Encrypted Traffic Classification with Graph Sketch-Based Traffic Representation

Sitong Yu[1], Ding Li[1, 2, 3*], Yanxing Guo[1], Ruyu Xie[1] and Yan Zhang[1, 2, 3]
[1]*School of Computer Science, Hubei University*
[2]*Hubei Key Laboratory of Big Data Intelligent Analysis and Application (Hubei University)*
[3]*Key Laboratory of Intelligent Sensing System and Security (Hubei University), Ministry of Education of PRC*
Wuhan, Hubei 430062, China
yst@stu.hubu.edu.cn, liding@hubu.edu.cn

2025 International Joint Conference on Neural Networks (IJCNN) | 979-8-3315-1042-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/IJCNN64981.2025.11228514

***Abstract*—Deep learning-based approaches have proven to be effective in encrypted traffic classification. However, existing methods did not take into consideration some critical challenges in this task. First, due to the fact that networking hardware often runs with limited computational resources, it is impractical to convert a raw traffic flow to a space-consuming and computation-expensive representation (e.g., byte-level representaion) or adopt large-scale models (e.g., transformer-based model) to conduct classification. Further, existing packet-level representation methods did not utilize adequate packet information since they constructed the representation with only a small number of packets for each traffic flow. Third, most methods represented raw traffic in only a single modality, which did not capture comprehensive traffic features. To solve the above challenges, in this paper, we propose a graph sketch-based traffic representation method which makes full use of packet information to convert raw traffic to two kinds of light-weight representations, time series and graph. Then, we adopt different deep learning models to extract features from the two kinds of representations, respectively. Finally, we fuse the features extracted from different representations to a final feature vector and input it to a fully-connected layer for classification. Extensive experiments are conducted on two public datasets, ISCX-nonVPN and ISCX-VPN, and experimental results show that our solution outperforms the state-of-the-arts in terms of classification accuracy.**

***Index Terms*—encrypted traffic classification, graph sketch, deep learning**

I. Introduction

Traffic classification plays an important role in many scenarios such as detecting threats in intrusion detection systems and improving service quality as well as user experience [1]. In recent years, various traffic encryption techniques such as Internet Protocol Security (IPSec) have been adopted to protect user privacy and anonymity. As a result, encrypted traffic classification is receiving vast attention from researchers as well as industry professionals.

The earliest port-based classification methods [2] are now ineffective due to the usage of dynamic ports. The subsequent deep packet inspection (DPI) approach [3] also does not work well since it is time-consuming and has low accuracy when handling encrypted traffic. To improve the classification performance, researchers proposed a series of machine learning-based methods [4] [5]. For instance, Liu et al. devised a feature selection method to find the most important statistical features of traffic flows (e.g., the largest packet length), and then utilized a random forest model to predict in-app services [6]. However, the feature selection process highly relies on expert experience, and the statistical features are less recognizable compared to deep features, resulting in relatively low accuracy.

To this end, researcher proposed various deep learning-based methods to improve the classification accuracy. These solutions mainly include two steps: first devising an approach to converting a raw traffic flow to a normalized representation, and then designing a deep neural network (DNN) based model to automatically extract hidden features from the representation for classification. For instance, Liu et al. proposed FS-Net [7] which adopted recurrent neural network (RNN) and a multi-layer encoder-decoder structure to mine hidden features from flow sequences. Subsequently, Lin et al. devised TSCRNN [8]. TSCRNN first converted a raw traffic flow to a standard format by vectorzing each packet in the flow, and then combined 1D convolutional neural network (CNN) and LSTM to conduct the classification. More recently, Lin et al. devised a traffic representation model called ET-BERT [9]. ET-BERT first pre-trained deep contextualized datagram-level representation from large-scale unlabeled data, and then fine-tuned the pre-trained model on a small number of task-specific labeled data. In 2023, Zhao et al. proposed YaTC [10] which represented traffic with hierachical flow information and conducted traffic classification using their designed Traffic Transformer. In the same year, Fauvel et al. designed LEXNet [11] which represented a traffic flow by a matrix of packet size and proposed a new residual block and prototype layer to conduct the efficient and explainable classification. Besides, Zhang proposed TFE-GNN [12] which first constructed a byte-level traffic graph and then extracted features from the traffic graph using a graph neural network (GNN) based traffic graph encoder.

However, the above works mainly have the following flaws: **1) Ignoring the fact that computational resources of networking hardware are limited.** Some methods [12] claimed to convert raw traffic to byte-level representations which were space-consuming and computation-expensive. In

*This work was supported by the Major Science and Technology Project of Hubei Province (No. 2024BAA008). The corresponding author is Ding Li. (E-mail: liding@hubu.edu.cn)

addition, for classification tasks, they constructed large-scale models such as transformer-based models [9] [10] which were not adaptive to networking environment. **2) Inadequate utilization of packet information.** Although some methods [8] [10] [11] constructed a light-weight representation for raw traffic, for each traffic flow, they only used a very small number of packets to construct the representation since most DNN based models require a fixed-size input. For instance, LEXNet [11] only used 20 packets to construct the representation for each traffic flow. However, these representation methods might lead to a serious loss of information since a short-time (e.g., 10-second) traffic flow may contain thousands of packets. **3) Representing raw traffic in a single modality.** Most methods converted a raw traffic flow in only one kind of representation, such as sequence [7] [8] or graph [12] [13] [14]. However, such representation methods could not capture comprehensive traffic features.

To address the above challenges, in this paper, we develop a novel encrypted traffic classification mechanism named ETC-GS. The framework of ETC-GS is shown in Fig. 1. First, with the summarization ability of graph sketches [15], we convert an encrypted traffic flow to several light-weight representations (i.e., graph sketches) by capturing spatiotemporal information of packets and potential correlations between packets. We emphasize that our method makes full use of all the packets in a traffic segment to construct the representations, which is quite different from the existing methods that used a very small number of packets. Next, we utilize long short-term memory (LSTM) models and graph convolution network (GCN) [16] models to respectively extract higher-level features from different kinds of representations. Finally, we fuse the multimodal features extracted from different sketches to a final feature vector and input it to a fully-connected layer to conduct classification.

