reasonable choice for DL-ETC [16,17]. CL enables models to learn new knowledge without forgetting old applications, thus making TC models adaptable to network environment changes [18–20]. For this purpose, scholars have researched the implementation of CL to maintain the precision of previous classifications while reducing training expenses [21–23].

It is important to note that the network environment is exceedingly complex, marked by the frequent emergence and removal of applications [24]. Google typically removes apps from its market quarterly, reducing the number of available Android apps. In this context, we categorize the removed applications as **silent applications** and those that remain as **active applications**. Fig. 1 shows the number of silent applications over the past year from AppBrain, with about one million apps declining in status daily. Moreover, in our study, we train a classifier to identify existing apps and test them in a new environment, focusing on category changes. Fig. 1 also clearly demonstrates that a significant number of active applications are mistakenly identified as silent ones, which substantially impairs the model's effectiveness. CL currently focuses on learning new applications and retaining old knowledge, but overlooks the ability to forget silent applications.

A fundamental challenge in CL lies in addressing the stability-plasticity dilemma [25]. In the field of ETC, we define 'stability' as a metric to assess the performance of active applications. Conversely, 'plasticity' serves as a measure for evaluating the effectiveness of new applications. Below, we elaborate on the prevailing challenges in CL-ETC, focusing particularly on the stability-plasticity dilemma and the implications of storage costs.

(1) *How to mitigate the decay of silent applications bring to the stability of the model?*

In early CL iterations, the conventional replay-based approach provided high accuracy for older applications. Unfortunately, silent applications occupied exemplar set space, affecting part neurons. The experimental chapter of this study elucidates this issue, revealing instances where active applications were mistakenly classified as silent. Importantly, it was observed that the presence of data and neurons associated with these silent applications had a profound impact on the model's accuracy and stability.

(2) *How to keep the balance between the accuracy of active applications and new applications to guarantee the plasticity of the model?*

In the process of CL's training, the prior model maintains all neurons and parameters from previous applications to prevent catastrophic forgetting. However, retaining the neurons and parameters of silent applications may weaken the model's classification performance for active and new applications. When the model identifies network traffic, neurons of silent applications may be activated, leading to a potential

significant effect on the accuracy of the network model classifier for new and active applications.

(3) *How can classifiers be trained more efficiently using fewer resources, while minimizing the impact of silent applications?*

The vast volume of network traffic poses a storage challenge for training purposes, and CL-based ETC often neglects training costs. Unlike image-based replay CL, where 10% of training samples are retained as exemplars [26], this benchmark is unsuitable for Traffic Classification (TC). Moreover, including silent applications in the exemplar set consumes valuable memory, limiting space for new and active applications.

Therefore, we propose a CL framework for ETC. This framework includes an optimization constraint-driven parameter discarding method with steps such as label mapping, freezing parts of the model, deleting neurons for silent applications, and unfreezing the model. This approach mitigates model stability decay by restricting neuron optimization. Additionally, a novel replay-based CL method computes distillation loss between old and new models and optimizes cross-entropy loss for new and active applications, enhancing adaptability to changing conditions. Additionally, to keep the latest model updates timely amid rapidly changing network traffic, we use distributed learning [27–29] methods for CL. (The dataset and code can be found at <https://github.com/sailorlee97/What-changes-you>.) The main contribution of this paper:

(1) This paper presents Multi-ARCL, a multimodal adaptive replay-based CL framework for ETC. Multi-ARCL consists of two parts: multimodal feature extraction and Adaptive Replay-based CL (ARCL). Utilizing distributed learning, the framework adjusts its structure and parameters to accommodate changes in applications, ensuring adaptability. Additionally, it prioritizes a multimodal representation of network traffic to enhance classifier accuracy.

(2) To mitigate the decay of the stability of the model, an innovative method for discarding parameters based on constrained optimization is proposed. This method automatically identifies the neural parameters and data of active applications and eliminates those of silent applications.

(3) In this paper, a novel replay-based CL method for training is proposed to improve the accuracy of application classification. The method ensures the accuracy of active applications in the new model by calculating the distillation loss of active applications in the old and new models. The method also maintains a high accuracy rate by calculating the cross-entropy loss of the new and active applications of the model.

(4) We also discuss the problem of the number of old data stores. ARCL tests the performance of the model on different numbers of stores. Finally, we conclude that the model stabilizes when the active application retains 1000 samples, and the model reaches its best performance when it retains around 3000–4000 pieces of data.

**Table 1**

Comparison of various research works based on dataset, input data, and incremental/unlearning approaches. • present, ◦ lacking.

Research	Dataset	Input Data	Incremental	Unlearning
Wang et al., 2020, IEEE ICC [30]	ISCX2012	packet	◦	◦
Wang et al., 2020, Elsevier C&S [31]	ISOT, ISCX, self-collected dataset	srcIP, srcPort, dstPort, tcpSeq, tcpFlag, L4 payload (784 B), 4 fields (32 packets).	◦	◦
Aceto et al., 2021, Elsevier JNCA [32]	ISCX VPN-nonVPN	packet length, packet time interval, byte rate, additional information.	◦	◦
Wu et al., 2022, Elsevier CN [33]	self-collected dataset	feature-based statistics	•	◦
Ma et al., 2023, Elsevier C&S [34]	ISCXVPN2016, self-collected dataset	feature-based statistics	•	◦
Bonvezi et al., 2024, IEEE TNSM [35]	MIRAGE19	feature-based statistics	•	◦
Zhu et al., 2023, Elsevier CN [36]	ISCX VPN-nonVPN dataset	traffic images	•	◦
Li et al., 2023, Elsevier C&S [37]	self-collected dataset	feature-based statistics	•	◦
This paper	ISCX2016, MIRAGE2019, NUPT2023	feature-based statistics	•	•

(5) The framework proposed in this paper is tested on public datasets and private datasets. After conducting extensive comparisons and analysis, the proposed solution achieves an 88.1% accuracy with a processing time of 809.99 seconds for the NJUPT2023 dataset. In contrast, the state-of-the-art solutions achieve an accuracy ranging from 48.22% to 79.46% for the same dataset.

The paper is structured as follows: Section 2 covers existing methods for ETC and CL. Section 3 provides a detailed description of the Multi-ARCL framework. Section 4 provides an overview of the experiment and evaluation, and discusses the results of the experiment. Finally, section 5 summarizes the paper and discusses future work.

## 2. Related works

### 2.1. Deep learning based traffic classification

DL, also known as deep structured learning or hierarchical learning, is achieved by learning data representation [38,39]. Compared to machine learning algorithms, DL can automatically extract features without human intervention, which makes it ideal for TC [40,41]. (See Table 1.)

There have been many scholars who have studied how to apply the DL to the TC [42–45]. Dicks and Chavula [46] explore the computational efficiency and accuracy of Long Short-Term Memory (LSTM) and Multi-Layer Perceptron (MLP) deep learning models for packet-based classification of traffic in a community network. Aceto et al. [32] study distiller: encrypted traffic classification via multimodal multitask deep learning. To this end, a novel multimodal multitask deep learning approach for traffic classification is proposed, leading to the distiller classifier. Guan et al. [47] propose a traffic classification method based on deep transfer learning for 5G IoT scenarios with scarce labeled data and limited computing capability and train the classification model by weight transferring and neural network fine-tuning. Pang et al. [48] present a multimodal classification method named MTCM to exploit the context for the classification task systematically. However, the following limitations remain: (1) the traffic representation is generated from raw packet bytes, resulting in the absence of critical information; (2) the model structure of directly applying deep learning algorithms does not take traffic features into account. The preprocessing phase of the proposed end-to-end encrypted traffic classification method by Wang et al. [49] uses the USTC-TL2016 tool to generate byte data of raw traffic in IDX3 files for input data of Convolutional Neural Networks (CNN) model, which is different from a traditional divide-and-conquer method. Wang et al. [31] propose a dynamical MLP-based detection method against DDoS attacks by combining sequential feature selection and feedback mechanism. The proposed method effectively perceives the detection errors when their saliency accumulates to a certain degree and then reconstructs the detector according to updated data. The results showed that the proposed method had comparable detection per-

formance on the popular benchmark data NSL-KDD compared with some related works.

However, these articles focus on accurately identifying applications and ignore the influence of new applications.

### 2.2. Continual learning based traffic classification

There are three primary categories of CL [50]: Task Incremental Learning (TIL), Domain Incremental Learning (DIL), and Continual Incremental Learning (CIL). TIL requires data for different tasks to arrive at different times. But current network applications change constantly, making TIL repeatedly train for various tasks, causing high resource overhead and being hard to fit the complex network environment. DIL involves CL of data arriving at different times, yet it's challenging for DIL to handle the continuous increase in new class classifications. CIL, a recent CL branch, gradually adds new classes or labels during model training to adapt to new classes. Hence, CIL is more suitable for complex and changing network scenarios.

Currently, CIL has many well-known methods widely used in the field of computer vision [51]. BiC is an approach that handles catastrophic forgetting by adding a linear model after the last fully connected layer to correct the bias toward new classes. Wu et al. [26] introduced a bias correction layer to address the imbalance responsible for catastrophic forgetting. iCaRL [52] method using fine-tuning with classification and distillation losses to prevent catastrophic forgetting. In addition, Zhou et al. [53] propose a python toolbox that implements several key algorithms for class-incremental learning to ease the burden of researchers in the machine learning community.

