VT-GAT: A Novel VPN Encrypted Traffic Classification Model Based on Graph Attention Neural Network

Hongbo Xu[1,2], Shuhao Li[1,2], Zhenyu Cheng[1(✉)], Rui Qin[1], Jiang Xie[1,2], and Peishuai Sun[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
{xuhongbo,lishuhao,chengzhenyu,qinrui,xiejiang,sunpeishuai}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract. Virtual Private Network (VPN) technology is now widely used in various scenarios such as telecommuting. The importance of VPN traffic identification for network security and management has increased significantly with the development of proxy technology. Unlike other tasks such as application classification, VPN traffic has only one flow problem. In addition, the development of encryption technology brings new challenges to VPN traffic identification.

This paper proposes VT-GAT, a VPN traffic graph classification model based on Graph Attention Networks (GAT), to solve the above problems. Compared with existing VPN encrypted traffic classification techniques, VT-GAT solves the problem that previous techniques ignore the graph connectivity information contained in traffic. VT-GAT first constructs traffic behavior graphs by characterizing raw traffic data at packet and flow levels. Then it combines graph neural networks and attention mechanisms to extract behavioral features in the traffic graph data automatically. Extensive experimental results on the Datacon21 dataset show that VT-GAT can achieve over 99% in all classification metrics. Compared to existing machine learning and deep learning methods, VT-GAT improves F1-Score by about 3.02%–63.55%. In addition, VT-GAT maintains good robustness when the number of classification categories varies. These results demonstrate the usefulness of VT-GAT in the VPN traffic classification.

Keywords: Traffic classification · VPN · Encrypted traffic · Graph attention networks · Graph classification

1 Introduction

Virtual private network (VPN), a type of encrypted communication technology, is now commonly used in various scenarios such as identity concealment, censorship avoidance, and telecommuting. Therefore, it is often used by criminals

H. Gao et al. (Eds.): CollaborateCom 2022, LNICST 461, pp. 437–456, 2022.
https://doi.org/10.1007/978-3-031-24386-8_24

to engage in pornography, network intrusion, data theft, and other illegal activities. Achieving accurate VPN traffic classification is essential to preventing and tracking cybercrime incidents.

In the early days, Deep Packet Inspection (DPI) method could obtain high accuracy by examining packets. However, with the popularity of encryption technology, traffic content becomes unresolvable, which leads to DPI methods becoming unusable. Soon after, the concept of flow is proposed. Packets with the same quintuple information (source IP, source port, destination IP, destination port, and IP protocol) are defined as a flow. Most of the existing encrypted traffic classification techniques cut the traffic at packet or flow level [1] and then use machine learning algorithms or deep learning algorithms to automatically learn the extracted traffic features, which can often achieve good results. However, when users use VPN applications for identity obfuscation, the number of flows extracted from the traffic will drop dramatically. As shown in Fig. 1, the server-side IP address and port of the packet sent by the user are replaced by the VPN application. Therefore, VPN traffic cannot be split into multiple flows by server IP address and port. This phenomenon is known as the only one flow problem [2]. Only one flow problem can significantly reduce the extractable traffic features, thus severely affecting the performance of existing traffic classification techniques.

Finding practical and robust features is a feasible way to solve the single-flow problem. We note that previous works mainly focused on the spatio-temporal features of traffic. Moreover, the graph connectivity behavioral features implied by the traffic are generally ignored. Using only traditional deep learning methods does not extract the connection behavior features of the flows quickly and effectively from the existing features. Therefore, In this paper, we propose VT-GAT, a graph neural network model that fuses graph behavior features and spatio-temporal traffic features to address the above problems. Firstly, we extract traditional spatio-temporal traffic features and traffic behavior graph features from the original traffic. We then use graph attention networks (GAT) [3] to learn the combined features automatically. Compared with existing techniques, VT-GAT enriches the traffic features by introducing traffic behavior graph information. We conduct extensive comparison experiments between our model and other models on the Datacon21 dataset to demonstrate the effectiveness of VT-GAT.

The contributions of this paper are summarized as follows.

