Journal of Systems Architecture 149 (2024) 103091

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Network traffic classification based on federated semi-supervised learning

ZiXuan Wang [a], ZeYi Li [b], MengYi Fu [c], YingChun Ye [a,d], Pan Wang [a,*]

[a] School of Modern Posts, Nanjing University of Posts & Telecommunications, Nanjing, China
[b] School of Computer Science, Nanjing University of Posts & Telecommunications, Nanjing, China
[c] School of Internet of Things, Nanjing University of Posts & Telecommunications, Nanjing, China
[d] Nanjing Future Network Industry Innovation Co., Ltd, Nanjing, China

ARTICLE INFO

Keywords:
Federated learning
Semi-supervised learning
Network traffic classification
Deep learning

ABSTRACT

Traffic Classification (TC) has been applied to a wide range of applications, from security monitoring to quality of service (QoS) provisioning in network Internet Service Providers (ISPs). In recent years, many researchers have applied Machine Learning (ML) or Deep Learning (DL) to TC, namely AI-TC. However, AI-TC methods face significant challenges, including high data dependency, exhaustively costly traffic labeling, and network subscribers' privacy. This paper proposes a TC framework for smart home networks using Federated Learning (FL) that protects traffic data privacy by performing local training and inference of TC models. Firstly, we design a DPI-based traffic labeling method on edge home gateways as FL nodes, which enables these nodes to have data labeling capability while protecting data privacy. Then, a semi-supervised TC model based on an autoencoder (AE) is proposed to reduce the dependence of the model on labeled traffic samples. Finally, an XAI-based method is utilized to interpret the model to ensure its explainability. We validate the proposed method on public and real datasets using benchmarking methods. The experimental results show that the method can achieve high performance using a small number of samples while protecting data privacy and improving the model's credibility. Experimental code can be found in the following url: https://github.com/PrinceXuan12138/HGW-TC-Experimental-code.

1. Introduction

The development of the Internet of Things, cloud computing, and big data technologies has laid a solid technical foundation for the rapid development of smart homes [1]. The fast development of smart appliances and devices has also further boosted the pace of smart homes. The home network has completely transformed from a simple interconnection of computers in the home to an "entrance" for home users to various intelligent applications and services outside. As shown in Fig. 1, in addition to high-definition video, online games, and other "standard" high-bandwidth applications in the home, there are more and more applications with high real-time requirements, including Virtual Reality/Augmented Reality (VR/AR), e-sports games, online education, live video, telemedicine, fire alarm smoke detection, home security monitoring and other services. These applications also pose very demanding requirements on the QoS of the network, including real-time, high reliability, and fast and flexible service customization. Therefore, it is necessary to classify the network traffic in the home and provide network packet forwarding strategies [2]. Data packets of network flows that are sensitive to delay are forwarded preferentially. In contrast, data packets of network flows that are not sensitive to delay are forwarded on a best-effort basis, thereby improving user experience. The most important thing to achieve this goal is accurately identifying network application traffic [3].

Traditional rule-based classification methods have a simple implementation principle and can meet the requirements of fast classification in high-speed networks. However, as more and more network applications adopt randomization and encryption to improve transmission security, traditional methods quickly become ineffective. Machine learning methods based on traffic statistical features overcome the limitations of traditional methods. However, flow features need to be manually designed, which significantly depends on the experience of domain experts, is exhaustive and costly, and cannot adapt to dynamically changing traffic features. In the past decade, industry and academia have shown unprecedented interest in deep learning, and many research efforts have been devoted to using deep learning techniques to solve problems in network systems [4]. Deep learning can characterize the hidden features between data through multiple neural networks, represent the essential data characteristics, and have strong generalization ability. Most deep learning solutions can perform better than traditional hand-made, rule-based heuristic solutions. However,

* Corresponding author.
E-mail address: wangpan@njupt.edu.cn (P. Wang).

https://doi.org/10.1016/j.sysarc.2024.103091
Received 29 June 2023; Received in revised form 3 February 2024; Accepted 16 February 2024
Available online 17 February 2024
1383-7621/© 2024 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Table 1
List of abbreviations and meanings of proper terms.

Abbreviation	Explanation
AE	Auto Encoder
AI-TC	AI based Traffic Classification
CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
DNS	Domain Name System
DPI	Deep Packet Inspection
FAM	Flow Attribute Matrix
FF	Flow Features
FL	Federated Learning
GAN	Generative Adversarial Networks
ByteSGAN	Byte Semi-supervised Generative Adversarial Network
QoS	Quality of Service
TC	Traffic Classification
SSTC	Semi-Supervised Traffic Classification
XAI	Model Interpretability

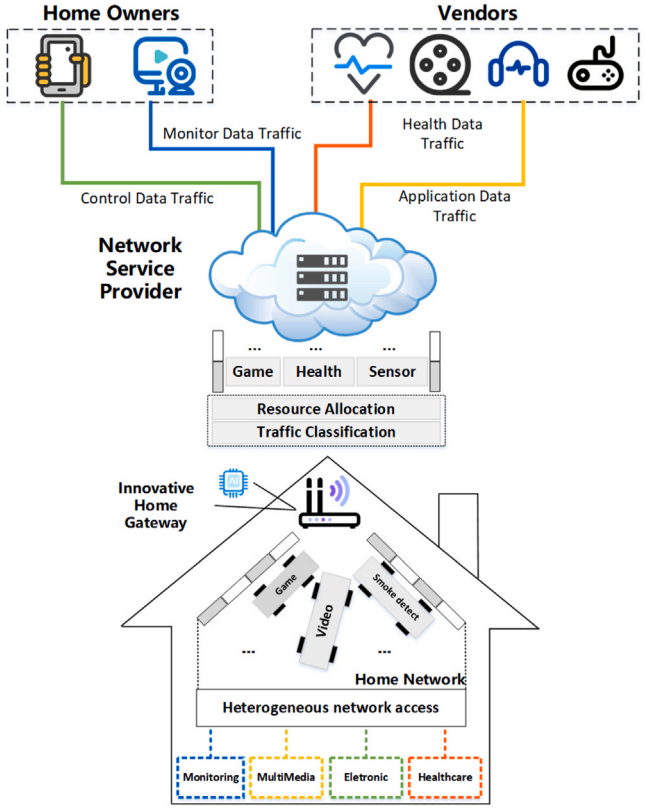


Fig. 1. Home network traffic management.

the training of deep learning models requires a large amount of data, and there are the following problems in practical applications:

- *Data Privacy*: Data privacy protection is increasingly important for society and individuals, and regulations are being implemented accordingly. However, collecting model training data may pose privacy risks. Home network traffic, which includes personal information, browsing records, asset information, and other private data, is especially hard to collect and label. Moreover, some business factors hinder data sharing among enterprises, leading to severe data silos.
- *Data Dependency*: Data labeling, which deep learning algorithms require a lot, is time-consuming and relies on expert experience. Therefore, the model needs to be designed uniquely to classify traffic with high precision using little labeled data.
- *Sustainability*: The applications of smart homes are updated frequently. When the applications are updated, the characteristics of their network flows may change, resulting in a decrease in the model's accuracy or even complete failure. It is necessary to have a certain data collection and labeling capability at the terminal side and realize the continuous update of the model.

Federated learning is an emerging deep learning paradigm that enables user data sharing without uploading it to the central server [5], thus breaking data silos and protecting data privacy [6]. We use smart home gateways and distributed edge devices to deploy deep learning models with federated architecture. This paper presents a secure, efficient, sustainable home traffic classification scheme based on federated learning. Our contributions are:

1. A novel home edge gateway based federated learning encrypted traffic classification framework is proposed for smart home networks.
2. A semi-supervised traffic classification model is proposed using a small number of labeled and many unlabeled samples to train the model. It can significantly reduce the model's dependence on labeled data.
3. A traffic labeling method based on DPI technology is proposed, which can label some unlabeled flows using DPI and DNS. Then, a semi-supervised model exploits the characteristics of the rest of the unlabeled samples to improve the traffic classification task.
4. A private traffic dataset for mobile device application classification is built in a real network scenario. Using benchmarking and XAI methods, we verify and explain the model's performance on public and private datasets to ensure its trustworthiness.

This paper is organized as follows: Section 1 introduces the motivation and contribution of this paper. Section 2 reviews and compares the existing research on semi-supervised learning and federated semi-supervised learning and analyzes the significance of this paper's work. Section 3 presents the method and process proposed in this paper. Section 4 describes the experiment's purpose and setting. Section 5 discusses the results, and Section 6 concludes the paper and suggests future directions. Table 1 lists the abbreviations used in this paper.

2. Related works

2.1. Network traffic classification

Semi-supervised traffic classification (SSTC) combines the advantages of supervised and unsupervised methods. It improves the model's recognition performance and reduces the data labeling workload. Table 2 shows the main methods of SSTC.

Methods based on cluster-label. These methods cluster all data and map clusters to categories based on labeled data. Then, they assign unlabeled data to categories. The cluster-label methods include K-means, Spectral Clustering, Hierarchical Clustering, etc.

Lin et al. [7] used improved K-Means for network flows with scarce labels. They initialized cluster centers with flow attribute variance and mapped clusters to traffic classes. Glennan et al. [8] classified traffic at the protocol level with a fuzzy clustering label algorithm. They combined packet header and flow statistical features and optimized the model with a feature selection algorithm.

These cluster-based methods are computationally efficient but sensitive to data distribution. Since the size of the dataset is difficult to cover the distribution characteristics of actual traffic, the clustering model may be unstable in practical use.

Methods based on generative models. These methods use generative models, such as the Gaussian Mixture Model, Hidden Markov Model, Naive Bayes, etc. They assume traffic categories have different

Table 2
Related works: Semi-supervised learning based traffic classification methods.

Method name	Technique	Semi-supervised type	Dataset	Target classes	Labeled data amount	Accuracy
Improved K-Means [7]	k-means	Cluster-label	Private	5 (Service-level)	2000flows	91.80%
Fuzzy-cluster [8]	k-means	Cluster-label	MAWI working group	5 (Protocol-level)	100flows	95.00%
(GMM)-based model [9]	GMM	Generative model	Private	8 (Service-level)	10%flows	70%
Clustering hybrid based method [10]	CFS, X-means, NB, J48	Generative model	Moore	10 (Service-level)	20%flows	95%
DCGAN [11]	DCGAN	GAN	ISCX VPN-NonVPN+Private	4 (Service-level)	10%flows	89%
ByteSGAN [12]	SGAN	GAN	ISCX 2012+CrossMarket	15 (Application-level)	1000label	92%
StackedAutoencoder Approach [13]	SAE	Discriminative model	Dataset Unicauca	54 (Application-level)	70%label	89%

probability distributions and estimate distribution parameters by maximum likelihood or Bayesian inference. Then, they divide unlabeled data by posterior probability.

Qian et al. [9] applied GMM to semi-supervised traffic classification and identified different network applications. Noorbehbahani et al. [10] used x-means clustering and label propagation for semi-supervised traffic classification. They automatically determined cluster numbers, calculated similarity, propagated labels with j48 and naive Bayes, and mapped clusters to predefined categories.

These methods based on generative models are computationally moderate. They can effectively deal with unknown or dynamically changing network applications. However, they require prior knowledge to select appropriate statistical features and clustering parameters, which may affect the stability and generalization of classification results.

Methods based on generative adversarial networks (GANs). These methods use a generator and discriminator networks to compete with each other. The generator network tries to generate fake traffic close to the actual traffic distribution. The discriminator network tries to distinguish between real and fake traffic and perform traffic classification simultaneously. In this way, unlabeled data can be used to enhance the classification ability of the discriminator network, and data imbalance problems can be handled [14].

