Hindawi
Discrete Dynamics in Nature and Society
Volume 2023, Article ID 2879563, 13 pages
https://doi.org/10.1155/2023/2879563

Research Article

Semisupervised Graph Neural Networks for Traffic Classification in Edge Networks

Yang Yang, Rui Lyu, Zhipeng Gao, Lanlan Rui, and Yu Yan

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China

Correspondence should be addressed to Yang Yang; yyang@bupt.edu.cn

Received 14 October 2022; Revised 6 April 2023; Accepted 23 May 2023; Published 3 July 2023

Academic Editor: Sarangapani Jagannathan

Edge networking brings computation and data storage as close to the point of request as possible. Various intelligent devices are connected to the edge nodes where traffic packets flow. Traffic classification tasks are thought to be a keystone for network management; researchers can analyze packets captured to understand the traffic as it hits their network. However, the existing traffic classification framework needs to conduct a unified analysis, which leads to the huge bandwidth resources required in the process of transferring all captured packet files to train a global classifier. In this paper, a semisupervised graph neural network traffic classifier is proposed for cloud-edge architecture so that cloud servers and edge nodes could cooperate to perform the traffic classification tasks in order to deliver low latency and save bandwidth on the edge nodes. To preserve the structural information and interrelationships conveyed in packets within a session, we transform traffic sessions into graphs. We segment the frequently combined consecutive packets into granules, which are later transformed into the nodes in graphs. Edges could extract the adjacency of the granules in the sessions; the edge node side then selects the highly representative samples and sends them to the cloud server; the server side uses graph neural networks to perform semisupervised classification tasks on the selected training set. Our method has been trained and tested on several datasets, such as the VPN-nonVPN dataset, and the experimental results show good performance on accuracy, recall, and F-score.

1. Introduction

In the edge network environment, tens of millions of edge nodes are linked together through countless network nodes for data interaction and analysis. At the same time, more and more edge devices are also joining the Internet of Things. Each network application has its own corresponding traffic behavior characteristics. With the continuous emergence of various new network applications and network application layer protocols, the complexity of network traffic is increasing and becoming more changeable, dynamic, and heterogeneous. To meet network specification requirements for a given type of service, traffic data need to be classified with a high degree of accuracy to satisfy the QoS requirements. Nowadays, the mainstream traffic classification method always arranges its models on the centralized cloud server [1], and the edge terminals only take responsibility for sending the collected traffic to the cloud server to train the traffic classifier. That will result in subpar real-time performance and raise edge nodes' bandwidth overhead.

The centralized processing mode based on cloud computing models has successfully aggregated computing power and storage capacity and performed unified network management. Due to the limitations of the edge node's hardware resources, it is often necessary to provide relevant services to users through remote cloud computing resources, and the cloud server still bears a huge computing load. Therefore, it is a new trend to mount neural networks on edge nodes. With more and more edge terminals joining the network, the existing centralized cloud computing service has delegated computing resources to the edge, allowing more data processing tasks to be completed nearby. The current trend in technology development is more inclined to perform tasks on flexible but resource-constrained terminal devices. However, for AI technology, most of the intelligent algorithms are computationally intensive and require strong computing power for support. Due to the limitations of the hardware performance of the edge node device itself and the network communication environment, we are faced with the problem that it is difficult to realize all of them at the same time.

In this paper, we paid attention on how to perform traffic classification tasks in edge networks. The existing traffic analysis and identification framework needs to conduct unified analysis [2–4], which leads to the huge bandwidth resources required in the process of transferring all captured packet files to train a global classifier. When faced with the current situation of edge networks featuring high dynamic, large scale, low bandwidth resources, and weak links, the depth model with a large parameter scale finds it difficult to play a role on edge nodes. In order to maintain the efficiency of edge-side traffic identification and traceability, it is necessary to design lightweight traffic analysis models and traceability suitable for cloud-edge end-to-end collaboration scenarios and be able to match and analyze the service QoS.

Network traffic classification maps the traffic flow through the network according to its type; thus, the managers can have an overview of the network conditions, which is also thought to be a prerequisite for subsequent management decisions. There are usually three types of classifiers applied in the traffic classification fields [5]: port-based classification methods that group traffic's kinds according to port information; machine learning-based methods that classify traffic according to the statistical features (e.g., conventional machine-based approaches connect the statistical signatures from traffic samples to each application type); and expert labor to select the features to fit the models. The features that are usually used are packet length, packet direction, packet arrival time, and so on. However, in recent years, researchers have focused on automatically learning feature-based trained deep neural network models for well-known application kinds. Deep learning-based methods make up for the limitations brought by classic machine learning methods as they do not incorporate human labors. Many deep learning models have been applied in the traffic classification field [6–8]. Recently, Pang et al. [6] proposed a method that applied graph neural networks to traffic classification. To generate graphs, session' packets are extracted as nodes, and edges are used to record the order information for the traffic sessions; however, the chained graph model only considers the order arrangement of the packets and does not explore the interrelationships between the packets in a session.

In this paper, we propose a novel semi-supervised traffic classification method based on graph convolutional neural networks. We process the traffic packets uploaded and transform them into graphs to convey their structural information. Then we use graph neural networks to further extract the features of the traffic data. Finally, we have used multiple GCNNs to expand the training set for the cloud server. On the publicly available network traffic dataset "ISCX VPN and Non-VPN dataset," we verify the efficacy of our model. The experiment's findings show that it accomplished outstanding classification.

This paper is to address the existing issues for traffic classification tasks in edge networks as the existing methods like deep packet inspection (DPI) [9] remain, which require unified traffic identification analysis after port mirroring, leading to huge bandwidth resources occupied during the process of copying messages from one or more ports (the source port) of the device to a monitoring port (the destination port) of the device. The upload of the complete traffic data from edge nodes to a cloud server that is frequently used in traditional methods brings a lot of problems, as the explosive growth of traffic in its volume and complexity will result in consuming a large number of bandwidth resources and poor responsiveness of the edge system, thus it cannot assume real-time performance and impede other normal services. Semi-supervised methods can be a good choice to solve the problem that it is easy to collect data but labeling it is cumbersome, especially when given the massive amount of traffic per second forwarding in and out of edge gateways. And also, to relieve the bandwidth pressure of transferring the traffic data from the edge nodes to the cloud server, the edge nodes will select some samples to upload for the training of the cloud server.

Besides, to better extract the features when generating the raw captured traffic sessions into graphs. If we consider the packets in a session as nodes, all nodes have their own feature information (e.g., sequential features of raw packet streams and statistical features concluded from packet bytes) and structural information (e.g., its relative position in the session and the structure of its byte-length sequence), we can abstract a session to a graph to cover the inter- and intra-relationships it conveys.

The following is the paper's primary contributions:

(1) The designed model is typically applied to the cloud-edge architecture as the edge nodes extract the features from the raw capture files that come in from the terminal side and select the samples for the training set, while the server side performs the semisupervised learning that tries to finish graph classification jobs with just a few samples of labeled graphs. In our framework, edge nodes use several GCNNs to choose highly representative graph instances from the newly collected data, and after pseudo-labeling, add them to the training set.

(2) Rather than only regarding the packets in a session as the nodes in the graph, in this paper, we borrow the concept of "flow granules" to cover the internal information between data packets. Individual packets in sessions are extracted and packed into several granules which are incorporated into the

graph as nodes. The relative positions and structural information of the granules are transformed into edges. A graph that represents a session and is labeled as the session's traffic type is later used in classification.

(3) Our model uses graph convolutional neural networks to capture the traffic data's structural information. Our solution surpasses various state-of-the-art approaches and produces great results on labeled network traffic datasets that are publicly available for traffic classification.

The arrangement of the remaining sections in this paper is as listed: The proposed traffic classification algorithms for semisupervised jobs are then discussed, starting with an introduction to the associated work and a description of the preliminary steps. The following section of this research paper introduces the datasets utilized and assesses the performance of the proposed model in comparison to previous network traffic classification methods. Finally, the paper comes to a conclusion.