The main contributions of this paper are summarized as follows.

- We propose a novel mechanism named ETC-GS for encrypted traffic classification. ETC-GS breaks through the existing traffic analysis methods in two perspectives: raw traffic representation and classifier structure.
- We design a graph sketch-based traffic representation method to represent a raw traffic flow in multiple modalities by capturing spatiotemporal information of packets and potential correlation between packets. To the best of our knowledge, we are the first to utilize graph sketches to construct traffic representations.
- Based on our designed traffic representations, we devise a traffic classifier which first uses LSTM models and GCN models to respectively extract hidden features from different representations, and then fuse the extracted features to a final feature vector for classification task.
- We conduct extensive experiments on two public encrypted traffic datasets, and experimental results show that our ETC-GS achieves better performance compared to the state-of-the-art methods.

II. Related Works

A. Rule-Based Method

In the early stage, traffic classification mainly relied on rule-based methods [2] [3] [17] [18]. The rules were designed by domain experts, and common attributes for classification include communication protocol, port number, etc. However, the fundamental features are now unable to fulfill the current traffic analysis requirement due to more and more complicated network environments [4].

B. Machine Learning-Based Method

Due to the ineffectiveness of rule-based methods in complicated network environments, researchers applied machine learning methods to encrypted traffic classification. For instance, AppScanner [4] proposed in 2016 adopted a random forest model to analyze the statistical features of traffic for real-time identification of Android apps. Afterwards, Liu et al. presented FRF [6] which also used a random forest model to conduct classification. The difference is that Liu el al. devised a novel feature selection method to find out the most distinguishable statistical features. Xiao et al. introduced DMTCS [19] which treated protocols as topics, and then applied topic models to cluster packets. Subsequently, Ede et al. proffered FlowPrint [5], a semi-supervised approach which automatically found temporal correlations among destination-related features of network traffic.

C. Deep Learning-Based Method

To avoid the performance degradation caused by unreliable hand-crafted features, researchers proposed deep learning-based methods which analyzed encrypted traffic based on raw packets. In 2020, Zheng et al. introduced RBRN [20] that learned representative features from the raw flows and then classified them in a unified framework. Subsequently, Shen et al. presented GraphDApp [21], a decentralized application fingerprinting method using GNN models. More recently, Luxemburk et al. trained a multimodal neural network consisting of 1D convolutions and linear layers for identifying TLS services in encrypted traffic [22]. Yun et al. were interested in encrypted TLS traffic classification on cloud platforms and proposed NeuTic [23]. NeuTic took the packet sequence of each TLS flow as the input, and classified raw TLS flows generated by cloud applications. In 2024, Chen et al designed MPAF [24] which used three phases to separately leverage attributes that emerged at different time periods, and conducted the classification based on a leaf node masking tree. Deng et al. devised AN-Net [25] which aimed to combat per-packet attribute noise and improve the classification accuracy in anonymous network. There also exist a lot of other works. Since we have discussed these work in section I, we do not provide repeated introductions here due to page limit.

III. Preliminaries

Before illustrating our proposed method, in this section, we first introduce the rationale of graph sketch, give some formal definitions, and formulate the problem that we focus on.

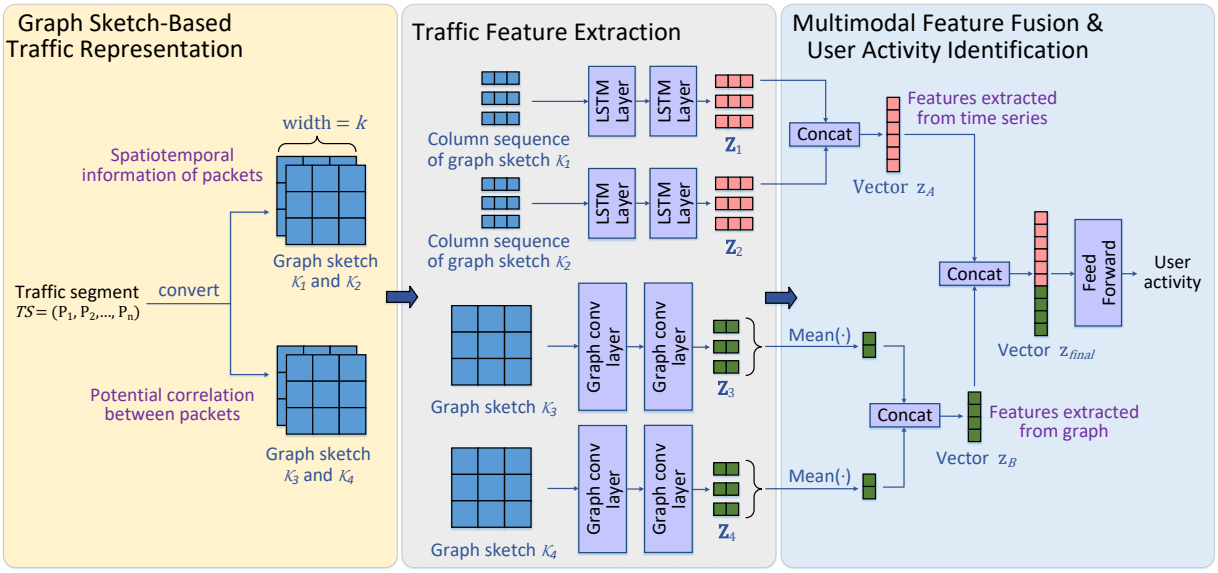


Fig. 1. The framework of ETC-GS mechanism.

A. Definitions

Graph sketch is a compact data structure. It was originally proposed for efficiently recording large-scale graph streams, and the definition of graph stream is introduced as follows.