Iliyasu et al. [11] proposed a semi-supervised learning approach using Deep Convolutional Generative Adversarial Network (DCGAN). Their idea is to utilize the samples generated by DCGAN generators and unlabeled data to improve the performance of a classifier trained on a few labeled samples. Wang et al. [12] proposed a Generative Adversarial Network (GAN)-based Semi-Supervised Learning Encrypted Traffic Classification method called ByteSGAN. ByteSGAN can use a small number of labeled traffic samples to achieve a good traffic classification performance by modifying the GAN discriminator network's structure and loss function.

These methods based on GAN can rely on neural networks to automatically extract high-dimensional features without the need for a large amount of labeled data or expert knowledge. At the same time, the generative ability of GAN can enhance the dataset's diversity and quality and improve the model's generalization performance. However, GAN models have complex structures and many parameters, which are very difficult to train and unsuitable for edge device deployment.

Methods based on discriminative models. These methods directly learn a mapping function from feature space to category space and optimize the model parameters by minimizing the classification error of labeled data and the regularization term of unlabeled data, dividing the unlabeled data according to the prediction results. The methods based on discriminative models mainly include Support Vector Machine (SVM), Neural Network (NN), Decision Tree (DT), etc.

Aouedi et al. [13] used a semi-supervised stacked autoencoder (SSAE) based method. It uses the encoder to extract feature vectors automatically and uses the decoder and classifier of SSAE to perform classification. The idea of this paper is similar to ours, but they build the model based on the idea of transfer learning and achieve semi-supervised classification in two steps. First, they use unlabeled data to train the SSAE model. Then, they concatenate the SSAE and classification models and use labeled data for supervised training. The model we propose uses unlabeled and labeled data to train the model simultaneously, which is more simplified in the training steps.

2.2. Semi-supervised federated learning traffic classification

Although there are a large number of studies that use deep learning to achieve traffic classification tasks in a supervised or semi-supervised manner, these methods always train data in a centralized way, which has the following limitations:

- Network traffic is complex, and a single traffic may not reflect the existing network's real data distribution. Also, some business factors prevent enterprises from sharing data completely [15], causing severe data island [16].
- In the centralized deep learning framework, sending data to the central server risks privacy leakage, and the centralized data model may expose data information to malicious attackers.

Google first proposed federated learning (FL) in 2016 [17], which provides a new solution for user data sharing. Federated learning has been widely used in medical imaging, smart terminals, computer vision, and other fields [18], [19]. In recent years, some scholars have combined federated learning and deep learning to achieve traffic classification and achieved good results, as shown in Table 3.

Zhu et al. [20] proposed a traffic classification method based on federated learning (DTCFL). They realized the labeling of application traffic collection by associating port numbers. Similar to our scenario, He et al. [21] proposed an Edge Device Identification (EDI) method based on federated learning called (FedeEDI). FedeEDI has faster training speed and shorter training time than centralized learning-based EDI (CentEDI), which is more suitable for edge devices.

Guo et al. [22] proposed a federated approach for privacy-preserving network traffic classification in heterogeneous environments called FEAT, which is a client selection algorithm to proactively and intelligently select the clients with low skewness to participate in the model aggregation in FL. With the help of FEAT, classification accuracy is improved, and convergence is faster than average FL in low and high heterogeneity environments.

Although these methods have achieved good classification results, their models are supervised and rely on many labeled samples for training, which are complex to obtain and more difficult to label in real scenarios. He et al. [21] and Guo et al. [22] did not explicitly state how to obtain labeled data on edge devices. Zhu et al. [20] proposed to use port association for data labeling, which cannot be applied in our scenario because, on the home edge devices (such as routers), we cannot use the port association of application processes to label the traffic of terminals (such as mobile phones). Therefore, in the last two years, some scholars have studied combining semi-supervised and federated learning to achieve traffic classification.

Aouedi et al. [23] proposed a semi-supervised traffic intrusion detection system based on federated learning called FLUIDS. They first use the unlabeled data on the sub-nodes to train AE in an unsupervised way, then adjust the model parameters of AE through federated learning, concatenating the trained AE model with a classifier and fine-tune it on the sub-nodes using labeled data, and finally detect the malicious traffic with an accuracy of about 85% on the UNSW-NB15 dataset.

The method proposed by Aouedi et al. [24] (SSFLIDS) and the method proposed by Jin et al. [25] (FSSL) have the same idea. They both proposed a semi-supervised federated learning traffic classification

Table 3
Related works: Federated learning based traffic classification methods.

Method name	Technique	Training mode	Implement semi-supervised method	Dataset	Target class	Labeled data amount	Accuracy
DTCFL [20]	NN	Supervised	/	Private	7 (Application-level)	/	96%
FedeEDI [21]	CNN/LSTM+MLP	Supervised	/	ProfilIoT	10 (Service-level)	/	88.20%
FEAT [22]	CNN	Supervised	/	QUIC, ISCX	8 (Service-level)	/	93%
FLUIDS [23]	AE+CNN	Semi-supervised	Train locally with unlabeled samples, aggregate globally and then fine-tune with local labeled samples	UNSW-NB15	2 (Application-level)	40%	85%
SSFLIDS [24]	AE+NN	Semi-supervised	Client-side training with unlabeled data, server-side fine-tuning with labeled data	UNSW-NB15	2 (Application-level)	30%	84%
FSSL [25]	CNN	Semi-supervised	Client-side training with unlabeled data, server-side fine-tuning with labeled data	QUIC	5 (Service-level)	2000	97.81%
FSRA [26]	MLP	Semi-supervised	Server-side supervised training, client-side output pseudo-labeling with global model, fine-tuning with pseudo-labeled samples	QUIC	5 (Service -level)	20%	0.0133(rmse)

framework. The main idea is to train the model on the sub-nodes in an unsupervised way, then fine-tune the model with the labeled data on the server during aggregation, and finally achieve semi-supervised traffic classification.

To achieve multimedia traffic classification, Bano et al. [26] proposed a federated semi-supervised regression algorithm (FSRA). Their idea is to train a global model on the server using labeled data in a supervised way, and the sub-nodes use the global model to identify unlabeled samples, obtain pseudo-labels, fine-tune the local model through a feature selection algorithm, and finally achieve traffic classification.

Few works combine federated learning and semi-supervised learning for traffic classification. Our survey reveals two main ideas for federated semi-supervised learning [27]. The first one, similar to Aouedi et al. [23] and Bano et al. [26], aims to enable sub-nodes to label data. However, Aouedi et al. [23] do not explain how sub-nodes obtain labeled data. Bano et al. [26] depend on the server's classification model for pseudo-labels' accuracy. However, the model is trained centrally, which has problems and may affect the semi-supervised model's performance. The second idea, similar to Aouedi et al. [24] and Jin et al. [25], transforms the model structure and objective so that sub-nodes train the model with unlabeled data and the server fine-tunes the model with labeled data, achieving semi-supervised traffic classification. This idea avoids data labeling on sub-nodes but increases data labeling on the server side, and a single institution cannot collect more data than many terminals. The best way is that sub-nodes can continuously label data and provide high-quality labeled data, which is very difficult for traffic classification. Therefore, we propose an edge device traffic labeling method based on DNS and DPI, which gives edge devices data labeling capabilities. Also, we propose a semi-supervised traffic classification model based on AE-CNN, which combines federated learning and trains models on edge devices, alleviating the above problems.

3. Deep FL-based TC method on home edge gateway

3.1. Goals

After the investigation in the previous chapter, the federated learning framework for family-oriented network traffic classification should have the following capabilities:

1. It must have some traffic labeling ability on the edge devices while protecting data privacy. It can provide continuous labeled data for the model, leverage the advantages of federated learning, expand data distribution, and improve the model's performance.
2. While achieving the abovementioned goal, the traffic classification model needs to be carefully designed to complete the model training and achieve good results with only a small amount of labeled samples, further reducing the workload of data labeling.
3. The training method should be easy to operate (minimize steps other than model iteration, such as outputting pseudo-labels and adjusting the model structure to work well on edge devices with low computing power like routers.

3.2. Overall architecture

The general AI pipeline includes five steps: model design, data preparation, model training, model validation, and model serving. In order to achieve the design objectives of the previous section, combined with the general AI pipeline, we propose a deep federated traffic classification framework based on edge devices, as shown in Fig. 2. The overall architecture is divided into two parts: the management layer and the node layer. The management layer is deployed in the cloud and mainly completes the model design, model aggregation and evaluation, and router forwarding policy management. The node layer consists of edge devices in each family and mainly completes data collection, model training, and traffic forwarding. Specifically, it includes the following five steps:

(1) Initial model design and initial file distribution: The initial model refers to a model that has only fixed the model structure but has not trained the neurons' weights. The initial file contains two parts: the target application's DNS feature file and the target application's forwarding priority file. First, the goal of the deep learning classification task needs to be determined according to the demand, that is, to determine the number of applications to be identified, the dimension of classification, and the model structure according to the task goal. According to the goal in the previous section, we designed a stacked neural network based on AE and CNN, which can receive labeled and unlabeled data for training and output classification results. The details of the model will be introduced in detail in Section 3.3. When the model

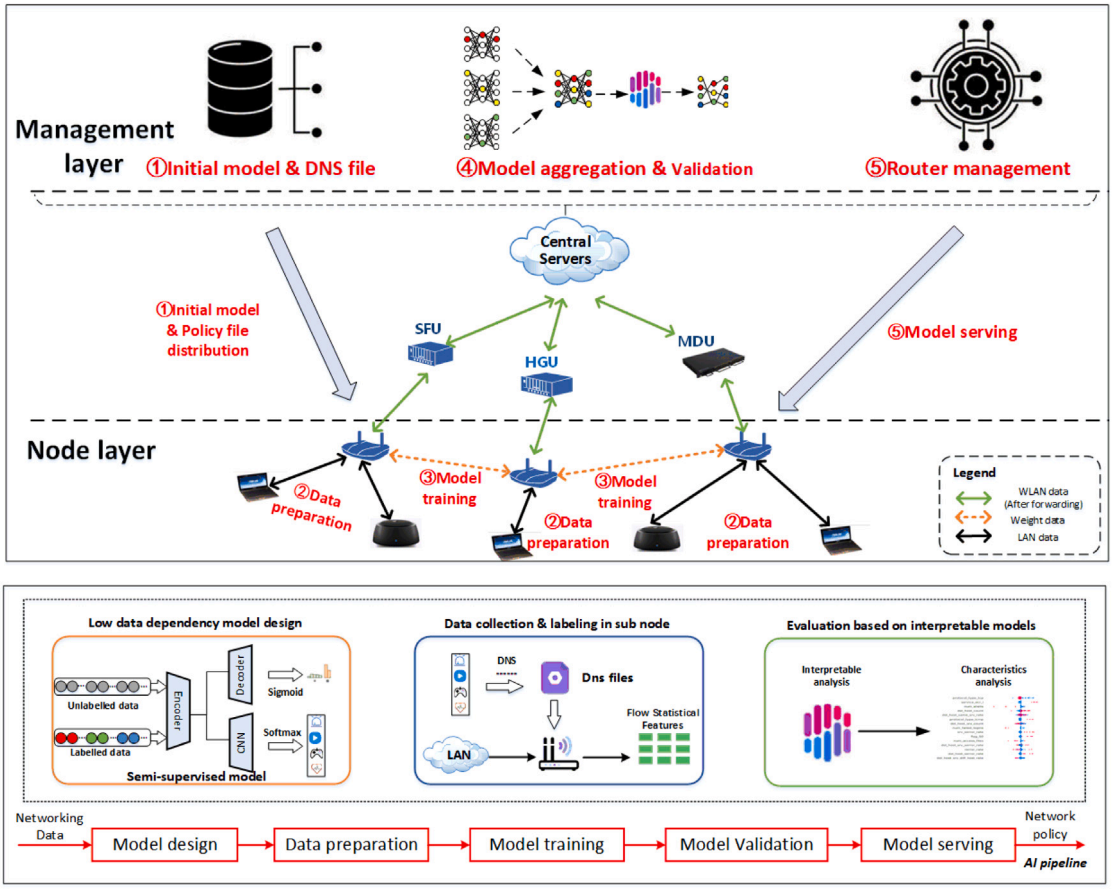


Fig. 2. Overall architecture.

is designed, samples of the target application need to be collected at the management layer, and representative DNS domain names are analyzed and extracted from them to form a DNS feature file. At the same time, forwarding priority needs to be set for the target application to form a flow forwarding policy file. Finally, the initial model and files are distributed to each edge device, waiting for the next operation step.