2. Related Works

2.1. Traffic Classification. In light of the swift advancement of network technologies and the explosive growth of the scale of network traffic, network traffic of diverse types requires different underlying network resources. Therefore, in order to achieve efficient network management and improve the quality of network service, it is necessary to effectively monitor and classify network traffic.

In recent years, as a result of the rapid development of deep learning in artificial intelligence and other domains, many academics have started attempting to use deep learning to solve the problem of network traffic classification, thus achieving the purpose of online intelligent identification of network traffic. Because they extract features without the help of experts, deep learning-based approaches are different from conventional machine learning-based methods or packet inspection-based methods. Additionally, deep learning-based methods are more capable of learning than conventional machine learning methods, which allow them to perform better overall [4].

Wang et al. [10] proposed a 1D-CNN-based encrypted traffic classifier extracting features directly from bytes of raw traffic. Lotfollahi et al. [11] proposed a method called deep-packet which used the first 1480 bytes of each IP packet as model input to perform packet-level traffic classification tasks and accomplished excellent performance. Lopez-Martin et al. [12] combined recurrent neural networks (RNNs) and CNNs to categorize traffic for every packet in the session using six extracted statistical features. An RNN-based technique for traffic classification termed BSNN was proposed by Li et al. [13]. Long short-term memory (LSTM) or gated recurrent units (GRUs) serve as the foundation for the RNN component of BSNN. Network datagrams are treated as input by BSNN, which provides the categorization outcomes immediately. Liu et al. [14] later introduced the FS-Net, an end-to-end traffic classification model in which a multilayer encoder-decoder structure fed with flows' sequential features as packet length sequence was used to further enhance the RNN-based encrypted traffic classifier. In [15], multimodal multitask cutting-edge deep learning approaches are applied in a systematic framework to create a viable mobile traffic classifier, which can jointly learn the shared representation of the sequential features (payload bytes) and statistical features (informative protocol header fields) of sessions. That work has been further improved in [16], and explainable artificial intelligence (XAI) is employed to extrapolate the categorization process of the improved version on the state-of-the-art multimodal traffic classifier [17]. In [18], hybrid neural networks are used to analyze the dual-mode features that are extracted from the raw traffic data.

In Tables 1 and 2, the related work section is complemented by a table categorizing the reviewed works along with their primary distinguishing features so as to position the present contribution effectively. In Table 1, the single-modal traffic classifier is presented, and within each category, the works are presented in order of publication. The following definitions are provided for acronyms and columns. Column "Research" listed the research for comparison. Column "Input Data" means the input data for the deep learning models; in the entries, LX means the Xth layer of the ISO/OSI model. Column "Traffic object," briefly known as TO, means the traffic classification granularity adopted, Entry "B" means biflow/session, "F" means flow, "P" means packet, and "D" means IP datagram. Column "DL Classifier" means the deep learning models adopted for the traffic classifier, in the entries, BiGRU means bi-directional gated recurrent unit, CNN means convolutional neural network, LSTM means long short-term memory, MLP means multilayer perceptron, and SAE means stacked auto encoder. Column "Open," briefly known as O, means whether the publicly available dataset has been adopted, Entry "Y" means Yes, "N" means No, and "P" means partial. In Table 2, multimodal traffic classification architectures are listed. The following definitions are provided for columns and acronyms in Table 2 that are absent from Table 1. Column "Multimodal," briefly as known as MM, means whether the multimodal deep learning techniques are employed. Column "Multitask," briefly known as MT, means whether the multitask deep learning techniques are employed. Column "Supervised Shared Representation" and Column "Training-Phase Specification," briefly known as SSR and TPS, respectively, clarify whether the traffic classifier uses those techniques. For all above columns, "Y" means Yes, "N" means No, "P" means Partial, and "-" means not applicable.

We can find that most classifiers only handled single types of features for classifiers and few of them dealt with multimodal inputs (MM columns) with specific subsets of the heterogeneous inputs being trained on the lowest layers. So, we concluded the single-model and multimodal architectures, respectively, in Tables 1 and 2.

A biflow is the most frequently used traffic object (TO column), both for extracting input data and for assigning classification labels. Also, deep learning models tend to

Table 1: Related works based on deep learning models (single-modal architectures).

Research	Input data	TO	DL classifiers	O
Wang et al. [10], *proc. IEEE ISI*	(1) PCAP trace (2) First 784 bytes of L4 payload	F/B	1D-CNN	Y
Lopez-Martin et al. [12], *IEEE access*	6 fields extracted per packets and first 20 packets per biflow	B	2D-CNN +LSTM	N
Li et al. [13], *IWQoS*	IP datagram	D	RNN	N
Liu et al. [14], *proc. IEEE INFOCOM*	IP packet lengths	B	BiGRU	N
Lotfollahi et al. [11], *soft computing*	First 1500 bytes of L2 payload	P	SAE/1D-CNN	Y

Table 2: Related works based on deep learning models (multimodal architectures).

Research	Input data	TO	DL classifier	MM	MT	SSR	TPS	O
Aceto et al. [17], *Elsevier ComNet*	(1) First 576 bytes of L4 payload (2) 4 fields for the first 12 packets	B	1D-CNN&BiGRU	Y	N	—	Y	P
Aceto et al. [15], *JNCA*	(1) First 784 bytes of L4 payload (2) 4 fields for the first 32 packets	B	1D-CNN&BiGRU	Y	Y	Y	Y	Y
Nascita et al. [16], *TNSM*	(1) First 576 bytes of L4 payload (2) 4 fields for the first 12 packets	B	1D-CNN&BiGRU	Y	N	—	Y	Y

extract features from the raw input in an end-to-end way. It is the way that deep learning methods learn automatically and do not involve expert labors so that they gain popularity than machine learning methods in traffic classification fields. Numerous alternative methods have been developed for the classification tasks, including Deep neural networks (DNNs), various autoEncoders (AEs), one- and two-dimensional convolutional neural networks (1D and 2D-CNNs), and various recurrent neural networks (RNNs) (DL classifier column). Besides, it showed that a proportion of these publications validate and assess the performance of their classifiers using publicly accessible datasets (open columns).

In this paper, to learn the different views of the classification object, we have tried to combine the sequential features and statistical features to extract the features from the raw PCAP files; that is, we have tried to take advantage of the packet length sequence to segment the packet-byte sequence into granules (which could be transformed into the nodes in the graphs). Then graph neural networks are employed for traffic classification.

2.2. Semisupervised Traffic Classification. In fact, we cannot label all samples, as it will require a lot of labor. In order to solve the problem of insufficient network traffic data labels, methods based on semisupervised deep learning have gained popularity. Semisupervised learning (SSL) is a key problem in the fields of machine learning and pattern recognition. It is a technique for learning that combines supervised and unsupervised learning. Regarding semisupervised learning, pattern recognition is performed on the basis of both labeled and unlabeled data in huge quantities. Semisupervised learning can produce relatively good accuracy while also requiring the fewest number of workers possible. Consequently, semisupervised learning is receiving increased attention.

In recent decades, numerous deep learning-based semisupervised methods have proven their efficiency and effectiveness in the traffic classification field. Deep convolutional generated adversarial networks (DCGAN) have been employed in [19]. The accuracy of their method is almost the same as that of the supervised method for the labeled large datasets. The authors in [20, 21] adopted autoencoders, which are thought to be a common technology in semisupervised learning. In [22], the author used stacked sparse autoencoders (SSAEs). The results obtained demonstrate better performance than the traditional model. In [23], the author proposed a variational automatic encoder (VAE)-based model for anomaly detection. The model is superior to other semisupervised learning models, and the evaluation index increases by 5–10%.

Wang et al. [24] proposed a SDN edge gateway-embedded semisupervised traffic classifier based on generative adversarial networks (GANs). By training and testing on the public dataset “ISCX2012 VPN-nonVPN,” the experimental results demonstrate that the ByteSGAN can effectively outperform other supervised-learning based methods such as CNN. In this paper, the graph convolutional neural networks (GCNNs) are further utilized in the traffic categorization model. GCNN is a kind of convolutional neural network that can directly act on graphs and utilize their structural information.