Scientists are already trying to use CIL for ETC. Bovenzi et al. [35] explore CIL techniques in DL-TC, evaluate their performance, and discuss their research potential. The authors focus on reviewing a large number of state-of-the-art CIL methods for DL-based TC, comparing their design principles and providing their working mechanisms, and evaluating the performance of the CIL methods through experiments on the MIRAGE2019 dataset. Wu et al. [33] focuses on how to use CNN for online multimedia traffic categorization. The article proposes a class-incremental learning model based on convolutional neural networks to retain knowledge of old classes while learning new classes of data. The framework is capable of fast and accurate traffic categorization and remains efficient even in the presence of increasing traffic. The paper also describes the use of the sliding window technique for feature extraction and the use of techniques such as knowledge distillation and bias correction for incremental learning of traffic categories. The proposed incremental learning framework by Tian et al. [54] addresses the challenge of adding new applications to the classification system while preserving the learned knowledge of the existing classifier. The framework is based on the one vs rest (OvR) strategy and neural network classifiers, and it uses a sample selection algorithm to balance training effort and classification accuracy. Experimental results show

that the proposed framework achieves incremental learning with high classification accuracy and significantly reduces training efforts. Zhu et al. [36] propose an incremental learning method called ILETc for encrypted traffic classification. ILETc replays the knowledge of old classes using generated samples and exemplars when learning from new classes, mitigating catastrophic forgetting. The performance of ILETc is evaluated using the ISCX VPN-nonVPN dataset and a self-collected dataset, demonstrating superior accuracy compared to state-of-the-art methods. Li et al. [37] propose an Incremental Learning (IL) framework called Multi-view Sequences FuSion (MISS) to keep models evolving with new applications.

As shown in Table 1, there are many scholars working on the sustainable classification of network traffic and achieving good results [55]. However, researchers focus on accurately identifying new applications and ignore the influence of silent applications.

### 3. Methodology

This section comprehensively details the proposed Multi-ARCL framework. Our discussion covers several key areas: a formalized depiction of the problem, an overview of the method's workflow, a multimodal feature extraction method, and an adaptive relay-based continual learning method.

#### 3.1. Problem formulation

This paper focuses on the implementation of ETC from the perspective of traffic management and service requirements. In order to provide a clear and comprehensible explanation, the following definitions are given:

The packet set is formed by extracting TCP and UDP packets from the application traffic [15],  $P = \{p_1, p_2, \dots, p_i\}$ ,  $0 \leq i \leq M$ ,  $M$  is the number of packets in the flow. The packets in the packet set are subsequently utilized to form the flow set using quintuples.  $F = \{f_1, f_2, f_3, \dots, f_i\}$ ,  $0 \leq i \leq K$ ,  $K$  is the number of flows. Next, we use the quintuple as a token to represent the flow and store it in the hash table. Additionally, we record the timestamp of the first and last packet of the flow. If the flow's duration surpasses 120 seconds, the flow is truncated and the information in the hash table is cleared. Subsequent packets will be treated as new flows. To extract session characteristics, we establish the source-destination IP of the first packet as the flow direction. The features are extracted from the forward and backward flows  $SS = \{f_1, f'_1, f_2, f'_2, \dots\}$ . The computation of features is performed on the edge device using the method of feature extraction. In each session, bi-directional flow feature is defined as  $FF = \{ff_1, ff_2, ff_3, \dots\}$ . By definition, there are three levels, which are packet level, flow level, and statistical features. The flow feature vector  $FF$  is transformed into  $sFF$  after feature scaling, which is the fundamental unit of the model input. A batch of  $sFF$  serves as input to the model, which becomes  $x$ .

A list of abbreviations is presented in Table 2 to facilitate understanding of the notations adopted in what follows.

#### 3.2. Workflow of multi-ARCL

The framework comprises two parts: multimodal feature extraction and ARCL.

As shown in Fig. 2, this framework prioritizes the use of multimodal feature extraction methods. This method extracts packet features, flow features, and payload semantic information simultaneously. After processing the semantic information with word vectors, it combines with the packet-level and flow-level features to create a new multimodal feature.

ARCL's solution is constructed as a two-stage process: a model is acquired in the  $i - th$  stage, and an updated model is created in the  $(i + 1) - th$  stage. The first stage aims to guarantee a good initial model in the  $i - th$  stage, while the second stage removes the parameters and data

**Table 2**

List of abbreviation.

Notation	Remark
$P$	A set of packets, where each packet serves as the fundamental unit of a flow.
$F$	A set of flows filtered based on quintuples.
$SS$	A set of session flows.
$FF$	The set of features of a bi-directional flow.
$sFF$	Scaled set of features of a bi-directional flow
$f_i$	The $i - th$ forward flow
$f'_i$	The $i - th$ backward flow
$x$	A batch of $sFF$ serves as input to the model
$y$	Label of the flow feature
$o$	Category of the classifier output
$M$	Number of new applications
$N$	Number of old applications
$J$	Number of silent applications
$m$	Number of labels for new applications
$n$	Number of labels for old applications
$j$	Number of labels for silent applications
$D_n$	The set of labels for old applications
$D_j$	The set of labels for silent applications
$D_m$	The set of labels for new applications
$FN$	Number of packets in a flow
$\alpha$	A smoothing parameter in Smooth Inverse Frequency(SIF).
$w_i$	SIF weight of the $i$ th word
$v_i$	Word vector of the $i$ th word
$MultiFF$	Multimodal data as input to the model
$A$	The original label set of active applications
$B$	The new label set of active applications

of silent applications, reduces the forgetting of the active application, and keeps the new application in the  $(i + 1) - th$  stage.

In the  $i - th$  stage of the model, the initial classification capability is crucial in determining the subsequent capability. This capability provides accurate and sufficient information for stable inheritance in the next stage. In the  $i - th$  stage, the model extracts the flow feature matrix  $FF$  first by the flow feature extraction module, and the feature selection module first selects the features with higher contribution from  $FF$  to get a new subset of features. The feature scaling module normalizes flow features to obtain  $sFF$ .  $sFF$  is then used to train an initialized classification model during the first training session, and the resulting model is saved.

During the updating process, CL faces two challenges. The first challenge is to discard the parameters of silent apps and mitigate the catastrophic forgetting of active apps. The second challenge is to improve the accuracy of new apps. To address these challenges, we discard the data from inactive applications and select representative samples of active applications and all new applications to reform the exemplar dataset. Since forgetting can only be slowed down but not avoided, in the  $(i + 1) - th$  stage, the model will first freeze the parameters of the previous convolutional layer, automatically find the parameters of the active applications in each iteration, and then unfreeze the parameters of the convolutional layer.

#### 3.3. Multimodal feature extraction method

Processing and extracting information from  $P$  is a crucial step. Convert the information in  $P$  to decimal format, and then calculate the temporal features, protocol headers, and payload features. For protocol features, if the number of packets in a flow is less than  $FN$ , the remaining values are padded with zeros. Temporal features include the mean, variance, and standard deviation of the inter-packet time intervals within a flow, as well as the mean, variance, and standard deviation of packet lengths within a flow. The payload features represent the part of the network packet that carries the actual data. Each packet's payload can contain multiple bytes, and each byte value can be considered a word.



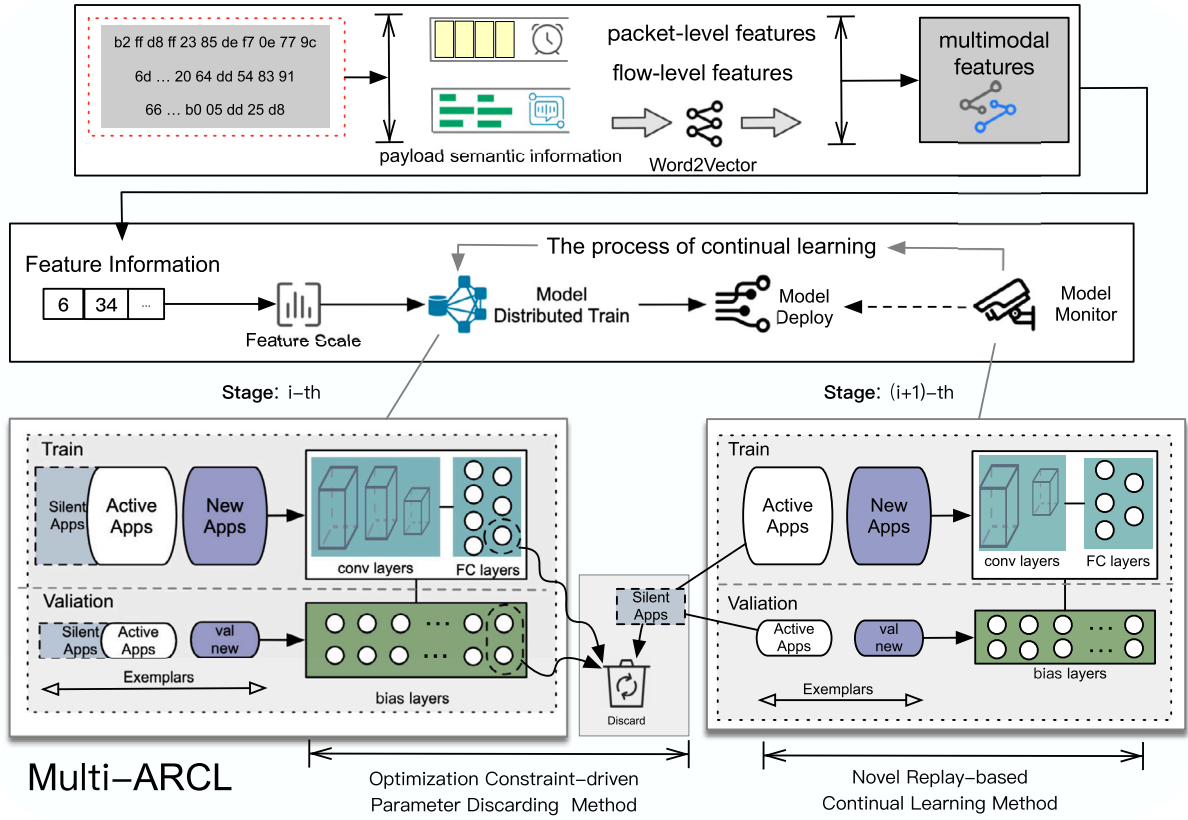


Fig. 2. The framework of the proposed method.