Definition 1 (Graph stream): A graph stream (a.k.a. edge stream) is a consecutive sequence of items $S = \{e_1, e_2, \ldots, e_n\}$, where each item $e_i = (n_s, n_d, t, w)$ denotes a directed or undirected edge from node n_s to node n_d arriving at timestamp t with weight w. A graph stream can form a dynamic graph that changes with every arrival of an edge. Note that since an edge e_i may appear multiple times at different timestamps, the final weight of e_i is usually computed by the aggregation function $\mathrm{SUM}(\cdot)$.

Definition 2 (Graph sketch [15]): Assuming that $G = (V, E)$ is the graph formed by a given graph stream S, graph sketch $\mathcal{K}$ is defined as a compressed graph $\mathcal{K} = (V_\mathcal{K}, E_\mathcal{K})$ whose size is smaller than G (i.e., $|V_\mathcal{K}| \leq |V|$ and $|E_\mathcal{K}| \leq |E|$), where a random hash function $H(\cdot)$ is adopted to map each node in V to a node in $V_\mathcal{K}$. Correspondingly, each edge (n_s, n_d) in E is mapped to the edge $(H(n_s), H(n_d))$ in $E_\mathcal{K}$.

In practice, an adjacency matrix $\mathbf{M} \in \mathbb{R}^{k \times k}$ is leveraged to implement a graph sketch $\mathcal{K}$, where k is named the *width* of graph sketch. At the beginning, all the values stored in the matrix $\mathbf{M}$ are 0. To store a graph stream S in graph sketch $\mathcal{K}$, for each edge $e = (n_s, n_d, t, w)$ in S, it conducts an *edge update* operation which is explained as follows.

Edge update: The graph sketch $\mathcal{K}$ first calculates the hash values $(H(n_s), H(n_d))$ and then locates the corresponding position $\mathbf{M}[H(n_s)][H(n_d)]$ in the adjacency matrix. Finally, the value in the located position is added by weight w.

An encrypted traffic flow can be informally defined as a sequence of packets. Similar to [12], we extend the concept of traffic flows by introducing time-induced *traffic segments*.

Definition 3 (Traffic segment): A traffic segment refers to a consecutive subsequence of an encrypted traffic flow. Formally, a traffic segment TS can be defined as $TS = \{P_1, P_2, \cdots, P_m\}$, where $P_i = (l_i, t_i, d_i)$ denotes a single packet where l_i denotes the size of packet (the number of bytes); t_i denotes the timestamp of packet; and d_i denotes the direction of packet (sent or received). The *length* of a traffic segment TS is defined as $length(TS) = t_m - t_1$.

Since traffic is encrypted, we assume that only three kinds of packet information, i.e., packet size, packet timestamp and packet direction, are available for classification tasks.

B. Problem Formulation

The encrypted traffic classifcation task aims to recognize the various sources (e.g., applications, web pages or services) that generated the traffic based on the information of traffic packets. In this paper, we concentrate on *in-app user activity* classification for a given encrypted traffic flow TF. In other words, we aim to identify the user activity that generated TF. Typical in-app user activities include text chat, sending emails, etc. To achieve this goal, we need to solve the following two sub-problems: traffic segment representation and user activity identification.

- *Traffic segment representation:* Different traffic flows may have different lengths. In order to transform them to normalized representations, for each traffic flow, we can truncate α-second traffic from the middle of it, obtaining a set of traffic segments with the same length. With the obtained traffic segments, traffic segment representation aims to convert each of them to a normalized representation for further analysis.
- *User activity identification:* With the normalized representation of each traffic segment, user activity identification is to design a classifier which is able to map each

normalized traffic representation to a fine-grained in-app user activity such as text chat or file transfer.

IV. ETC-GS Mechanism

In this section, we detailly introduce our proposed ETC-GS mechanism. The details of the ETC-GS framework is shown in Fig. 1, and its workflow is presented as follows. As discussed in section III-B, since different traffic flows have different lengths, we should first unify their lengths before converting them to the normalized representation. Thus, for each traffic flow, we first truncate α-second traffic from the middle of it, obtaining a set of traffic segments with the same length. ETC-GS mechanism mainly consists of three stages. In the first stage, we convert a traffic segment to a set of graph sketches by considering two aspects, i.e., spatiotemporal information of packets and potential correlation between packets. In the second stage, we adopt LSTM models and GCN models to extract features from different graph sketches, respectively. In the final stage, we fuse the extracted multimodal features to a final vector which contains comprehensive traffic information. Based on it, we use a fully-connected layer to identify user activity. The three stages of ETC-GS mechanism, i.e., graph sketch-based traffic representation, traffic feature extraction, multimodal feature fusion and user activity identification, are introduced as follows.

A. Graph Sketch-Based Traffic Representation

As previously mentioned, we propose a graph sketch-based method to solve the problem of traffic segmentation representation. The details of our representation method is explained in this section.

1) Representation Based on Spatiotemporal Information of Packets: As presented in Definition 3, each packet in a traffic segment is a triplet $P_i = (l_i, t_i, d_i)$, where l_i denotes the size of packet (i.e., the number of bytes); t_i denotes the timestamp of packet; and d_i denotes the direction of packet (sent or received). To convert a traffic segment TS to a normalized representation, we employ two graph sketches $\mathcal{K}_1$ and $\mathcal{K}_2$ (implemented by matrix $\mathbf{M}_1$ and $\mathbf{M}_2$) to record all the packets in TS, and the widths of $\mathcal{K}_1$ and $\mathcal{K}_2$ are both k. For each packet $P_i = (l_i, t_i, d_i)$ in TS, we first inspect its direction. The sent packet will be recorded in $\mathcal{K}_1$, and the received packet will be recorded in $\mathcal{K}_2$. The process of recording a packet is similar to the *edge update* operation introduced in III-A. Different from the basic graph sketch which only adopts a random hash function for edge recording, we carefully design two hash functions for packet recording, and the designed hash functions are formulated as follows:

$$H_1(l_i) = \lceil \frac{l_i \times k}{\mathrm{MTU}} \rceil \tag{1}$$

$$H_2(t_i) = \lceil \frac{t_i \times k}{length(TS)} \rceil \tag{2}$$

where l_i and t_i denotes the size and the timestamp of packet P_i, respectively; k denotes the width of $\mathcal{K}_1$ (and $\mathcal{K}_2$); $length(TS)$ denotes the length of a traffic segment; and MTU denotes the maximum transmission unit, i.e., $\mathrm{MTU} = 1500$. With the above hash functions, to record a packet P_i, we can first locate P_i's corresponding position in the graph sketch, i.e., $\mathbf{M_1}[H_1(l_i)][H_2(t_i)]$ (or $\mathbf{M_2}[H_1(l_i)][H_2(t_i)]$), and then add the value in the position by 1. After processing all the packets, we can obtain two graph sketches which contain the spatiotemporal information of packets in the traffic segment.