2.3. Traffic Classification in Edge Networks. In the edge network environment, where network traffic of various heterogeneous types grows exponentially, how to effectively perform traffic classification tasks in edge scenarios remains a problem for researchers. As mentioned, when dealing with SDN edge gateways, Wang et al. [24] proposed a semi-supervised traffic classifier using GANs. In SDN edge gateways, various intelligent devices are connected to the edge gateway through wireless access technologies, and all data packets from these smart devices will be queued on the WAN interface, waiting for the edge gateway to forward them out of order. The traffic classification process is mostly concentrated on the SDN controller. Obviously, SDN controllers will suffer from huge flow processing pressure. Though in [24] only traffic classifiers that are applied in SDN edge networks are considered, it can still give some inspiration for traffic classifiers applied in edge networks.

In an edge environment, there is often just a virtualized resource pool made up of many servers. However, when a number of terminal devices are linked to the edge platform via the edge side, there is frequently significant resource demand on the edge side. Numerous terminals and sensors are networked to the edge platform in numerous contexts, including medical, industrial, and the Internet of vehicles. Higher standards are needed to be presented for edge clouds. As shown in Figure 1, in the process of edge-side network traffic classification and recognition, the DL models are trained based on the labeled network traffic data. However, in the actual situation, the classifier often receives the network traffic of unmarked categories, resulting in misclassification and other problems. In addition, because the labeled sample size is too small, the model trained with small samples is easy to fall into over-fitting of small samples and under-fitting of target tasks.

With the increasing complexity of network topology and the explosive growth of network applications, the network traffic on the edge side presents features like nonlinearity, high complexity, and auto-correlation. At the same time, the network traffic of different applications varies greatly, which brings difficulties and challenges to the accurate marking of network traffic. The traditional traffic analysis and identification framework is based on the cloud server for unified analysis, which leads to the need for huge bandwidth resources in the process of transmitting all captured packet files to the cloud server. Therefore, current model applications are more and more inclined to deploy from the cloud to the edge to reduce bandwidth consumption.

In the cloud-edge integration system, the edge gateway can realize local linkage between the device and data processing and analysis without networking [25]. However, the edge node deployment traffic analysis model still has problems. When faced with low-bandwidth resources and weak-link edge networks, the deep learning model with huge parameters is unable to play a role on the edge nodes. In order to perform accurate traffic classification tasks on the edge-side network, cloud-edge collaboration can be used to balance the processing pressure of edge nodes and maximize the advantages of cloud computing and edge computing with high processing efficiency and low latency.

The edge gateways and cloud server can cooperate to share the pressure of the central cloud node. Some of the data computation and storage work is carried out by the edge

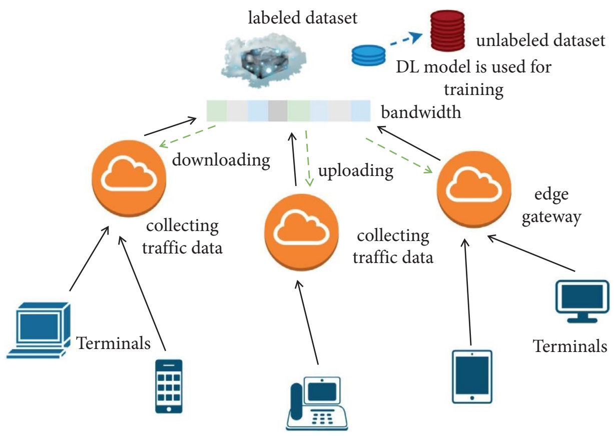


Figure 1: The traffic classification process in edge gateways.

computing node, reducing the computing processing pressure of the edge cloud server to aggregate the traffic data of each node for unified traffic analysis. Based on this, this paper proposes a semisupervised network traffic classification and identification method for cloud-side collaboration scenarios. This method distributes part of the traffic analysis tasks on the central cloud server and the edge gateway to jointly complete the edge-side traffic identification task and realize the efficient use of computing resources.

3. Graph-Based Traffic Classification in Edge Computing Networks

To deal with bandwidth shortage problems brought by traditional methods, we are considering a cloud-edge integrated collaborative system with several edge gateways and cloud servers. Traditional traffic classifiers tend to collect all captured raw traffic files together, and that may eat up the network bandwidth while transmitting raw files. In a cloud-edge integrated system, we split the traffic classification tasks into several phases and put them on edge nodes and a cloud server to fully utilize all computing resources and prevent latency and bandwidth insufficiency when putting all models and traffic raw captured files on one side. The detail is depicted in Figure 2.

At the edge gateway layer, there are mainly two stages as feature extraction and graph generation. The specific process is as follows:

(1) The edge-side gateway captures and processes the features of the traffic packets uploaded by the terminal node to avoid a large number of complete packets' information being uploaded to the cloud center, so as to reduce the data processing delay and relieve the resource pressure on the cloud server.

(2) At the same time, the edge gateway will use the graph neural networks to further transform the traffic data into graphs, using "granules" to further extract the interrelationship between individual packets within a session. Multiple GCNNs are also used to select and transmit the samples with high confidence to the cloud server.

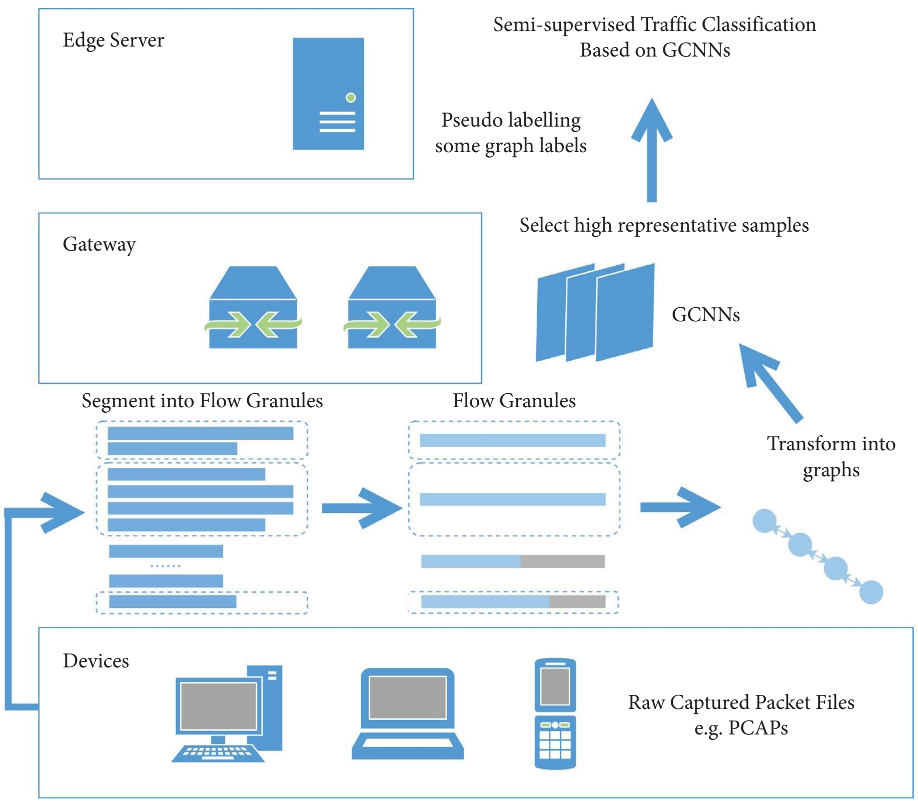

Edge Server
Semi-supervised Traffic Classification Based on GCNNs
Pseudo labelling some graph labels
Gateway
Select high representative samples
GCNNs
Segment into Flow Granules
Flow Granules
Transform into graphs
Devices
Raw Captured Packet Files e.g. PCAPs

FIGURE 2: The flow diagram of the semisupervised traffic classifier.