1) We present a method for extracting traffic behavior graphs from VPN encrypted traffic. It can transform the traffic classification problem into a graph classification problem. Through experimental validation, this method can effectively improve the model's classification accuracy.
2) We propose the VT-GAT model based on graph attention networks. As far as we know, this is the first model that uses graph neural networks to achieve VPN traffic classification. VT-GAT integrates spatio-temporal features of traffic and graph behavioral features to achieve classification, which makes up for the shortcomings of existing techniques. Furthermore, VT-GAT enhances

the robustness of the model by aggregating the features of neighboring nodes based on the graph attention mechanism.

3) We present a traffic graph dataset[1] suitable for VPN encrypted traffic classification. Moreover, we implement a prototype system based on the VT-GAT model and conduct experiments on the recently released dataset Datacon21. The experimental results show that VT-GAT achieves the best performance in all evaluation metrics (Accuracy, Recall, Precision, and F1-Score) compared to existing machine learning and deep learning methods.

The rest of the paper is organized as follows. Section 2 presents background and summarises related work on traffic classification. Section 3 describes the construction process of VT-GAT in detail. Section 4 describes the experimental setup and analyzes the experimental results. After a brief discussion in Sect. 5, Sect. 6 concludes the paper.

2 Background and Related Work

2.1 The only One Flow Problem Challenge

As shown in Fig. 1, the VPN application takes over the user's traffic data in the VPN proxy environment. In the traffic captured at the gateway, the destination address of the traffic packets sent by the user is the VPN proxy server instead of the actual destination address. At the VPN proxy server, the actual address of the packet is decrypted. And these traffic packets are then sent by the proxy server to the actual destination address. The VPN proxy is a middleman between the user and the target server. The destination address of all traffic packets sent by the user is the VPN proxy server. This phenomenon is the only one flow problem [2]. As shown in Fig. 2, only one flow problem causes the traffic data captured at the gateway to form a star-shaped network. Flow-based traffic classification methods are severely impacted due to the reduction of classifiable flows. Only one flow problem brings a great challenge to VPN traffic classification.

2.2 Related Work

Network traffic classification techniques have been extensively studied [4]. Based on the chronological order of technology development, we classify the related work on traffic classification into the following three categories.

Methods Based on Rules. Rules-based traffic identification methods focus on identifying traffic by matching feature fields. One of the most commonly used methods is DPI. This method is widely used in traffic classification tasks such as application identification and intrusion detection. The core principle of DPI

[1] The dataset can be found at https://anonymous.4open.science/r/VPN_Traffic_Graph_Dataset-EDA0. Researchers who use the dataset should indicate the source of data by citing this paper.

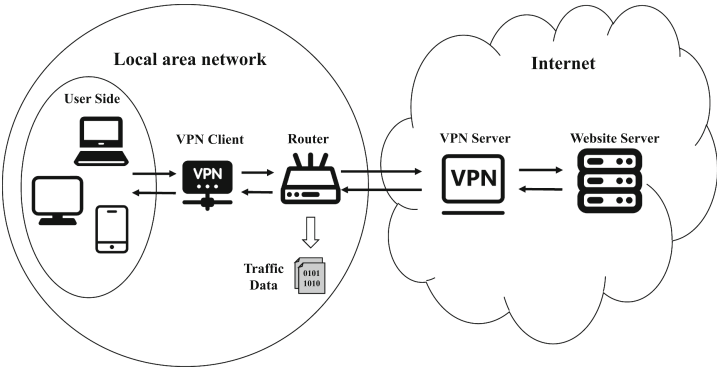

Fig. 1. The only one flow problem challenge

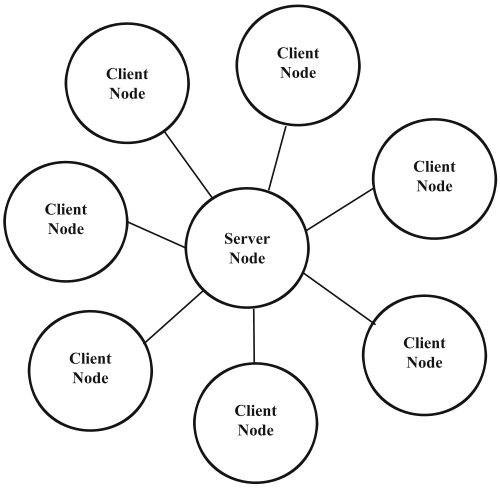

Fig. 2. Star communication network

is to use pattern matching to locate traffic. It does this by searching for specific strings or regular expressions in the content of traffic packets, which relies on the unencrypted transmission of traffic data [5].

