FG-SAT: Efficient Flow Graph for Encrypted Traffic Classification under Environment Shifts

Susu Cui, Xueying Han, Dongqi Han, Zhiliang Wang, *Member, IEEE,*
Weihang Wang, Yun Li, Bo Jiang, Baoxu Liu and Zhigang Lu

arXiv:2408.14122v1 [cs.CR] 26 Aug 2024

Abstract—Encrypted traffic classification plays a critical role in network security and management. Currently, mining deep patterns from side-channel contents and plaintext fields through neural networks is a major solution. However, existing methods have two major limitations: (1) They fail to recognize the critical link between transport layer mechanisms and applications, missing the opportunity to learn internal structure features for accurate traffic classification. (2) They assume network traffic in an unrealistically stable and singular environment, making it difficult to effectively classify real-world traffic under environment shifts. In this paper, we propose FG-SAT, the first end-to-end method for encrypted traffic analysis under environment shifts. We propose a key abstraction, the *Flow Graph*, to represent flow internal relationship structures and rich node attributes, which enables robust and generalized representation. Additionally, to address the problem of inconsistent data distribution under environment shifts, we introduce a novel feature selection algorithm based on Jensen-Shannon divergence (JSD) to select robust node attributes. Finally, we design a classifier, GraphSAT, which integrates Graph-SAGE and GAT to deeply learn Flow Graph features, enabling accurate encrypted traffic identification. FG-SAT exhibits both efficient and robust classification performance under environment shifts and outperforms state-of-the-art methods in encrypted attack detection and application classification.

I. INTRODUCTION

Traffic encryption technology plays a significant role in enhancing network security and privacy. By employing encryption protocols, users can have greater confidence in the sensitive transactions and communications they engage in. To ensure data security, numerous internet services and applications employ various encryption techniques to encrypt transmitted content. However, traffic encryption technology also poses challenges for security managers and internet service providers (ISPs). Firstly, encryption does not guarantee the security of the content itself. Malware can be encrypted and transmitted as easily as legitimate files. In fact, over 80% of malware spreads through TLS protocol [1]. To bypass firewall and security software detection, many malicious network attacks also employ encryption technology to conceal communication content [2]. Therefore, security managers need to identify encrypted traffic in order to inspect malicious encrypted traffic. Secondly, due to the widespread use of smart devices and the emergence of various new application services, network traffic is showing a geometric growth trend [3]. While ensuring reliable data transmission, ISPs need to identify encrypted traffic in order to provide personalized services to users [4], thereby improving network efficiency and enhancing the quality of user services.

As the traffic payload is encrypted, traditional classification methods are no longer effective for encrypted traffic. With the development of machine learning and neural networks, a considerable number of studies show that analyzing the statistical, byte and sequential features of encrypted traffic can also achieve effective classification. According to the feature analysis dimension, we summarize encrypted traffic classification into three methods: (1) Statistics-based methods [5]–[13] extract the side-channel statistical features and header fields to construct machine learning classifiers for traffic classification. (2) Byte-based methods [14]–[18] utilize the raw bytes of encrypted traffic and transform the classification task into an image classification task, using neural networks such as convolutional neural network (CNN) or capsule neural network (CapsNet) to learn the spatial features of raw bytes in traffic. (3) Sequence-based methods [19]–[22] treat the traffic as the sequence of packets and extract the packet length and arrival time as the sequential features, and use the recurrent neural network (RNN) or Encoder-Decoder for classification.

Unfortunately, despite the progress made, there are still two primary limitations in existing works as follows:

Overlook the critical link between transport layer mechanisms and applications. Transport layer features are intrinsically linked to applications. To communicate more efficiently, the TCP protocol uses a sliding window and acknowledgment mechanism to send multiple packets at the same time and a single acknowledgment number to confirm receipt of multiple packets. As a result, larger sliding windows are common for streaming applications to complete data transfer quickly, but instant messaging applications typically use a balanced exchange of received and acknowledgment packets. Therefore, encoding structural relationships between packets can be beneficial for encrypted traffic classification. However, existing

Susu Cui, Xueying Han, Yun Liu, Bo Jiang, Baoxu Liu and Zhigang Lu are with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, and also with the School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China (e-mail: cuisusu@iie.ac.cn, hanxueying@iie.ac.cn, liyun1996@iie.ac.cn, jiangbo@iie.ac.cn, liubaoxu@iie.ac.cn, luzhigang@iie.ac.cn).

Dongqi Han, Zhiliang Wang are with the Institute for Network Sciences and Cyberspace, BNRist, Tsinghua University, Beijing, China (e-mail: handq19@mails.tsinghua.edu.cn, wzl@cernet.edu.cn).

Weihang Wang is with the University of Southern California, Los Angeles, CA, USA (e-mail: weihangw@usc.edu).
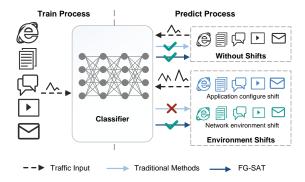
Fig. 1. The comparison with the traditional encrypted traffic classification methods. Traditional methods can only identify traffic without shifts that is consistent with the distribution of the training set. In contrast, FG-SAT can identify traffic that does not match the distribution of the training set under environment shifts.

works either do not consider the structural relationships between packets [23], [24] or are limited to specific classification scenarios and deployment locations [25]–[27], missing the opportunity to learn transport layer features for accurate and generic classification of the encrypted traffic.

Unrealistic stable and singular network environment. Current methods mainly focus on traffic classification in a stable and singular network environment, which is unrealistic in representing real-world traffic. As shown in Figure 1, network traffic is susceptible to environmental influences and undergoes long-term dynamic change [24], [28]–[30], which we call *environment shifts* in this paper. We define environment shifts as changes in application configuration and/or network environment that cause changes in statistics and feature distribution of traffic within the same class. The application configuration shift includes, but is not limited to, changes in user behaviors (e.g., actions when using a particular application), and the emergence of new software. The network environments shift includes, but is not limited to, changes in protocols and network quality, such as bandwidth. Consider "browsing" applications as an example, their environment shifts can include new browser software, different browsing contents, and a change in network bandwidth, compared with the training data environment. If not handled properly, these changes can cause shifts in the distribution of traffic features, leading to poor classification performance.

To address the abovementioned limitations, we investigate encrypted traffic classification methods with the following challenging goals:

1) It is critical to design a flow representation that discovers the intrinsic link between transport layer mechanisms and applications, and learn the relevant features of packets within a flow tied to application.
2) The flow representation should be i) general enough to represent diverse traffic types, ii) adaptive to feature selection in environment shifts, and iii) independent of encryption protocols.
3) Since collecting traffic from all possible network en-

vironments is impractical, it is necessary to design robust algorithms that can select stable features when the environment shifts to maintain high classification performance.

In this paper, we propose FG-SAT, the first end-to-end method for encrypted traffic classification in environment shifts. We define a key abstraction, the *Flow Graph*, to characterize the internal relationship structure based on the transport layer mechanisms inside a flow. The Flow Graph provides several key promises: (1) It treats packets as nodes and identifies structural relationships between packets using edges of multiple types that characterize the window and acknowledgment mechanisms. (2) It features a general representation that can represent diverse traffic types, include rich features, and adapt to environment shifts. (3) It also augments node attributes with header fields extracted from the 2-4 layers that are independent of encryption protocols.

To achieve high classification performance in environment shifts, we propose a robust feature selection algorithm to solve the problem of inconsistent data distribution. Specifically, our feature selection algorithm evaluates packet header fields and automatically selects those stable fields as node attributes in the face of environment shifts. We use Jensen-Shannon divergence (JSD), a method for measuring similarity between two probability distributions, to assess the stability of features. Specifically, we compare the distribution differences between inter-class (with environment shifts) and extra-class (with varying traffic types) using JSD. When the extra-class JSD is greater than the inter-class JSD, we consider the features to be stable under environment shifts. Finally, we build a classifier based on graph neural networks (GNN), named GraphSAT, to identify encrypted traffic types. GraphSAT combines GraphSAGE [31] and GAT [32] to deeply learn the structural relationships and rich node attributes of the Flow Graph, enabling efficient encrypted traffic classification.