(3) At the edge-server layer, the cloud server conducts semi-supervised traffic classification on the traffic information selected and uploaded by the edge node, then unifies the training model, and then delegates the model weight and other information to the edge gateway.

3.1. Graph Generation Based on Granules in Edge Nodes. The edge nodes (here we mean edge computing gateways) will collect all raw packet files (PCAPs) captured by edge devices that are abstracted as edge nodes in the network. The edge nodes will further process the PCAP files and put them into the graph neural networks to classify the traffic session.

The definition of the flow is provided first. Flow separation is the initial step after receiving the raw traffic files (PCAPs). Traffic is made up of flows, and a flow is a collection of packets that can be uniquely identified by the traditional five-tuple notation (source IP, destination IP, source port, destination port, and protocol). Sessions are made up of packets with an exchangeable pair of network source and destination, and they include all packets sent between two hosts during the course of a session. This paper uses a session as the traffic classification granularity.

We typically cover 2 types of features: session-level features and the packets' sequential features. Packet length sequence (i.e., the number of bytes per packet) is used as a session-level features. Besides, we typically select the first **M** bytes for every packet to compose the byte sequence of this session (packets with length less than **M** will be pad with zeros to reach **M**-byte sequence, while the packets' bytes exceeding the range of **M** will be truncated). As depicted in Figure 3, a biflow classification object can be transformed into a vector of $X \in \mathbb{R}^{n \times M}$, where n is the size of packets of the session. For the packet level, the individual packet can be vertically segmented into several granules, as shown in Figure 3(a), that would be transformed into vertices of the graphs. When segmenting the neighboring packets into the granules in the next part, we use the session' statistical features (a session's packet length sequence) to perform the packet segmentation.

Secondly, the edge gateway further processes the traffic data uploaded by the edge devices. As depicted in Figures 3(a) and 3(b), each packet-byte sequence is vertically segmented according to the session-level information.

To fully explain how to transform the session's packet-byte sequence according to the packet length sequence. At first, we introduce the concept of "flow granule." The concept of "flow granule" used in this paper is inspired and derived from [26]. The term "flow granule" was initially used in [26], which explained how neighborhood data packets with the same packet length might be aggregated to create granules. As a result, an aggregated packet sequence rather than a single, unique packet now represents the information to be processed. Here, we try to segment the packet sequence to extract the internal relationship between the packets themselves. If consecutive interarrival data packets tend to show similarities in their length or very probably appear in the same neighborhood, they will be combined to become a "granule." The session is composed of a sequence of

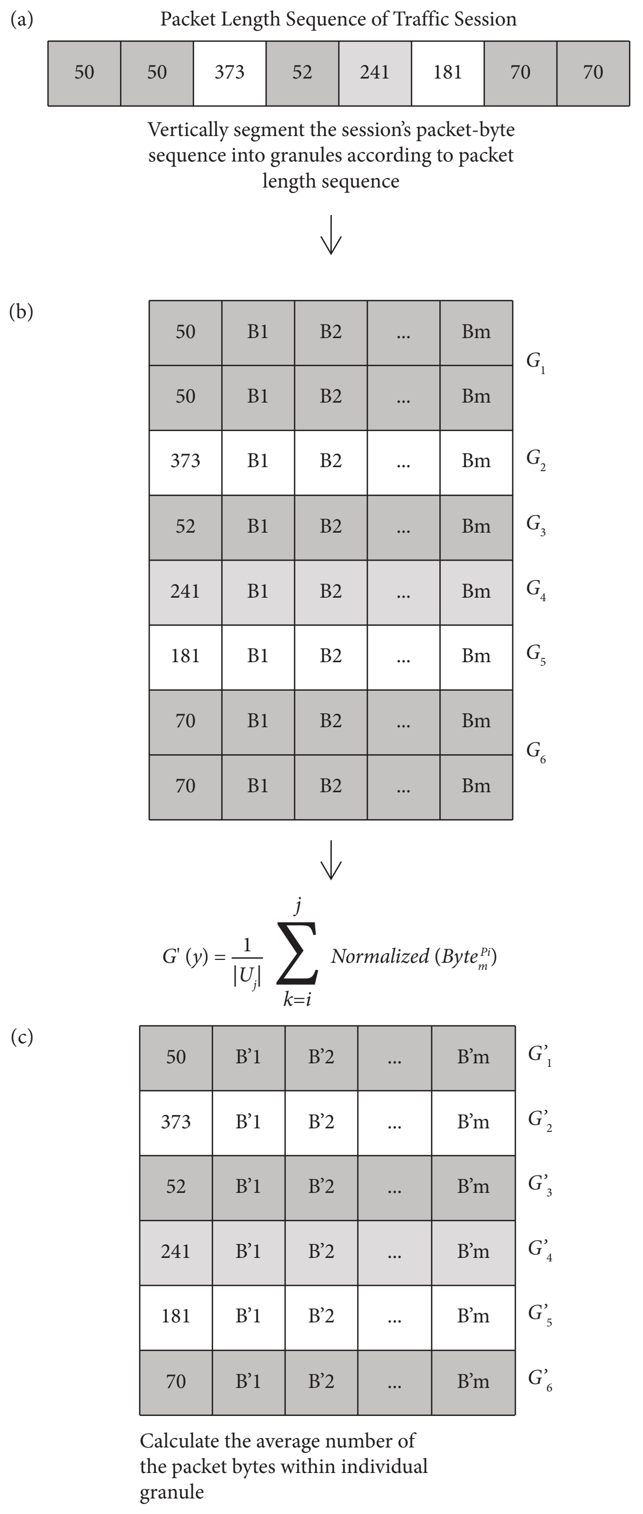

(a)
Packet Length Sequence of Traffic Session
50
50
373
52
241
181
70
70
Vertically segment the session's packet-byte sequence into granules according to packet length sequence
(b)
B1
B2
...
Bm
G1
G2
G3
G4
G5
G6
G' (y) = 1/|Uj| Σ_{k=i}^{j} Normalized (Byte_m^{Pi})
(c)
B'1
B'2
B'm
G'1
G'2
G'3
G'4
G'5
G'6
Calculate the average number of the packet bytes within individual granule

FIGURE 3: Transforming the session's packet-byte sequence into graphs.

granules that will still keep the order within the granule itself and within the session. Now suppose, we segment the session sequence into several subsequences to find the granule segments. Our goal is to segment the session' packet sequence $\mathbf{X} = \mathbf{x}_{1:T} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$, where T is the packet size for that session. Neighboring packets with similar sizes can be combined together to make a granule that later can be transformed into a node in graphs. After vertical segmentation, we get the granule $G(\mathbf{y})|_{y=1.2\ldots Y}$ per session as formula (1); Y is the size of the granules per session.

$$\begin{aligned} G(\mathbf{y}) &= \bigcup_{k=i}^{j} \mathbf{P_i} \in U, \\ s.t. &\left|P_i - P_{i+1}\right| < \text{threshold}_v. \end{aligned} \tag{1}$$

After vertically segmenting the packets to make up the flow granules, to further transform the flow granules into graph nodes, the packets within the flow granule are combined, as depicted in Figure 3(c), by calculating the average number of the packet bytes.

$$G'(\mathbf{y}) = \frac{1}{\left|U_j\right|} \sum_{k=i}^{j} \text{Normalized}\left(\text{Byte}_m^{P_i}\right), \tag{2}$$

where $m = 1, 2, \ldots, M$, and M means the length of the packet sequence. $|U_j|$ means the packet size of that granule.

Nodes correspond to granules, while edges refer to the adjacency between the granules when transforming every session's packet-byte sequence into graphs. As opposed to the proposed methods in [6], the features of every packet in the session are extracted to represent the graph's nodes and then a chained graph is created according to the session's packet order. In this step, we not only cover the packet-byte sequence itself but also explore the interrelationship between the consecutive data packets, as an individual granule can be represented as a node in graphs.