In [6], the authors first develop a system called nDPI for protocol classification using packet headers and payloads. This system supports the detection of more than 170 protocols. In [7], the authors propose a pattern language similar to regular expressions to summarize the rules of packet sequences, which is then integrated into the DPI system to enhance its identification capability.

Methods Based on Machine Learning. With the popularity of encryption technology, DPI techniques are no longer suitable for encrypting traffic because they match packet contents based on payload. The popularity of CDN technology has also drastically reduced the accuracy of IP-based and port-based methods. In addition, DPI techniques require constant manual updating of rules to adapt to traffic changes. To solve the above problems, machine learning methods are commonly applied to encrypted traffic classification. After feature extraction and selection, machine learning algorithms can often achieve good classification results.

In [8], the authors convert the number of occurrences of flow-level features (e.g., TCP stream length, SSL/TLS handshake information type, etc.) into frequency distributions. The frequency distribution data extracted from the encrypted traffic is then used to identify applications by incorporating the Random Forest. In [9], by re-labeling the data misjudged by the model through reinforcement learning, the authors improve the classification accuracy of AppScanner [10]. In [11], the authors propose a more robust feature called the sliding window JS divergence feature. They then use machine learning algorithms (SVM, Random Forest, Bayesian) to learn traditional and newly discovered traffic classification features. In [12], the authors propose a semi-supervised clustering technique FlowPrint. The main principle of Flowprint is to classify network flows based on the correlation of temporal features between flow destinations.

Methods Based on Deep Learning. Machine learning methods can address the shortcomings of rules-based methods and enable the detection of encrypted traffic. However, machine learning methods often rely on expert experience to filter traffic features, and feature selection needs to be updated according to different data. Deep learning has gradually become a research hotspot favored by scholars with its powerful automatic feature extraction capability [13]. Many scholars use deep learning for automatic traffic feature extraction to solve the above problem.

In [14], the authors extract the header bytes of network traffic packets and achieve the accurate real-time traffic classification through a self-attentive mechanism, thus optimizing the feature extraction steps. In [15], the authors combine convolutional auto-encoding (CAE) and convolutional neural network (CNN) to classify encrypted traffic. In [1], the authors apply LSTM and CNN to learn traffic's temporal and spatial features by combining packet-level and flow-level features, respectively. Chen et al. [2] propose a novel method called AI-FlowDet. It leverages the CNN model to find behavior change points in traffic data.

Deep learning methods have achieved good results in some tasks such as application classification, intrusion detection, etc. Unfortunately, the above work lacks a targeted design for VPN traffic. There is still much room for improvement in classification effectiveness for VPN traffic due to the only one flow problem.

In summary, with the development of encryption technology and the existence of only one flow problem, existing techniques cannot meet the VPN encrypted traffic classification requirements. We note that existing techniques focus only on

the traffic's spatio-temporal features while ignoring the traffic's graph interaction behavior features. Moreover, extracting graph interaction behavior features can help improve the classification performance of VPN traffic. Therefore, we propose VT-GAT, a VPN encrypted traffic classification method that combines spatio-temporal features of traffic and graph interaction behavior features.

3 Methodology

In this section, we introduce the construction method of VT-GAT. First, we introduce the method of constructing traffic behavior graphs from the original traffic. Then we briefly describe the basic principles of graph neural networks and graph attention networks. Finally, we show the overall structure of VT-GAT.