**Payload Feature Extraction.** In the payload part of a packet, each byte is defined as a packet word (similar to a word in text, for the sake of understanding, we define it as a packet word), and all the words in the payload are sequentially composed into a packet sentence (similar to a sentence in text, for the sake of understanding, we define it as a packet sentence). The packet sentences of the first  $FN$  packets of each flow are concatenated in order to form a flow paragraph (similar to a paragraph in text, for the sake of understanding, we define it as a flow paragraph). First, extract the sentences of the first  $FN$  packets of each flow and concatenate them in order to form the flow paragraph. If there is no sentence in the first  $FN$  packets, then the value of its flow paragraph is set to -1. Then, use word2vec to learn the flow paragraphs of all flows in the dataset to obtain the word vector for each packet word. To enhance the model's detection speed, we utilize a process of 20 packets before a stream, denoted as  $FN$  equals 20.

For each vocabulary in the payload, calculate the probability of its occurrence in the entire payload, and based on this probability, calculate the SIF weight. The higher the occurrence probability, the smaller the weight, reducing the impact of common words on the results. The calculation formula is as follows, where  $w_i$  is the SIF weight of the  $i$ -th word,  $a$  is a smoothing parameter, and  $p(i)$  is the word frequency.

$$w_i = \frac{a}{a + p(i)} \quad (1)$$

Next, calculate the SIF weighted average word vector. The calculation formula is as follows, where  $V$  is the weighted average word vector, and  $v_i$  is the word vector of the  $i$ -th word.

$$V = \frac{\sum_{i=1}^n w_i v_i}{\sum_{i=1}^n w_i} \quad (2)$$

Finally, subtract the projection on the first principal component of the flow paragraph from the weighted average word vector,  $V' = V -$

$PV_1$ , where  $V_1$  is the first principal component and  $P$  is the projection vector of  $V$  on  $V_1$ .

The length of the principal component is the dimension  $N_2$  of the principal component, so after this calculation, the dimension of the sentence vector is fixed at  $N_2$ . Therefore, the payload feature dimension of each flow is  $(1, N_2)$ .

Before extracting features, it is necessary to normalize the byte information. This involves converting each byte from hexadecimal to decimal within the range of 0 to 255 and then dividing by 255 to ensure that each byte falls within the range of 0 to 1.

#### 3.4. Adaptive relay-based continual learning

The convolutional neural network serves as the foundation of our model, and it utilizes convolutional operations to capture local features in a systematic manner. Moreover, the convolutional layer can perform a dot product of each feature in the input data with a convolutional kernel [56].

The model's design is presented in Fig. 3. We connect multiple convolutional layers before adding an application classifier layer that maps features to labels. For outputting predicted probabilities, we utilize a fully connected layer (FC) as the classifier layer in this paper.

Currently, continual learning encounters two significant issues. Initially, the model has a tendency to forget previous classes when it learns new ones as model weights are updated. Additionally, old applications are still disappearing, which leads to a significant loss of accuracy. In addition, continual learning requires saving historical data, resulting in increased storage usage due to silent applications and additional costs. To address these problems, we discard parameters and constrain optimization to address the issue of silent applications faced by CL.

The following algorithm shows that a classifier with  $n$  number of classifications has been trained so far. When the set of silent applications and new applications is known, it becomes critical to retain the

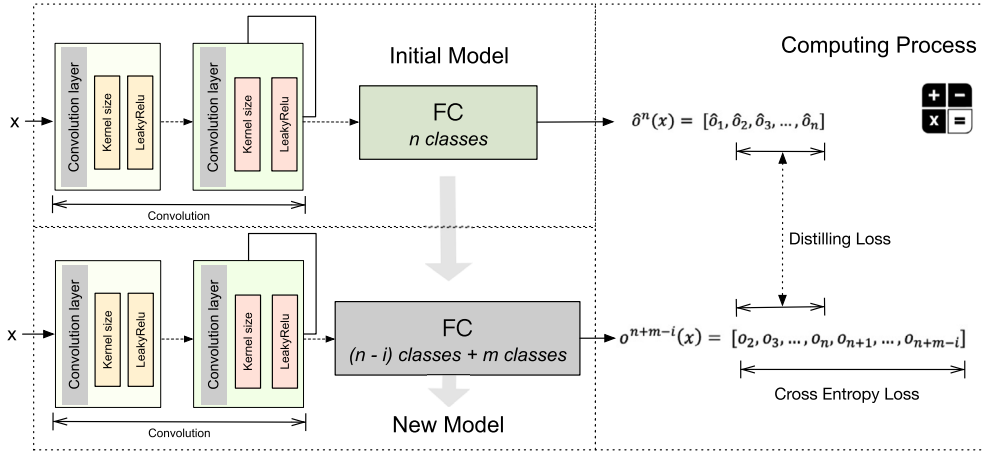


Fig. 3. Comparison of model structures before and after classes change.

knowledge of the active applications in the original classifier while incorporating the knowledge of new applications.

**Algorithm 1** Adaptive Relay-based Continual Learning.

**Input:**

Old applications,  $X^n = (x^i, y^i)$ ,  $0 \leq i \leq N$ ,  $y \in D_n$ ; Silent applications,  $X^j = (x^i, y^i)$ ,  $0 \leq i \leq J$ ,  $y \in D_j$ ; New applications,  $X^m = (x^i, y^i)$ ,  $0 \leq i \leq M$ ,  $y \in D_m$ ; Parameters of TC,  $H^i = \theta_c, \theta_f$ ;  $\theta_c$  represents the parameter of the convolutional layer;  $\theta_f$  represents the parameter of FC;  $epoch$  is the number of training.

**Output:** New TC-classifier  $H^{i+1}$ .

- 1: Active applications data:  $X_{active} = X^n - X^j = \{x | x \in X^n \wedge x \notin X^j\}$ ;
- 2: New Applications data:  $X_{new} = X_{active} \cup X^m$ ;
- 3: freeze  $\theta_c$ , unfreeze  $\theta_f$ ;
- 4: **for** epoch in  $num\_epochs$  **do**
- 5:  $p, y \leftarrow H^i(x)$
- 6:  $\theta_f^* \leftarrow \text{argmin}(-\frac{1}{S} \sum_{i=1}^S \sum_{c=1}^{n-j} y_{ic} \log(p_{ic}))$ ;
- 7: **end for**
- 8: unfreeze  $\theta_c$ ;
- 9: **for** epoch in  $num\_epochs$  **do**
- 10:  $\theta_f^*, \theta_c^* \leftarrow \text{argmin}(\frac{n-j}{n+m-j} \times loss_{soft-target} + \frac{m}{n+m-j} \times loss_{hard-target})$ ;
- 11: **end for**

**3.4.1. Optimization constraint-driven parameter discarding method**

This method involves several steps, including label mapping, freezing a portion of the model, deleting neurons for silent applications, and unfreezing the model.

The first step of this method is to map the label domain of the active application to the new label domain. This involves establishing a correspondence between the original label collection and the rearranged label collection. The mapping relationship is represented in the following form:  $f: A \rightarrow B$ . The given equation represents a labeled mapping function, where  $A$  is the definition domain and  $B$  is the accompanying domain. For each element  $x$  in  $A$ , there is a unique corresponding element  $y$  in  $B$ , determined by the function  $f$ . We assume that Category 1 and Category 2 are silent applications. Since the number of silent applications is 2, this correspondence is denoted by the equation:

$$y = f(x) = \begin{cases} 0, & x = 0 \\ 1, & x = 3 \\ 2, & x = 4 \\ \dots \\ n-2, & x = n \end{cases}, \quad x \in A \quad (3)$$

The parameters of the neurons in the first half of the neural network must be frozen. This means that during training, the parameters will not be updated and the gradient will not be calculated. The formula for

freezing the parameters is denoted as follows:  $\frac{\partial L}{\partial \theta} = 0$ . The loss function ( $L$ ) and frozen parameter ( $\theta$ ) are used to eliminate neurons of silent applications in the discriminative layer before performing fine-tuning. Once fine-tuning is complete, the first half of the neuron parameters are unfrozen. The formula for unfreezing the parameters is denoted as follows:  $\frac{\partial L}{\partial \theta} = \alpha \frac{\partial L}{\partial \theta}$ ,  $\alpha$  is the degree of the unfreezing parameter. When  $\alpha = 1$ , the unfreezing is complete. Finally, the labels are remapped to the original domain using an inverse function,  $x = f^{-1}(y)$ .

Generally, our elimination of neuron parameters for silent applications mitigates model stability decay and provides a good foundation for the next stage of learning new applications for the model.

**3.4.2. Novel replay-based continual learning method**

We divide the entire continual learning process into two phases to ensure effective implementation. In the initial phase, an initial network classification model is trained with the assumption that the number of classes is  $n$ . The original training data is then separated into a training set and a validation set. Once an initial model is obtained, the second phase of continual learning is executed. This phase is based on the classes that disappear and new classes that must be added. During the second phase, the model learns new applications while disregarding silent applications. In the traditional incremental process, the former model would recognize  $n$  old classes and still need to learn  $m$  new classes. Therefore, we will consider the model's need to recognize  $m + n$  classes as a new class. We define the set of classes that need to be newly learned:  $X^m = \{(x^i, y^i), 0 \leq i \leq M, y \in D_m\}$ , define old classes,  $X^n = \{(x^i, y^i), 0 \leq i \leq N, y \in D_n\}$ . The outputs of the old and new classifiers are defined as follows:  $\delta^n = [o_1, \dots, o_n]$ ,  $o^n = [o_1, \dots, o_n, o_{n+1}, \dots, o_m]$ . The previous class incremental continual learning looks for the neuron parameters of old applications in the new model and computes the distillation loss between the two. Since there is a downgrading of old applications, the neuron parameters used to recognize the silent applications in the old model become invalid. So we define the set of active applications:  $X^{(n-j)} = (x^i, y^i)$ ,  $0 \leq i \leq N - J$ ,  $y \in D_n \wedge y \notin D_j$ .  $J$  is the number of silent applications. Definition of old and new classifier:  $\delta^n = [o_1, \dots, o_n]$ ,  $o^n = [o_1, \dots, o_{n-j}, o_{n-j+1}, \dots, o_{n+m-j}]$ .