Our contributions are summarized as follows:

- We define a key abstraction, the *Flow Graph*, to represent encrypted flows, which features a general representation for diverse traffic, rich features and encryption protocol independence.
- We propose a feature selection algorithm, which measures the distribution differences of features by evaluating header fields and selecting stable fields in environment shifts.
- We design an encrypted traffic classifier based on GNN, which accurately learns the internal structure and node attributes of the Flow Graph from the raw traffic.
- We conduct experiments using publicly available datasets for attack traffic detection and our own collected dataset for application classification. Our method outperforms state-of-the-art methods, increasing accuracy by 6.85% over pre-training methods and by 15.84% over traditional deep learning methods.
- We evaluate the effects of environment shifts on encrypted traffic classification. The results show that as the environment shifts, all other methods' accuracy decreases, whereas our JSD-based feature selection algorithm increases accuracy by 7.44%.

II. RELATED WORK

In this section, we summarize the related work of encrypted traffic classification and graph-based methods on traffic analysis. Moreover, we compare the classification methods with proposed FG-SAT.

A. Methods on Encrypted Traffic Classification

1) Statistics-based Methods: Statistics-based methods utilize features like flow duration and packet count for encrypted traffic classification, applying machine learning to distinguish between traffic types. Draper-Gil et al. [5] use time-related features and the C4.5 algorithm for classifying 12 service types, including VPN traffic. Other works propose features like packet count, size, peak, and time for encrypted web classification [6], [7], and enrich analysis with unencrypted field data for identifying malicious applications [8], [9] and classifying applications through TLS handshake and certificate information [10]–[12]. Ede et al. [13] highlight the use of temporal correlations among destination-related features for generating application fingerprints.

Statistics-based methods face limitations including high time latency from processing complete flows, strong feature dependency limiting cross-protocol classification, and a lack of structural feature consideration, reducing accuracy under environment shifts.

2) Byte-based Methods: Byte-based methods input encrypted traffic's raw bytes into neural networks to classify traffic without manual feature extraction, relying on preprocessing and deep learning to uncover traffic byte features. Wang et al. [14], [15] utilize the first 784 bytes and CNNs for feature learning in malware and encrypted application classification. Cui et al. [16] enhance feature learning with CapsNet, surpassing CNNs in spatial and byte feature analysis. Han et al. [17] use Transformers on n-gram frequency vectors for flow feature learning. Lin et al. [18] introduce ET-BERT, a pre-training model for generic traffic representation learning in a few-shot context.

Byte-based methods simplify feature extraction but face challenges in cross-protocol classification due to plaintext fields in application protocols. These methods may not fully utilize byte information, including timestamps that risk overfitting and hamper generalization across environment shifts. Additionally, while focusing on spatial and temporal traffic byte features, they neglect structural aspects.

3) Sequence-based Methods: Sequence-based methods utilize packet length and time interval sequences, applying models like LSTM and Encoder-Decoder to understand sequence relationships. Ramezani et al. [19] use the server name from Client Hello packets for fingerprints. Shapira et al. [20] develop FlowPic, an image from packet size and arrival times, analyzed with CNN. Liu et al. [21] introduce FS-Net, leveraging LSTM-based Encoder-Decoder to explore packet length sequences for classification. However, extracting sequence information from complete flows usually results in high time latency.

Sequence-based methods analyze packet relationships in a flow but face challenges with environment shifts like network congestion, bandwidth changes, and application updates, affecting sequence consistency and accuracy across environments. Additionally, they sort by packet arrival time, missing varied packet structure representations.

B. Graph-based Methods on Traffic Analysis

The graph construction methods are closely related to the specific analysis task. In the field of intrusion detection, IPs or domain names are usually used as nodes, and edges are designed based on the communication between nodes to construct a network interaction graph for anomaly detection. However, these methods are not applicable to the general encrypted traffic classification task addressed in this paper. For example, Fu et al. [25] implement unsupervised anomaly detection based on network interaction graphs, identifying abnormal edges through a clustering loss function, but it is not suitable for multi-classification tasks. Fu et al. [26] construct a heterogeneous graph, characterizing the communication behavior of hosts and servers through their temporal and spatial features, and then identifying whether the host is infected. However, it has limitations for application classification since a single host can generate various types of application traffic. In application classification, graphs represent communication relationships to classify traffic. Pham et al. [27] use IPs and ports as nodes, turning mobile app traffic into a graph for fingerprinting, but this method assumes uniform traffic labels, limiting it to endpoint classification and not suitable for gateway traffic analysis. Shen et al. [22] offer a detailed approach, creating flow-specific graphs with packet length and direction, using a GNN to learn flow structures. However, the simplicity of its node features limits effectiveness in complex networks.

In the graph-based traffic analysis, IPs are mainly used as nodes to construct network interaction graphs for analysis. However, these methods has many limitations in general traffic classification tasks, such as classification scenarios and deployment locations. In addition, although Shen et al. [22] propose using graphs to represent relationships between packets, it only constructed one burst relationship, and the node features are simplistic, still lacking effective structural relationships and node feature representations.

III. PROBLEM DESCRIPTION AND CORE IDEA

In this section, we present the problem description of encrypted traffic classification, and then we introduce the framework and the core idea of FG-SAT.

A. Problem Description

In this paper, we classify encrypted traffic into different types, focusing on scenarios like application and attack type classification. Encrypted traffic consists of packets between clients and servers, encrypted using protocols like TLS, SSH, and Tor, masking application layer content but leaving some header information visible. In this paper, encrypted traffic is composed of multiple flows, where packets of the same flow have the same five-tuple information (source IP, destination IP, source port, destination port, transport layer protocol). We consider flow as the direct object of encrypted traffic
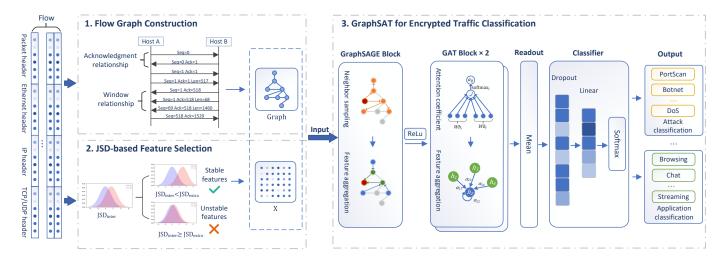
Fig. 2. The framework of FG-SAT, including 1. Flow Graph construction, 2. JSD-based feature selection, 3. GraphSAT for encrypted traffic classification.

classification, and model each flow as a graph for classification using a GNN-based approach, focusing on the structure rather than content. Moreover, we focus on the traffic classification at the firewall and local ISP network nodes, which are typically deployed in enterprise networks. Additionally, our goal is to design an end-to-end system that captures simple traffic information at the edge of the enterprise network for traffic classification. It is worth noting that we only achieve classification through the side-channel information of encrypted traffic without decrypting the traffic. Ultimately, the system can be utilized to optimize resource allocation and perform security monitoring of the enterprise network through encrypted traffic classification.

B. Core Idea

We introduce FG-SAT to achieve efficient encrypted traffic classification, as shown in Figure 2. Firstly, we define a key abstraction, the *Flow Graph*, that can represents multiple relationships between packets within the same flow and contains rich node attributes. It comprehensively mines the statistical, sequential, and structural features of encrypted traffic. Secondly, we propose a novel JSD-based feature selection algorithm for environment shifts. It can evaluate and select a stable set of features in dynamic and variable traffic environments. Finally, we establish a GNN-based encrypted traffic classifier, named GraphSAT, which fully explores the internal relationships between different nodes within the Flow Graph and rich node attributes, achieving efficient encrypted traffic classification.