(2) Data preparation: When receiving the initial model and initial files from the management node, the edge device will load the DNS feature file, start the capture module, perform DNS association on the captured packets, and realize the collection and labeling of the original packets. The specific method of data collection and labeling will be introduced in Section 3.4. After completing the collection and labeling of packets, preprocess the collected original packets, extract flow features according to the five-tuple, and organize them into flow feature vectors to input to the model for training.

(3) Federated Learning: Integrate the resources of multiple families to achieve horizontal federated learning. Specifically, it is divided into three steps: (a) Each sub-node uses the locally collected data for local training. (b) Each node uploads the gradient of the local model to the global node. (c) The global node aggregates the gradients of each sub-node model to form a global model and returns the global model to each sub-node for retraining. Repeat the above three steps multiple times until global convergence stops, obtain the final global model, and complete federated training. Federated learning will be introduced in detail in Section 3.5

(4) Model validation: After federated learning is completed, use model explainable methods to verify the global model and select trustworthy models to distribute to edge devices for deployment.

(5) Model serving: After data preprocessing on the traffic passing through the edge device, input it to the deep learning model to obtain recognition labels. According to the recognition label, use a five-tuple to extract network flow from packets and divide them into application flows according to application. According to the obtained application flow forwarding priority policy file, queue the recognized application flows according to priority and forward them to realize traffic management.

3.3. Semi-supervised model design

3.3.1. Definition

(1) Packet: Packet is the smallest unit in the flow. The set P is the set of Packets, and M is the number of Packets in a flow.

$$P = \{p_1, p_2, p_3 \ldots\ldots p_{i-1}, p_i\}, 0 \le \mathrm{i} \le \mathrm{M} \tag{1}$$

(2) Session: The session is also called bi-directional flow, which refers to the up-flow and down-flow with the same source address, destination address, source port, destination port, and protocol. The set S represents the session, and N is the number of sessions the application interacts with. Application traffic may contain multiple sessions, such as a game application that contains multiple sessions between the cell phone and the domain name server, the leading game site, the advertising server, etc.

$$S = \{S_1, S_2, S_3 \ldots\ldots S_{i-1}, S_i\}, 0 \le 1 \le N \tag{2}$$

(3) Flow: The difference between Flow and Session is that flow is unidirectional. The set F represents flow, and K represents the number of flows in application interactions.

$$F = \{f_1, f_2, f_3 \ldots\ldots f_{i-1}, f_i\}, 0 \le i \le K \tag{3}$$

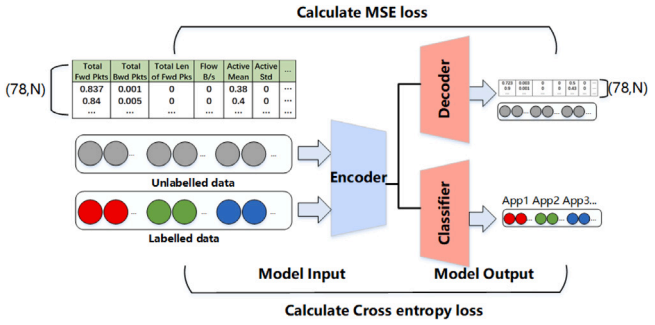


Fig. 3. Model input design.

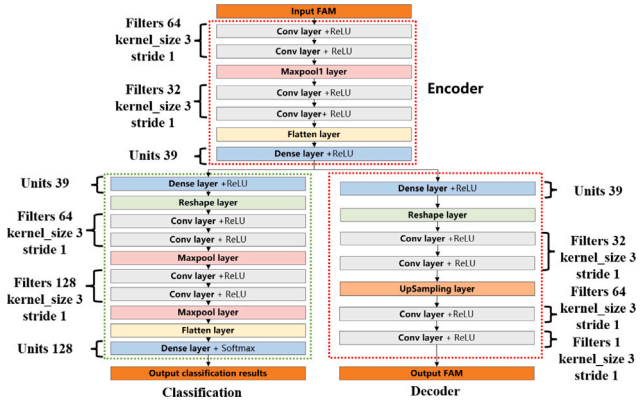


Fig. 4. Model structure.

(4) Flow features: This paper computes Flow Features(FF) for Packet and Flow to form the feature set according to the above definition.

$$FF = \{ff_1, ff_2, ff_3 \dots ff_j\} \tag{4}$$

3.3.2. Model input design

As shown in Fig. 3, the FAM obtained in Section 3.4.2 is first subjected to data normalization. Then, the FAMs are cropped according to different lengths to form a set H containing multiple FAM subsets as the input to the model. $H = [FAM_1, FAM_2, FAM_3, \dots]$ where the shape of each FAM subset is 78*N. During training, N is the size of the set batch. During identification, N is set to 10 000, which means when the edge device captures 10,000 flows, they are sent to the model for detection once.

3.3.3. Model structure

In order to further reduce the model's dependence on labeled data, we designed a semi-supervised traffic classification model based on a convolutional autoencoder. As shown in Fig. 4, the model consists of an encoder, a decoder, and a classifier. The input of the encoder is FAM, and the output is the high-dimensional feature after dimensionality reduction; the decoder receives the high-dimensional feature after dimensionality reduction and outputs the restored FAM; the classifier receives the high-dimensional feature after dimensionality reduction and outputs the recognized application label. Through the encoder, high-dimensional feature extraction based on restoration loss is realized with unlabeled samples; through the classifier, supervised known traffic classification is realized with labeled samples.

The encoder consists of two 1D CNN layers, one max pooling layer, two 1D CNN layers, one flattened layer, and one fully connected layer connected in series. The activation functions of both CNN layers and fully connected layers are ReLU.

The encoder learns and outputs features after the reduction of dimensionality from the fused features' high-dimensional features. The decoder consists of one fully connected layer, a transpose layer, two 1D CNN layers, one upsampling layer, and two 1D CNN layers connected in series. The activation functions of both CNN layers and fully connected layers are ReLU. The decoder learns from the high-dimensional features after dimensionality reduction by the encoder and tries to restore them, outputting a restored vector with the same dimension as FF.

The classifier consists of one fully connected layer, a transpose layer, two 1D CNN layers, one max pooling layer, two 1D CNN layers, one max pooling layer, one flattened layer, and one fully connected layer connected in series. The activation function of the last fully connected layer uses Softmax, and the activation functions of the other fully connected layers and CNN layers use ReLU. The classifier learns from the fused features after dimensionality reduction by the encoder and outputs the classification label.

The goal of model training is achieved by using an alternating training method. The model has two training goals. The first is using labeled samples to classify applications through supervised learning. The second is to use unlabeled samples to achieve high-dimensional feature extraction of unlabeled traffic based on restoration loss through unsupervised learning and help the classifier learn potential features.

First, unsupervised training is performed, and the number of training rounds is set. The encoder and decoder are trained with unlabeled samples, using MSE as the loss function, as shown in Eq. (5). Where $\hat{y}_i$ is the restored FAM output by the decoder, and y_i is the original FAM input to the encoder. The Adam optimizer is used to optimize, and in each round of training, the weights of the neurons in the encoder and decoder are adjusted until the number of unsupervised training rounds is met and stopped.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5}$$

Then, supervised training is performed, and the number of supervised training rounds is set. Using the labeling method in 3.4, label FAM with y_n to form labeled FAMy.

$$FAMy = \begin{pmatrix} FF_1 \\ FF_2 \\ FF_{3,} \\ \dots \\ FF_n \end{pmatrix} = \begin{pmatrix} ff_{1_1}, ff_{1_2} \dots ff_{1_{78}}, y_1 \\ ff_{2_1}, ff_{2_2} \dots ff_{2_{78}}, y_2 \\ ff_{3_1}, ff_{3_2} \dots ff_{3_{78}}, y_3 \\ \dots \\ ff_{n_1}, ff_{n_2} \dots ff_{n_{78}}, y_n \end{pmatrix} \tag{6}$$

Using the labeled FAMy, train the encoder and classifier, using cross entropy as the loss function, as shown in Eq. (7). Where $\hat{y}_n$ is the predicted label output by the classifier for the nth ff, and y_n is the actual label of the nth ff. The Adam optimizer is used to optimize, and in each round of training, the weights of the neurons in the encoder are fixed, and only the weights of the neurons in the classifier are adjusted until the number of supervised training rounds is met and stopped.

Alternate training of unsupervised and supervised models until the model is convergent.

$$\text{Loss} = -\sum_{i=1}^{\text{output-}} y_n \cdot \log \hat{y}_n \tag{7}$$

3.4. Edge traffic labeling method based on DPI and DNS

3.4.1. Traffic capture and labeling method based on edge devices

In order to enable edge devices to have some traffic labeling ability, we propose an edge-side traffic labeling scheme based on DPI and DNS. As shown in Fig. 5, the method is divided into two steps: (1) making DNS configuration files on the management node. (2) Capturing packets on the edge device and labeling them based on the DNS configuration file.

(1) DNS configuration file making. We build an Android emulator based on Anbox on the management node and monitor the virtual

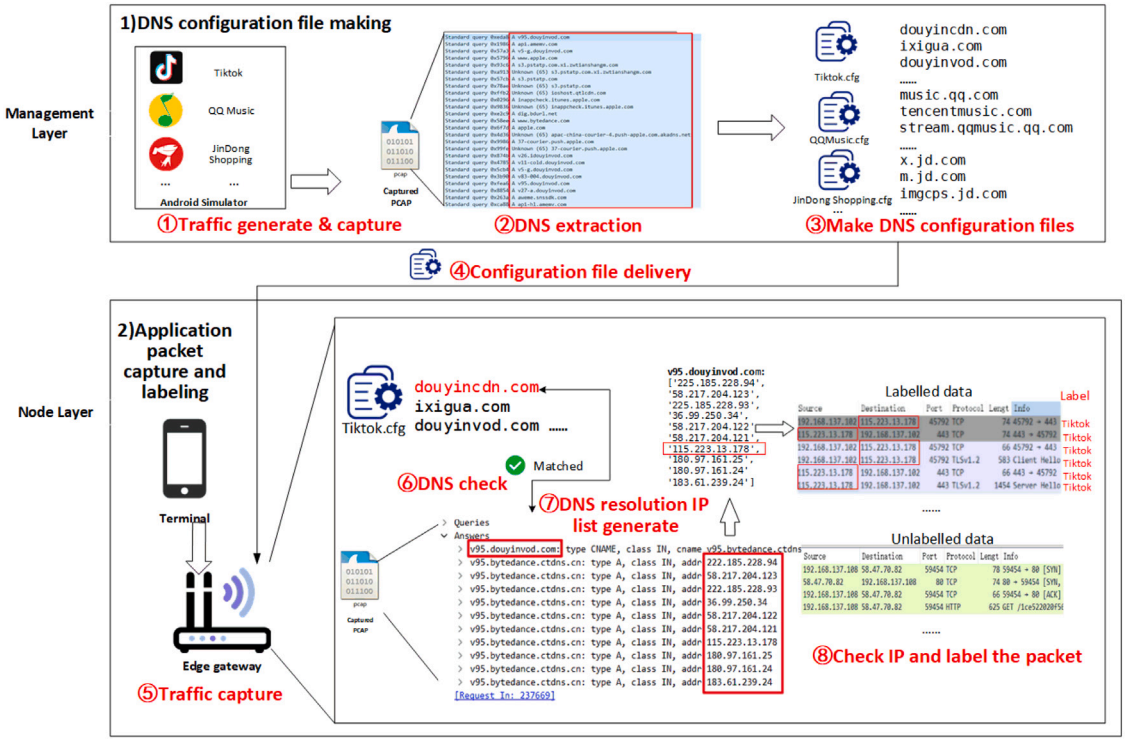


Fig. 5. Traffic capture and labeling method.