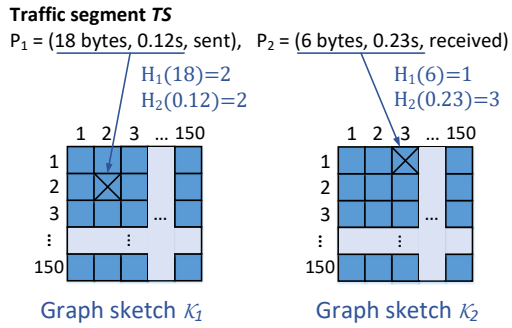


Fig. 2. Graph sketch-based traffic representation which captures spatiotemporal information of packets.

For ease of calculation, in practice, we set $k = 150$ and $length(TS) = 15$ (seconds). Thus, the hash functions in Equation 1 and Equation 2 can be simplified as $H_1(l_i) = \lceil \frac{l}{10} \rceil$ and $H_2(t_i) = \lceil t \times 10 \rceil$. To illustrate this representation method more clearly, we give a concrete example in Fig. 2. In this example, a traffic segment TS contains two packets $P_1 = (18\,bytes, 0.12s, sent)$ and $P_2 = (6\,bytes, 0.23s, received)$. To record packet p_1, we first inspect its direction. Since P_1 is a sent packet, we record it in graph sketch $\mathcal{K}_1$. Then, we locate its position by calculating two hash values. Since $H_1(18) = \lceil \frac{18 \times 150}{1500} \rceil = 2$ and $H_2(0.12) = \lceil \frac{0.12 \times 150}{15} \rceil = 2$, packet P_1 should be recorded in $\mathbf{M_1}[2][2]$ (marked by a symbol "$\times$" in Fig. 2), and the value stored in this position will be added by 1. Similarly, packet P_2 should be recorded in $\mathbf{M_2}[1][3]$ (marked by a symbol "$\times$" in Fig. 2) and the value stored in this position will also be added by 1.

Now we explain the advantages of such representation. First, considering the example in Fig. 2, it is easy to see that $\mathbf{M_1}[a][b]$ (or $\mathbf{M_2}[a][b]$) refers to the number of packets whose sizes range from $(10a - 9)$ to $10a$ and whose timestamps range from $(0.1a - 0.1)$ to $0.1a$. Since the second dimension of matrix $\mathbf{M_1}$ (or $\mathbf{M_2}$) represents timestamp, we can further split the matrix $\mathbf{M_1}$ (or $\mathbf{M_2}$) to a sequence of columns (vectors) $\{\mathbf{M_1}[:][1]; \mathbf{M_1}[:][2]; \cdots; \mathbf{M_1}[:][150]\}$ which is essentially a time series. Thus, the hidden features in this kind of traffic representation can be easily extracted by a RNN model. Second, compared with existing works [7] [11] [23] which also construct traffic representation on packet-level, we make full use of packet information by recording all the packets in the graph sketch and meanwhile keep the representation light-weight. In contract, these existing works only used a small number of packets in a traffic flow to construct the representation, which led to a serious loss of information.

2) Representation Based on Potential Correlation between Packets: Besides representing a traffic segment using the spatiotemporal information of packets, we also devise a graph-

based representation method to capture potential correlation between packets in traffic segment TS, aiming to mine more recognizable features. Different from existing approaches [13] that used a node to denote a single packet, we utilize a node to denote a set of packets whose sizes are in the same range. Edges between two nodes are created based on the order of packets in the traffic segment. Concretely, for each pair of adjacent packets P_i and P_{i+1}, we create a connection between the node that contains P_i and the node that contains P_{i+1}. To convert a traffic segment to the graph described above, we utilize a graph sketch $\mathcal{K}_3$ (implemented by matrix $\mathbf{M}_3$) whose width is k. For each pair of adjacent packets (P_i, P_{i+1}), we first create an edge $e = (l_i, l_{i+1})$ where l_i denotes the size of packet P_i and l_{i+1} denotes the size of packet P_{i+1}. Then, we conduct an edge update for edge e in graph sketch $\mathcal{K}_3$. Here, we still leverage function $H_1(\cdot)$ (see Equation 1) as the hash function associated with graph sketch $\mathcal{K}_3$. Specifically, to conduct the edge update with $e = (l_i, l_{i+1})$, we first locate the edge e's position in the graph sketch, i.e., $\mathbf{M_3}[H_1(l_i)][H_1(l_{i+1})]$, and then add the value in that position by 1. Similar to section IV-A1, after processing all the edges, we can obtain a graph sketch which captures the correlation between packets in traffic segment TS.

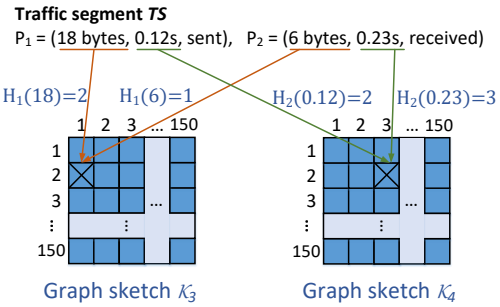


Fig. 3. Graph sketch-based traffic representation which captures potential correlation between packets.