During Flow Graph construction, each flow is viewed as a graph, capturing its statistical, sequential, and structural features for a comprehensive feature representation. Packets serve as nodes, with relationships defined by transport layer mechanisms and node attributes derived from header fields like packet length, arrival time, direction, TTL, and window size. These attributes are essential for encrypted traffic classification [33]. The transport layer's window size, vital for

classification, indicates the communication type, with the TCP protocol's sliding window mechanism adjusting for efficient, reliable transmission tailored to the application's needs, such as larger windows for streaming traffic to ensure high throughput. To depict the flow's internal structure, we establish two types of relationships:

- **Window relationship:** The client or server sends multiple packets continuously, and these packets are connected in sequence based on their arrival time to form the window relationship.
- **Acknowledgment relationship:** The client or server acknowledges the received packets so that the packet and its corresponding acknowledgment packet form the acknowledgment relationship.

Due to the environment shifts, including changes in application configure and network environment, network traffic is in a state of constant change, causing traditional encrypted traffic classification methods to lose accuracy [29], [30]. To address this, we introduce a JSD-based feature selection algorithm that assesses feature stability across different environments by measuring inter-class and extra-class distribution differences with JSD. This algorithm helps select stable distribution features as final node attributes amidst environment shifts.

We also develop GraphSAT, a GNN-based classifier, combining GraphSAGE and GAT to analyze Flow Graph's structure and node attributes. GraphSAGE samples neighbors to enhance generalization and reduce memory use, while GAT assesses neighbor importance. GraphSAT effectively and accurately classifies encrypted traffic types.

IV. CONSTRUCTION AND ANALYSIS OF FLOW GRAPH

In this section, we provide a detailed description of the process of Flow Graph construction. Additionally, we illustrate the differences between Flow Graphs among various types of traffic through analysis of specific examples.
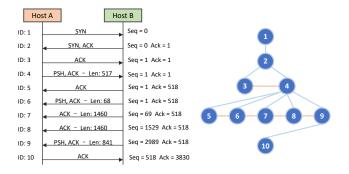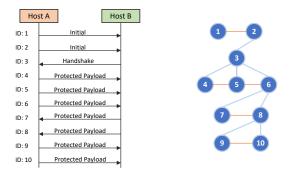
Fig. 3. An example of Flow Graph on TCP flow.

Fig. 4. An example of Flow Graph on UDP flow.

A. Flow Graph Construction

To achieve an end-to-end encrypted traffic classification, we convert the raw traffic into the Flow Graph. Firstly, we aggregate traffic based on flow granularity. In long-term services such as file transfers or malicious C&C communication, the duration of a flow can be as long as several hours. In order to perform efficient traffic classification, we only extract the first n packets of a flow to establish a Flow Graph, where n is considered as a hyperparameter of our method.

After traffic aggregation, we construct the Flow Graph consisting of no more than n packets. In this paper, a Flow Graph is defined as $G(V, E)$ consisting of n nodes and m edges. $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, with each node representing a packet within the flow. The node attributes of the node v, denoted as x_v, is a vector representation of the header fields, including packet header, ethernet header, IP header and TCP/UDP header. However, some fields in the packet header may not be effective in environment shifts. Therefore, we use the JSD-based feature selection algorithm to evaluate all the fields and select the stable fields as the final node attributes. More details on the JSD-based feature selection algorithm are presented in Section V.

$E = \{e_1, e_2, \ldots, e_m\}$ is the set of edges, where each $e \in E$ represents the relationship between packets, including window relationship and acknowledgment relationship, defined

as follows:

- **Window relationship:** When multiple packets are continuously sent from the sender to the receiver, they are connected in sequence according to the arrival time and form a window relationship. All packets in a single window are contiguous and in the same direction. In addition, in TCP packets, they also have the same ACK number.

- **Acknowledgment relationship:** When the receiver confirms and responds to the received packets, the packets and their corresponding acknowledgment packet form a acknowledgment relationship. They are in opposite directions. In TCP packets, the acknowledgment relationship can be associated based on the SEQ and ACK number. UDP packets consider adjacent packets, which are sent in opposite directions, as having an acknowledgment relationship.

We analyze TCP and UDP flows to understand their transmission behaviors and Flow Graph construction. For TCP flow, depicted in Figure 3, The packets inside the flow are the nodes of the graph, each represented by header field patterns. Packets in the same window share direction, continuity, and ACK number, forming window relationships based on arrival times. Acknowledgment relationships are built when a receiver's ACK number matches the sender's SEQ number plus TCP packet length. Hence, nodes within the same window, such as nodes 5-9, share both window and acknowledgment relationships, typically with a common node.

For UDP flow, shown in Figure 4, the absence of SEQ and ACK fields means reliability mechanisms of TCP are not present. Here, packets that share direction and continuity are considered in the same window, and adjacent packets in opposite directions have acknowledgment relationships. As with TCP, nodes in window relationships share acknowledgment relationships with a specific node, simplifying the construction of UDP Flow Graphs.

B. Analysis on Flow Graph

In this section, we describe and compare the Flow Graphs of different encrypted traffic types. We focus on the overall graph structure rather than node attributes. Based on the definition of Flow Graph mentioned above, we construct Flow Graphs for four types of encrypted traffic, Bot, DoS, Email, and Streaming, as shown in Figure 5. It can be seen that different types of encrypted traffic have different graph structures. Bot traffic is primarily controlled by streamlined commands, so it has a small window size and frequent acknowledgment. DoS sends packets with large window size at the beginning of the TCP connection. Emails generally transmit plaintext content, thus, the upload and download windows are similar in size. Streaming servers typically send large window size packets to clients because they download data more than they upload. Thus, different types of traffic exhibit different internal transmission structures based on the content of their upper-layer applications. Flow Graphs can clearly represent the internal structure of traffic by proposed edge relationships.
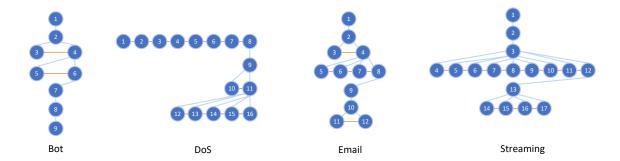
Fig. 5.   Examples of Flow Graph with different labels.

V. JSD-BASED ALGORITHM FOR FEATURE SELECTION

In this section, we introduce a feature selection algorithm based on JSD. It aims to evaluate the inter-class distribution differences and extra-class distribution differences of features in environment shifts, and selects stable features to provide accurate and stable feature representation. JSD is a method for measuring the similarity between two probability distributions. It is a variation of the KL divergence that solves the problem of KL divergence's asymmetry [34]. Assuming there are two probability distributions, P and Q, with probability density functions of $p(x)$ and $q(x)$ at point x, respectively. The definition of JSD is defined as follows:

$$JSD(P,Q) = \frac{1}{2}(D_{KL}(P||M) + D_{KL}(Q||M)) \quad (1)$$

$$D_{KL}(P,Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \quad (2)$$

$$M = \frac{1}{2}(P + Q) \quad (3)$$

where D_{KL} denotes the KL divergence, and M is the intermediate distribution between P and Q. The value of JSD ranges from 0 to 1, with 0 indicating that the two distributions are identical and 1 signifying that both distributions are completely different. The greater the similarity between two probability distributions, the closer the value of JSD is to 0.

The JSD-based feature selection algorithm is described in Algorithm 1 for multi-class classification tasks. It is important to note that we only use labeled data from the training set for feature evaluation and selection. We evaluate the inter-class and extra-class JSD of each feature for every class. We create three datasets from the training set, where T_I and T_{II} are used to evaluate the inter-class JSD of the traffic and T_I and T_{III} are used to evaluate the extra-class JSD. For example, when we evaluate the difference in feature distribution between "browsing" and other labels, we can divide "browsing" traffic into two non-overlapping datasets T_I and T_{II} based on environment shifts. Specifically, we randomly split the factors of "browsing" according to the environment shift factors, as shown in Table I, into two non-overlapping sets: EL_I and EL_{II}. Subsequently, we allocate all the traffic belonging to EL_I into T_I and all the traffic associated belonging to EL_{II} into T_{II}. If there is no environment shifts label, T_I and T_{II} are obtained by randomly dividing "browsing" traffic. In contrast, T_{III} is the traffic of other labels traffic, in this example, other labels include chat, emial file and streaming.