Table 4
Flow Attribute Matrix (FAM).

Packet level					Flow-level					Statistics				
Total Fwd Pkts	Total Bwd Pkts	Total len of Fwd Pks	Total len of Bwd Pkts	…	Flow duration	Flow B/s	Flow IAT Max	Flow IAT Min	…	Mean (Pkt-len)	Average (Pkt-size)	Active mean	Active std	…
32	16	6448	1152	…	112 740 690	67.411331	16 400 000	3	…	163.3265	166.7292	359.4286	11.99802	…
32	16	6448	5056	…	112 740 560	102.03959	16 400 000	2	…	243	248.0625	320.2857	15.74499	…
545	0	0	0	…	113 757 377	0	20 800 000	0	…	0	0	9 361 829	7 324 646	…
…	…	…	…	…	…	…	…	…	…	…	…	…	…	

network card of the emulator to capture the traffic generated by the emulator. We start each target app individually in the virtual machine and parse the DNS request packets in the captured packets. We use DPI technology to extract DNS domain names and filter out the characteristic domain names of the target application based on expert experience to form a DNS configuration file. The configuration file is sent to the edge device for traffic labeling at the beginning of the classification task.

(2) Application packet capture and label based on DNS configuration file. We developed a data collection module on the edge devices. It uses hook functions to copy network packets to the ring buffer, achieving the capture of raw packets. After capturing the raw packets, we use DPI to parse the DNS in the PCAP and associate it with the DNS configuration file issued by the management platform. First, we analyze the DNS response packets, check whether there is a domain name consistent with the domain name in the configuration file, and if there is, then generate the DNS resolution IP list of all valid IPs for that domain name in the DNS response packet, and record the application name. Then, we parse the subsequent UDP and TCP packets and check whether the source IP and destination IP are in the above DNS resolution IP list; if yes, label all the packets of that source–destination IP with the corresponding application tag.

Next, we use an example to illustrate the whole process. As shown in Fig. 5, we first generate applications in the simulator environment and capture the packets they produce. Then, we analyze the packets generated by each application and obtain the domain name configuration file for each application. For example, the domain name configuration file of TikTok contains domain names: “douyincdn.com”, “ixigua.com”, “douyinvod.com”, and “...”. Then, we send these configuration files to the edge gateway and perform raw packet capture on the edge gateway. When the raw packet is captured, we parse the DNS response packet. As shown in Fig. 5, when the domain name “douyinvod.com” appears in the response packet, it is consistent with the domain name recorded in “Tiktok.cfg”, then we record all the IPs parsed from it to form a list. Then, we perform IP detection on all subsequent packets; if any are consistent with the list, we consider these packets to be coming from Tiktok.

3.4.2. Data preprocessing method

After completing the labeling of some packets, all packets need to be preprocessed. First, we extract the TCP and UDP packets from the obtained application interaction traffic to form a packet set $P = \{p_1, p_2, p_3 \ldots\ldots p_{i-1}, p_i\}, 0 \leq i \leq M$. Then, the packets in the packet set are organized using the five-tuple to form the flow set $F = \{f_1, f_2, f_3 \ldots\ldots f_{i-1}, f_i\}$. When extracting, we use a five-tuple as the flow’s key, which is stored in the hash table, along with the timestamp of the first packet and the timestamp of the last flow packet. If the duration of a flow exceeds 120 s, truncate the flow and clear the information in the hash table. The subsequent packet will be considered the new flow to compute the FF; the finish (FIN) mark of TCP is considered the end mark of the flow. The flow features are output, and the information in the hash table is cleared as soon as the FIN

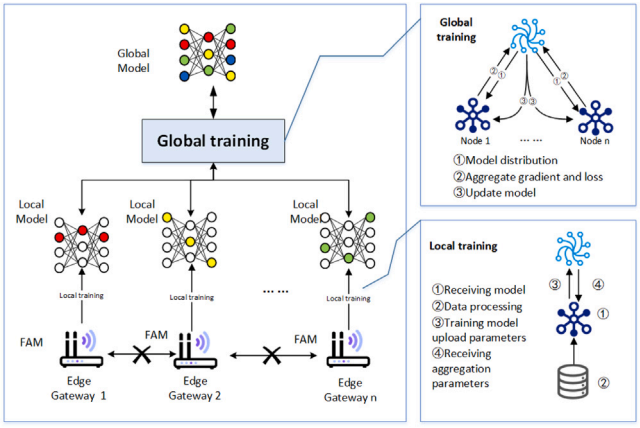


Fig. 6. Federated learning process.

mark appears. To extract the Session, TCP Flow is analyzed. The packet in the TCP flow that is consistent with the srcIP and dstIP of the Synchronize Sequence Numbers (SYN) packet will be recorded as the forward flow, and the opposite will be recorded as the backward flow. For those Flows without SYN packets, the srcIP and dstIP of the first packet are used as a basis for judgment. The set containing forward and backward flows is formed for feature extraction $S_j = \{f_1, f_1', f_2, f_2' \cdots\}$. We write the feature extraction code based on the feature calculation method in CICFLOWMETER [28] for edge-based devices. Extract each session's bi-directional flow features to form a flow feature vector $FF = \{ff_1, ff_2, ff_3 \cdots, ff_j\}$. By definition, there are three levels, including packet-level features, flow-level features, and statistical features. Each flow feature vector FF is stitched into a flow attributes matrix FAM according to the packet, as shown in Table 4.

3.5. Federated learning

As shown in Fig. 6, the entire training process consists of global and local training. In each round of training, each sub-node independently calculates the update of the current model based on its local data and transmits this update to the central server. The sub-node updates are aggregated at the central server to calculate a new global model. The central server plays the role of the global node in federated learning. Before starting federated learning, the central server first detects idle edge gateways. Then, it initiates federated training requests to the idle edge gateways. If the edge gateways agree to participate, they return receipts to the central server. The central server aggregates all the edge gateways that agree to join the training, forming the edge gateway list for this training. Then, the central server sends the DNS configuration file and the initial model to each edge gateway in the list and starts federated learning. In each global epoch of federated learning, the central server waits for all the edge gateways to finish local training, collects the local models uploaded by each edge gateway, and aggregates them until the training is completed.

The model local training includes four steps: (1) First, receive the initial intrusion detection model based on the autoencoder issued by the central server. (2) Then collect multiple FAMs to form a set H = $\{\mathrm{FAM_i}, \mathrm{FAM_j}\}$ as the training dataset, where FAM_i is a set of unlabeled FAMs, and FAM_j is a set of labeled FAMs. (3) Each time training, take one FAM from FAM_i and FAM_j respectively and use gradient descent algorithm to train the model. First, use unlabeled FAM to train the Encoder and Decoder, then fix the weights of the Encoder and use labeled FAM to adjust the weights of the CNN classifier until all FAMs in set H are trained. (4) Upload the weights and biases of the trained model to the central server. Receive the updated weights and biases returned by the central server. Repeat steps (3)–(4) multiple times until the global node judges that it stops after global convergence.

Algorithm 1 The FL-SSTC method

Require: The set H containing multiple FAM subsets
Input: $H = [FAM_i, FAM_j]$. H is the set of training; FAM_i are FAM_s without label; FAM_j are FAM_s with label.
Output: global model M_{t+1}.
Set the relevant initialization parameters. The activation function $\alpha = RELU$; e_g is the global epoch; e_l is the local epoch; k is the number of nodes; η is the learning rate; τ is the batch_size.
for each Global Round **t** =**1,2,3,...,** e_g **do**
  for each node **k** = **1,2,3,...** **do**
    if global model exists **then**
      Load global model;
    else
      Get an initial model from a central server.
    end if
    while FAM_i and FAM_j in **H do**
      The unlabeled training set FAM_i **are feed as input to the Encoder and Decoder model**
      Input: Feature vectors in $FAM_i, F = \{f_1, f_2, \cdots, f_{78}\}$
      Output: $\hat{F} = \{\hat{f}_1, \hat{f}_2, \cdots, \hat{f}_{78}\}$, Updated weights ω_t^k, b_t^k
      for local epoch e_l **do**
        Calculate and minimize losses $min(M_{MSE}) = min(\frac{1}{n}\sum_{i=1}^{\tau}(F_i - \hat{F}_i)^2)$
        Backpropagation to update the weights of AE model: $\omega_t^k = \omega_t^k - \eta\frac{\partial L}{\partial \omega}$; $b_t^k = b_t^k - \eta\frac{\partial L}{\partial b}$
      end for
      The labeled training set FAM_j **are feed as input to the Encoder and CNN model**
      Input: Feature vectors in $FAM_j, F = \{f_1, f_2, \cdots, f_{78}\}$
      Output: $\hat{F} = \{\hat{f}_1, \hat{f}_2, \cdots, \hat{f}_{78}\}$, Updated weights ω_t^k, b_t^k
      for local epoch e_l **do**
        Calculate and minimize losses $min(L_{TC}) = min(Loss = -\sum_{i=1}^{output\_size} y_i \cdot \log \hat{y}_i)$
        Backpropagation to update the weights of CNN model: $\omega_t^k = \omega_t^k - \eta\frac{\partial L}{\partial \omega}$; $b_t^k = b_t^k - \eta\frac{\partial L}{\partial b}$
      end for
    end while
    if update model parameters **then**
      Average the model parameters: $\omega_{t+1} = \frac{1}{k}\sum_{k=1}^{k}\frac{d_k}{d}\omega_t^k$; $b_{t+1} = \frac{1}{k}\sum_{k=1}^{k}\frac{d_k}{d}b_t^k$
      Load the new model parameters ω_{t+1}, b_{t+1} and get the new global model M_{t+1}
      Send the new global model to the node.
    end if
  end for
end for

The model global training includes three steps: (1) Design a semi-supervised classification model according to the classification goal and distribute it to each node. (2) Collect the model parameters and losses of each node's local training. (3) Summarize the parameters and losses of each node. Use FedAvg [17] to calculate the parameters and losses of the global model and distribute the model to each node. (2) and (3) multiple times until the global loss reaches the set convergence threshold. When aggregating the model globally, considering that the data volume of different sub-nodes is different, use the data volume of each node and all nodes as weights for aggregation. Algorithm 1 shows the specific training process. Where k represents k nodes, ω_t represents the global model parameters after t iterations. ω_t^k represents the model parameters of the kth node after t iterations.

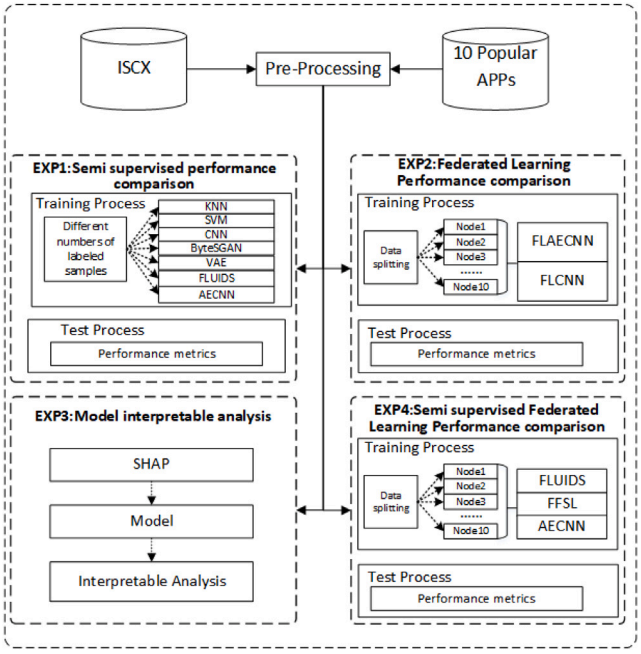


Fig. 7. Experimental setup.