To better explain this representation method, we also give a concrete example in Fig. 3. In this example, since packet P_1 and P_2 are adjacent, we will create an edge between the node that contains P_1 and the node that contains P_2. To accomplish this, we first find the above two nodes in graph sketch $\mathcal{K}_3$ by calculating two hash values. Since $H_1(18) = \lceil \frac{18\times150}{1500} \rceil = 2$ and $H_1(6) = \lceil \frac{6\times150}{1500} \rceil = 1$, a directed edge between node #2 and node #1 (marked by a symbol "$\times$" in Fig. 3) will be created in $\mathcal{K}_3$. In other words, the value in $\mathbf{M_3}[2][1]$ will be added by 1. Besides using packet sizes to capture the potential correlation between packets, we also use packet timestamps to capture another kind of correlation by constructing graph sketch $\mathcal{K}_4$ associated with hash function $H_2(\cdot)$ as shown in Fig. 3. Similarly, a directed edge between node #2 and node #3 will be created in $\mathcal{K}_4$ in this example.

Our designed graph-based traffic representation has the following advantage. In contrast to some existing methods which created a very dense graph whose topological structure lacked distinguishability, our method creates a sparse and distinguishable graph. For instance, the sizes of most packets generated by user activity *streaming* range from 1101 to 1300 (bytes). In our method, if we use one node n_i to represent the packets whose sizes range from 1101 to 1200, and use another node n_j to represent the packets whose sizes range from 1201 to 1300, there will be a self-loop with a large weight for node n_i, and another self-loop with a large weight for node n_j. Also, there will be an edge with a medium weight between n_i and n_j and few other edges in our constructed graph. Thus, the graph described above is quite recognizable, and its hidden features can be easily extracted by a GNN model.

B. Traffic Feature Extraction

To address the problem of user activity identification, we should first extract high-level features from the traffic representations obtained in section IV-A.

1) Spatiotemporal Feature Extraction: As mentioned in section IV-A1, since graph sketch $\mathcal{K}_1$ (or $\mathcal{K}_2$) records the spatiotemporal information of a traffic segment and it can be splitted into a time series, we can adopt a LSTM model to extract features from it as shown in Fig. 1. To be concrete, we first split graph sketch $\mathcal{K}_1$ into a vector sequence $\mathbf{Z}_0 = \{\mathbf{v}_1; \mathbf{v}_2; \cdots; \mathbf{v}_k\}$, where each vector $\mathbf{v}_i (i = 1, 2, \ldots, k)$ denotes the i-th column of matrix $\mathbf{M}_1$, i.e., $\mathbf{v}_i = \mathbf{M}_1[:][i]$. Then, we input vector sequence $\mathbf{Z}_0$ to a two-layer LSTM model for temporal feature extraction, obtaining another vector sequence $\mathbf{Z}_1 = \mathrm{LSTM}(\mathbf{Z}_0)$, where $\mathrm{LSTM}(\cdot)$ denotes a two-layer LSTM model. We also split graph sketch $\mathcal{K}_2$ to a vector sequence, and apply the similar feature extraction method, obtaining a hidden feature vector sequence $\mathbf{Z}_2$.

2) Packet Correlation Feature Extraction: Since graph sketch $\mathcal{K}_3$ stores the information of correlation between packets and graph sketch is essentially a compressed graph, we can utilize a GNN model to extract features from it. Specifically, we adopted a two-layer GCN [16] model to extract node-level features from $\mathcal{K}_3$, obtaining node embeddings $\mathbf{Z}_3 = \mathrm{GCN}(\mathbf{X}, \mathbf{M}_3)$, where $\mathbf{X}$ denotes a matrix of node feature vectors; $\mathbf{M}_3$ denotes the adjacency matrix of graph sketch $\mathcal{K}_3$; and $\mathbf{Z}_3 \in \mathbb{R}^{k\times d}$ (d is the dimension of node embeddings) denotes a matrix of node embeddings. We also apply the same feature extraction method to graph sketch $\mathcal{K}_4$, obtaining a matrix of node embeddings $\mathbf{Z}_4$.

C. Multimodal Feature Fusion and User Activity Identification

Before fusing the features extracted from different graph sketches, we should conduct some preprocessing as shown in Fig. 1. First, we only need the last vector in vector sequence $\mathbf{Z}_1$ and that in $\mathbf{Z}_2$ since these two vectors contain the global information of the original time series. Thus, we concatenate the last vector in $\mathbf{Z}_1$ and the last vector in $\mathbf{Z}_2$, obtaining a new feature vector $\mathbf{z}_A = \mathrm{Concat}(\mathbf{Z}_1[k], \mathbf{Z}_2[k])$, where $\mathrm{Concat}(\cdot)$ denotes the concatenation operation.

Second, since $\mathbf{Z}_3$ and $\mathbf{Z}_4$ are node-level features, we convert them to a graph-level feature vector by concatenating the elementwise mean of the vectors in $\{\mathbf{Z}_3[i], \forall i \in \{1, 2, \ldots, k\}\}$

and that of the vectors in $\{\mathbf{Z}_4[i], \forall i \in \{1, 2, \ldots, k\}\}$, obtaining $\mathbf{z}_B$:

$$\mathbf{z}_B = \text{Concat}(\text{Mean}(\{\mathbf{Z}_3[i], \forall i \in \{1, 2, \ldots, k\}\}), \text{Mean}(\{\mathbf{Z}_4[i], \forall i \in \{1, 2, \ldots, k\}\})) \quad (3)$$

where $\text{Mean}(\cdot)$ denotes calculating the element-wise mean.

Finally, we fuse feature vector $\mathbf{z}_A$ and $\mathbf{z}_B$ by concatenation:

$$\mathbf{z}_{final} = \text{Concat}(\mathbf{z}_A, \mathbf{z}_B) \quad (4)$$

Note that any other multimodal feature fusion methods [25] [26] such as average pooling with data augmentation can be used here. We simply adopt concatenation to avoid bringing extra computation cost.