We then calculate the inter-class and extra-class JSD for each feature using T_I, T_{II}, and T_{III}, obtaining the distribution stability of each feature in varying environments. If the JSD of the inter-class distributions is smaller than the JSD of extra-class ones, the feature is considered stable in varying environments and is retained for subsequent traffic classification. To perform multi-class encrypted traffic classification, we also need to evaluate the comprehensive stability of features across multiple classes, weighted according to the number of samples for each class.

In this paper, we use header fields as node attributes, noting their distribution can vary across different environments. For instance, the type of content in instant messaging, like text or multimedia, impacts packet length distribution, and network bandwidth speed influences arrival time fields. To address this, we apply a feature selection algorithm described in Algorithm 1 for encrypted traffic classification tasks, choosing stable header fields as final node attributes to accurately characterize encrypted traffic.

VI. GRAPHSAT CLASSIFIER FOR ENCRYPTED TRAFFIC CLASSIFICATION

GraphSAT takes the Flow Graph as input and learns the internal structure features and rich node attributes to achieve efficient and accurate encrypted traffic classification. The structure of GraphSAT mainly consists of GraphSAGE Block, GAT Block, Readout, and Classifier.

A. GraphSAGE Block

We use GraphSAGE to collect local information on nodes in the Flow Graph and calculate node embeddings. GraphSAGE is a technique used to learn a node representation method that involves sampling and aggregating features from its local neighbors. The GraphSAGE block comprises two processes, neighbor sampling, and feature aggregation.

Neighbor sampling: It samples a fixed number of neighbors S, for each node to ensure efficient computation, regardless of the node's degree. For nodes with fewer than S neighbors,

---

Algorithm 1 JSD-based Feature Selection Algorithm

1: **Input:** Training set data T, label L, candidate feature set F, environment label EL (optional), number of samples for each class Len.
2: **Output:** Stable feature set under environment shifts Z.
3: **Variables:**
4:     T_I, T_{II}, T_{III} ← subsets of T, all three mutually exclusive.
5:     EL_I, EL_{II} ←: subsets of EL, two are mutually exclusive.
6:     FD ←: JSD difference matrix, $FD[f, l]$ denotes the JSD difference of label $l \in L$ in feature $f \in F$.
7: **procedure** JSD DIFFERENCE CALCULATION(T, L, EL, F)
8:     EL_I, EL_{II} ← two empty lists.
9:     **if** EL is True **then**
10:       Randomly divide EL into EL_I and EL_{II}, which of them have different environment settings.
11:     **end if**
12:     **for** l in L **do**
13:       T_I ← data labeled as l and el belongs to EL_I.
14:       T_{II} ← data whose label is l and el belongs to EL_{II}.
15:       T_{III} ← data whose label is not l.
16:       **for** f in F **do**
17:         $FD[f][l] \leftarrow JSD(T_I[f], T_{II}[f]) - JSD(T_I[f], T_{III}[f])$
18:       **end for**
19:     **end for**
20:     **return** FD.
21: **end procedure**
22: **procedure** STABLE FEATURE SELECTION(FD, Len)
23:     **for** fd in FD **do**
24:       $diff \leftarrow \sum_{i=1}^{n} fd_i \times Len_i$, where n is the number of types.
25:       **if** $diff < 0$ **then**
26:         Add the feature f to Z.
27:       **end if**
28:     **end for**
29:     **return** Z.
30: **end procedure**

---

sampling with replacement (SWR) [35] ensures S nodes are sampled; for those with more, sampling without replacement (SWOR) [36] is applied. GraphSAGE uses a constant K to define the hop number for neighbor sampling, enhancing distant neighbor information capture. We set $K = 2$, sampling S_1 first-order and S_2 second-order neighbors for each target node.

Feature aggregation: It creates the target node's embedding by averaging the feature vectors of sampled neighbors, moving from second-order to first-order neighbors before reaching the target node. Average aggregation is employed to combine neighbor embeddings dimension-wise, followed by a non-linear transformation. The definition is given as follows:

$$h_v^k = \sigma(W \cdot \text{MEAN}(h_v^{k-1} \cup h_u^{k-1}), \forall u \in \mathcal{N}_v) \quad (4)$$

where h_v^k is the k-th layer node v feature vector, $\mathcal{N}_v$ is v's neighbor set, σ denotes the activation function, and W is the k-th layer's trainable weight matrix.

B. GAT Block

We obtain the first layer of node embeddings using Graph-SAGE. Next, we further calculate the importance of nodes and generate new node embeddings using GAT. It assigns different learning weights to different neighbors for learning the interrelationships between nodes. GAT Block consists of two processes, calculating attention coefficients and feature aggregation.

Attention coefficient: Each attention coefficient is learned through the self-attention mechanism, where each node in the graph learns the weight for each of its neighbors based on their respective feature vectors. The definition of the attention coefficient is as follows:

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(a[Wh_i||Whj]\right)\right)}{\sum k \in \mathcal{N}_i \exp\left(\text{LeakyReLU}\left(a^T[Wh_i||Wh_k]\right)\right)} \quad (5)$$

where $\alpha_{i,j}$ represents the attention coefficient for the edge between nodes i and j, W is a weight matrix, h_i and h_j are the feature vectors for nodes i and j, respectively, $\mathcal{N}_i$ is the set of neighbors of node i, $||$ represents concatenation, and LeakyReLU is the leaky rectified linear unit activation function. The vector a is a learnable parameter vector that is shared across all nodes and is used to calculate the compatibility score between node attributes.

Feature aggregation: According to the attention coefficient, we aggregate the node attributes by weighting and summing them to obtain the embedded representation of the aggregated node. It is defined as follows:

$$h_i' = \sigma\left(\sum_{j \in N_i} \alpha_{ij} Wh_j \right) \quad (6)$$

where h_i' is the new feature output by GAT for each node i, which incorporates neighbors information, and σ is the activation function. In addition, we use multi-head attention to enhance the node attributes, which is defined as follows:

$$\overrightarrow{h_i} = \big\|_{k=1}^{K} h_i^{(k)} \quad (7)$$

where K is the number of attention heads. The final output $\overrightarrow{h_i}$ is the concatenation of the outputs from all attention heads.

C. Readout

After the GraphSAGE and GAT processing, we generate node embeddings for the Flow Graph. To represent the whole graph, we use a global mean pool to derive the graph's overall representation vector, defined as:

$$\mathcal{R}(\mathbf{H}) = \sigma\left(\frac{1}{N} \sum_{i=1}^{N} \mathbf{h}_i\right) \quad (8)$$

where $\mathcal{R}(\mathbf{H})$ is the whole graph's embedding, $\mathbf{h}_i$ represents the i-th node's feature, and σ refers to the sigmoid function.

TABLE I.   THE STATISTICS OF EXPERIMENTAL DATASETS.

Attack Detection			Application Classification				
Label	Count	Ratio	Label	Factor		Count	Ratio
Benign	50000	15.86%	Browsing	**Content**: [Blog, Map, Picture, Video], **Bandwidth**: [20Mbps, 100Mbps] **Browser**: [Google, Edge], **Speed of opening new pages**: [1s, 10s]		12272	66.88%
DDoS	50000	15.86%	Chat	**Content**: [Picture, Video, Text, Voice], **APP**: [Facebook, WeChat]		1518	8.27%
DoS	50000	15.86%	Email	**APP**: [Outlook, Google], **Action**: [Send, Read]		4810	13.11%
Spoofing	50000	15.86%	File	**Content**: [MP4, Word, Zip file], **Action**: [Upload, Download]		1179	6.43%
Recon	50000	15.86%	Streaming	**Resolution**: [270, 480, 720, 1080], **Playback**: [0.5, 1.0, 1.25, 1.5, 2.0]		1686	5.31%
Mirai	50000	15.86%					
Web attack	15239	4.83%					
Overall	315239	100%	Overall			18349	100%

D. Classifier

The classifier consists of dropout, linear, and softmax layers to prevent overfitting and classify encrypted traffic. Dropout layers randomly omit nodes to reduce interactions and over-fitting, serving as regularization. Linear layers map high-dimensional data to a lower-dimensional label space, with the softmax layer finalizing classification. We concentrate on classifying encrypted traffic by application type (such as browsing, email) and attack type (such as DoS, PortScan).