3.1 Traffic Behavior Graph Construction

Extraction of Node Features. Flow is one of the commonly used concepts in traffic classification. It comprises a sequence of packets with the same IP quintet characteristics (source IP, source port, destination IP, destination port, and protocol). The interval between consecutive packets in a flow is less than a fixed threshold, e.g., 10s. We obtain the node features of the traffic graph at the flow level. First, we use a combination of IP addresses and application ports to identify nodes uniquely. Then, we use CICFlowMeter [16,17] to extract flow level features. For each network flow, the source and destination addresses are determined based on the transmission direction of its first packet. After removing irrelevant information such as IP addresses and ports, we retain seventy-seven-dimensional features, which can be classified into the following four categories.

- Aggregate features: These features are the overall features of traffic obtained in network flows, including total duration, the total number of packets, total packet length, etc.
- Time features: These features mainly contain original and statistical features associated with time, including the average time between packets sent, the total time between packets sent, etc.
- Statistical features: These features are obtained from statistics based on packet size (excluding aggregation features), including the number of upstream packets per second, packet length mean value, packet length standard deviation, etc.
- Content features: These features are the features of packet content fields, including the number of FIN packets, the number of SYN packets, the number of ACK packets, etc.

The details of the extracted features are shown in Table 1. The flow ID identifies these features. Specifically, for a particular node p, its node features are equal to the average of all the associated flow features. We perform Min-Max normalization on the node features to accelerate the model training convergence. After normalization, all data are converted to float type data in the range 0–1.

Table 1. Flow-level features

Index	Name	Attributes	Direction	Category
1	Flow duration	–	All	Aggregated features
2–3	Total Packet	–	Fwd, Bwd	Aggregated features
4–5	Total Length of Packet	–	Fwd, Bwd	Aggregated features
6–7	Header Length	–	Fwd, Bwd	Aggregated features
8	Down/Up Ratio	–	All	Aggregated features
9	Average Packet Size	–	All	Aggregated features
10–11	Init Win bytes	–	Fwd, Bwd	Aggregated features
12–13	PSH Flags	–	Fwd, Bwd	Content features
14–15	URG Flags	–	Fwd, Bwd	Content features
16–23	Flag Count	FIN, SYN, RST, PSH, ACK, URG, CWR, ECE	All	Content features
24	Protocol	–	All v	Content features
25–28	Flow IAT	Max, Min, Mean, Std	Fwd, Bwd, All	Temporal features
29–38	IAT	Max, Min, Mean, Std, Sum	Fwd, Bwd, All	Temporal features
39–42	Active	Max, Min, Mean, Std	All	Temporal features
43–46	Idle	Max, Min, Mean, Std	All	Temporal features
47–54	Packet Length	Max, Min, Mean, Std	Fwd, Bwd, All	Statistical features
55	Flow Bytes/s	–	All	Statistical features
56	Flow Packets/s	–	All	Statistical features
57–58	Packets/s	–	Fwd,Bwd	Statistical features
59–63	Packet Length	Max,Min, Mean, Std, Variance	All	Statistical features
64–65	Segment Size	Avg	Fwd, Bwd	Statistical features
66–67	Bytes/Bulk	Avg	Fwd, Bwd	Statistical features
68–69	Packet/Bulk	Avg	Fwd, Bwd	Statistical features
70–71	Bulk Rate	Avg	Fwd, Bwd	Statistical features
72–73	Subflow Packets	–	Fwd, Bwd	Statistical features
74–75	Subflow Bytes	–	Fwd, Bwd	Statistical features
76	Act Data Pkts	-	Fwd	Statistical features
77	Seg Size	Min	Fwd	Statistical features

Edge Construction Method. To extract graph structure features more comprehensively, we use a sliding window approach to sample the graph data. Within each traffic sliding window, we count the number of packet connections between any two nodes. Next, an edge is constructed between the nodes with more than one connection. VT-GAT automatically learns the weights between different nodes through the attention mechanism. Thus the weights of all edges are set to be equal. The combined edge data and node data are the corresponding traffic graph data. We input all node features and edge data into the graph neural network (GNN) [18] to reduce the reliance on expert knowledge. The complex feature selection process can be omitted by the graph neural network's powerful automatic feature selection capability. As shown in Algorithm 1, after generating the sliding traffic windows, all the sliding traffic windows are processed to generate traffic graphs sequentially. Each window corresponds to the generation of a traffic graph sample.