4. Experiment setup

4.1. Experiment goal

We want to verify the feasibility of the proposed architecture from three aspects: accuracy, reliability, and performance. The questions we care about include three: (1) The performance of the proposed semi-supervised model traffic classification model. (2) The performance of the proposed federated semi-supervised classification architecture. (3) Whether the features learned by the model are consistent with the behavioral features of the application. As shown in Fig. 7, we conducted four experiments on public and private datasets to verify the above questions.

4.2. Experiment design

4.2.1. Dataset

(1) Public dataset: We use the "ISCX VPN2016" public dataset as the benchmark for model comparison. First, we divide the traffic in the "NonVPN-PACP" dataset into five categories according to the service type: "chat", "email", "file", "streaming", and "VoIP" for service-level traffic classification. Then, we use the method described in Section 3.4.2 to extract flow features and the smote method to balance the dataset.

(2) Private dataset: The self-built dataset is formed by ten users using Android phones, typically in the experimental environment for two weeks. The dataset contains popular applications in five categories: video, music, games, chat, and shopping in the Android app store. We selected ten applications with more captured packets among them for application-level traffic classification experiments, namely "Tiktok", "Snack Video", "iQiyi Video", "QQ Music", "Netease Cloud Music", "Jingdong Shopping", "Taobao Shopping", "Arena of Valor", "QQ" and "WeChat". We use the data collection and labeling method described in Section 3.4.1 to construct labels and use the method described in Section 3.4.2 to perform feature extraction. The details of the datasets are shown in Table 5. On both datasets, 90% of the data is used for training and 10% for testing. In semi-supervised experiments, unlabeled data samples are randomly selected from labeled samples, and their labels are removed.

Table 5
ISCX dataset and 10 popular APPs dataset.

ISCX dataset		
ClassName	Original records	Records after balance
Chat	6873	20 000
Email	5684	20 000
File	36 832	20 000
Streaming	3014	20 000
Voip	126 707	20 000
10 popular APPs dataset		
ClassName	Original records	Records after balance
Tiktok	20 065	18 000
iQiyi video	19 687	18 000
Jingdong shopping	28 174	18 000
Snack video	36 176	18 000
QQmusic	40 738	18 000
QQ	25 204	18 000
Taobao shopping	24 864	18 000
NetEase cloud music	18 791	18 000
Arena Of valor	20 310	18 000
WeChat	24 143	18 000

Table 6
Hyper parameters of the deep learning models.

	CNN	ByteSGAN	VAE	FLUIDS	AECNN
Optimizer	Rmsprop	Adam	Adam	Adam	Adam
Learning rate	0.0001	0.0001	0.0001	0.0001	0.0001
Batchsize	128	128	128	128	128
Epochs	100	5000	100	100	100

Table 7
FLCNN, FLAECNN hyper parameters.

	FL-CNN	FL-AECNN
Optimizer	Adam	Adam
Learning rate	0.0001	0.0001
Batchsize	256	256
local epochs	5	5
total epochs	50	50

4.2.2. Semi supervised performance comparison (Experiment 1)

To evaluate the performance of the proposed semi-supervised classification method, we conducted comparative experiments on the ISCX dataset and the Test 10 Popular APPs dataset with seven models (KNN [29], SVM [30], CNN [2], ByteSGAN [12], VAE [31], FLUIDS [23], and our proposed AECNN). KNN and SVM are machine learning models, while CNN, ByteSGAN, VAE, FLUIDS, and AECNN are deep learning models. KNN, SVM, and CNN are trained in a supervised manner. ByteSGAN, VAE, FLUIDS, and AECNN are trained semi-supervised (Only use the model structure of FLUIDS and train it in a centralized way). We used 1000, 2000, 3000, and 4000 labeled samples to train the models on public and private datasets to evaluate their performance metrics. The hyperparameters of the deep learning models are shown in Table 6.

4.2.3. Federated learning performance comparison (Experiment 2)

In order to verify the performance of the proposed federated semi-supervised classification architecture, we conducted experiments with the semi-supervised model and the baseline CNN model, respectively, using the federated learning architecture. We simulated ten sub-nodes and one central node using virtualization technology and split the two datasets evenly as the collected data of each sub-node for federated learning. As shown in Table 7, each model trains five epochs on subnodes, then 50 epochs globally, and uses the FedAvg method to aggregate the parameters of each model. Each sub-node uses an AMD R7 5800H CPU, 16G RAM, and RTX3070 GPU for training and validation.

Table 8
Exp4 experimental setup.

		FLUIDS	FFSL	AECNN
Number of sub nodes		10	10	10
Local training	Epoch	5	5	5
	Batchsize	64	64	64
	Optimizer	Adam	Adam	Adam
Global training	Epoch	20	20	20
	Batchsize	64	64	64
	Optimizer	Adam	Adam	Adam
Labeled data (per class)	ISCX	1800	1800	1500
	10 Popular Apps	1620	1500	1500

4.2.4. Model interpretable analysis (Experiment 3)

In order to verify the reliability of the model, we used the SHAP method to perform an explainable analysis of the model trained in Experiment 2. We ranked the feature contributions learned by the model according to the application output to ensure the reliability of the model.

4.2.5. Semi supervised federated learning performance comparison (Experiment 4)

To evaluate the performance of our proposed semi-supervised federated learning method, we carefully studied FLUIDS [23], SSFLIDS [24], and FSSL [25]. We reproduced them according to the description of the papers. The work of FLUIDS [23] and SSFLIDS [24] is consistent, so we wrote the experimental code according to the description of FLUIDS [23]. FLUIDS and FFSL perform unsupervised learning on the sub-nodes and fine-tuning on the global node with supervised learning. However, our method is to use labeled data for semi-supervised learning on the sub-nodes directly. This experiment compares the differences between these two training methods. The experimental parameters are shown in the Table 8.

4.3. Evaluation metrics

The performance evaluation indicators in this article include Precision, Recall, F1-score, and Acc, which are derived from four parameter indicators: TP, TN, FP, and FN. The following is a detailed description of this article's performance evaluation indicators.

1. Precision, also known as positive predictive value or precision rate, refers to the proportion of true positive samples among the samples that we predicted as positive. The formula is (8), where TP is the number of true positives, and FP is the number of false positives.

$$Precision = \frac{TP}{TP + FN} \tag{8}$$

2. Recall, also known as sensitivity or true positive rate, refers to the proportion of positive samples we predicted as positive among the positive samples. The formula is (9), where TP is the number of true positives, and FN is the number of false negatives.

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

3. F1-score is a weighted average of precision and recall. It can reflect the model's ability to identify and distinguish positive samples. The formula is (10).

$$F1 - score = \frac{2 * \text{ Precision } * \text{ recall}}{\text{Precision } + \text{ recall}} \tag{10}$$

4. Acc, also known as accuracy or correct rate, refers to the proportion of correctly classified samples among the total number of samples. The formula is (11), where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

5. Experiment results

5.1. Semi supervised performance comparison (Exp 1)

We conducted experiments on the ISCX dataset and the 10Pupular APPs dataset, and the overall performance parameters of the experiments are shown in Table 9. It can be seen from the table that the AECNN model shows the best performance on both datasets, and it can achieve more than 0.96 on all indicators in any case, which indicates that it has strong generalization ability and robustness. The KNN model is the best-performing one among the supervised models, and it exceeds 0.9 on all indicators on both datasets. However, its performance improves with the increased number of labeled data, indicating that it strongly depends on the data volume. The CNN model is a medium-performing one among the supervised models, and it exceeds 0.9 on all indicators on the ISCX dataset. However, it is lower than 0.9 on all indicators on the Test 10 Popular APPs dataset, which indicates that it is difficult to achieve good recognition results when the number of labeled samples is small. However, from the results on the ISCX dataset, it can be seen that with the increase in the number of labeled data samples, the accuracy of CNN has been significantly improved, and it has surpassed KNN, indicating that it has more significant potential.

It can be seen from the table that the four semi-supervised methods can significantly improve the classification accuracy when the amount of labeled data is small. On both datasets, the accuracy of the four semi-supervised methods exceeds 0.9. In the case of a minimal number of labeled samples (only 1000 labeled samples per application class), the method proposed in this paper performs better. On the ISCX dataset, AECNN achieves an accuracy of 0.97, 0.01 higher than the second-ranked FLUIDS, and 0.09 higher than the deep learning baseline CNN model. On the 10 Popular APPs dataset, due to the increased difficulty of the classification task, the accuracy of each model has dropped somewhat. However, AECNN still achieves an accuracy of 0.965, which is 0.02 higher than FLUIDS and 0.22 higher than the baseline CNN. The proposed method can effectively use the labeled data samples combined with many unlabeled data to improve the model's overall performance. The Two medium-performing models are the ByteSGAN and the VAE models among the semi-supervised models. Their indicators on both datasets are around 0.9. Their performance improvement could be more evident with the increased amount of labeled data, indicating low data utilization efficiency.

It is difficult to fully reflect the performance of the model by just looking at the overall indicators. Figs. 8 and 9 show the experimental performance of the four semi-supervised models on the two datasets in terms of service classification and application classification respectively.

It can be seen from Fig. 8 that for different service types, the Precision, Recall, and F1-score indicators of the proposed AECNN model have reached 95%. It indicates that each type of service can be effectively identified, and there is no imbalance phenomenon. For the service flows of Chat, Email, File, and Streaming, the proposed model has a better recognition effect when the labeled samples are fewer and has improved more than 5% compared to VAE. It can be seen from Fig. 9 that due to the more complex application classification task, the differences between the four semi-supervised models are apparent. The AECNN model shows the best performance on all applications, and its precision, recall, and F1-score all exceed 0.9 and are close to 1 on most applications. This shows that it has strong generalization ability and robustness. The ByteSGAN and VAE models perform generally and significantly differ in identifying different applications. For example, their F1-Score on Tiktok, iQiyi Video, Jingdong Shopping, Snack Video,

Table 9
EXP1: Evaluation metrics.

Labeled data	Method	ISCX				10 Popular APPs			
		ACC	Precision	Recall	F1-score	ACC	Precision	Recall	F1-score
1000	KNN	0.934	0.934	0.934	0.934	0.921	0.921	0.920	0.921
	SVM	0.770	0.806	0.767	0.738	0.798	0.829	0.799	0.796
	CNN	0.882	0.886	0.882	0.881	0.795	0.738	0.798	0.758
	ByteSGAN	0.928	0.966	0.965	0.965	0.909	0.917	0.909	0.910
	VAE	0.928	0.928	0.927	0.928	0.907	0.907	0.907	0.907
	FLUIDS	0.967	0.968	0.967	0.967	0.941	0.943	0.941	0.941
	AECNN	**0.979**	**0.979**	**0.979**	**0.979**	**0.965**	**0.965**	**0.965**	**0.965**
2000	KNN	0.954	0.955	0.954	0.954	0.941	0.942	0.941	0.941
	SVM	0.795	0.824	0.792	0.774	0.833	0.843	0.834	0.833
	CNN	0.940	0.942	0.940	0.940	0.869	0.793	0.869	0.826
	ByteSGAN	0.947	0.971	0.971	0.971	0.921	0.926	0.921	0.921
	VAE	0.947	0.948	0.947	0.947	0.919	0.919	0.918	0.918
	FLUIDS	0.975	0.976	0.975	0.976	0.968	0.969	0.968	0.968
	AECNN	**0.982**	**0.983**	**0.982**	**0.982**	**0.970**	**0.972**	**0.970**	**0.970**
3000	KNN	0.964	0.965	0.964	0.964	0.950	0.950	0.950	0.950
	SVM	0.916	0.920	0.915	0.917	0.838	0.848	0.838	0.837
	CNN	0.959	0.960	0.959	0.959	0.872	0.797	0.872	0.829
	ByteSGAN	0.960	0.975	0.975	0.975	0.924	0.926	0.924	0.924
	VAE	0.960	0.960	0.960	0.960	0.930	0.932	0.930	0.930
	FLUIDS	0.982	0.982	0.982	0.982	0.968	0.969	0.968	0.968
	AECNN	**0.985**	**0.986**	**0.985**	**0.985**	**0.982**	**0.982**	**0.982**	**0.982**
4000	KNN	0.968	0.968	0.968	0.968	0.957	0.958	0.957	0.957
	SVM	0.920	0.924	0.920	0.921	0.854	0.855	0.855	0.855
	CNN	0.974	0.974	0.974	0.974	0.887	0.813	0.887	0.845
	ByteSGAN	0.957	0.979	0.979	0.979	0.932	0.935	0.932	0.932
	VAE	0.957	0.958	0.957	0.957	0.939	0.941	0.940	0.940
	FLUIDS	0.984	0.984	0.984	0.984	0.954	0.960	0.954	0.954
	AECNN	**0.986**	**0.986**	**0.986**	**0.986**	**0.987**	**0.987**	**0.987**	**0.987**

and Arena Of Valor are all over 0.9, but their F1-Score on QQmusic, QQ, and NetEase Cloud Music are low.