VII.   EXPERIMENT AND EVALUATION

In this section, we evaluate and compare the proposed FG-SAT. Firstly, we describe the experiment setting. Next, we perform a comparison with baseline methods, analysis on environment shifts and the JSD-based feature selection algorithm, analysis on GraphSAT, analysis on open-world classification and few-shot analysis to fully evaluate the FG-SAT.

A. Experiment Setting

1) Datasets: In this paper, we evaluate our proposed method through three encrypted traffic classification tasks: *encrypted traffic attack detection*, *encrypted traffic application classification*, and *encrypted traffic application classification under environment shifts*.

For the encrypted traffic attack detection, we use the publicly available CIC-IOT2023 [**?**] dataset, which contains benign and the most up-to-date common attacks resembling real-world traffic. To construct environment shifts in attack detection, we partition data according to the IP pair method, ensuring that traffic from the same IP pair exists only in either the training set or the test set. Additionally, we enhance the realism of environment shifts by supplementing the "Benign" class traffic with our own collected dataset (APP-SHIFTS dataset), which includes various environment shift factors.

To evaluate the environment shifts, we newly collect an application dataset, named APP-SHIFTS dataset, for encrypted traffic application classification and encrypted traffic application classification under environment shifts. We capture multiple encrypted application traffic from a campus network environment. Specifically, when the user accesses an encrypted application through the PC, the generated traffic is saved to the data server by the traffic capture tool. In addition, we construct environment shifts by setting different factors for each type of

traffic, such as content, bandwidth, and resolution, totaling 72 types of environment factors. The statistics of the datasets are shown in Table I.

2) Baseline Methods: In order to evaluate and compare the performance of FG-SAT, we summarize the following baseline methods:

- **FlowPrint:** It is a semi-supervised approach for mobile application fingerprinting using temporal correlations to generate application fingerprints.
- **1dCNN:** It extracts the first 784 bytes of encrypted flows, converting them into a picture and using 1dCNN to learn spatial features for classification.
- **CapsNet:** It utilizes CapsNet to learn spatial and location features of encrypted flows for classification.
- **GTID:** It computes n-gram frequency of packet payloads, converting them into vectors and applying transformers for feature learning.
- **ET-BERT:** It is a pre-training model that learns generic traffic representations from unlabeled traffic and fine-tunes for specific tasks.
- **FlowPic:** It generates an image for each flow from packet size and arrival time, using CNN for classification.
- **FS-Net:** It extracts packet length sequences and employs an LSTM-based Encoder-Decoder for classification.
- **Rosetta:** It applies TCP-aware augmentation and self-supervised learning to enhance TLS traffic classification in diverse networks.
- **GraphDApp:** It is a GNN-based method using packet length and direction to learn structural packet relationships for DAPP identification.

3) Evaluation Metrics and Implementation Details: We assess FG-SAT's performance using Accuracy (Acc), Precision (Pre), Recall (Rec), and F_1, employing Macro Average to mitigate bias from class imbalances. Prediction speed is gauged by the time to predict 100 flows, and model complexity and memory usage are evaluated through trainable parameters. Our experiments rely on 5-fold cross validation for robustness.

For Flow Graph construction, we limit flows to 20 packets. In GraphSAT, hidden layers for both GraphSAGE and GAT are set at 128. We use a batch size of 128, a learning rate of 0.003, and cap epochs at 100, incorporating a 0.7 dropout ratio. Parameter fine-tuning details are in section VII-F.

TABLE II.   THE RESULTS AND COMPARISON WITH BASELINE METHODS.

Categories	Method	Attack Detection				APP Classification				APP Classification under Environment Shifts					
		Pre	Rec	F_1	Acc	Pre	Rec	F_1	Acc	Pre	Rec	F_1	Acc	Time (ms)	Params
Statistics-based	FlowPrint	0.4896	0.4548	0.4421	0.4458	0.8125	0.8180	0.8082	0.8867	0.7211	0.4093	0.4720	0.7370	594.36	-
Byte-based	1dCNN	0.6095	0.5427	0.5440	0.5515	0.6728	0.7774	0.7105	0.8017	0.4781	0.5051	0.4913	0.7781	5.16	5825413
	CapsNet	0.6117	0.5605	0.5716	0.6126	0.7643	0.7961	0.7791	0.8460	0.7844	0.6998	0.7251	0.8160	185.57	7592976
	GTID	0.6489	0.6106	0.6229	0.6127	0.7836	0.8668	0.8088	0.8711	0.8202	0.7503	0.7728	0.8220	289.74	893797
	ET-BERT	0.7759	0.7766	0.7607	0.7823	0.9017	0.8834	0.8915	0.9214	0.8180	0.7928	0.8022	0.8614	396.23	132129797
Sequence-based	FlowPic	0.7058	0.6352	0.6515	0.6471	0.7749	0.7944	0.7828	0.8695	0.8296	0.6259	0.7022	0.8243	9.63	1597219
	FS-Net	0.7182	0.6409	0.6582	0.6528	0.8106	0.7313	0.7513	0.8597	0.7482	0.6972	0.7115	0.8377	210.27	2250451
	Rosetta	0.6606	0.6045	0.6197	0.6160	0.8746	0.7558	0.8033	0.8664	0.6409	0.5392	0.5587	0.7544	131.97	11402479
Graph-based	GraphDApp	0.7457	0.6783	0.6922	0.6924	0.8177	0.8775	0.8408	0.8618	0.6766	0.6026	0.5916	0.8154	9.42	**35205**
	FG-SAT full	0.7665	0.7705	0.7556	0.7764	0.8916	0.8897	0.8906	0.9468	0.7616	0.7210	0.7380	0.8406	5.03	44293
	FG-SAT	**0.8357**	**0.8427**	**0.8330**	**0.8508**	**0.9028**	**0.8956**	**0.8991**	**0.9516**	**0.8250**	**0.8049**	**0.8124**	**0.8979**	**4.82**	43013

* **Bold** denotes the best reuslt.

B. Comparison with Baseline Methods

We evaluate FG-SAT and baseline methods in attack detection, application classification, and application classification under environment shifts, focusing on accuracy, classification speed, and model parameters. Moreover, we compare *FG-SAT*, utilizing JSD-based feature selection, with *FG-SAT full* without feature selection. Accuracy reflects classification capability, while time indicates efficiency critical for encrypted traffic analysis. Model parameters gauge complexity and memory consumption, highlighting the importance of lightweight classifiers. Results are detailed in Table II.

In terms of accuracy, we evaluate the methods using macro average of four metrics, Pre, Rec, F_1, and Acc, which are cross-validated to average the results. We observe that FG-SAT shows the best performance in all encrypted traffic classification tasks, reaching an accuracy of 0.8508 in the attack classification, 6.85% higher than the best baseline method. The accuracy of 0.9516 in application classification is 3.02% higher than the best baseline method, and the accuracy of 0.8979 in the application classification under environment shifts is 3.65% higher than the best baseline method. However, the baseline method with the best results, ET-BERT, is a pre-training model for traffic representation learning, and its high-performance results from a large amount of traffic collected. Except for the pre-training method ET-BERT, FG-SAT is significantly better than all other traditional baseline methods.

In terms of time, we investigate the time it takes to predict a flow. FG-SAT takes only 4.82ms to predict 100 flows, which is the shortest time they take. It indicates that FG-SAT can simultaneously classify encrypted traffic quickly.