Algorithm 1: The construction of the traffic behavior graph

Input: A sequence of traffic packets P, A sequence of node features F
Output: The corresponding sequence of traffic behaviour graphs $G = (g_1, \ldots, g_n)$

Set the sliding time window T_s and the sliding interval M_s, then generate a list of sliding time window intervals $T = [(L_1, R_1), \ldots, (L_n, R_n)]$ using P ;

for $i = 1$ *to* n **do**
  Filter the eligible sequences $O = (O_1, \ldots, O_m)$ from the time window $(L_i, R_i) \in T$;
  Initialize the dictionary of edge weights D ;
  Initialize the set of edges E and the set of nodes V ;
  for $j = 1$ *to* m **do**
    Get the IP and Port information of O_j to form the source address identification IPS_1 and the destination address identification IPS_2;
    if $IPS_1 \notin V$ **then**
      Add a vertex with IPS_1 to V;
    end
    if $IPS_2 \notin V$ **then**
      Add a vertex with IPS_2 to V;
    end
    if $(IPS_1, IPS_2) \in D$ **then**
      $D(IPS_1, IPS_2) = D(IPS_1, IPS_2) + 1$
    else
      $D(IPS_1, IPS_2) = 1$
    end
  end
  for $(d_l, d_r) \in D$ **do**
    $E.append((d_l, d_r))$ `/* Add an edge between` d_l `and` d_r `to` E `*/`
  end
  for $v \in V_i$, $f \in F$ **do**
    update $v = (v, f)$ `/* Aggregate the feature` f `of node` v `by identification */`
  end
  G.append((E, V))
end

3.2 GAT Model Construction

Graph Neural Networks and Graph Classification Tasks. Graph neural networks are a series of machine learning techniques for analyzing graph data. Due to the mighty expressive power of GNN, the research of graph neural networks has gradually become a hotspot in recent years [19]. The graph convolutional neural network (GCN) [20] is the most classical graph neural network, which generates embeddings representation of nodes by aggregating features of neighbors. GAT [3] is an improved version of the GCN. It extends the basic features of nodes through the attention mechanism.

This paper transforms the traffic classification problem into a graph classification problem by traffic behavior graph construction. The graph classification task is defined as follows. We define a set of labeled graph datasets as $H = (G, L)$, where G is a list of graphs and L is a list of graph labels. The graph classification task aims to learn a mapping function ϕ on the dataset H for predicting the unlabeled graph sample labels.

Graph Attention Mechanism. GAT adds the attention mechanism to GCN. During the aggregation process of the GCN, the feature weights of all adjacent nodes are equal. In contrast, GAT uses the weighted representations of neighbor nodes to update nodes. The calculation process of the attention score is shown in Fig. 3. For a graph G with n nodes, the initial features of n nodes can be expressed as $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_n\}$, $\mathbf{h}_i \in \mathbb{R}^F, \quad i = 1, 2, \cdots, n$. F is the initial feature dimension of each node. For the central node i, first, through a shared weight matrix W of $F * F'$, the feature transformation of node i and its neighbor j is performed, and F' is the new node feature dimension. Then we calculate the attention scores of nodes i and j and obtain the final attention coefficient through the softmax function. This process can be expressed as follows.

$$\mathbf{z}_i = W\mathbf{h}_i \tag{1}$$

$$e_{ij} = \text{LeakeyReLU}\left(\mathbf{a}^T\left(\mathbf{z}_i \| \mathbf{z}_j\right)\right) \tag{2}$$

$$\alpha_{ij} = \text{softmax}_j\left(e_{ij}\right) = \frac{\exp\left(e_{ij}\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(e_{ik}\right)} \tag{3}$$

In Eq. 2, $\|$ represents the vector splicing operation, and $\mathbf{a}$ represents the attention vector, which is used to convert the vector into a scalar value. $(\cdot)^T$ represents the vector transpose operation, e_{ij} represents the attention score. In Eq. 3, α_{ij} represents the attention coefficient. $\mathcal{N}_i$ represents all first-order neighbors of node i and includes i itself. The representation of the central node can be updated with the weighted sum of neighbor features, as shown in Eq. 4.

$$\mathbf{h}_i' = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}\mathbf{z}_j\right) \tag{4}$$

In Eq. 4, σ is the nonlinear activation function, and $\mathbf{z}_j$ is the vector representation of node j after feature transformation.