After obtaining the final feature vector $\mathbf{z}_{final}$, we input it to a fully-connected layer to identify the corresponding user activity. Note that the whole framework presented in Fig. 1 can be trained through cross-entropy loss in an end-to-end manner.

V. Experiment

To validate the effectiveness of our proposed ETC-GS mechanism, we conduct extensive experiments on two public traffic datasets: ISCX-nonVPN and ISCX-VPN [27]. We compare ETC-GS with three state-of-the-art methods: TSCRNN [8], TFE-GNN [12], and LEXNet [11]. All experiments are performed on a laptop with Intel Core i9-14900HX processors (24 cores, 32 threads), 32 GB of memory, and NVIDIA GeForce RTX 4080 Laptop GPU. ETC-GS and TSCRNN are implemented using PyTorch library and PyG library. For LEXNet and TFE-GNN, we use the source code[1][2] provided on Github.

TABLE I
IN-APP USER ACTIVITIES IN DATASET ISCX-NONVPN.

User activity (label)	Application
Chat	AIM, Facebook, Gmail, Hangouts, ICQ
Email	Email (using SMPTS, POP3S and IMAPS)
File transfer	Skype, Filezilla (using FTP and SFTP)
Streaming	Netflix, Spotify, Vimeo, Youtube
Video call	Facebook, Hangouts, Skype, Voipbuster
Audio	Facebook, Hangouts, Skype

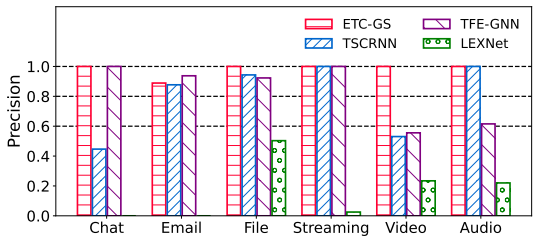
A. Dataset

We conduct our experiments on two public dataset: ISCX-nonVPN and ISCX-VPN [27]. The encrypted traffic contains a total amount of 25 GB of data. The ISCX-VPN dataset is collected over virtual private networks (VPNs), while traffic in ISCX-nonVPN dataset is regular and not collected over VPNs. There are totally 6 classes (in-app user activities) in dataset ISCX-nonVPN which are shown in table I. Note that compared with dataset ISCX-nonVPN, dataset ISCX-VPN contains an extra user activity, P2P using BitTorrent. Both datasets are randomly splitted into two subsets: 80% of samples are denoted as training set, while 20% of samples as test set.

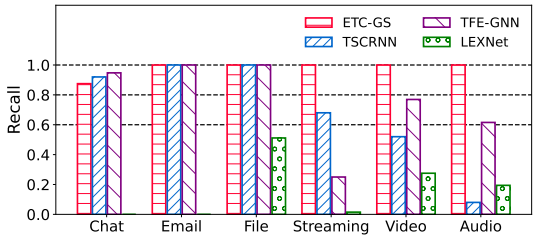
[1]https://github.com/ViktorAxelsen/TFE-GNN
[2]https://github.com/XAIseries/LEXNet

B. Numerical Results

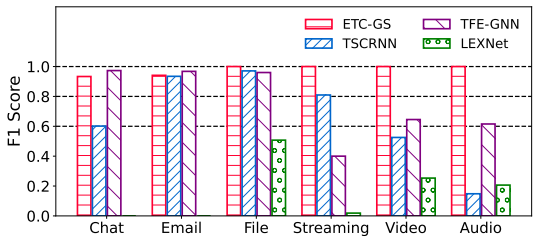
We adopt four evaluation metrics to evaluate the classification performance of our method as well as the baselines. The metrics include precision, recall, F1 score, and overall accuracy. The performance comparison in terms of precision, recall and F1 score are shown in Fig. 4 and Fig. 5, and the comprison of overall accuracy is presented in table II.



(a) Precision



(b) Recall



(c) F1 score

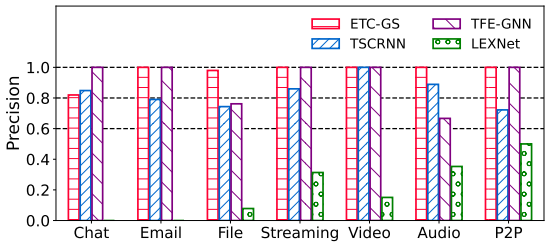
Fig. 4. Classificaiton results on dataset ISCX-nonVPN.

As shown in Fig. 4, on dataset ISCX-nonVPN, LEXNet has the worst performance. Concretely, LEXNet is unable to recognize chat and email, and it only achieves a F1 score of 0.0187 when identifying streaming. This is probably because LEXNet only uses 20 packets to represent a traffic flow, and thus does not have enough packet information for classification. Among all the baselines, TFE-GNN achieves the best performance. Specifically, TFE-GNN is good at recognizing chat, email, and file transfer. However, when identifying streaming, video call, and audio, TFE-GNN does not perform well. For instance, it only achieves a F1 score of 0.40 when identifying streaming. In contrast, our ETC-GS is able to accurately identify all kinds of user activity. For example, it respectively achieves a F1 score of 0.9412 and 1.0 when recognizing email and file transfer.

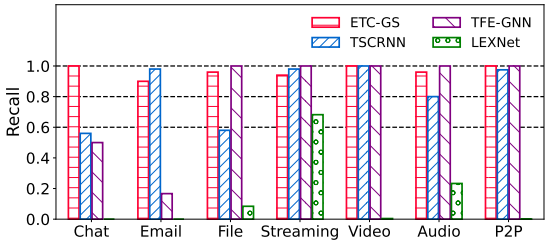
On dataset ISCX-VPN, our ETC-GS also outperforms all the baseline methods as presented in Fig. 5. As can be seen, LEXNet also has the worst performance on this dataset. TSCRNN is only good at identifying streaming and video

call. TFE-GNN again has the best performance among all the baselines. However, it is bad at identifying chat and email. To be specific, it only achieves a F1 score of 0.6667 and 0.2857 when recognizing chat and email, respectively. Comparably, our ETC-GS is able to perform well when identifying all kinds of user activity, and it achieves a F1 score of 0.9009 when recognizing chat, and a F1 score of 0.9474 when recognizing email.