Each node is associated with a feature vector for that granule. Then edges generated. By default, each edge is undirected. After obtaining the nodes, we extract the set of edges between nodes according to the adjacency between packets. Here we use undirected edges because undirected edges capture the relative sequence relationship between each granule better than directed edges. Suppose the node feature matrix at this point is $F \in \mathbb{R}^{y\times M}$, accordingly, we construct node correlation functions, which take a node feature matrix as input and produce the corresponding adjacency matrix. $\mathbf{A}_T \in \mathbb{R}^{y\times y}$: If two original messages are adjacent in the raw sequence, an undirected edge is established between the corresponding two nodes.

$$\mathbf{A}_T = \text{Corr}\left(\mathbf{X}_T\right), \tag{3}$$

where $\text{Corr}(\bullet)$ computes correlations or dependencies of each channels (nodes) on the basis of $\mathbf{X}_T$. Node correlation functions come in a variety of options; here, we calculate the correlations by

$$\text{Dif}\left(\boldsymbol{\alpha}_a, \boldsymbol{\alpha}_b\right) \triangleq 1 - \frac{2\cdot\boldsymbol{\alpha}_a\boldsymbol{\alpha}_b^T}{\boldsymbol{\alpha}_a\boldsymbol{\alpha}_a^T + \boldsymbol{\alpha}_b\boldsymbol{\alpha}_b^T}. \tag{4}$$

In this stage, we extract the relationships between the subsequent data packets in addition to using the isolated packet information and transform a session into graphs. Later, we can use the graph neural networks to model the traffic data and further mine the inner relationship between the traffic data to its types.

3.2. Semisupervised Traffic Classification in Edge Server and Centralized Server. In the last subsection, sessions have been transfered into graphs to convey the structural information between packets within the session. In this stage, our goal is to forecast the class labels of graphs in order to forecast the label of the traffic session that turned into the graph. A node in a graph often symbolizes an object in the real world; in this paper, it means the granules. Moreover, sessions can also be interconnected and abstracted as nodes in the global graph. Here we explore a more difficult but practically valuable scenario in which a node is a graph instance in and of itself. After the raw byte sequence of the traffic sessions has been transformed into several graphs to model the interdependency within the sessions' packets themselves. We can find that the session-level graphs show their similarity when the graphs are categorized in the same application types, which means, a set of graph instances can be modeled into a hierarchical graph connecting individual graphs with edges.

Regarding graph classification tasks, typical graph-based neural network algorithms often need a large number of labeled graph samples. However, since large-scale labeled graph datasets often come at a significant cost in terms of time and effort, graph classification jobs frequently encounter the issue of a lack of labeled graph samples. Besides, considering the edge networks, we cannot deploy all the neural networks on the edge nodes since only limited resources are allowed on edge nodes. If all the work is pursued on the centered servers, it would lead to high latency and wasting computing resources. In this paper, active learning techniques are employed to enhance the efficiency of semisupervised learning, which unifies the edge nodes and central server.

In this part, we first introduce a self-attentive graph embedding techniques to include graphs of any size into fixed-length vectors, which are frequently utilized as semisupervised classification input. In addition to greatly simplifying the representation of a hierarchical graph, the embedding approach also offers meaningful interpretations of an individual graph instance through a self-attentive mechanism that distinguishes their role in categorizing a graph instance. This phase that connects and unifies the edge nodes and cloud server can be simply separated into two parts as graph embedding and graph-based classification, which are depicted in Figure 4. The former is to transform the graph with variable node sizes into a fixed length vector, and latter can be the classification input in the latter part.

As follows, we give the descriptions for the graph embedding part that takes the processed samples from the previous stage as input. As depicted in Figure 5, the purpose of this part is to convert the graph with different number of nodes into a vector $\mathbf{e_n}$ with a unified dimension and then use it as the input for graph classification. Firstly, two layers of GCNs are applied, with the adjacency matrix $\mathbf{A} \in \mathbb{R}^{y\times y}$ and attribution matrix $\mathbf{F} \in \mathbb{R}^{y\times M}$ as input. Then, we get

$$\mathbf{H} = \widehat{\mathbf{A}}\mathrm{ReLU}\left(\widehat{\mathbf{A}}\mathbf{F}\mathbf{W^0}\right)\mathbf{W^1}. \tag{5}$$

In the former formula, $\widehat{\mathbf{A}} = \widetilde{\mathbf{D}}^{-(1/2)}(\mathbf{A} + \mathbf{I_n})\widetilde{\mathbf{D}}^{-(1/2)}$ is the normalized form of adjacency matrix $\mathbf{A}$ where $\mathbf{I_n}$ is identity matrix and $\widetilde{\mathbf{D}} = \sum_m (\mathbf{A} + \mathbf{I_n})_{\mathrm{im}}$. Here, $\mathbf{W^0} \in \mathbb{R}^{M\times h}$ and $\mathbf{W^1} \in \mathbb{R}^{M\times v}$ are two weight matrices.

Next, we use the self-attentive mechanism to assign different weights to nodes in the graph, to differentiate the nodes within a graph. After softmax, we can also get the predicted class probabilities ψ after a fully connected layer.

$$\mathbf{S} = \mathrm{softmax}\left(\mathbf{W_{s2}}\tanh\left(\mathbf{W_{s1}}\mathbf{H^T}\right)\right), \tag{6}$$

where $\mathbf{W_{s1}} \in \mathbb{R}^{d\times v}$ and $W_{s2} \in \mathbb{R}^{r\times d}$. The purpose of multiplying $\mathbf{W_{s1}}$ is to convert the node representation from a v- to a d-dimensional space linearly. After that, nonlinearity is added by coupling with the function tanh. $\mathbf{W_{s2}}$ is used to infer the importance assigned to each node within the graph.

Lastly, we get the $\mathbf{e} \in \mathbb{R}^{r\times v}$ by multiplying $\mathbf{S}$ and $\mathbf{H}$.

The predicted class probabilities $\boldsymbol{\psi}$ would be used for picking out the samples sent to the cloud server. To enhance graph classification performance, we should choose which samples may be successfully filtered out and applied to a graph neural network-based classifier on a cloud server afterwards. The framework determines which graph examples are often considered to be significant for enhancing the performance of the graph classification model by employing a number of supervised classifiers. The training set is then updated with these examples.

We introduce a unified classifier system, which uses weighted majority voting to combine the decisions of $\mathbf{P}$ classifiers to decide the final label of the graph sample after $\mathbf{P}$ isolated GCN-based classifiers and obtains the weight by maximizing the performance of the whole expert set. To be more specific, each classifier has the same set-up as Figure 4 but has a different kernel size. When the individual classifier gets its predicted class probabilities ψ, the final decisions could be calculated. If the results show the same label as the final voting results, then we add weights to that classifier. The weighted voting method assigns a certain weight to each classifier member, and the weight is obtained by measuring the classifier accuracy of each member on the training set. The weight is proportional to the accuracy; that is, the base classifier with good classification ability is given a larger weight coefficient, while the base classifier with relatively poor classification ability is given a smaller weight coefficient, and the integration result depends on the weighted sum.

Now, we need to select the samples with high performance gain which is mathematically described as the weighted mean value of the class output probability value calculated by the last phase. To be more specific, we define the graph samples that will be picked out as training samples with a weighted mean classification probability of $\mathbf{P}$ classifiers higher than a threshold. The selected samples would be annotated and added to the labeled training set to improve the effectiveness of the GCN-based classifier on the cloud server and further improve the accuracy of the graph classification tasks.