First, we initialize the model to get the classification probability.  $p = model_{pre}(X^{(n-j)})$ , where  $model_{pre}$  is the initial model and  $p$  is the predicted score. Then, the scores are corrected using the bias layer.  $y = A \times p + B$ .  $A$  and  $B$  are the two-parameter matrices of the correction layer, and  $y$  is the initialized model to obtain the final classification probability. The model design for continuous learning is shown in Fig. 3. The old model automatically finds the parameters of old applications that have not been downgraded at each iteration and calculates the distillation loss with respect to the output probability of the old application that has not disappeared in the new model. We define distillation loss:

$$y = \frac{e^{y_i}}{\sum_{i=1}^{n-j} e^{y_i}} \quad (4)$$

$$\log(y) = \log\left(\frac{e^{y_i}}{\sum_{i=1}^{n-j} e^{y_i}}\right) \quad (5)$$

$$loss_{soft-target} = -\frac{1}{S} \sum_{i=1}^S \sum_{k=1}^{n-j} y_{ik} \log(y_{ik}) \quad (6)$$

In this way, this distillation loss allows the deep network to retain as much knowledge as possible about the active applications in the classifier. We then use the cross entropy loss as the classification loss to ensure classification accuracy for new classes.

$$loss_{hard-target} = -\frac{1}{S} \sum_{i=1}^S \sum_{c=1}^{n+m-j} y_{ic} \log(p_{ic}) \quad (7)$$

Where  $S$  is the number of samples,  $n+m-j$  is the number of categories,  $y_{ic}$  is the true label of the  $i$ th sample, which is 1 if it belongs to the  $c$ -th category and 0 otherwise, and  $p_{ic}$  is the predicted probability that the  $i$ -th sample belongs to the  $c$ -th category. The total loss combines the distillation loss and the categorization loss as follows:

$$loss = \lambda \times loss_{soft-target} + (1 - \lambda) \times loss_{hard-target} \quad (8)$$

$\lambda$  is the equilibrium distillation and classification loss.  $\lambda = \frac{n-j}{n+m-j}$ , where  $n-j$  is the number of active applications and  $m$  is the number of new applications.

By optimizing the loss function training as described above, we obtain a classifier that can identify new classes and significantly reduce catastrophic forgetting. In the next section, we will validate our approach through experiments.

## 4. Experiments

### 4.1. Datasets and experimental settings

The MIRAGE2019 dataset [57] is designed for mobile traffic analysis and contains real data related to mobile applications, with the goal of advancing the state of the art in mobile app traffic analysis. The dataset covers multiple domains such as social, video, music, games, news, etc., and provides detailed information for each traffic packet, including timestamp, source and destination address, protocol, length, etc. Additionally, the dataset includes a label for every traffic packet, rendering it comprehensive and coherent. The MIRAGE2019 dataset is useful for examining various elements of mobile app traffic, including its characteristics, classification, privacy, optimization, and other related factors. The ISCX2016 dataset [58] is utilized for cybersecurity research and comprises two sub-datasets: ISCXVPN2016 and ISCX-Tor2016. ISCXVPN2016 encompasses network traffic with and without VPNs, spanning applications and protocols such as VoIP, P2P, FTP, and video streams. The dataset also includes labels for each stream, indicating details such as the application name and user action. This was an experiment to expand the original dataset with SMOTE [59] since the original dataset was small.

The private dataset utilized in this study is composed of more than 19 popular applications collected in the campus network scenario. Table 3 displays the part regarding applications, encompassing diverse types of apps including those designated for video, music, gaming, and social media. The dataset of NJUPT2023 contains 91 features. Fig. 4 shows the statistical distribution of features and feature outliers. The distribution of features was selected as the 29 most typical features. "fTcPwindow" dispersion, considered a typical feature, is prominent. The number of samples in the part close to 0 on the left is large. The vast region on the right also has a large number of samples.

The experimental environment is Intel Core, 32GB RAM, NVIDIA GTX 4080. In addition, four 3080 and one 3060 graphics cards were used

**Table 3**

The number of categories in the dataset is presented.

MIRAGE2019		NJUPT2023	
applications	amount	applications	amount
Accuweather	3991	QQ music	39465
Contextlogic	2792	Garena	21872
Dropbox	2635	VR	33611
Duolingo	3922	background	62228
Facebook	3195	MOOC	14842
Groupon	839	Peacekeeper	12120
Hypah	1583	bilibili	20014
Iconology	2806	Weibo	22061
Joelapenna	3443	TikTok	16640
Motain	5215	DOYU	15821
Pinterest	2152	NARUTO	27240
Spotify	3514	iQIYI	36740
Subito	3538	HOK	42734
Trello	1310	Tieba	18206
Tripadvisor	1260	ZH	16643
Twitter	2407	TencentVideo	11913
Viber	1112	LoLWR	19841
Waze	5278	HUYA	108757
Youtube	2507	TFT	23718

**Table 4**

Experimental equipment.

Name	Specification
CPU	13th Gen Intel Core i7-13700KF
Memory Capacity	32GB RAM
Graphics Card	NVIDIA GeForce RTX 4080, 3080 and 3060
Python Version	3.10

to test the training efficiency of distributed training. Specific parameters are shown in Table 4. In this paper, Python3 is the main programming language. The following evaluation metrics are described: Precision, Recall, F1, and Accuracy.

Since this experiment focuses on the impact of accuracy and consumption from the perspective of silent applications, we do not consider other hardware indicators.

### 4.2. Experimental operation

In this study, we examine the challenge of CL, or the ability to selectively unlearn irrelevant knowledge while retaining useful knowledge and acquiring new information over time. As shown in Fig. 5, we evaluate our proposed method using three datasets: MIRAGE2019, ISCX2016, and NJUPT2023.

At the initial stage of the experiment, we conducted comparative tests using different modalities to verify the superiority of multimodal feature representation. In the testing phase of ARCL and SOTA methods, we used a 4080 graphics card for training, and tested the accuracy and time efficiency of different methods in this equipment environment. We train an initial model based on the existing applications. Then, we simulate and test the class change scenario. In the MIRAGE2019 and NJUPT2023 datasets, we first train a model with 16 classifiers. Then, the simulated scenarios are the disappearance of two application streams and the appearance of three new applications. In the ISCX2016 dataset, we first train a model with 4 classifiers and then simulate a scenario where one application disappears and two new applications appear. In the following, we will test our proposed method and three other CL methods, Fine-tune, Bic, and iCaRL. Additionally, due to the complex and dynamic nature of network traffic, rapid updates are essential, and distributed learning can significantly enhance training efficiency. In this experiment, we aim to verify the optimal performance of distributed learning by varying the number of GPUs and the batch size.

Fig. 6 illustrates the entire process from distributed data collection by edge devices to its final integration into a distributed learning

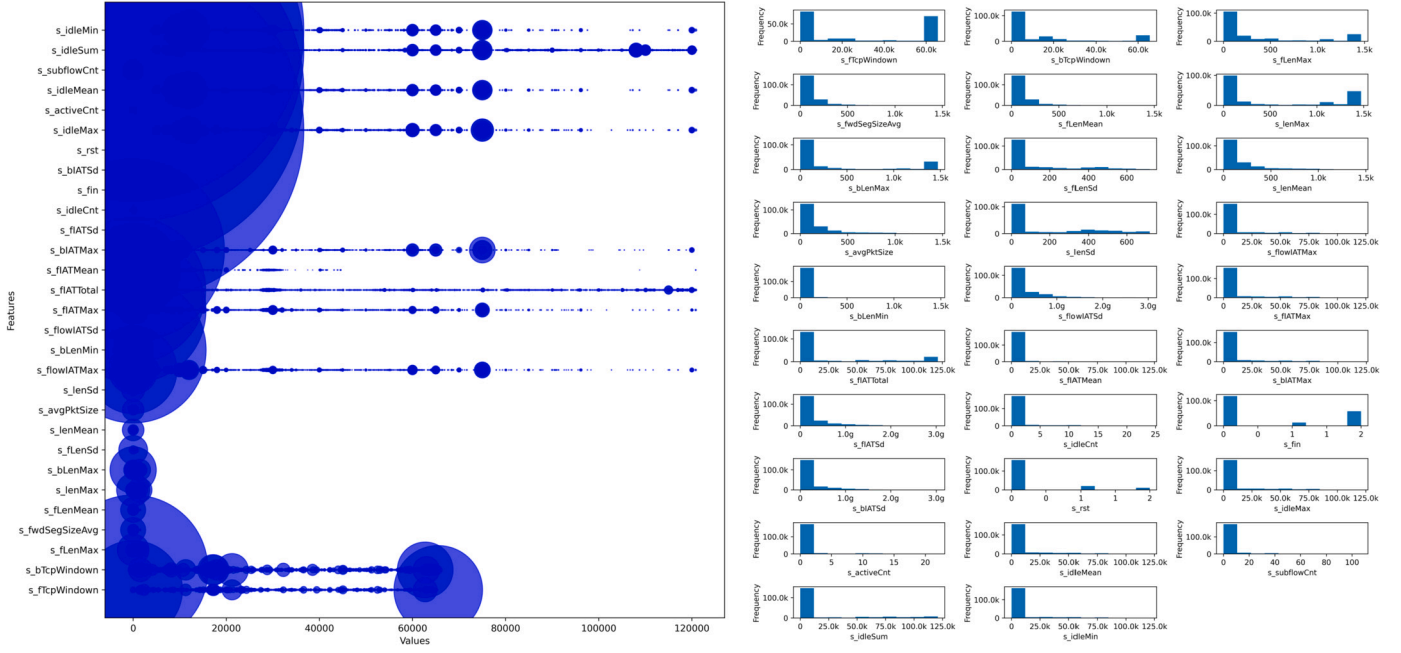


Fig. 4. Distribution of feature.