Fig. 10 is the confusion matrix of the four models using the 10 Popular APPs Dataset for classification, which can more intuitively reflect the performance of the models. It can be seen from the figure that ByteSGAN has more misidentification for QQ, NetEase Cloud Music, and WeChat, with 336, 305, and 116 records being misidentified, respectively. VAE and FLUIDS have significantly improved the recognition performance for these three applications. AECNN has further improved on their basis and only has more misidentification for QQ and NetEase Cloud Music, with 172 and 54 records being misidentified, respectively.

Therefore, whether from the overall indicators or the indicators of each application, our proposed AECNN model has better experimental performance in the case of fewer labeled samples.

5.2. Federated learning performance comparison (Exp 2)

We conducted experiments on the two datasets using the baseline CNN and the proposed AECNN model in a federated learning manner, and the overall indicators of the experiments are shown in Table 10. It can be seen from the table that the proposed federated AECNN method can effectively improve traffic classification performance. For service type classification (ISCX), the ACC, Precision, Recall, and F1-score indicators of federated AECNN all exceeded 95%, while the baseline CNN model only had about 75%, an increase of about 20%. For application classification (10Popular APPS dataset), the ACC, Precision, Recall, and F1-score of the proposed federated AECNN exceeded 85%, while the baseline CNN only had about 74%, an increase of about 15%. It is worth noting that when the number of labeled samples is small (500 labeled samples per class per node), the proposed federated semi-supervised method still performs well.

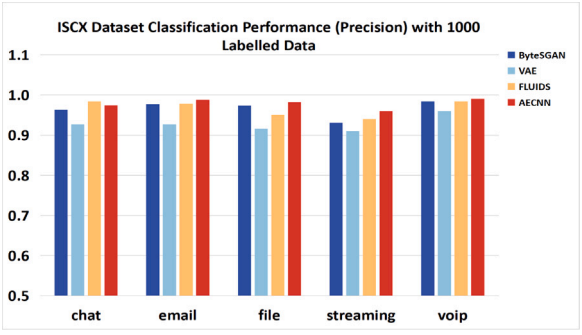
Fig. 11 shows the experimental performance of service classification and application classification on the two datasets, where the blue, red, and green series legends represent the Precision, Recall, and F1-score of the three models, respectively.

Table 10
EXP2: Evaluation metrics.

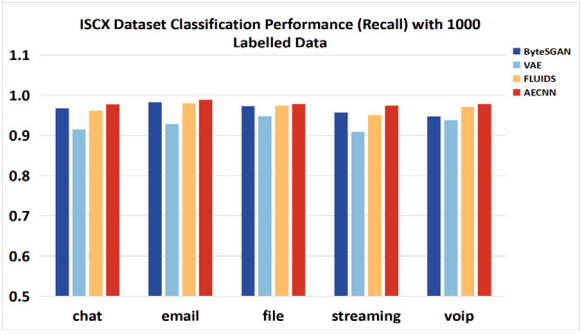
Labeled data	Method	ISCX			
		ACC	Precision	Recall	F1-score
500	FLCNN	0.742	0.745	0.741	0.720
	FLAECNN	**0.942**	**0.944**	**0.942**	**0.942**
800	FLCNN	0.751	0.764	0.750	0.729
	FLAECNN	**0.953**	**0.954**	**0.953**	**0.953**
1000	FLCNN	0.768	0.789	0.768	0.747
	FLAECNN	**0.961**	**0.962**	**0.961**	**0.961**
Labeled data	Method	10 Popular APPS			
		ACC	Precision	Recall	F1-score
500	FLCNN	0.711	0.738	0.709	0.701
	FLAECNN	**0.867**	**0.873**	**0.866**	**0.866**
800	FLCNN	0.748	0.765	0.747	0.745
	FLAECNN	**0.907**	**0.909**	**0.906**	**0.906**
1000	FLCNN	0.787	0.711	0.748	0.787
	FLAECNN	**0.919**	**0.920**	**0.918**	**0.918**

As can be seen from Fig. 11(a), for service types, the baseline CNN model shows different recognition performance, with good recognition performance for Chat and File but poor recognition performance for Email and Voip, resulting in an imbalance phenomenon. The proposed federated AECNN model effectively improves this problem and has high recognition performance for Email and VoIP services, with Precision, Recall, and F1-score all exceeding 90%.

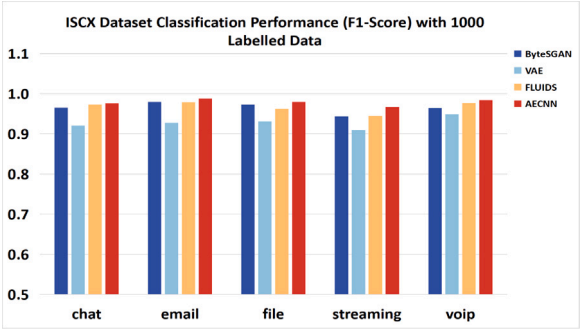
As can be seen from Fig. 11(b), in the case of only a small number of labeled samples, except for NetEase ClodMusic's Precision indicator, the remaining applications' Precision, Recall, and F1-score are better than the baseline CNN model, among which Tiktok's precision indicator has improved significantly, increasing by more than 30%. The baseline CNN model has a poor recognition effect for all applications except SnackVideo, with an F1-score lower than 85%, resulting in a severe



(a) The Precision on the ISCX dataset

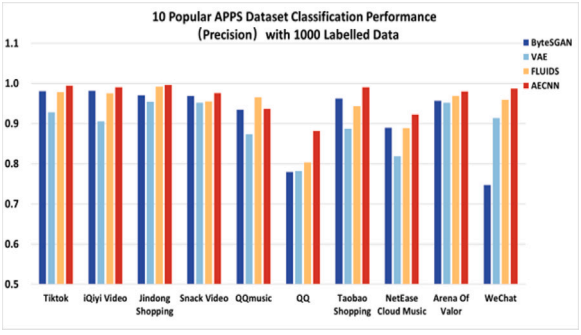


(b) The Recall on the ISCX dataset

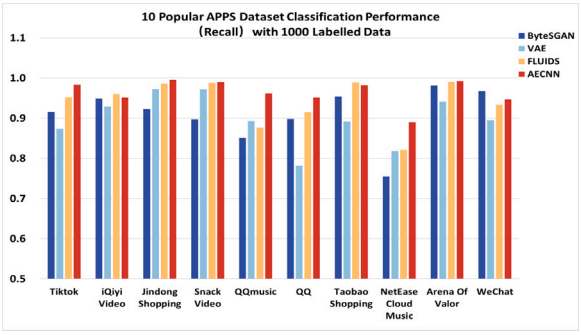


(c) The F1-score on the ISCX dataset

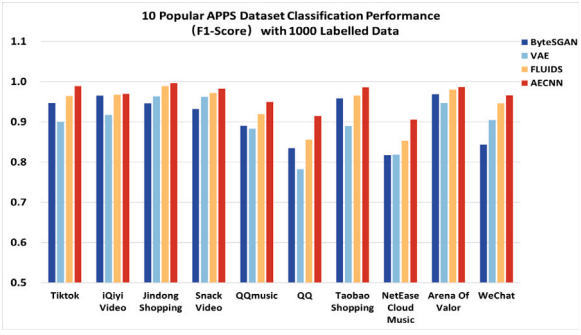
Fig. 8. ISCX dataset classification performance with 1000 labeled data.



(a) The Precision on the 10 Popular APPs dataset



(b) The Recall on the 10 Popular APPs dataset



(c) The F1-score on the 10 Popular APPs dataset

Fig. 9. 10 popular APPs dataset classification performance with 1000 labeled data.

imbalance problem. The proposed federated AECNN dramatically alleviates this problem, with only two applications having F1-score lower than 85%, and both exceeding 70%

Next, we focus on the accuracy indicators of each node. The accuracy of each node obtained by the two models is shown in Fig. 12. It can be seen from Fig. 12 that there is a significant difference in accuracy between the two models. For the ISCX and 10 Pupular APPs dataset, the accuracy of the semi-supervised model is above 85%. However, the accuracy of the baseline CNN classification model is below 85%. It can be seen from the two figures that the proposed federated AECNN method has high accuracy for each node, and the difference is negligible. However, the accuracy of the baseline CNN model varies greatly, indicating that the federated AECNN model can effectively alleviate the impact of data volume difference among nodes on model accuracy.

On the ten popular applications dataset, when the number of labeled feature samples decreases from 1000 to 800, the accuracy of the CNN model drops by an average of 2%, while the accuracy of the semi-supervised model drops by less than 1% on average. This shows that the semi-supervised model has stronger robustness to labeled feature samples, which means that the decrease in the number of labeled feature samples has little impact on model accuracy. It is worth mentioning that on the real collected application traffic dataset, the semi-supervised model only needs 500 labeled feature samples per category to achieve more than 85% recognition accuracy. In real scenarios, only about 20,000 traffic packets need to be labeled to achieve more than 80% accuracy, which is relatively easy.

Table 11
EXP4: Evaluation metrics.