Besides, GAT uses the multi-head attention mechanism to learn node features of different dimensions. Figure 4 describes the process of using attention heads $K = 3$. Lines of different styles represent different dimensions of graph attention calculations. Finally, GAT combines the results of K times to obtain the final

vector representation of this node $\mathbf{h}_i''$. The formula of the multi-head attention mechanism is as follows.

$$\mathbf{h}_i'' = \|_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{z}_j^k\right) \tag{5}$$

GAT adaptively learns the weights of different neighbor nodes from different dimensions, which enhances the expressive ability of the graph neural network.

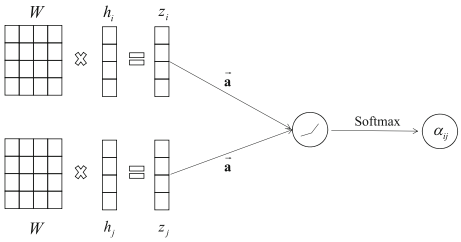

Fig. 3. Attention score calculation

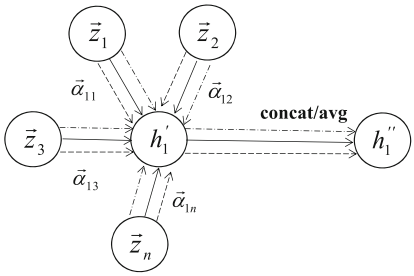

Fig. 4. Multi-head attention mechanism

Architecture of VT-GAT Model. As shown in Fig. 5, we obtained the optimal model structure after parameter optimization. The first two layers of the VT-GAT are two consistent-shaped graph attention convolution layers. Both layers have sixty-four hidden layer cells, eight attention heads, and Relu activation functions. Moreover, they are separated by a dropout layer with a dropout rate of 0.5. After two graph attention convolution layers is a global average pooling layer. Finally, there is a linear layer of size 100. The activation function of the output node is the LogSoftmax function. LogSoftmax function can speed up the convergence, improve data stability and prevent data overflow.

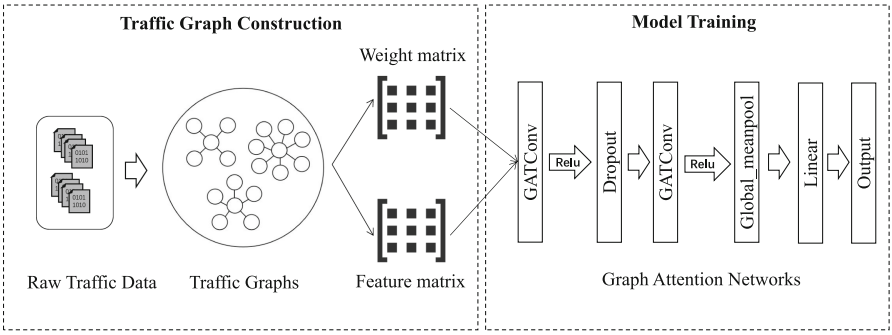

Fig. 5. Architecture of VT-GAT for VPN encrypted traffic classification

A complete VPN traffic classification process is as follows.

Step1: Input a PCAP file containing raw traffic. This file contains multiple packet-level traffic. Then we slice and dice the raw traffic into streams according to quintuple information (source IP, source port, destination IP, destination port, and IP protocol).

Step2: Select all packet and flow level data within a specific sliding time window to build the traffic behavior graphs.

Step3: Input the data into the VT-GAT model for training and predicting.