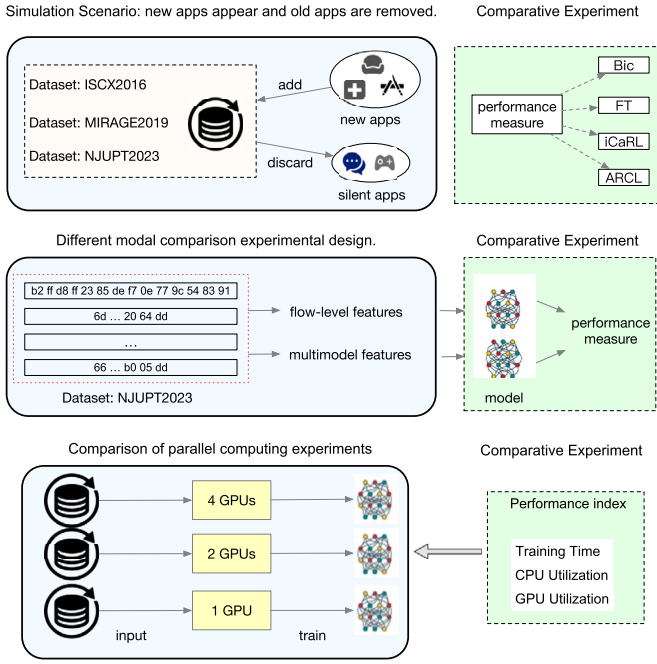


Fig. 5. Experimental design.

framework. Multiple mobile PCs, located in different positions, gather network traffic data. On individual terminals, we use PCAPdroid to label network traffic. Network traffic is captured by Wireshark on the router and then filtered and compared based on five-tuples. After filtering, the router feature calculation module computes network traffic features for each application's PCAP file and transmits them to the cloud data center for storage. Subsequently, in the distributed data parallel processing framework, different DDP processes synchronize model gradients through AllReduce operations, with these gradients allocated to different buckets. Finally, in the distributed training environment, each model utilizes the synchronized gradients to update parameters, gradually optimizing model performance.

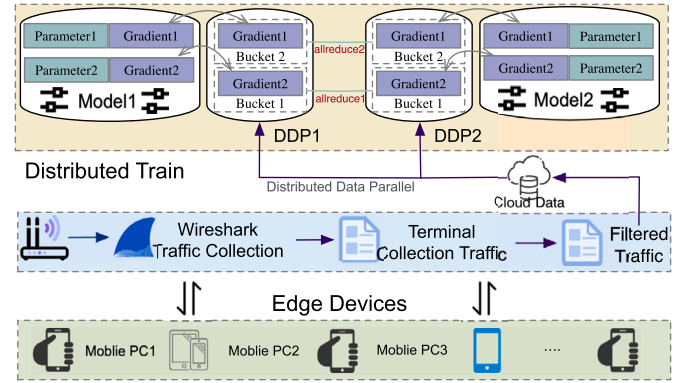


Fig. 6. The workflow of distributed edge computing and training.

### 4.3. Evaluation results

#### 4.3.1. Comparison with different modes

From Table 5, the multimodal model demonstrates high effectiveness across various applications, with particularly exceptional performance in “Eggy Party” (F1 score of 0.9761), “Sausage Party” (F1 score of 0.9635), and “HOK” (F1 score of 0.9542). “LoLWR” presents the lowest scores in precision (0.6098), recall (0.6667), and F1 (0.6369), indicating this area might require specific attention or adjustments in the model for better performance. From Table 6, the application domain where the single-modal model performs best is “Eggy Party”, which has the highest F1 score of 0.9078. The worst-performing application domain is “LoLWR”, which has the lowest F1 score of 0.3771. Due to the interactive nature of the LoLWR application, network traffic is difficult to capture. As a result, both single-modal and multimodal models perform poorly. The overall performance of the multimodal model, with an average F1 score of 0.9356 and an accuracy of 0.9353, is considerably higher than that of the single-modal model, which has an average F1 score of 0.8141 and an accuracy of 0.8136. This demonstrates the effectiveness of integrating multiple modes of data, which helps in capturing a broader range of features and improving the model's robustness and generalization.



**Table 5**  
Multimodal experimental results.

applications	precision	recall	f1
HOK	0.9640	0.9446	0.9542
LoLWR	0.6098	0.6667	0.6369
Honkai	0.8578	0.9257	0.8905
Peacekeeper	0.9193	0.954	0.9363
Sausage Party	0.9670	0.9600	0.9635
Eggy Party	0.9866	0.9658	0.9761
Genshin	0.9019	0.9197	0.9107
background	0.8747	0.8329	0.8533
avg	0.9363	0.9353	0.9356
acc	0.9353		

**Table 6**  
Single-modal experimental results.

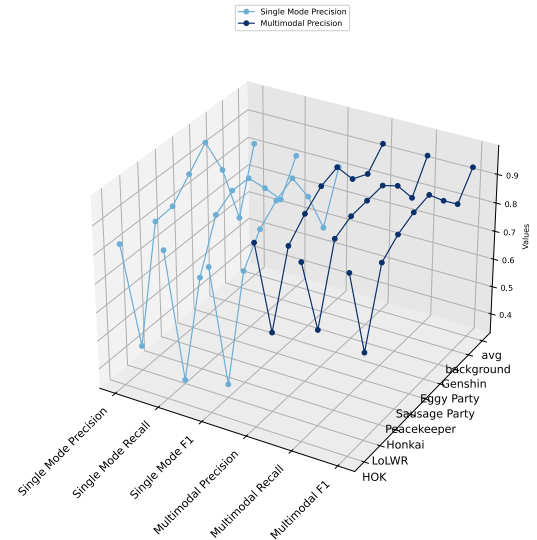
applications	precision	recall	f1
HOK	0.827	0.8505	0.8386
LoLWR	0.4179	0.3436	0.3771
Honkai	0.8107	0.6589	0.7269
Peacekeeper	0.8173	0.83	0.8236
Sausage Party	0.884	0.8686	0.8762
Eggy Party	0.9507	0.8686	0.9078
Genshin	0.8102	0.7867	0.7983
background	0.5913	0.7006	0.6413
avg	0.8165	0.8136	0.8141
acc	0.8136		

Fig. 7 displays a comparison of precision, recall, and F1 scores obtained after training with different modal features. Within the single-modal model, we can observe that the model's performance across the "HOK", "Peacekeeper", "Sausage Party", "Eggy Party", and "Genshin" application domains are relatively balanced, with precision, recall, and F1 scores that are closely aligned and comparatively high. In contrast, the scores for "LoLWR" and "background" are significantly lower, with "LoLWR" performing particularly poorly in terms of recall. In the multimodal model, we can see that the precision, recall, and F1 scores across all applications are very close to each other and are generally higher than those of the single-modal model, indicating that model performance is enhanced when combining traditional statistical features with encrypted information. The performance in "LoLWR" remains the lowest for both models, but the multimodal model shows a marked improvement in this application compared to the single-modal model. The performance in "background" also improved in the multimodal model compared to the single-modal model, but it still scored lower than other applications.

Overall, the multimodal model is more effective at processing and integrating information from different sources, thereby enhancing the model's overall performance, especially in areas where the single-modal model underperforms. This also suggests that for complex or diverse datasets, adopting a multimodal approach could be an effective strategy to enhance performance.

#### 4.3.2. Ablation study

Table 7 presents performance metrics for models on three datasets in relation to changes in class distributions. "Class changed" refers to scenarios involving the addition of new classes and the removal of old ones. Analyzing these metrics reveals that the "Initial model" consistently achieves superior performance when class distributions are stable, with impressive accuracy scores such as 0.9753 for MIRAGE2019 and similarly high marks for the other datasets. However, when faced with changes—such as the addition of new applications and the retirement of old ones—the performance of the "Initial model" drops significantly across all datasets. In contrast, ARCL stands out for its robustness, consistently maintaining high accuracy rates close to those of the "Initial model". This resilience to classes change, as demonstrated by ARCL,

**Fig. 7.** Comparison with different modals.**Table 7**  
Improved model comparison.

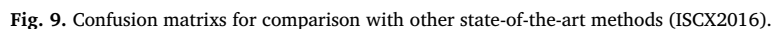
Dataset	Model	Precision	Recall	F1	Accuracy
MIRAGE	Initial model	0.9666	0.9631	0.9641	0.9753
	Class changed	0.8608	0.8014	0.8300	0.7967
	ARCL	0.8945	0.8809	0.8813	0.9157
ISCX	Initial model	0.9579	0.9572	0.9572	0.9573
	Class changed	0.6560	0.5818	0.6167	0.5818
	ARCL	0.9236	0.9218	0.9213	0.9218
NJUPT	Initial model	0.9158	0.9139	0.9136	0.9139
	Class changed	0.7919	0.7339	0.7618	0.7339
	ARCL	0.8839	0.8810	0.8790	0.8810

is particularly valuable in real-world applications. Overall, the "Initial model" shows excellent performance across all three datasets under stable class conditions, demonstrating its effectiveness in a static environment. Upon classes changing, all models experience some degree of performance degradation. However, ARCL exhibits a less pronounced decline in performance, indicating its superior capability to handle dynamic class variations. This robustness is a valuable attribute for practical applications, as real-world data often changes over time.

#### 4.3.3. Comparison with other state-of-the-art methods

We use four different incremental learning models, Fine-tune, Bic, iCaRL, and ARCL, to compare their classification accuracies. Fine-tune is a basic method that trains new knowledge directly on top of the original model, but it is prone to the forgetting phenomenon. Bic is a model based on knowledge distillation, which trains new knowledge while retaining the important features of the old knowledge to minimize forgetting. iCaRL is a model based on memory replay, which trains new knowledge while repeatedly training some representative samples of the old knowledge to keep the knowledge balanced.