Method	ISCX				10 Popular APPs			
	ACC	Precision	Recall	F1-score	ACC	Precision	Recall	F1-score
FLUIDS	0.979	0.979	0.979	0.979	0.853	0.886	0.854	0.854
FFSL	**0.981**	**0.982**	**0.981**	**0.981**	0.950	0.951	0.950	0.950
AECNN	0.980	0.980	0.980	0.980	**0.956**	**0.958**	**0.957**	**0.957**

5.3. Model interpretable analysis (Exp 3)

We performed an interpretable model analysis on both datasets, and the results are shown in Fig. 13. The top ten contributing features in both datasets are mostly related to packet size and length, and

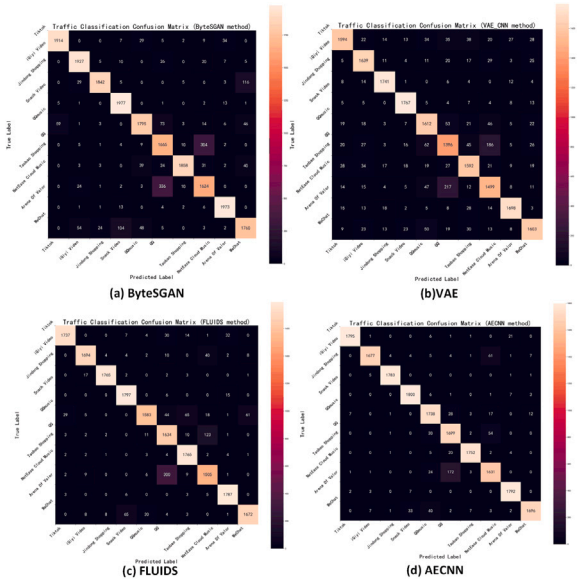
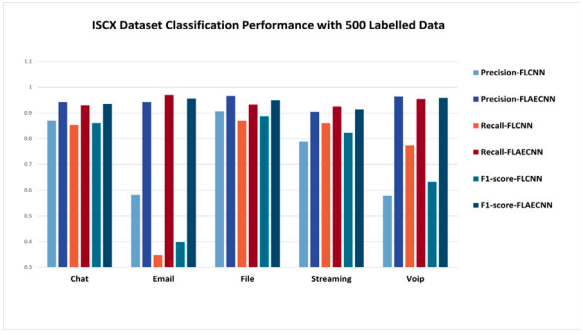
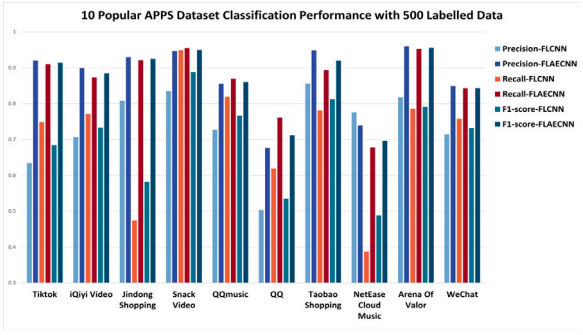


Fig. 10. 10 popular APPs dataset model classification confusion matrix.



(a) ISCX Dataset Classification Performance with 1000 Labeled Data



(b) 10 Popular APPS Dataset Classification Performance with 1000 Labeled Data

Fig. 11. Exp2: Model performance comparison. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the most contributing feature is the "Down/up Ratio". This feature describes the downloaded data ratio to the application flow's uploaded data. On the ISCX dataset, since file transmission is almost uplink or downlink traffic, it is consistent with the subjective perception that this feature has the most significant impact on identifying file transmission categories. For 10 popular applications datasets, resources must be constantly obtained from service providers when using video and shopping software. Almost all of them are down-flow, so this feature significantly contributes to classification, which aligns with expectations. Observing the data packets of the original PCAP, it is found that most video applications are TCP packets, most game applications are UDP packets, and most shopping applications are HTTPS packets. Therefore, the protocol type is also crucial to the application classification. In general, through interpretable analysis, it can be found that the features learned by the model are consistent with subjective cognition, indicating the reliability of the proposed model.

5.4. Semi supervised federated learning performance comparison (Exp 4)

We compared the performance of three methods on the ISCX dataset and the 10 Popular APPs dataset, and the results are shown in the Table 11. The performance of the three models on the ISCX dataset is very close; their indicators are around 0.98, and the difference between them does not exceed 0.001, which indicates that they can identify and distinguish the traffic types in the ISCX dataset well.

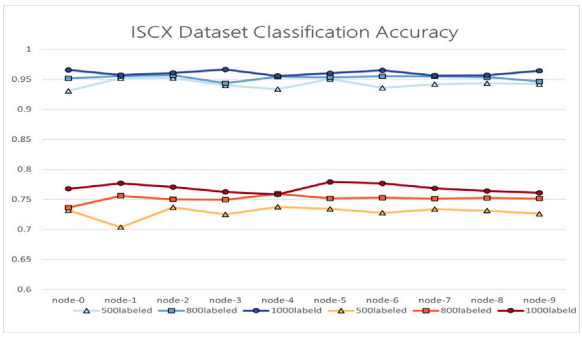
The performance of the three models on the Test 10 Popular APPs dataset is different, among which the AECNN model shows the best performance, and its indicators all exceed 0.95. In contrast, the FLUIDS model shows the worst performance. Its indicators are all lower than 0.9, which indicates that AECNN has better generalization performance and can better handle more diverse and complex classification tasks.

Fig. 14 reflects the experimental performance of the three methods on the two datasets regarding service classification and application classification. The performance of the three models on different applications varies, and the FLUIDS model performs poorly; its indicators on all applications are lower than 0.9, and its F1-score on Tiktok, QQmusic, QQ, and NetEase Cloud Music are lower than 0.7, which indicates that it has insufficient adaptability to the features and complexity of the dataset, which also reflects the limitations of the FLUIDS model. The FFSL model is an up-and-coming model. Its performance on all applications is stable, and its indicators are above 0.9.

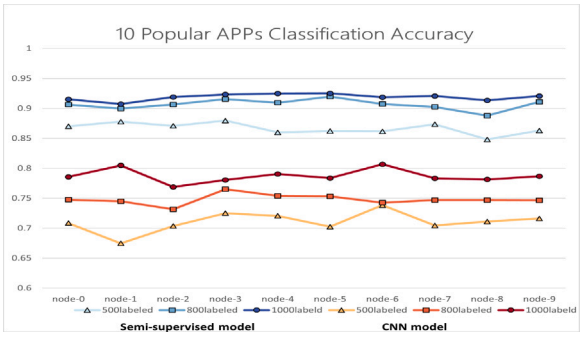
The performance of AECNN and FFSL is close, but from the F1-score indicator, except for iQiyi Video and Taobao Shopping, AECNN's F1-score is slightly higher than FFSL. This indicates that AECNN has stronger generalization ability and robustness.

Table 12
Exp4: Model resource consumption.

		AECNN	FFSL	FLUIDS
Global aggregation	avg_GPU_load	20.2	50.6	27.4
	avg_Memory_used	22.3	15.1	15.0
	avg_used_time (μs) (per global epoch)	4 178 139.5	3 419 772.5	3 373 981.2
Sub node training	avg_GPU_load	30.0	51.7	31.3
	avg_Memory_used	22.7	14.9	14.8
	avg_used_time (μs) (per node)	13 263 529.9	14 170 570.8	6 421 578.8
Global supervised training	avg_GPU_load	\	31.5	30.7
	avg_Memory_used	\	15.7	15.0
	avg_used_time (μs) (per global epoch)	\	12 683 591.3	16 902 534.8
Total used time		6840.67	7877.8	4204.3

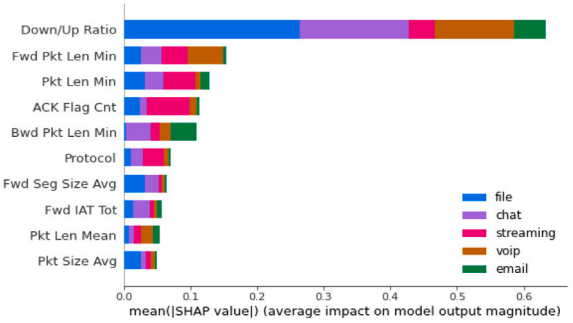


(a) ISCX Dataset Classification ACC with 1000 Labeled Data

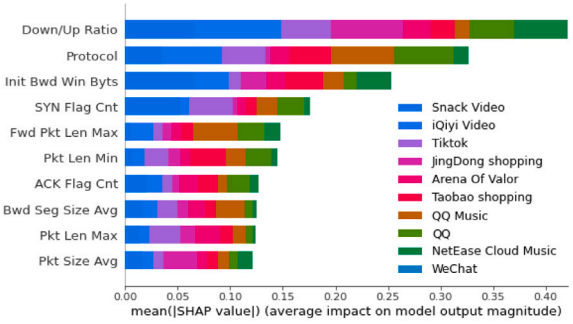


(b) 10 Popular APPS Dataset Classification ACC with 1000 Labeled Data

Fig. 12. Exp2: Model ACC comparison.



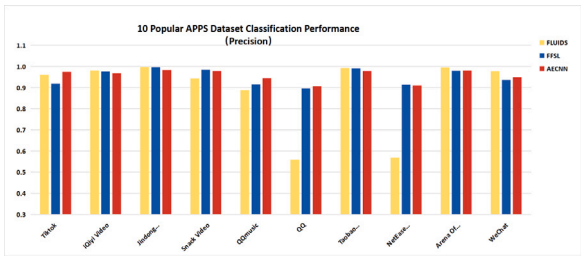
(a) ISCX DatasetFeature Contribution Ranking



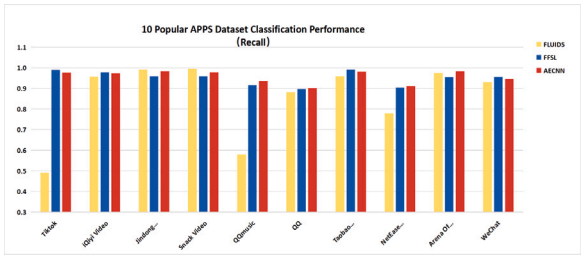
(b) 10 Popular APPS Dataset Feature Contribution Ranking

Fig. 13. Exp3: Feature contribution ranking.

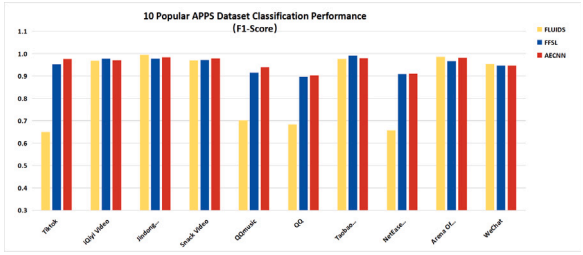
Table 12 reflects the resource consumption of the three methods. It can be seen from the table that FLUIDS has the lowest resource consumption in global aggregation and sub-node training, low GPU load and memory usage, and the shortest overall training time. FFSL has a high GPU load during local training and global aggregation, exceeding 50. Compared with AECNN, it has an extra global supervised training step, so the overall training time is also higher than AECNN. Therefore, considering the performance and resource consumption of the model, our proposed AECNN method performs better overall.



(a) The Precision on the 10 Popular APPs dataset



(b) The Recall on the 10 Popular APPs dataset



(c) The F1-Score on the 10 Popular APPs dataset

Fig. 14. EXP4: 10 popular APPs dataset classification performance.

6. Summary and future work

In this paper, we combine federated learning with deep learning to achieve data privacy protection while performing home traffic classification tasks. We reduce data dependency and improve accuracy by using semi-supervised models. We design a data collection method based on edge devices, which enables edge devices to have data collection and labeling capabilities. We analyze the final model after federated learning using explainable methods to ensure the model's reliability. We conducted experiments on two datasets, and the experimental results show that the proposed method performs well on various indicators. Although the proposed method has achieved some breakthroughs, there are still the following problems, which we will further explore in future research:

1. **Data non-independent and identically distributed.** Since each household user has different usage habits [32], the data distribution of each sub-node may also be inconsistent. When the data is non-IID [33], the performance of federated learning will drop significantly, and it needs to be alleviated by designing unique aggregation algorithms [34].

2. **Node selection.** Since the quality of traffic data generated by each household varies, different nodes provide different contributions to the global model [35]. It is necessary to evaluate the quality of each sub-node model before each aggregation and formulate and implement a dropout strategy to improve the overall performance of federated learning [36].
3. **Asynchronous training.** Since the available computing, communication resources and data volume on each edge device are usually different, there is a difference in the time for working nodes to submit model parameters after completing each round of local training. This will cause the parameter server to extend the training time due to waiting for slow nodes to upload parameters, greatly affecting the overall performance. Therefore, an asynchronous mechanism needs to be established to limit the impact of fast node high-frequency updates on the overall weights [37], improve the update efficiency of slow nodes, and make the model on the parameter server converge faster [38].

CRediT authorship contribution statement

ZiXuan Wang: Software, Resources, Conceptualization. **ZeYi Li:** Formal analysis, Data curation. **MengYi Fu:** Writing – original draft, Visualization, Validation, Conceptualization. **YingChun Ye:** Writing – review & editing, Visualization, Funding acquisition. **Pan Wang:** Writing – review & editing, Writing – original draft, Project administration.