In terms of parameters, we count the number of trainable parameters during the model run, and the number of parameters in FG-SAT is only second to GraphDAPP, which has just one node attribute. Moreover, the number of parameters in FG-SAT is 3/10,000 of that in ET-BERT. Thus, FG-SAT can achieve the lightweight classification of encrypted traffic, which greatly reduces the memory requirements of devices in practical deployments.

Furthermore, comparing FG-SAT with FG-SAT full, we observe that after JSD-based feature selection, FG-SAT is able to achieve better performance capabilities. In particular, the accuracy of FG-SAT is 7.44% higher than that of FG-SAT

full under environment shifts. Thus, the JSD-based feature selection algorithm is able to select robust features and improve the generalization ability of the method.

Overall, FG-SAT achieves the best performance in terms of combined accuracy, time and parameters.

C. Analysis on Environment Shifts and JSD-based Algorithm

In this section, we first analyze the impact of environment shifts on the data distribution, and then perform the analysis of the JSD-based feature selection algorithm for a specific "browsing" traffic identification scenario. Finally, we compare the JSD-based algorithm with existing feature selection algorithms to evaluate the performance.

1) Analysis on Environment Shifts: In the collected encrypted traffic application classification dataset, environment shifts are constructed by altering various factors. In this section, we analyze the impact of these factors on data distribution. Different shift settings are implemented for five types of encrypted traffic, such as bandwidth, application, and action. Header fields are selected as node attributes in this paper, therefore, the distribution status of the header fields of packets is compared to clearly reflect data distribution differences. Since header fields represent multi-dimensional features, data distribution is measured by comparing the distance of packets to the cluster centroid through clustering methods. Note that we only use clustering to obtain the centroid for calculating data distribution, rather than employing clustering for classification. Therefore, we set the number of clusters to 1. For instance, under the content shift of "browsing" traffic, K-means [37] clustering is performed on the traffic of two groups of content, the cluster centroid is obtained, and then the Euclidean distance of the data packet to the cluster centroid is calculated. This distance characterizes the distribution status of packets. Ultimately, kernel density estimation curves [38] are employed to statistically analyze the distribution of Euclidean distances of the two types of content traffic, representing the distribution differences of traffic under content shift.

The distribution statistics of traffic for each environment shift are illustrated in Figure 6. Among them, *none* represents the distribution of two groups of traffic without environment shift, and it can be observed that the data distribution almost overlaps in all categories of traffic. However, this phenomenon
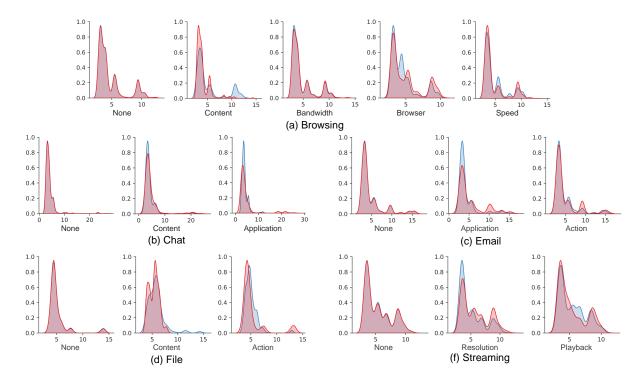
Fig. 6. Differences in data distribution under various environment shifts.

is nearly impossible to achieve in the real world. In all shift settings, the distributions of the two groups of data exhibit differences to varying degrees. Comparatively, content shift, application shift, resolution shift, and playback shift all induce significant data distribution differences. Consequently, these common environment shifts lead to different data distributions, further diminishing the accuracy of encrypted traffic classification.

2) Analysis on JSD-based Algorithm: We examine the JSD-based algorithm's response to environment shifts in "browsing" traffic, focusing on content, bandwidth, browser, and speed shift factors. We assess feature robustness using inter-class and extra-class JSD comparisons, illustrated in Figure 7 with blue for stable (inter-class JSD ¡ extra-class JSD) and red for unstable features (inter-class JSD ¿ extra-class JSD). Without environment shifts, all features appear stable for browsing traffic identification. However, environment shifts introduce unstable features, notably affecting packet length and time.

We observe that IP header fields, such as *IP DS field, IP id, IP flags, IP TTL*, remain robust across shifts, highlighting their significance for browsing traffic identification. Content shifts impact features the most, leaving only 20 robust fields, whereas bandwidth shifts have the least effect.

3) Comparison with Baseline Feature Selection: The aim of the JSD feature selection algorithm is to evaluate the differences in feature distribution and select robust features under environment shifts. In this section, we compare the performance of the JSD feature selection algorithm with other feature selection algorithms as described below. (1) Chi-Squared Test [39], which evaluates the dependence between

TABLE III.   COMPARISON RESULTS WITH BASELINE FEATURE SELECTION ALGORITHMS.

Algorithm	Pre	Rec	F_1	Acc	Dimension
Chi-Squared Test	0.7179	0.7390	0.7056	0.8107	25
L1-LR	0.7216	0.7478	0.7247	0.8511	32
RFE	0.7251	0.7562	0.7219	0.8441	25
RFFI	0.7123	0.7684	0.7313	0.8366	25
None	0.7616	0.7210	0.7380	0.8406	39
JSD Algorithm	**0.8250**	**0.8049**	**0.8124**	**0.8979**	**25**

features and target variables using chi-square statistics. (2) L1-Regularized Logistic Regression (L1-LR) [40], which applies L1 regularization to logistic regression models, inducing sparsity and performing feature selection. (3) Recursive Feature Elimination (RFE) [41], which iteratively removes the weakest features while building a model to find the optimal feature subset. and (4) Random Forest Feature Importance (RFFI) [42], which ranks features based on their importance in an ensemble of decision trees. The Chi-Squared Test, RFE and RFFI aim to score and rank features. To clearly compare the performance of them with that of the JSD algorithm, we ensure that the dimensions of their respective features remain consistent with the dimensions of the JSD features. While the L1-LR indirectly achieves feature selection by reducing the coefficients of some features to zero, therefore, we maintain its original feature selection dimensions.

Comparison results, as shown in Table III, indicate that none of these feature selection algorithms can outperform the JSD algorithm under environment shifts. Additionally, only L1-LR provides better classification results than those obtained

	frame.len	frame.cap_len	frame.time_relative	eth.type	ip.version	ip.hdr_len	ip.tos	ip.dsfield	ip.len	ip.id	ip.flags	ip.flags.rb	ip.flags.df
Content	0.009	0.009	-0.005	0.000	-0.163	0.000	0.000	-0.063	-0.015	-0.137	-0.021	-0.163	-0.021
Bandwidth	-0.076	-0.076	-0.038	0.000	-0.163	0.000	0.000	-0.134	-0.072	-0.088	-0.137	-0.163	-0.137
Browser	0.002	0.002	-0.031	0.000	-0.163	0.000	0.000	-0.104	0.017	-0.083	-0.134	-0.163	-0.134
Speed	-0.030	-0.030	0.021	0.000	-0.164	0.000	0.000	-0.098	-0.027	-0.127	-0.129	-0.164	-0.129
None	-0.088	-0.088	-0.139	0.000	-0.163	0.000	0.000	-0.128	-0.094	-0.163	-0.152	-0.163	-0.152

	ip.flags.mf	ip.flags.sf	ip.frag_offset	ip.ttl	ip.proto	ip.checksum	tcp.srcport	tcp.dstport	tcp.seq	tcp.ack	tcp.len	tcp.hdr_len	tcp.flags
Content	-0.163	0.000	-0.163	-0.074	-0.015	-0.365	-0.132	-0.136	0.037	0.013	-0.019	-0.042	0.008
Bandwidth	-0.163	0.000	-0.163	-0.152	-0.149	-0.363	-0.190	-0.212	-0.092	-0.128	-0.050	-0.043	-0.045
Browser	-0.163	0.000	-0.163	-0.120	-0.140	-0.336	-0.078	-0.120	-0.026	0.009	-0.025	0.045	0.013
Speed	-0.164	0.000	-0.164	-0.137	-0.120	-0.377	-0.018	-0.127	-0.005	-0.046	0.003	-0.020	0.001
None	-0.163	0.000	-0.163	-0.171	-0.163	-0.377	-0.289	-0.261	-0.131	-0.162	-0.069	-0.043	-0.052