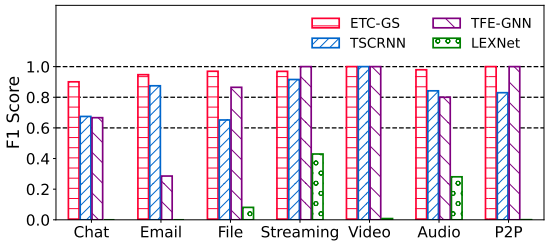
In summary, ETC-GS outperforms all the baselines in terms of overall accuracy as shown in table II. To be concrete, ETC-GS achieves an overall accuracy of 0.9657 and 0.9792 on dataset ISCX-nonVPN and ISCX-VPN, respectively.



(a) Precision



(b) Recall



(c) F1 score

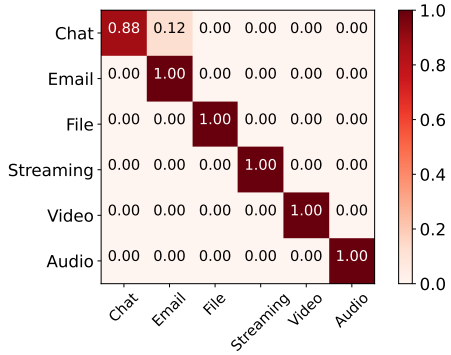
Fig. 5. Classificaiton results on dataset ISCX-VPN.

TABLE II
PERFORMANCE COMPARISON IN TERMS OF OVERALL ACCURACY.

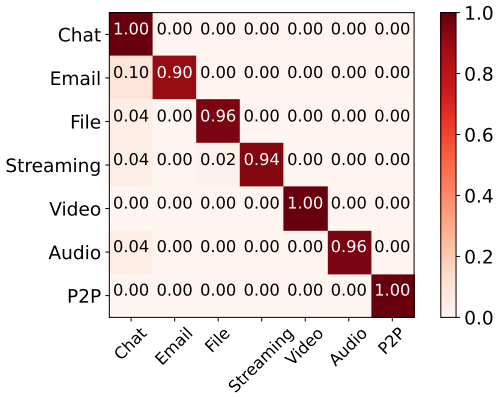
Method	Dataset	
	ISCX-nonVPN	ISCX-VPN
TSCRNN	0.7000	0.8353
TFE-GNN	0.8125	0.8701
LEXNet	0.3643	0.3022
ETC-GS (ours)	**0.9792**	**0.9657**

C. Confusion Matrix

As illustrated in Fig. 6, we draw the confusion matrices based on the classification results achieved by our ETC-GS. As shown in Fig. 6(a), our method can very accurately recognize all kinds of user activities except chat on dataset ISCX-nonVPN. Specifically, 12% of traffic flows generated by chat are falsely identified as the traffic flows generated by email. In addition, on dataset ISCX-VPN, a small amount of traffic flows generated by several activities such as file transfer are also falsely classified as the traffic flows generated by chat as shown in Fig. 6(b). A possible explanation is that in the traffic flows generated by chat, *time intervals between packets* are highly related to users (e.g. typing speed), which brings too much randomness and makes chat difficult to identify.



(a) Dataset ISCX-nonVPN



(b) Dataset ISCX-VPN

Fig. 6. Confusion matrices based on the classification results by ETC-GS.

D. Ablation Study

To further evaluate the effectiveness of the key components of ETC-GS, we perform ablation study on the same datasets, and the evaluation results are presented in table III and table IV. Note that abbreviation SI represents spatiotemporal information of packets, and abbreviation PC represents potential correlation between packets in both tables. Also note that precision, recall, and F1 score respectively represent macro-precision, macro-recall, and macro-F1 score. As can be seen, when sololy using spatiotemporal information of packets (or potential correlation between packets), our method can still maintain high performance. In addition, fusing both kinds of packet information indeed improves the classification accuracy. For example, as shown in III, ECT-GS (default) achieves an accuracy of 0.9792 which is higher than

the accuracy achieved by using only one kind of packet information to conduct classification.

TABLE III
ABLATION STUDY OF KEY COMPONENTS IN ECT-GS ON DATASET ISCX-NONVPN.

Method	Precision	Recall	F1 score	Accuracy
ETC-GS w/o SI	0.9451	0.9375	0.9362	0.9375
ETC-GS w/o PC	0.9214	0.9208	0.9198	0.9208
ETC-GS (default)	**0.9815**	**0.9792**	**0.9791**	**0.9792**

TABLE IV
ABLATION STUDY OF KEY COMPONENTS IN ECT-GS ON DATASET ISCX-VPN.

Method	Precision	Recall	F1 score	Accuracy
ETC-GS w/o SI	0.9112	0.8886	0.8908	0.8886
ETC-GS w/o PC	0.9306	0.9086	0.9075	0.9086
ETC-GS (default)	**0.9713**	**0.9657**	**0.9667**	**0.9657**

VI. CONCLUSION

Existing methods for encrypted traffic classification either constructed a heavy-weight representation for raw traffic or did not utilize adequate packet information when constructing the traffic representation. To this end, in this paper, we develop a novel mechanism named ETC-GS for indentifying in-app user activities in encrypted traffic. ETC-GS first converts a raw traffic flow to several light-weight graph sketches by capturing spatiotemporal information of packets and potential correlation between packets. Next, it uses LSTM models and GCN models to respectively extract high-level features from different graph sketches, respectively. Finally, ETC-GS fuses the extracted features to a final feature vector, and input it to a fully-connected layer to recognize the user activity that generates the traffic. Extensive experiments on two public datasets, ISCX-nonVPN and ISCX-VPN, verify the effectiveness of ETC-GS in terms of classification accuracy.