Fig. 8 and 9 show that Fine-tune does not work well on the public dataset ISCX2016. In particular, Fine-tune detects new classes with a large number of misidentifications. Fine-tune suffers from catastrophic forgetting on both NJUPT2023 and MIRAGE2019. Pinterest is misidentified as Viber. Tripadvisor is misidentified as Groupon and Twitter. Bic on the MIRAGE2019 dataset showed very clear catastrophic forgetting. Hypah was misidentified as an Accuweather application because the method is unable to forget silent applications. The Bic method also suffered from catastrophic forgetting on NJUPT2023. Most active applications were misidentified as silent applications. iCaRL performed about the same as Bic on ISCX2016 and MIRAGE2019. However, iCaRL per-



As can be seen from Fig. 10, we can conclude that ARCL has the optimal performance on all datasets and all metrics, indicating that it is an effective and robust ETC method that can accurately identify dif-

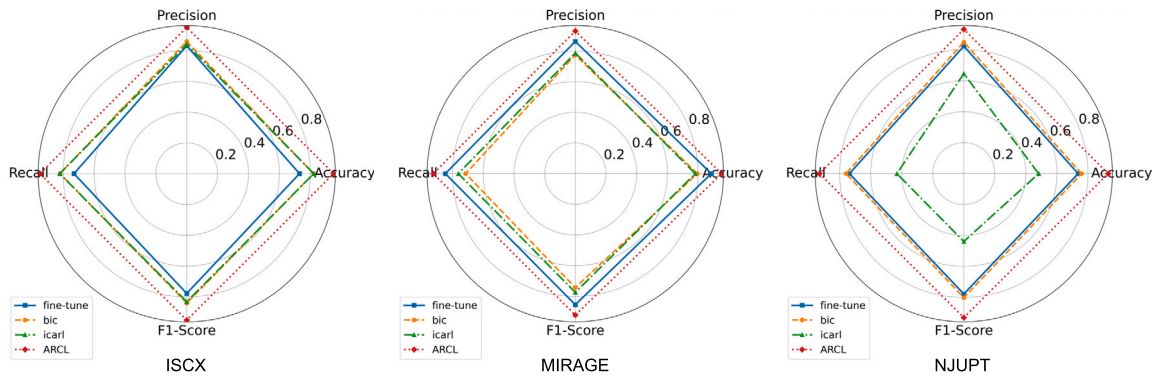


Fig. 10. Radar charts for comparison with other state-of-the-art methods (ISCX2016, NJUPT2023 and MIRAGE2019).

Table 9

The amount of old data that needs to be trained for different methods (the retrain model, traditional replay- based CL methods and ARCL).

	the retrain model	traditional replay- based CL methods	ARCL
ISCX	30000	4000	3000
MIRAGE	30256	8000	7000
private	140000	16000	14000

Table 10

Comparison results of our method with the retraining model.

Dataset	Model	Precision	Recall	F1	Accuracy
MIRAGE	Retraining	0.9041	0.8950	0.8958	0.9302
	ARCL	0.8945	0.8809	0.8813	0.9157
ISCX	Retraining	0.9390	0.9390	0.9390	0.9390
	ARCL	0.9236	0.9218	0.9213	0.9218
NJUPT	Retraining	0.9204	0.9186	0.9184	0.9186
	ARCL	0.8839	0.8810	0.8790	0.8810

ferent application services under different datasets. Fine-tune has the worst performance and all metrics on the ISCX2016 dataset, showing that it is a simple and basic ETC method that is prone to overfitting or underfitting problems and cannot adapt to intricate scenarios. Bic and iCaRL have moderate performance on all datasets and all metrics, which shows that they have improved network traffic detection capabilities. The mechanisms of knowledge distillation and memory replay are used, which can improve the detection ability to some extent, but there are some problems of information loss and data imbalance.

#### 4.4. Evaluation discussion

We first discuss the resource expenditure of different methods during training, which is mainly divided into two parts: training duration and storage consumption. As shown in Table 9, retraining models require all data from old applications, resulting in substantial storage consumption. Traditional continual learning methods do not need to store all the data from old applications, thereby somewhat alleviating storage resource consumption. ARCL further reduces storage expenditure by removing silent applications from the exemplar set. Although Table 10 shows that the retraining model consistently outperforms ARCL, it achieves the highest accuracy on each dataset, with 0.9302 for MIRAGE2019, 0.9390 for ISCX2016, and 0.9186 for NJUPT2023, indicating a superior ability to correctly label all classes. Despite this, ARCL demonstrates robust performance at a much lower cost.

Table 11 displays the performance of the ARCL's model as it is trained with varying sizes of a dataset specific to "old applications". The columns represent the model's performance when trained with incre-

mentally increasing amounts of replayed data: 500, 1000, 1500, 2000, 3000, 4000, and 5000 samples. As expected, the training time increases with the size of the dataset. It starts at 718.49 seconds for 500 samples and grows to 1555.51 seconds for 5000 samples. The GPU usage remains relatively stable across different dataset sizes, with a slight trend of increasing as more data is used. It starts at 88.4% for 500 samples and slightly increases to around 91.3% when the model is trained with 5000 samples. From Fig. 11, the accuracy improvement appears to be rapid initially as the training size increases from 500 to 1000, after which the rate of improvement slows down.

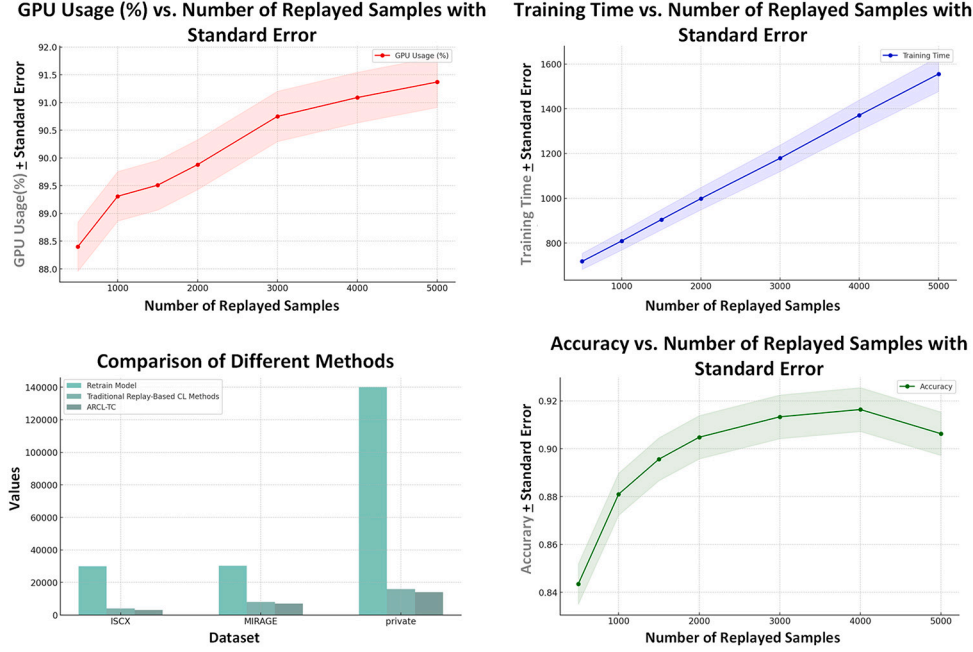
Fig. 11 illustrates a marked increase in the ARCL model's accuracy as the volume of replayed data expands from 500 to 1000. Beyond this point, the accuracy growth becomes more gradual. Upon reaching 3000, the accuracy rate stabilizes. We infer that at the 1000 data quantity mark, the model not only utilizes resources more efficiently but also achieves higher accuracy, making it the most cost-effective option.

Table 12 displays data on the resource usage and computational complexity of different models trained on three datasets. The models compared are Fine-tune, Bic, iCaRL, Retraining, and ARCL. Retraining the model generally takes the longest, particularly on the NJUPT2023 dataset, where it exceeds 1500 seconds. Fine-tune is the fastest across all datasets, suggesting a more efficient learning process. CPU utilization is relatively low for Fine-tune, Bic, and ARCL, typically around 4%. However, iCaRL exhibits a notably higher CPU usage on all datasets, with a particularly high usage of 26.75% on ISCX2016, indicating a potentially more CPU-intensive learning algorithm or less optimization for CPU usage. GPU usage is high for most models, often exceeding 90%, which is typical for deep learning tasks. Bic and Retraining models consistently show high GPU usage across all datasets, while iCaRL uses the GPU less intensively, which could be due to the nature of the incremental learning process iCaRL implements. From this data, we can infer that Retraining is the most resource-intensive approach, as it likely involves retraining the model from scratch or significantly updating it with new data. Fine-tuning is the most resource-efficient, suggesting that adjusting pre-trained models to new data or tasks requires less computational effort. ARCL strikes a balance between resource usage and training time, suggesting it is a more efficient incremental learning method compared to full retraining.