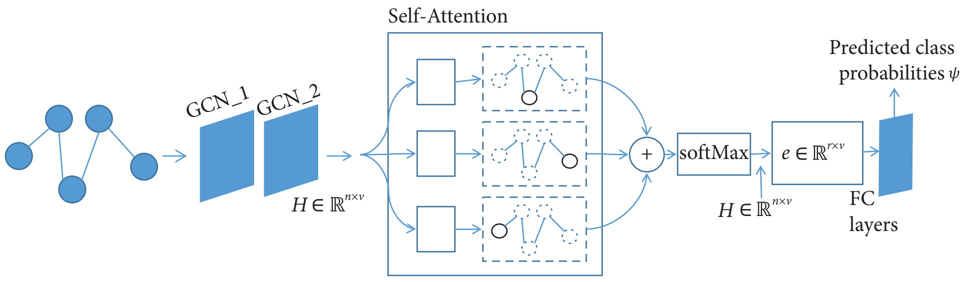

Self-Attention
GCN_1
GCN_2
$H \in \mathbb{R}^{n\times v}$
softMax
$e \in \mathbb{R}^{r\times v}$
$H \in \mathbb{R}^{n\times v}$
Predicted class probabilities ψ
FC layers

Figure 4: Graph embedding by the self-attention mechanism.

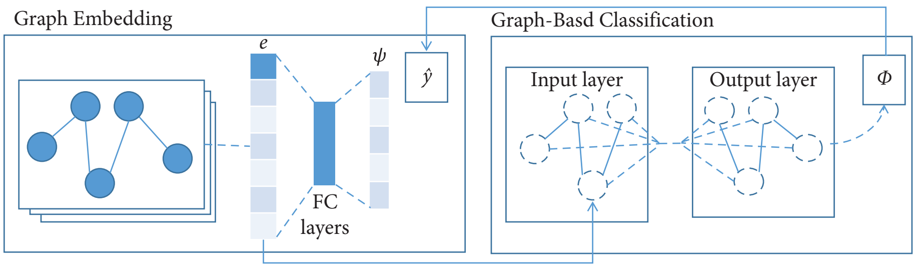


Figure 5: The process of graph-based classification.

The Graph Embedding phase tends to enlarge the labeled training set and produce the fix length $\mathbf{e} \in \mathbb{R}^{r\times v}$ that would be the model input in the cloud server. The semi-supervised classification phase is set up on cloud servers.

The definitions of the problems are given at first. Graphs are represented as $\mathscr{G}_m = (V, E)$, V is the set of nodes while E is the set of edges that define a graph. The goal is to map the graph to its class label as function $f\colon \{\mathscr{G}_m\}_{m=1}^{M} \longrightarrow \mathscr{Y}$ given the set of graphs $\{\mathscr{G}_m\}_{m=1}^{M}$. We incorporate the active graph classification phase to select a set of graphs $G_{\text{select}} = \{\mathscr{G}_{l+1}, \ldots, \mathscr{G}_{l+k}\}$ from the unlabeled samples G_U to the labeled training set G_L after annotation so that the new training set in the center server can have a better ability to predict the unlabeled class labels. In order to enhance graph classification outcomes for semi-supervised learning, our method chooses unlabeled graph samples with high confidence in multiple GCNs' clustering and adds them to the training set after pseudo labeling.

GCN-based models are chosen to be employed in semi-supervised training on the cloud server. Now the graph embedding $E = \{e\}_{i=1}^{L+U}$ and the adjacency matrix $\Theta \in \mathbb{R}^{(L+U)\times(L+U)}$ have been given, which are calculated according to formula (4). Two GCN layers are later used here; the classification probability of each graph example will be represented by the Softmax layer as follows:

$$\Phi = \operatorname{softmax}\Big(\widehat{\Theta}\operatorname{ReLU}\Big(\widehat{\Theta}\mathrm{FW}_{\Theta}^{0}\Big)\mathrm{W}_{\Theta}^{1}\Big). \tag{7}$$

$\widehat{\Theta} = \widetilde{\mathrm{D}}^{-(1/2)}(\Theta + I_n)\widetilde{\mathrm{D}}^{-(1/2)}$ is the normalized form of adjacency matrix Θ and I_n is represented as identity matrix and $\widetilde{\mathrm{D}} = \sum_m (\Theta + I_n)_{\text{im}}$. W_{Θ}^{0} and W_{Θ}^{1} are weight matrices.

The parameters in edge nodes are not retrained but rather fine-tuned depending on the parameters gained in the previous iteration to further increase the efficiency.

In the cloud server, the graph neural network-based semi-supervised traffic classifier uses pseudo labeled samples to further train itself and a softmax layer to get its outputs.

4. Results and Discussion

4.1. Dataset. Based on the dataset of packet capture (PCAP) files from the University of New Brunswick (UNB): "ISCX VPN-nonVPN traffic dataset" [27] and "ISCX Tor-nonTor dataset" (ISCX-Tor) [28], which are publicly accessible labeled datasets, we analyzed and examined our methodology. The traffic of fourteen programs, including Skype, VPN-VOIP, VPN-P2P, and others, is covered within the ISCX-VPN dataset. ISCX-nonVPN includes traffic from fifteen applications, including Skype, FTPS/SFTP, and others. ISCX-nonTOR carries the traffic of sixteen apps, classified into several categories as Web Browsing, e-mail, Chat, and others. Initially, these traffics are identified by the acts that caused them.

4.2. Evaluation Metrics. All approaches are evaluated based on their accuracy (A.), recall (R.), and F1-score (F1). The following are the definitions:

$$\begin{aligned}
\text{Accuracy} &= \frac{\sum_{i\in\text{classes}} \mathrm{TP}_i}{\sum_{i\in\text{classes}} \left(\mathrm{TP}_i + \mathrm{FP}_i\right)}, \\
\text{Recall} &= \frac{\sum_{i\in\text{classes}} \mathrm{TP}_i}{\sum_{i\in\text{classes}} \left(\mathrm{TP}_i + \mathrm{FN}_i\right)}, \\
\mathrm{F1} &= \frac{2 \times \text{recall} \times \text{accuracy}}{\text{recall} + \text{accuracy}}.
\end{aligned} \tag{8}$$

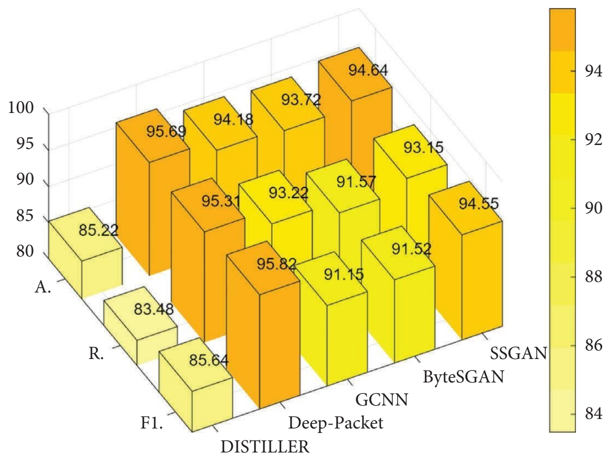


Figure 6: The experimental comparison for the VPN dataset.

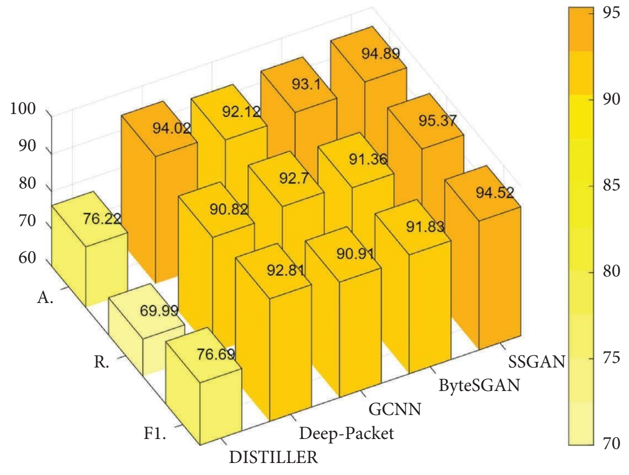


Figure 7: The experimental comparison for the non-VPN dataset.