Declaration of competing interest

The author(s) declared no potential conflicts of interest with respect to the research, authorship or publication of this article.

Acknowledgment

The paper is supported by National Natural Science Foundation (General Program) of China under Grant 61972211; Future Network Innovation Research and Application Projects under Grant No. 2021FNA02006 and 2021 Jiangsu Postgraduate Research Innovation Plan under Grant No. KYCX210794.

References

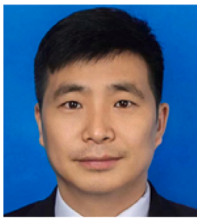
[1] Jielun Zhang, Fuhao Li, Feng Ye, Sustaining the high performance of AI-based network traffic classification models, IEEE/ACM Trans. Netw. 31 (2) (2022) 816–827.
[2] Pan Wang, Feng Ye, Xuejiao Chen, Yi Qian, Datanet: Deep learning based encrypted network traffic classification in sdn home gateway, IEEE Access 6 (2018) 55380–55391.
[3] Xiaokang Zhou, Wei Liang, I Kevin, Kai Wang, Laurence T Yang, Deep correlation mining based on hierarchical hybrid networks for heterogeneous big data recommendations, IEEE Trans. Comput. Soc. Syst. 8 (1) (2020) 171–178.
[4] Hongye He, Zhiguo Yang, Xiangning Chen, Payload encoding representation from transformer for encrypted traffic classification, ZTE Commun. 19 (4) (2022) 90–97.
[5] Xiaokang Zhou, Wang Huang, Wei Liang, Zheng Yan, Jianhua Ma, Yi Pan, I Kevin, Kai Wang, Federated distillation and blockchain empowered secure knowledge sharing for internet of medical things, Inf. Sci. (2024) 120217.
[6] Jielun Zhang, Shicong Liang, Feng Ye, Rose Qingyang Hu, Yi Qian, Towards detection of zero-day botnet attack in iot networks using federated learning, in: ICC 2023-IEEE International Conference on Communications, IEEE, 2023, pp. 7–12.
[7] Guan-Zhou Lin, Xin Yang, Xin-xin Niu, Hui-bai Jiang, Network traffic classification based on semi-supervised clustering, J. China Univ. Posts Telecommun. 17 (2010) 84–88.
[8] Timothy Glennan, Christopher Leckie, Sarah M. Erfani, Improved classification of known and unknown network traffic flows using semi-supervised machine learning, in: Information Security and Privacy: 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part II 21, Springer, 2016, pp. 493–501.
[9] Feng Qian, Guang-min Hu, Xing-miao Yao, Semi-supervised internet network traffic classification using a Gaussian mixture model, AEU-Int. J. Electron. Commun. 62 (7) (2008) 557–564.
[10] Fakhroddin Noorbehbahani, Sadeq Mansoori, A new semi-supervised method for network traffic classification based on X-means clustering and label propagation, in: 2018 8th International Conference on Computer and Knowledge Engineering, ICCKE, IEEE, 2018, pp. 120–125.
[11] Auwal Sani Iliyasu, Huifang Deng, Semi-supervised encrypted traffic classification with deep convolutional generative adversarial networks, IEEE Access 8 (2019) 118–126.
[12] Pan Wang, Zixuan Wang, Feng Ye, Xuejiao Chen, Bytesgan: A semi-supervised generative adversarial network for encrypted traffic classification in SDN edge gateway, Comput. Netw. 200 (2021) 108535.
[13] Ons Aouedi, Kandaraj Piamrat, Dhruvjyoti Bagadthey, A semi-supervised stacked autoencoder approach for network traffic classification, in: 2020 IEEE 28th International Conference on Network Protocols, ICNP, IEEE, 2020, pp. 1–6.
[14] Xiaokang Zhou, Xuzhe Zheng, Tian Shu, Wei Liang, I Kevin, Kai Wang, Lianyong Qi, Shohei Shimizu, Qun Jin, Information theoretic learning-enhanced dual-generative adversarial networks with causal representation for robust ood generalization, IEEE Trans. Neural Netw. Learn. Syst. (2023).
[15] Qiang Yang, Yang Liu, Tianjian Chen, Yongxin Tong, Federated machine learning: Concept and applications, ACM Trans. Intell. Syst. Technol. 10 (2) (2019) 1–19.
[16] Xiaokang Zhou, Wei Liang, I Kevin, Kai Wang, Zheng Yan, Laurence T Yang, Wei Wei, Jianhua Ma, Qun Jin, Decentralized p2p federated learning for privacy-preserving and resilient mobile robotic systems, IEEE Wirel. Commun. 30 (2) (2023) 82–89.
[17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Aguera y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282.
[18] Howard H. Yang, Zhao Zhongyuan, T.Q. Quek, Enabling intelligence at network edge network edge: An overview of federated learning, ZTE Commun. 18 (2) (2020) 2–10.
[19] Xiaokang Zhou, Xiaozhou Ye, I Kevin, Kai Wang, Wei Liang, Nirmal Kumar C Nair, Shohei Shimizu, Zheng Yan, Qun Jin, Hierarchical federated learning with social context clustering-based participant selection for internet of medical things applications, IEEE Trans. Comput. Soc. Syst. (2023).
[20] Xufeng Zhu, Nina Shu, Huaixi Wang, Tao Wu, A distributed traffic classification model based on Federated Learning, in: 2021 7th International Conference on Big Data Computing and Communications, BigCom, IEEE, 2021, pp. 75–81.
[21] Zhimin He, Jie Yin, Yu Wang, Guan Gui, Bamidele Adebisi, Tomoaki Ohtsuki, Haris Gacanin, Hikmet Sari, Edge device identification based on federated learning and network traffic feature engineering, IEEE Trans. Cogn. Commun. Netw. 8 (4) (2021) 1898–1909.
[22] Yingya Guo, Dan Wang, FEAT: A federated approach for privacy-preserving network traffic classification in heterogeneous environments, IEEE Internet Things J. 10 (2) (2022) 1274–1285.
[23] Ons Aouedi, Kandaraj Piamrat, Guillaume Muller, Kamal Singh, FLUIDS: Federated Learning with semi-supervised approach for Intrusion Detection System, in: 2022 IEEE 19th Annual Consumer Communications & Networking Conference, CCNC, IEEE, 2022, pp. 523–524.
[24] Ons Aouedi, Kandaraj Piamrat, Guillaume Muller, Kamal Singh, Intrusion detection for softwarized networks with semi-supervised federated learning, in: ICC 2022-IEEE International Conference on Communications, IEEE, 2022, pp. 5244–5249.
[25] Zhiping Jin, Zhibiao Liang, Meirong He, Yao Peng, Hanxiao Xue, Yu Wang, A federated semi-supervised learning approach for network traffic classification, Int. J. Netw. Manage. 33 (3) (2023) e2222.
[26] Saira Bano, Achilles Machumilane, Lorenzo Valerio, Pietro Cassarà, Alberto Gotta, Federated semi-supervised classification of multimedia flows for 3D networks, in: 2022 IEEE 21st Mediterranean Electrotechnical Conference, MELECON, IEEE, 2022, pp. 165–170.
[27] Wonyong Jeong, Jaehong Yoon, Eunho Yang, Sung Ju Hwang, Federated semi-supervised learning with inter-client consistency & disjoint learning, 2020, arXiv preprint arXiv:2006.12097.
[28] Iman Sharafaldin, Arash Habibi Lashkari, Ali A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization, in: ICISSp, Vol. 1, 2018, pp. 108–116.
[29] Chencheng Ma, Xuehui Du, Lifeng Cao, Improved KNN algorithm for fine-grained classification of encrypted network flow, Electronics 9 (2) (2020) 324.
[30] Ruixi Yuan, Zhu Li, Xiaohong Guan, Li Xu, An SVM-based machine learning method for accurate internet traffic classification, Inf. Syst. Front. 12 (2010) 149–156.
[31] Sultan Zavrak, Murat İskefiyeli, Anomaly-based intrusion detection from network flow features using variational autoencoder, IEEE Access 8 (2020) 108346–108358.
[32] Xiaokang Zhou, Qiuyue Yang, Xuzhe Zheng, Wei Liang, I Kevin, Kai Wang, Jianhua Ma, Yi Pan, Qun Jin, Personalized Federation Learning with Model-Contrastive Learning for Multi-Modal User Modeling in Human-Centric Metaverse, IEEE J. Sel. Areas Commun. (2024).
[33] Zhiyuan Wu, Sheng Sun, Yuwei Wang, Min Liu, Ke Xu, Wen Wang, Xuefeng Jiang, Bo Gao, Jinda Lu, Fedcache: A knowledge cache-driven federated learning architecture for personalized edge intelligence, IEEE Trans. Mob. Comput. (2024).

[34] Wenyu Zhang, Xiumin Wang, Pan Zhou, Weiwei Wu, Xinglin Zhang, Client selection for federated learning with non-iid data in mobile edge computing, IEEE Access 9 (2021) 24462–24474.
[35] Xiaokang Zhou, Xuzhe Zheng, Xuesong Cui, Jiashuai Shi, Wei Liang, Zheng Yan, Laurance T Yang, Shohei Shimizu, I Kevin, Kai Wang, Digital twin enhanced federated reinforcement learning with lightweight knowledge distillation in mobile networks, IEEE J. Sel. Areas Commun. (2023).
[36] Dong Yang, Weiting Zhang, Qiang Ye, Chuan Zhang, Ning Zhang, Chuan Huang, Hongke Zhang, Xuemin Shen, Detfed: Dynamic resource scheduling for deterministic federated learning over time-sensitive networks, IEEE Trans. Mob. Comput. (2023).
[37] Chenhao Xu, Youyang Qu, Yong Xiang, Longxiang Gao, Asynchronous federated learning on heterogeneous devices: A survey, Comput. Sci. Rev. 50 (2023) 100595.
[38] Xiaokang Zhou, Qiuyue Yang, Qiang Liu, Wei Liang, Kevin Wang, Zhi Liu, Jianhua Ma, Qun Jin, Spatial–Temporal Federated Transfer Learning with multi-sensor data fusion for cooperative positioning, Inf. Fusion 105 (2024) 102182.

ZiXuan Wang was born in Nanjing, Jiangsu, China, in 1994. He obtained a master's degree in logistics engineering at Nanjing University of Posts and Telecommunications in 2020, He is currently pursuing a Phd degree at Nanjing University of Posts and Telecommunications. His research interests include encrypted traffic identification and federated learning.

Zeyi Li is currently pursuing a Ph.D. degree in Cyberspace Security at Nanjing University of Posts and Telecommunications. He was born in Soochow, Jiangsu, China, in 1997. He received his bachelor'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.

Mengyi Fu is currently pursuing a Ph.D. degree at Nanjing University of Posts and Telecommunications. She received her B.Sc from Nanjing University of Posts and Telecommunications, Nanjing China, in 2022. Her research includes encrypted traffic identification and deep learning.



YingChun Ye obtained his master's degree from the University of Science and Technology of China in 2012. He is currently the executive vice president of Jiangsu Future Network Group Co., Ltd. and an industrial professor at Xi'an Jiaotong-Liverpool University. As the main person in charge of the projects, he has undertaken five key projects at the provincial and ministerial levels and more than 20 projects at the municipal and district levels, with a total project scale of over 500 million yuan. His main research direction is the next-generation network architecture, including deterministic networks, time-sensitive networks, etc.

Pan Wang (M'18) received a BS degree from the Department of Communication Engineering, Nanjing University of Posts & Telecommunications, Nanjing, China, in 2001 and a Ph.D. degree in Electrical & Computer Engineering from Nanjing University of Posts&Telecommunications, Nanjing, China, in 2013. He is currently an Associate Professor in the School of Modern Posts, Nanjing University of Posts & 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 at the University of Dayton (UD) in the Department of Electrical and Computer Engineering.