From Fig. 12 and Table 13, it is evident that the processing time varies significantly under different GPU configurations. Generally, with the same batch size, the more GPUs used, the shorter the processing time. For instance, with a batch size of 1024, the processing time with four 3080 GPUs is 656.49 seconds, while it increases to 887.17 seconds with two 3080 GPUs, and further to 1229.93 seconds with a single 3060 GPU. However, if the batch size is too small, the performance advantages of distributed computing cannot be fully utilized. For example, with a batch size of 256, the processing time for the two 3080 GPUs configurations is 2426.36 seconds, compared to 1654.97 seconds with a single 3060 GPU. Therefore, the batch size also has a significant impact on the processing time. Similarly, with the two 3080 GPUs configurations,

**Table 11**  
Comparison results of our method with different replayed sizes.

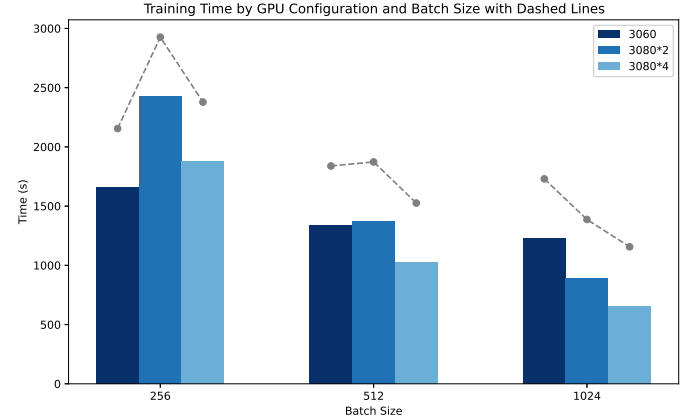
Replayed size	500	1000	1500	2000	3000	4000	5000
Accuracy	84.35%	88.10%	89.56%	90.48%	91.34%	91.64%	90.63%
Training time	718.49	809.99	904.99	998.91	1179.07	1371.16	1555.51
GPU usage	88.40%	89.31%	89.51%	89.88%	90.75%	91.09%	91.37%



**Fig. 11.** Amount of the exemplar dataset for comparison with the resource consumption.

**Table 12**  
Computing resource consumption comparison.

Dataset	Model	Time (s)	CPU (%)	GPU (%)
MIRAGE	Fine-tune	51.06	3.92	77.08
	Bic	196.51	4.12	91.00
	iCaRL	220.99	19.53	66.24
	Retraining	333.57	4.23	90.49
	ARCL	234.74	4.21	88.22
ISCX	Fine-tune	78.61	4.02	78.93
	Bic	309.72	4.15	92.79
	iCaRL	125.57	26.75	59.94
	Retraining	458.71	4.20	91.37
	ARCL	348.88	4.16	91.09
NJUPT	Fine-tune	135.52	4.24	79.76
	Bic	585.27	4.20	93.09
	iCaRL	543.05	12.73	77.93
	Retraining	1,573.21	4.17	91.66
	ARCL	809.99	4.31	89.31



**Fig. 12.** Training time by GPU configuration and batch size.

the processing time is 887.17 seconds for a batch size of 1024, 1373.05 seconds for a batch size of 512, and 2426.36 seconds for a batch size of 256. This indicates that a larger batch size can improve processing efficiency and reduce processing time, especially under multi-GPU configurations. Table 13 indicates that while GPUs are heavily influenced by the number of GPUs and batch sizes, CPUs remain more consistent in their usage, suggesting that the CPU is less of a bottleneck in this distributed learning scenario.

## 5. Conclusions and future work

In this paper, we proposed a novel CL framework called Multi-ARCL, which used multimodal features to train an accurate classifier.

The framework was able to reduce the training overhead of CL by automatically identifying and eliminating silent application data when new applications were acquired. Furthermore, it updated the neural network parameters of active applications, significantly improving classification performance. We evaluated ARCL on three datasets ISCX2016, MIRAGE2019, and self-collected NJUPT2023 and the results showed that ARCL outperformed state-of-the-art methods such as Bic and iCaRL in all metrics. However, our framework still has some limitations, such as low efficiency, and future work will be devoted to accelerating the search process of neural networks and improving the training speed. In addition, our framework does not address another important challenge in open environments, which is how to filter unknown traffic data. In future work, the unknown traffic filtering function will be integrated



**Table 13**  
Distributed learning resource consumption comparison.

		3080*4	3080*2	3060
Batch 1024	time	656.49 s	887.17 s	1229.93
	gpu	56.40% 78.39%	62.33%	88.0604%
	cpu	83.26% 82.84%	43.80%	88.3755%
Batch 512	time	1025.96 s	1373.05 s	1338.88 s
	gpu	77.72% 78.11%	69.89%	87.5677%
	cpu	77.16% 56.35%	52.15%	53.49%
Batch 256	time	1878.23 s	2426.36 s	1654.97 s
	gpu	77.76% 77.26%	63.33%	88.46%
	cpu	77.61% 51.52%	43.80%	11.61%

into the ARCL framework so that it can be adapted to more application scenarios.

### CRedit authorship contribution statement

**Zeyi Li:** Writing – original draft, Software, Methodology, Conceptualization. **Minyao Liu:** Validation, Data curation. **Pan Wang:** Writing – review & editing, Supervision. **Wangyu Su:** Visualization. **Tianshui Chang:** Writing – review & editing. **Xuejiao Chen:** Supervision. **Xi-aokang Zhou:** Supervision.

### Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Pan Wang reports financial support was provided by National Natural Science Foundation of China. If there are other authors, they 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] E. Papadogiannaki, S. Ioannidis, A survey on encrypted network traffic analysis applications, techniques, and countermeasures, *ACM Comput. Surv.* 54 (2021) 1–35.
- [2] X. Zhou, Q. Yang, Q. Liu, W. Liang, K. Wang, Z. Liu, J. Ma, Q. Jin, Spatial-temporal federated transfer learning with multi-sensor data fusion for cooperative positioning, *Inf. Fusion* 105 (2024) 102182.
- [3] M.A. Jamshed, J. Lee, S. Moon, I. Yun, D. Kim, S. Lee, Y. Yi, K. Park, Kargus: a highly-scalable software-based intrusion detection system, in: *Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS '12*, Association for Computing Machinery, New York, NY, USA, 2012, pp. 317–328.
- [4] J. Nam, M. Jamshed, B. Choi, D. Han, K. Park, Haetae: scaling the performance of network intrusion detection with many-core processors, in: *Research in Attacks, Intrusions, and Defenses: 18th International Symposium, RAID 2015, Kyoto, Japan, November 2–4, 2015*, in: *Proceedings 18*, Springer, 2015, pp. 89–110.
- [5] L. Bernaille, R. Teixeira, K. Salamati, Early application identification, in: *Proceedings of the 2006 ACM CoNEXT Conference*, 2006, pp. 1–12.
- [6] T. Karagiannis, K. Papagiannaki, M. Faloutsos, Blinc: multilevel traffic classification in the dark, in: *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2005, pp. 229–240.
- [7] T. Van Ede, R. Bortolameotti, A. Continella, J. Ren, D.J. Dubois, M. Lindorfer, D. Choffnes, M. van Steen, A. Peter, Flowprint: semi-supervised mobile-app fingerprinting on encrypted network traffic, in: *Network and Distributed System Security Symposium (NDSS)*, vol. 27, 2020.
- [8] Z. Wang, Z. Li, M. Fu, Y. Ye, P. Wang, Network traffic classification based on federated semi-supervised learning, *J. Syst. Archit.* 149 (2024) 103091.
- [9] Z. Li, P. Wang, Z. Wang, Flowganomaly: flow-based anomaly network intrusion detection with adversarial learning, *Chin. J. Electron.* 33 (2024) 1–14.
- [10] P. Wang, Z. Wang, F. Ye, X. Chen, Bytesgan: a semi-supervised generative adversarial network for encrypted traffic classification in sdn edge gateway, *Comput. Netw.* 200 (2021) 108535.
- [11] P. Wang, F. Ye, X. Chen, Y. Qian, Datanet: deep learning based encrypted network traffic classification in sdn home gateway, *IEEE Access* 6 (2018) 55380–55391.
- [12] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapé, Toward effective mobile encrypted traffic classification through deep learning, *Neurocomputing* 409 (2020) 306–315.