	tcp.flags.fin	tcp.flags.syn	tcp.flags.reset	tcp.flags.push	tcp.flags.ack	tcp.flags.urg	tcp.options.mss_val	tcp.window_size	tcp.checksum	tcp.urgent_pointer	udp.srcport	udp.dstport	udp.length
Content	0.013	0.031	0.041	0.040	0.032	0.041	-0.052	-0.053	-0.080	0.041	0.040	0.029	0.025
Bandwidth	-0.036	-0.020	-0.001	-0.007	-0.020	-0.001	-0.045	-0.142	-0.171	-0.001	-0.055	-0.063	-0.137
Browser	-0.017	0.016	0.008	-0.009	0.000	0.002	0.044	-0.078	-0.180	0.002	-0.077	-0.072	-0.139
Speed	0.016	-0.010	0.030	0.036	-0.011	0.026	-0.042	-0.103	-0.041	0.026	-0.008	-0.033	-0.109
None	-0.053	-0.029	-0.001	-0.017	-0.029	-0.001	-0.053	-0.129	-0.258	-0.001	-0.088	-0.110	-0.158

Fig. 7. The results on JSD-based feature selection. The horizontal axis denotes the header fields and the vertical axis denotes specific environment shifts. The color gradient indicates the magnitude of the JSD difference, blue denotes the robust fields and red denotes the unstable fields, and the larger the difference, the stronger the color.

TABLE IV. THE RESULTS OF ANALYSIS ON GRAPHSAT AND COMPARISION WITH GCN [43], GRAPHSAGE [31], AND GAT [32].

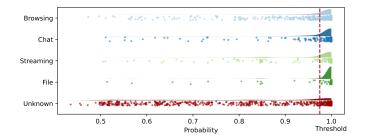| Model	Pre	Rec	F_1	Acc	Time (ms)	Params
GCN	0.8799	0.8711	0.8752	0.9358	4.99	**38149**
GraphSAGE	0.8998	0.8953	0.8973	0.9478	**4.17**	75269
GAT	0.8892	0.8988	0.8928	0.9462	5.15	38917
GraphSAT	**0.9028**	**0.8956**	**0.8991**	**0.9516**	4.82	43013

Fig. 8. The results on open-world classification, where the email class is removed from the training set but kept in the test set as an unknown class. The results show that 70% of the traffic falls below the threshold, while over 85% of the known traffic is above the threshold.

without feature selection, whereas the other two feature selection algorithms provide lower accuracy. Consequently, the JSD algorithm outperforms state-of-the-art algorithms under environment shifts.

D. Analysis on GraphSAT

In this section, we conduct a comprehensive analysis of the GraphSAT model in relation to other traditional GNN models, specifically, GCN, GraphSAGE, and GAT. Our goal is to evaluate the efficacy and efficiency of our proposed model in the context of encrypted traffic application classification. The results present in Table IV demonstrate that GraphSAT exhibits better performance across various evaluation metrics, including Pre, Rec, F_1, and Acc, highlighting its effectiveness in handling encrypted traffic classification tasks.

Furthermore, our analysis reveals that GraphSAT offers a competitive balance between accuracy and computational demands. The optimal combination of accuracy, time, and parameters count positions GraphSAT as a more favorable choice for encrypted traffic classification in comparison to the other GNN models.

E. Analysis on Open-World Classification

In the open-world, challenges arise not only from environment shifts but also from the presence of unknown class

traffic, which is not seen during training. To investigate the performance of FG-SAT in such open-world scenarios, we incorporate an additional unknown class in the test set that is absent from the training set. The experimental results, as depicted in Figure 8, demonstrate some noteworthy patterns.

We observe that the prediction probability for known classes is relatively high, with 85% of them exceeding 0.975. In contrast, the prediction probability for 70% of the unknown traffic falls below 0.975. This finding indicates that FG-SAT is generally adept at detecting known classes with high confidence. However, when confronted with unknown traffic, the feature patterns of unknown traffic often exhibit substantial dissimilarities compared to those of known classes, which consequently leads FG-SAT to struggle with predicting them as one of the known classes. This results in lower classification probabilities for unknown traffic.

Given these findings, we propose that in open-world scenarios, the identification of unknown traffic can be effectively achieved by setting an appropriate threshold for the classifier.
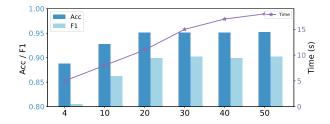
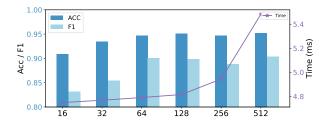Fig. 9.   The results on the maximum packet number of FG-SAT.

Fig. 10.   The results on the dimension of the hidden channel of FG-SAT.

This approach allows FG-SAT to discriminate between known and unknown classes based on their classification probabilities.

F. Parameter Tuning

In this section, we introduce the selection of parameters, including (1) the maximum packet number, and (2) the dimension of the hidden channel.

1) The Maximum Packet Number: Flow duration can vary widely, impacting model efficiency. We adjust flow aggregation time by setting a maximum packet number, the results of which are illustrated in Figure 9. Experiments with different packet counts show accuracy improves with higher counts up to a threshold. Analysis indicates accuracy gains level off beyond 20 packets, while aggregation time increases significantly. Therefore, we set the maximum packet count at 20.

2) The Dimension of Hidden Channel: A larger hidden channel dimension enhances flow graph feature extraction and accuracy but increases computation time. Balancing accuracy and computational complexity is key. Through a series of experiments, as depicted in Figure 10, we observe show that as the hidden channel dimension grows, accuracy improves up to a point. At a dimension of 128, accuracy plateaus, indicating an optimal balance between accuracy and processing time. Thus, we set the hidden channel dimension at 128 for efficient, effective encrypted traffic classification.

VIII.   CONCLUSION

In this paper, we propose the Flow Graph for efficient encrypted traffic classification, which constructs flows as graphs to characterize the internal structure relationships and rich node attributes. To improve the generalization of the model to the real world, we propose a JSD-based feature selection algorithm that is able to select robust features under environment shifts. In addition, we design a fusion GraphSAGE and GAT classifier

GraphSAT, which can efficiently and deeply learn Flow Graph features to achieve fast and accurate classification. We comprehensively evaluate the FG-SAT in three different scenarios of encrypted traffic classification tasks. It shows outstanding performance and outperforms state-of-the-art methods in terms of accuracy, time, parameters, and generalization ability.

In our future work, we plan to improve the Flow Graph by investigating the influence of different edge relationships and assigning them varying weights. Additionally, we will further investigate the ability of FG-SAT to identify unknown categories. These efforts will enhance the effectiveness and robustness of our proposed method, as well as contribute to the advancement of the field of network traffic analysis.

REFERENCES

[1]   C. Oh, J. Ha, and H. Roh, "A survey on tls-encrypted malware network traffic analysis applicable to security operations centers," *Applied Sciences*, vol. 12, no. 1, p. 155, 2021.

[2]   J. Liu, Y. Zeng, J. Shi, Y. Yang, R. Wang, and L. He, "Maldetect: A structure of encrypted malware traffic detection," *Computers, Materials and Continua*, vol. 60, no. 2, pp. 721–739, 2019.

[3]   M. Abbasi, A. Shahraki, and A. Taherkordi, "Deep learning for network traffic monitoring and analysis (NTMA): A survey," *Comput. Commun.*, vol. 170, pp. 19–41, 2021.

[4]   M. Shafiq, Z. Tian, A. K. Bashir, A. Jolfaei, and X. Yu, "Data mining and machine learning methods for sustainable smart cities traffic classification: A survey," *Sustainable Cities and Society*, vol. 60, p. 102177, 2020.