REFERENCES

[1] E. Papadogiannaki and S. Ioannidis, "A survey on encrypted network traffic analysis applications, techniques, and countermeasures," *ACM Computing Surveys*, vol. 54, no. 6, pp. 123:1–123:35, 2022.

[2] Q. Zhang, Y. Ma, J. Wang, and X. Li, "UDP traffic classification using most distinguished port," in *APNOMS'14: The 16th Asia-Pacific Network Operations and Management Symposium*, 2014, pp. 1–4.

[3] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "Blindbox: Deep packet inspection over encrypted traffic," in *SIGCOMM'15: The 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 213–226.

[4] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic," in *EuroS&P'16: 2016 IEEE European Symposium on Security and Privacy*, 2016, pp. 439–454.

[5] T. van Ede, R. Bortolameotti, A. Continella, J. Ren, D. J. Dubois, M. Lindorfer, D. R. Choffnes, M. van Steen, and A. Peter, "Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic," in *NDSS'20: The 27th Annual Network and Distributed System Security Symposium*, 2020.

[6] J. Liu, Y. Fu, J. Ming, Y. Ren, L. Sun, and H. Xiong, "Effective and real-time in-app activity analysis in encrypted internet traffic streams," in *KDD'17: The 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 335–344.

[7] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "Fs-net: A flow sequence network for encrypted traffic classification," in *INFOCOM'19: 2019 IEEE Conference on Computer Communications*, 2019, pp. 1171–1179.

[8] K. Lin, X. Xu, and H. Gao, "TSCRNN: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of iiot," *Computer Networks*, vol. 190, p. 107974, 2021.

[9] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Yu, "ET-BERT: A contextualized datagram representation with pre-training transformers for encrypted traffic classification," in *WWW'22: The ACM Web Conference 2022*, 2022, pp. 633–642.

[10] R. Zhao, M. Zhan, X. Deng, Y. Wang, Y. Wang, G. Gui, and Z. Xue, "Yet another traffic classifier: A masked autoencoder based traffic transformer with multi-level flow representation," in *AAAI'23: Thirty-Seventh AAAI Conference on Artificial Intelligence*, 2023, pp. 5420–5427.

[11] K. Fauvel, F. Chen, and D. Rossi, "A lightweight, efficient and explainable-by-design convolutional neural network for internet traffic classification," in *KDD'23: The 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 4013–4023.

[12] H. Zhang, L. Yu, X. Xiao, Q. Li, F. Mercaldo, X. Luo, and Q. Liu, "TFE-GNN: A temporal fusion encoder using graph neural networks for fine-grained encrypted traffic classification," in *WWW'23: The ACM Web Conference 2023*, 2023, pp. 2066–2075.

[13] T. Huoh, Y. Luo, and T. Zhang, "Encrypted network traffic classification using a geometric learning model," in *IM'21: The 17th IFIP/IEEE International Symposium on Integrated Network Management*, 2021, pp. 376–383.

[14] G. Wu, X. Wang, and J. Zhang, "Peerg: A P2P botnet detection method based on representation learning and graph contrastive learning," *Computers & Security*, vol. 140, p. 103775, 2024. [Online]. Available: https://doi.org/10.1016/j.cose.2024.103775

[15] N. Tang, Q. Chen, and P. Mitra, "Graph stream summarization: From big bang to big crunch," in *SIGMOD'16: The 2016 International Conference on Management of Data*, 2016, pp. 1481–1496.

[16] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR'17: The 5th International Conference on Learning Representations*, 2017.

[17] H. Yao, G. Ranjan, A. Tongaonkar, Y. Liao, and Z. M. Mao, "SAMPLES: self adaptive mining of persistent lexical snippets for classifying mobile application traffic," in *MobiCom'15: The 21st Annual International Conference on Mobile Computing and Networking*, 2015, pp. 439–451.

[18] G. Ranjan, A. Tongaonkar, and R. Torres, "Approximate matching of persistent lexicon using search-engines for classifying mobile app traffic," in *INFOCOM'16: The 35th Annual IEEE International Conference on Computer Communications*, 2016, pp. 1–9.

[19] X. Xiao, R. Li, H. Zheng, R. Ye, A. K. Sangaiah, and S. Xia, "Novel dynamic multiple classification system for network traffic," *Information Sciences*, vol. 479, pp. 526–541, 2019.

[20] W. Zheng, C. Gou, L. Yan, and S. Mo, "Learning to classify: A flow-based relation network for encrypted traffic classification," in *WWW'20: The Web Conference 2020*, 2020, pp. 13–22.

[21] M. Shen, J. Zhang, L. Zhu, K. Xu, and X. Du, "Accurate decentralized application identification via encrypted traffic analysis using graph neural networks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2367–2380, 2021.

[22] J. Luxemburk and T. Cejka, "Fine-grained TLS services classification with reject option," *Computer Networks*, vol. 220, p. 109467, 2023.

[23] X. Yun, Y. Wang, Y. Zhang, C. Zhao, and Z. Zhao, "Encrypted TLS traffic classification on cloud platforms," *IEEE/ACM Transactions on Networking*, vol. 31, no. 1, pp. 164–177, 2023.

[24] Y. Chen and Y. Wang, "MPAF: encrypted traffic classification with multi-phase attribute fingerprint," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 7091–7105, 2024.

[25] X. Deng, Y. Wang, and Z. Xue, "AN-Net: an anti-noise network for anonymous traffic classification," in *WWW'24: The ACM on Web Conference 2024*, 2024, pp. 4417–4428.

[26] G. Wu, X. Wang, Q. Lu, and H. Zhang, "Bot-dm: A dual-modal botnet detection method based on the combination of implicit semantic expression and graphical expression," *Expert Systems Applications*, vol. 248, p. 123384, 2024.

[27] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *ICISSP'16: The 2nd International Conference on Information Systems Security and Privacy*, 2016, pp. 407–414.