- [13] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, A. Pescapé, Ai-powered Internet traffic classification: past, present, and future, *IEEE Commun. Mag.* (2023) 1–7.
- [14] Z. Nazari, M. Noferesti, R. Jalili, Dsca: an inline and adaptive application identification approach in encrypted network traffic, in: *Proceedings of the 3rd International Conference on Cryptography, Security and Privacy, ICCSP '19*, Association for Computing Machinery, New York, NY, USA, 2019, pp. 39–43.
- [15] J. Holland, P. Schmitt, N. Feamster, P. Mittal, New directions in automated traffic analysis, in: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, CCS '21*, Association for Computing Machinery, New York, NY, USA, 2021, pp. 3366–3383.
- [16] X. Zhou, W. Liang, I. Kevin, K. Wang, S. Shimizu, Multi-modality behavioral influence analysis for personalized recommendations in health social media environment, *IEEE Trans. Comput. Soc. Syst.* 6 (2019) 888–897.
- [17] X. Zhou, Q. Yang, X. Zheng, W. Liang, I. Kevin, K. Wang, J. Ma, Y. Pan, Q. Jin, Personalized federated learning with model-contrastive learning for multi-modal user modeling in human-centric metaverse, *IEEE J. Sel. Areas Commun.* (2024).
- [18] H. Xu, B. Liu, L. Shu, P. Yu, Open-world learning and application to product classification, in: *The World Wide Web Conference*, 2019, pp. 3413–3419.
- [19] M. Biagiola, P. Tonella, Testing the plasticity of reinforcement learning-based systems, *ACM Trans. Softw. Eng. Methodol.* 31 (2022).
- [20] M. Du, Z. Chen, C. Liu, R. Oak, D. Song, Lifelong anomaly detection through unlearning, in: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS '19*, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1283–1297.
- [21] Y. Sang, M. Tian, Y. Zhang, P. Chang, S. Zhao, Increaibmf: incremental learning for encrypted mobile application identification, in: *Algorithms and Architectures for Parallel Processing: 20th International Conference, ICA3PP 2020, New York City, NY, USA, October 2–4, 2020*, *Proceedings, Part III 20*, Springer, 2020, pp. 494–508.
- [22] J. Zhang, F. Li, F. Ye, H. Wu, Autonomous unknown-application filtering and labeling for dl-based traffic classifier update, in: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 397–405.
- [23] J. Zhang, F. Li, H. Wu, F. Ye, Autonomous model update scheme for deep learning based network traffic classifiers, in: *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.
- [24] V.N. Inukollu, D.D. Keshamoni, T. Kang, M. Inukollu, Factors influencing quality of mobile apps: role of mobile app development life cycle, preprint, arXiv:1410.4537, 2014.
- [25] M. Mermillod, A. Bugaiska, P. Bonin, The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects, 2013.
- [26] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Y. Fu, Large scale incremental learning, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 374–382.
- [27] S. Li, Y. Zhao, R. Varma, O. Salpekar, P. Noordhuis, T. Li, A. Paszke, J. Smith, B. Vaughan, P. Damania, et al., Pytorch distributed: experiences on accelerating data parallel training, preprint, arXiv:2006.15704, 2020.
- [28] R. Kozik, M. Choraś, M. Ficco, F. Palmieri, A scalable distributed machine learning approach for attack detection in edge computing environments, *J. Parallel Distrib. Comput.* 119 (2018) 18–26.
- [29] J. Fang, H. Fu, G. Yang, C.-J. Hsieh, Redsync: reducing synchronization bandwidth for distributed deep learning training system, *J. Parallel Distrib. Comput.* 133 (2019) 30–39.
- [30] P. Wang, S. Li, F. Ye, Z. Wang, M. Zhang, Packetcgan: exploratory study of class imbalance for encrypted traffic classification using cgan, in: *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.
- [31] M. Wang, Y. Lu, J. Qin, A dynamic mlp-based ddos attack detection method using feature selection and feedback, *Comput. Secur.* 88 (2020) 101645.
- [32] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapé, Distiller: encrypted traffic classification via multimodal multitask deep learning, *J. Netw. Comput. Appl.* 183–184 (2021) 102985.
- [33] Z. Wu, Y.-n. Dong, X. Qiu, J. Jin, Online multimedia traffic classification from the qos perspective using deep learning, *Comput. Netw.* 204 (2022) 108716.
- [34] X. Ma, W. Zhu, J. Wei, Y. Jin, D. Gu, R. Wang, Eetc: an extended encrypted traffic classification algorithm based on variant resnet network, *Comput. Secur.* 128 (2023) 103175.
- [35] G. Bovenzi, A. Nascita, L. Yang, A. Finamore, G. Aceto, D. Ciuonzo, A. Pescapé, D. Rossi, Benchmarking class incremental learning in deep learning traffic classification, *IEEE Trans. Netw. Serv. Manag.* 21 (2024) 51–69.
- [36] W. Zhu, X. Ma, Y. Jin, R. Wang, Iletc: incremental learning for encrypted traffic classification using generative replay and exemplar, *Comput. Netw.* 224 (2023) 109602.
- [37] X. Li, J. Xie, Q. Song, Y. Sang, Y. Zhang, S. Li, T. Zang, Let model keep evolving: incremental learning for encrypted traffic classification, *Comput. Secur.* (2023) 103624.
- [38] W. Liang, Y. Hu, X. Zhou, Y. Pan, I. Kevin, K. Wang, Variational few-shot learning for microservice-oriented intrusion detection in distributed industrial iot, *IEEE Trans. Ind. Inform.* 18 (2021) 5087–5095.
- [39] X. Zhou, Y. Hu, J. Wu, W. Liang, J. Ma, Q. Jin, Distribution bias aware collaborative generative adversarial network for imbalanced deep learning in industrial iot, *IEEE Trans. Ind. Inform.* 19 (2022) 570–580.
- [40] R. Bozkır, M. Cicioğlu, A. Çalhan, C. Toğay, A new platform for machine-learning-based network traffic classification, *Comput. Commun.* 208 (2023) 1–14.
- [41] M. Abbasi, A. Shahraki, A. Taherkordi, Deep learning for network traffic monitoring and analysis (ntma): a survey, *Comput. Commun.* 170 (2021) 19–41.

- [42] J. Cao, Z. Yang, K. Sun, Q. Li, M. Xu, P. Han, Fingerprinting {SDN} applications via encrypted control traffic, in: 22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019), 2019, pp. 501–515.
- [43] M. Nasr, A. Houmansadr, A. Mazumdar, Compressive traffic analysis: a new paradigm for scalable traffic analysis, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 2053–2069.
- [44] G. Aceto, D. Ciunzo, A. Montieri, A. Pescapé, Mobile encrypted traffic classification using deep learning: experimental evaluation, lessons learned, and challenges, *IEEE Trans. Netw. Serv. Manag.* 16 (2019) 445–458.
- [45] M. Shen, Y. Liu, L. Zhu, X. Du, J. Hu, Fine-grained webpage fingerprinting using only packet length information of encrypted traffic, *IEEE Trans. Inf. Forensics Secur.* 16 (2020) 2046–2059.
- [46] M. Dicks, J. Chavula, Deep learning traffic classification in resource-constrained community networks, in: 2021 IEEE AFRICON, 2021, pp. 1–7.
- [47] J. Guan, J. Cai, H. Bai, I. You, Deep transfer learning-based network traffic classification for scarce dataset in 5g iot systems, *Int. J. Mach. Learn. Cybern.* 12 (2021) 3351–3365.
- [48] B. Pang, Y. Fu, S. Ren, S. Shen, Y. Wang, Q. Liao, Y. Jia, A multi-modal approach for context-aware network traffic classification, in: ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2023, pp. 1–5.
- [49] W. Wang, M. Zhu, J. Wang, X. Zeng, Z. Yang, End-to-end encrypted traffic classification with one-dimensional convolution neural networks, in: 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), IEEE, 2017, pp. 43–48.
- [50] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, T. Tuytelaars, A continual learning survey: defying forgetting in classification tasks, *IEEE Trans. Pattern Anal. Mach. Intell.* 44 (2022) 3366–3385.
- [51] D.-W. Zhou, Q.-W. Wang, Z.-H. Qi, H.-J. Ye, D.-C. Zhan, Z. Liu, Deep class-incremental learning: a survey, preprint, arXiv:2302.03648, 2023.
- [52] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, C.H. Lampert, icarl: incremental classifier and representation learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2001–2010.
- [53] D.-W. Zhou, F.-Y. Wang, H.-J. Ye, D.-C. Zhan, Pycil: a python toolbox for class-incremental learning, *Sci. China Inf. Sci.* 66 (2023) 197101.
- [54] M. Tian, P. Chang, Y. Sang, Y. Zhang, S. Li, Mobile application identification over https traffic based on multi-view features, in: 2019 26th International Conference on Telecommunications (ICT), IEEE, 2019, pp. 73–79.
- [55] J. Zhang, F. Li, F. Ye, Sustaining the high performance of ai-based network traffic classification models, *IEEE/ACM Trans. Netw.* 31 (2023) 816–827.
- [56] Z. Li, Z. Zhang, M. Fu, P. Wang, A novel network flow feature scaling method based on cloud-edge collaboration, in: 2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2023, pp. 1947–1953.
- [57] G. Aceto, D. Ciunzo, A. Montieri, V. Persico, A. Pescapé, Mirage: mobile-app traffic capture and ground-truth creation, in: 2019 4th International Conference on Computing, Communications and Security (ICCCS), IEEE, 2019, pp. 1–8.
- [58] G. Draper-Gil, A.H. Lashkari, M.S.I. Mamun, A.A. Ghorbani, Characterization of encrypted and vpn traffic using time-related, in: Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP), 2016, pp. 407–414.
- [59] A. Fernández, S. Garcia, F. Herrera, N.V. Chawla, Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary, *J. Artif. Intell. Res.* 61 (2018) 863–905.



**Zeyi Li** is currently pursuing the Ph.D. degree in Cyberspace Security at Nanjing University of Posts and Telecommunications. He received his B.S. degree in mathematics in 2019 and received M.S. degree in computer science in 2022. His research interests include network security, communication network security, anomaly detection and analysis, deep packet inspection, and graph neural networks.



**Minyao Liu** was born in Ganzou, Jiangxi, China, in 2000. She is currently pursuing her master's degree at Nanjing Post and Telecommunications University. She received her bachelor's degree in Management from NUPT in 2022. Her research areas include traffic identification, deep learning and anomaly detection.



**Pan Wang** received the BS degree from the Department of Communication Engineering, Nanjing University of Posts and Telecommunications, Nanjing, China, in 2001, and the PhD degree in Electrical and Computer Engineering from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2013. He is currently a Professor in the School of Modern Posts, Nanjing University of Posts and Telecommunications, Nanjing, China. His research interests include cyber security and communication network security, network measurements, Quality of Service, Deep Packet Inspection, SDN, big data analytics and applications. From 2017 to 2018, he was a visiting scholar of University of Dayton (UD) in the Department of Electrical and Computer Engineering.



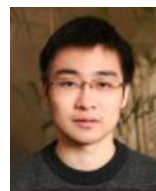
**Wangyu Su** is currently pursuing the Bachelor degree in Computer Science at University of Wisconsin, Madison. He will receive his B.S. degree in Computer Science in 2027. His research interests include network security, communication network security, data analysis, machine learning, and software engineering.



**Tianshui Chang** is currently pursuing a B.A. degree in Information Network at Nanjing University of Posts and Telecommunications, Nanjing, China. He has been involved in multiple research projects focusing on computer networks, network security, and secure communication protocols. His long-term goal is to contribute to cutting-edge network security research.



**Xuejiao Chen** received the B.E. and M.E. degrees from Nanjing University of Posts and Telecommunications (NUPT), majoring in communication and information systems, in 2001 and 2006, respectively. Now she is an associate professor at the Nanjing Institute of Information Vocational Technology. She has been a visiting scholar of the University of Dayton (OH, USA) from 2017 to 2018. Her research areas are B5G/6G network security and artificial intelligence.



**Xiaokang Zhou** is currently an associate professor with the Faculty of Business Data Science, Kansai University, Japan. He received the Ph.D. degree in human sciences from Waseda University, Japan, in 2014. From 2012 to 2015, he was a research associate with the Faculty of Human Sciences, Waseda University, Japan. He was a lecturer/associate professor with the Faculty of Data Science, Shiga University, Japan, from 2016 to 2024. He also works as a visiting researcher with the RIKEN Center for Advanced Intelligence Project (AIP), RIKEN, Japan, since 2017. Dr. Zhou has been engaged in interdisciplinary research works in the fields of computer science and engineering, information systems, and social and human informatics. His recent research interests include ubiquitous computing, big data, machine learning, behavior and cognitive informatics, cyber-physical-social systems, and cyber intelligence and security. Dr. Zhou is a member of the IEEE CS, and ACM, USA, IPSJ, and JSAP, Japan, and CCF, China.