[5]   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 *Proceedings of the 2nd International Conference on Information Systems Security and Privacy, ICISSP 2016, Rome, Italy, February 19-21, 2016*, O. Camp, S. Furnell, and P. Mori, Eds. SciTePress, 2016, pp. 407–414.

[6]   T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*, K. Fu and J. Jung, Eds.   USENIX Association, 2014, pp. 143–157.

[7]   J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, T. Holz and S. Savage, Eds.   USENIX Association, 2016, pp. 1187–1203.

[8]   B. Anderson, S. Paul, and D. A. McGrew, "Deciphering malware's use of TLS (without decryption)," *J. Comput. Virol. Hacking Tech.*, vol. 14, no. 3, pp. 195–211, 2018.

[9]   B. Anderson and D. A. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2016, Vienna, Austria, October 28, 2016*, D. M. Freeman, A. Mitrokotsa, and A. Sinha, Eds.   ACM, 2016, pp. 35–46.

[10]   M. Korczynski and A. Duda, "Markov chain fingerprinting to classify encrypted traffic," in *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*. IEEE, 2014, pp. 781–789.

[11]   M. Shen, M. Wei, L. Zhu, M. Wang, and F. Li, "Certificate-aware encrypted traffic classification using second-order markov chain," in *24th IEEE/ACM International Symposium on Quality of Service, IWQoS 2016, Beijing, China, June 20-21, 2016*.   IEEE, 2016, pp. 1–10.

[12]   Y. Chen, T. Zang, Y. Zhang, Y. Zhou, and Y. Wang, "Rethinking encrypted traffic classification: A multi-attribute associated fingerprint approach," in *27th IEEE International Conference on Network Protocols, ICNP 2019, Chicago, IL, USA, October 8-10, 2019*.   IEEE, pp. 1–11.

[13] 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 *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.

[14] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International Conference on Information Networking, ICOIN 2017, Da Nang, Vietnam, January 11-13, 2017*. IEEE, 2017, pp. 712–717.

[15] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE International Conference on Intelligence and Security Informatics, ISI 2017, Beijing, China, July 22-24, 2017*. IEEE, 2017, pp. 43–48.

[16] S. Cui, B. Jiang, Z. Cai, Z. Lu, S. Liu, and J. Liu, "A session-packets-based encrypted traffic classification using capsule neural networks," in *21st IEEE International Conference on High Performance Computing and Communications; 17th IEEE International Conference on Smart City; 5th IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2019, Zhangjiajie, China, August 10-12, 2019*, Z. Xiao, L. T. Yang, P. Balaji, T. Li, K. Li, and A. Y. Zomaya, Eds. IEEE, 2019, pp. 429–436.

[17] X. Han, S. Cui, S. Liu, C. Zhang, B. Jiang, and Z. Lu, "Network intrusion detection based on n-gram frequency and time-aware transformer," *Computers & Security*, p. 103171, 2023.

[18] 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, Virtual Event, Lyon, France, April 25 - 29, 2022*, F. Laforest, R. Troncy, E. Simperl, D. Agarwal, A. Gionis, I. Herman, and L. Médini, Eds. ACM, 2022, pp. 633–642.

[19] A. Ramezani, A. Khajehpour, and M. J. Siavoshani, "On multi-session website fingerprinting over TLS handshake," in *10th International Symposium on Telecommunications, IST 2020, Tehran, Iran, December 15-17, 2020*. IEEE, 2020, pp. 211–216.

[20] T. Shapira and Y. Shavitt, "Flowpic: A generic representation for encrypted traffic classification and applications identification," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 2, pp. 1218–1232, 2021.

[21] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "Fs-net: A flow sequence network for encrypted traffic classification," in *2019 IEEE Conference on Computer Communications, INFOCOM 2019, Paris, France, April 29 - May 2, 2019*. IEEE, 2019, pp. 1171–1179.

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

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

[24] M. Shen, K. Ye, X. Liu, L. Zhu, J. Kang, S. Yu, Q. Li, and K. Xu, "Machine learning-powered encrypted network traffic analysis: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 791–824, 2023.

[25] C. Fu, Q. Li, and K. Xu, "Detecting unknown encrypted malicious traffic in real time via flow interaction graph analysis," in *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023.

[26] Z. Fu, M. Liu, Y. Qin, J. Zhang, Y. Zou, Q. Yin, Q. Li, and H. Duan, "Encrypted malware traffic detection via graph-based network analysis," in *25th International Symposium on Research in Attacks, Intrusions and Defenses, RAID 2022, Limassol, Cyprus, October 26-28, 2022*. ACM, 2022, pp. 495–509.

[27] T. Pham, T. Ho, T. T. Huu, T. Cao, and H. L. Truong, "Mappgraph: Mobile-app classification on encrypted network traffic using deep graph

convolution neural networks," in *ACSAC '21: Annual Computer Security Applications Conference, Virtual Event, USA, December 6 - 10, 2021*. ACM, 2021, pp. 1025–1038.

[28] N. Malekghaini, E. Akbari, M. A. Salahuddin, N. Limam, R. Boutaba, B. Mathieu, S. Moteau, and S. Tuffin, "Data drift in dl: Lessons learned from encrypted traffic classification," in *2022 IFIP Networking Conference (IFIP Networking)*, 2022, pp. 1–9.

[29] B. Anderson and D. A. McGrew, "Machine learning for encrypted malware traffic classification: Accounting for noisy labels and non-stationarity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 2017, pp. 1723–1732.

[30] N. Malekghaini, E. Akbari, M. A. Salahuddin, N. Limam, R. Boutaba, B. Mathieu, S. Moteau, and S. Tuffin, "Deep learning for encrypted traffic classification in the face of data drift: An empirical study," *Comput. Networks*, vol. 225, p. 109648, 2023.

[31] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 1024–1034.

[32] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[33] S. Cui, J. Liu, C. Dong, Z. Lu, and D. Du, "Only header: a reliable encrypted traffic classification framework without privacy risk," *Soft Computing*, vol. 26, no. 24, pp. 13 391–13 403, 2022.

[34] F. Nielsen, "On a generalization of the jensen-shannon divergence and the jensen-shannon centroid," *Entropy*, vol. 22, no. 2, p. 221, 2020.

[35] R. Razavi-Far, M. Farajzadeh-Zanajni, B. Wang, M. Saif, and S. Chakrabarti, "Imputation-based ensemble techniques for class imbalance learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 5, pp. 1988–2001, 2019.

[36] R. Zmigrod, T. Vieira, and R. Cotterell, "Efficient sampling of dependency structures," *arXiv preprint arXiv:2109.06521*, 2021.

[37] M. Ahmed, R. Seraj, and S. M. S. Islam, "The k-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, vol. 9, no. 8, p. 1295, 2020.

[38] F. Kamalov, "Kernel density estimation based sampling for imbalanced class distribution," *Information Sciences*, vol. 512, pp. 1192–1201, 2020.

[39] L. Hakim, R. Fatma *et al.*, "Influence analysis of feature selection to network intrusion detection system performance using nsl-kdd dataset," in *2019 International conference on computer science, information technology, and electrical engineering (ICOMITEE)*. IEEE, 2019, pp. 217–220.

[40] W. Li and J. Lederer, "Tuning parameter calibration for l1-regularized logistic regression," *Journal of Statistical Planning and Inference*, vol. 202, pp. 80–98, 2019.

[41] H. Jeon and S. Oh, "Hybrid-recursive feature elimination for efficient feature selection," *Applied Sciences*, vol. 10, no. 9, p. 3211, 2020.

[42] J. L. Speiser, M. E. Miller, J. Tooze, and E. Ip, "A comparison of random forest variable selection methods for classification prediction modeling," *Expert systems with applications*, vol. 134, pp. 93–101, 2019.

[43] B. Jiang, Z. Zhang, D. Lin, J. Tang, and B. Luo, "Semi-supervised learning with graph learning-convolutional networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 11 313–11 320.