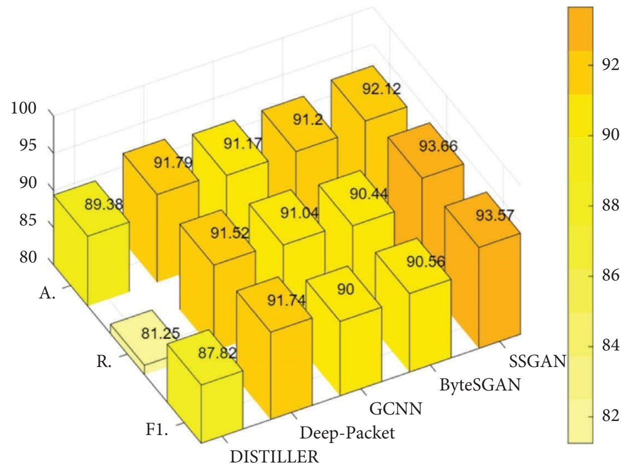


Figure 8: The experimental comparison for the TOR dataset.

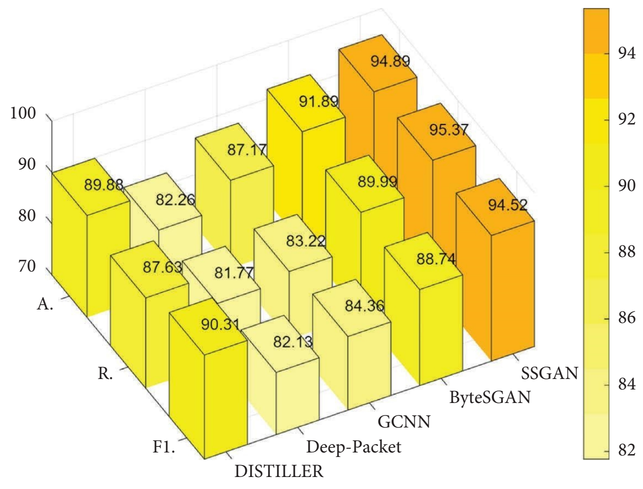


Figure 9: The experimental comparison for the non-TOR dataset.

The number of packets accurately classified into each class is represented by the TP for that class. The amount of instances that are erroneously classified into a specific class is known as FP. The number FN indicates how many occurrences are incorrectly categorized into one class but truly belong to another. In further detail, examples of class A are considered the positive class for computing F1 for that class, whilst instances of all other classes are considered the negative class.

4.3. Baselines. To completely illustrate the usefulness and efficiency of our suggested approaches, we compare our model with four traffic categorization methods, including DeepPacket [11], GCNN [6], DISTILLER [15], and the semisupervised classifier ByteSGAN [24]. The detailed explanations and comparison are given as follows.

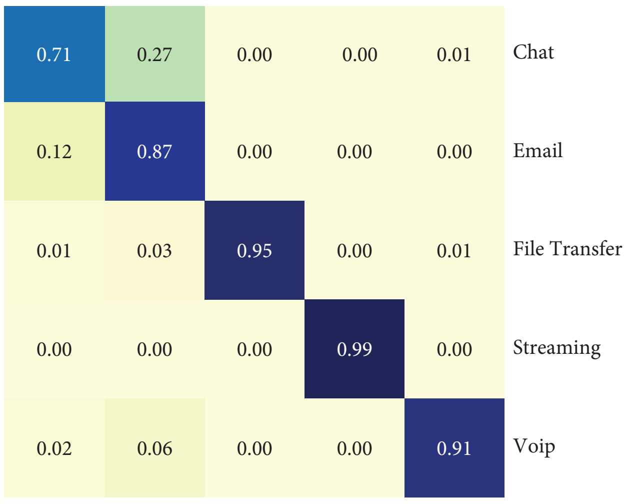
Deep-Packet: This method combines the steps of feature extraction and classification into a single system and is based on CNNs to process the byte sequence of a packet. On the UNB ISCX VPN-nonVPN dataset, it performs admirably.

GCNN: This method uses a chained graph model on the traffic packet data and performs supervised traffic classification tasks on graph neural networks over automatically extracted features over the chained graphs.

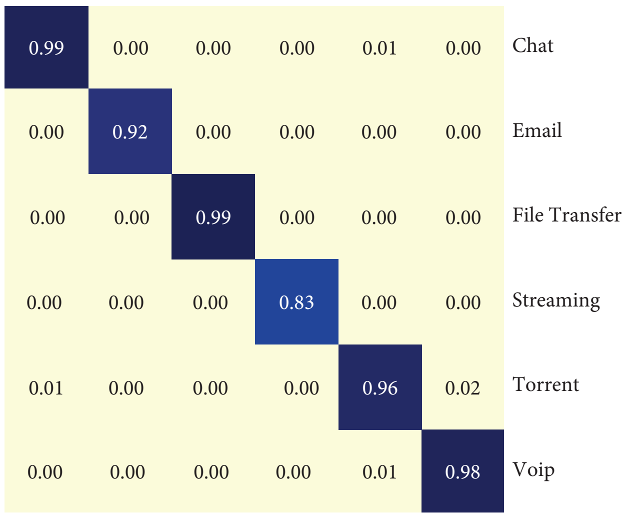
DISTILLER: This method leverages the combined and efficient use of multitasking and multimodal deep learning techniques. It handles three jobs at once: encapsulation, traffic kinds, and traffic application categorization tasks. Additionally, it offers the traffic object from the packet-level and flow-level viewpoints. However, in this study, we just focus on the problem of traffic type categorization.

ByteSGAN: This method typically employs semi-supervised learning approaches based on generative adversarial networks (GAN) for the categorization of encrypted data. It is intended to be incorporated in the SDN edge gateways. The method uses the packet-byte data for the model input.

4.4. Effectiveness Analysis. It is evident that the enhanced algorithm performs well. The findings will be displayed in Figures 6, 7, 8, and 9 in order to more clearly illustrate how this method has improved things.



(a)



(b)

FIGURE 10: The confusion matrix for the VPN and non-VPN dataset.

4.4.1. Experimental Setup. We set $P = 3$ graph convolutional neural networks built on edge nodes to vote for the last label outcome and set a 1500-byte input sequence length limit; if length is less than 1500, pad zeros; if more, then truncate the byte sequence. We set 2 layers of graph convolutional networks for each classifier and their input and output dimensions which are $n \times 64$, $n \times 128$, $n \times 256$, where n represents the number of nodes. To perform semi-supervised traffic classification learning, we randomly choose 30% of labeled datasets to train the ByteSGAN and SSGAN (our semi-supervised classifier).

4.4.2. Experimental Results of VPN Dataset. Figure 6 shows that our method shows excellent performance when applied to the VPN dataset. The accuracy, recall, and F1-score all rise when compared to the DISTILLER method by 9.42%, 9.67%, and 8.91%, respectively. Contrasting with the GCNN algorithm, the accuracy and F1-score are improved by the method we use by 0.46% and 3.40%, respectively. Compared to the ByteSGAN semi-supervised algorithm, by 0.92%, 1.57%, and 3.03%, respectively, our technique raises accuracy, recall, and the F1-score.

4.4.3. Experimental Results of Non-VPN Dataset. Figure 7 shows that our method applied to the non-VPN dataset performs best. Compared to the DISTILLER algorithm, the accuracy improves by 18.67%, the recall increases by 25.38%, and there is a 17.83% boost in the F1-score. In contrast to the deep-packet algorithm, the accuracy, recall, and F1-score are improved by our approach by 0.87%, 4.55%, and 1.71%, respectively. In contrast to the GCNN algorithm, our method increases accuracy, recall, and the F1-score by 2.77%, 5.10%, and 3.61%, respectively. Compared to the ByteSGAN semi-supervised algorithm, our method increases by 1.79%, 4.01%, and 2.69% in terms of accuracy, recall, and the F1-score.

4.4.4. Experimental Results of TOR Dataset. Figure 8 shows that our method shows excellent performance when applied to the TOR dataset. The accuracy rises by 2.74%, the recall rises by 12.41%, and the F1-score rises by 5.75% in comparison to the DISTILLER method. In contrast to the Deep-Packet algorithm, our method increases accuracy, recall, and the F1-score by 0.33%, 2.14%, and 1.83%, respectively. In contrast to the GCNN algorithm, our method increases accuracy, recall, and the F1-score by 0.95%, 2.62%, and 3.57%, respectively. Compared to the ByteSGAN semi-supervised algorithm, our method increases accuracy, recall, and the F1-score by 0.92%, 3.22%, and 3.01%, respectively.

4.4.5. Experimental Results of Non-TOR Dataset. Figure 9 shows that our method applied to the non-TOR dataset performs best. Compared to the DISTILLER algorithm, the accuracy improves by 5.01%, the recall increases by 7.74%, and the F1-score increases by 4.21%. Compared to the deep-packet algorithm, our method increases accuracy, recall, and F1-score by 12.63%, 13.6%, and 12.39%, respectively. When compared to the GCNN algorithm, our method increases accuracy, recall, and F1-score by 7.72%, 12.15%, and 10.16%, respectively. Compared to the ByteSGAN semisupervised algorithm, our method improves accuracy, recall, and F1-score by 3.0%, 5.38%, and 5.78%, respectively.

Figure 10 shows the confusion matrix for the VPN and non-VPN dataset, and we can see that almost all the traffic types have good performance. In the graph generation part, since we extract the traffic features by grouping packets as granules according to their sequential features, we can have better performance than simply transforming the simple packets into nodes and generating chained graphs when we use these graphs to perform traffic classification later. Also, we compare our methods with the fully-supervised methods since we want to prove that the SSGCN can have the same efficacy as the supervised methods such as deep-packet using only 30% part of the labeled dataset. Besides, SSGCN also shows its effectiveness when compared to the semi-supervised methods like ByteSGAN, as we employ the graphs to extract the structural information for the traffic data.

5. Conclusions

In this paper, we have presented a novel semi-supervised traffic classification approach based on improved graph convolutional neural networks. In the edge-server integrated-system, the traffic packets uploaded are processed and transformed into graphs. We have used multiple GCNNs to enlarge the training set for the cloud server. The cloud server performs the semisupervised traffic classification tasks based on graph convolutional networks. On publicly available network traffic datasets, we verify the efficacy of our model. The experiment's findings show that it is possible to accomplish outstanding classification.

In further study, we will investigate these aspects of the suggested methodology:

(1) The majority of current traffic classifiers operate inside the predefined traffic categories. These techniques cannot handle unrecognized traffic from unrecognized classes. Zero-day applications are traffic classifications for which the classifier has not been trained. Just a small number of recent studies, many of which rely on locating unlabeled clusters and later classifying them, have offered solutions for zero-day applications.

(2) The procedure for deploying the network will be extended to the real environment. More metrics will be introduced to measure the traffic classifier' performance.

Data Availability

The training dataset is from https://www.unb.ca/cic/datasets/vpn.html.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key R&D Program of China (2019YFB2103202).

References

[1] A. Bianco, P. Giaccone, S. Kelki, N. M. Campos, S. Traverso, and T. Zhang, "On-the-fly traffic classification and control with a stateful SDN approach," in *Proceedings of the 2017 IEEE International Conference on Communications (ICC)*, pp. 1–6, Paris, France, May 2017.

[2] P. Amaral, J. Dinis, P. Pinto, L. Bernardo, J. Tavares, and H. S. Mamede, "Machine learning in software defined networks: data collection and traffic classification," in *Proceedings of the 2016 IEEE 24th International Conference on Network Protocols (ICNP)*, pp. 1–5, Singapore, November 2016.

[3] P. Amaral, P. F. Pinto, L. Bernardo, and A. Mazandarani, "Application aware SDN architecture using semi-supervised traffic classification," in *Proceedings of the 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 1–6, Verona, Italy, November 2018.

[4] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[5] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: an overview," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 76–81, 2019.

[6] B. Pang, Y. Fu, S. Ren, Y. Wang, Q. Liao, and Y. Jia, "CGNN: traffic classification with graph neural network," 2021, https://arxiv.org/abs/2110.09726.

[7] P. Wang, F. Ye, X. Chen, and Y. Qian, "Datanet: deep learning based encrypted network traffic classification in SDN home gateway," *IEEE Access*, vol. 6, pp. 55380–55391, 2018.

[8] X. Chen, J. Yu, F. Ye, and P. Wang, "A hierarchical approach to encrypted data packet classification in smart home gateways," in *Proceedings of the 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pp. 41–45, Athens, Greece, August 2018.

[9] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "Blindbox: deep packet inspection over encrypted traffic," in *Proceedings of the 2015 ACM conference on special interest group on data communication*, pp. 213–226, New York, NY, USA, October 2015.

[10] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proceedings of the 2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 43–48, Beijing, China, July 2017.

[11] M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, and M. Saberian, "Deep packet: a novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.

[12] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for Internet of Things," *IEEE Access*, vol. 5, pp. 18042–18050, 2017.

[13] R. Li, X. Xiao, S. Ni, Z. Haitao, and X. Shutao, "Byte segment neural network for network traffic classification," in *Proceedings of the 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pp. 1–10, Banff, Canada, June 2018.

[14] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "Fs-net: a flow sequence network for encrypted traffic classification," in *Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 1171–1179, Paris, France, May 2019.

[15] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "DISTILLER: encrypted traffic classification via multimodal multitask deep learning," *Journal of Network and Computer Applications*, vol. 183-184, Article ID 102985, 2021.

[16] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescape, "XAI meets mobile traffic classification: understanding and improving multimodal deep learning

architectures," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4225–4246, 2021.

[17] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "MIMETIC: mobile encrypted traffic classification using multimodal deep learning," *Computer Networks*, vol. 165, Article ID 106944, 2019.

[18] Y. Yang, Y. Yu, G. Zhipeng et al., "A network traffic classification method based on dual-mode feature extraction and hybrid neural networks," *IEEE Transactions on Network and Service Management*, 2023.

[19] D. Zhao, J. Weng, and Y. Liu, "Generating traffic scene with deep convolutional generative adversarial networks," in *Proceedings of the 2017 Chinese Automation Congress (CAC)*, pp. 6612–6617, Jinan, China, October 2017.

[20] D. Li, Y. Zhu, and W. Lin, "Traffic identification of mobile apps based on variational autoencoder network," in *Proceedings of the 2017 13th International Conference on Computational Intelligence and Security (CIS)*, pp. 287–291, Hong Kong, China, December 2017.

[21] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Offline/realtime traffic classification using semi-supervised learning," *Performance Evaluation*, vol. 64, no. 9-12, pp. 1194–1213, 2007.

[22] O. Aouedi, K. Piamrat, and D. Bagadthey, "A semi-supervised stacked autoencoder approach for network traffic classification," in *Proceedings of the 2020 IEEE 28th International Conference on Network Protocols (ICNP)*, pp. 1–6, Madrid, Spain, October 2020.

[23] T. Li, S. Chen, Z. Yao, X. Chen, and J. Yang, "Semi-supervised network traffic classification using deep generative models," in *Proceedings of the 2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pp. 1282–1288, Huangshan, China, July 2018.

[24] P. Wang, Z. Wang, F. Ye, and X. Chen, "ByteSGAN: a semi-supervised generative adversarial network for encrypted traffic classification in SDN edge gateway," *Computer Networks*, vol. 200, Article ID 108535, 2021.

[25] C. H. Chen, M. Y. Lin, and C. C. Liu, "Edge computing gateway of the industrial internet of things using multiple collaborative microcontrollers," *IEEE Network*, vol. 32, no. 1, pp. 24–32, 2018.

[26] P. Tang, Y. Dong, and S. Mao, "Online traffic classification using granules," in *Proceedings of the IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1135–1140, Toronto, Canada, July 2020.

[27] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and 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)*, pp. 407–414, Lisbon, Portugal, February 2016.

[28] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features," in *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 253–262, Lisbon, Portugal, September 2017.