Step4: VT-GAT outputs a label to the traffic behavior graph composed of flows within each time window. This label represents the type of VPN application for the multiclassification task and whether the flow is VPN application traffic for the binary classification task.

4 Experiment Setup and Implementation

In this section, we design five VPN encrypted traffic classification experiments to demonstrate the applicability and efficiency of VT-GAT. We compare VT-GAT with other models and explore its robustness.

4.1 Experimental Setup

Dataset. The current dataset commonly used in VPN traffic classification is mainly ISCX VPN-nonVPN Dataset 2016 [17], but this dataset is too obsolete. Due to the continuous updates of technologies such as encryption technology and CDN technology in recent years, the traffic data in the ISCX VPN-nonVPN Dataset 2016 significantly differs from the current traffic characteristics. Therefore, we additionally choose the Datacon21 dataset. Datacon21 dataset [21] is released in 2021 in the DataCon2021 Big Data Security Analytics Competition. It consists of a total of one hundred categories and 218,168 flow samples. The traffic of this dataset is generated by automatically visiting the list of websites in the windows platform using the selenium library. The same encrypted proxy

software is used during the visits, and the traffic is captured simultaneously using Tshark. We process the dataset according to the method described in Chap. 3.1 and make the processed graph dataset publicly available on the Anonymous GitHub[2].

Metrics. Accuracy, Recall, Precision, and F1-Score are common evaluation metrics used in data classification tasks. Moreover, Micro Metric and Macro Metric are two commonly used evaluation metrics in multi-classification tasks. Micro Metric is directly calculated using the overall sample, and its results may be influenced by categories with a high number of samples. Macro Metric is calculated by category, and rare categories may influence its results. Because Accuracy is equal to each micro-metric (Micro-F1, Micro-Recall, Micro-Precision) in a multi-classification task, we choose the Macro method for Recall, Precision, and F1-Score in this paper. Assuming that there are n categories of samples $0, 1, 2, \cdots, N$ in the dataset, we need to calculate Precision and Recall for each category and then average them to obtain Macro-Precision and Macro-Recall. We then calculate the F1-Score for each category and average them to obtain the Macro-F1. Equations 6, 7, 8, 9 and 10 are the calculation method of Macro Metric for Accuracy, Precision, Recall, and F1-Score, respectively.

– Accuracy :

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{6}$$

– Precision :

$$Precision_i = \frac{TP_i}{TP_i + FP_i}, Precision = \frac{1}{n} \times \sum_{i=1}^{n} Precision_i \tag{7}$$

– Recall:

$$Recall_i = \frac{TP_i}{TP_i + FN_i}, Recall = \frac{1}{n} \times \sum_{i=1}^{n} Recall_i \tag{8}$$

– F1-Score:

$$F1\text{-}Score_i = 2 \times \frac{\text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}, \quad i = 0, 1, 2, \cdots, N \tag{9}$$

$$F1\text{-}Score = \frac{1}{n} \times \sum_{i=1}^{N} F1\text{-}Score_i \tag{10}$$

In Eqs. 6, 7, and 8, TP refers to the number of samples whose actual and predicted results are both positive. TN refers to the number of samples whose actual and predicted values are negative. FP refers to the number of samples that are predicted to be positive but whose actual value is negative. FN represents the number of samples in which the actual value is positive, but the predicted value is negative.

[2] https://anonymous.4open.science/r/VPN_Traffic_Graph_Dataset-EDA0.

Experimental Settings. VT-GAT is implemented in Python 3.8 based on PyTorch Geometric[3]. Other comparison models are implemented by Keras and sklearn. The experiments are carried out on a Ubuntu server with Intel(R) Xeon(R) Silver 4110 8-core processor @ 2.10 GHz, 128 GB of RAM, and three NVIDIA TITAN XPs GPUs, each having 12 GB of memory.

Baseline Models. To fully evaluate the performance of VT-GAT, we select six classical machine learning methods (SVM, KNN, NaiveBayes, Decision Tree, Random Forest, Logistic Regression) and three deep learning methods (DNN, CNN, LSTM) [22] for comparison. Although these machine learning models were proposed relatively early, these models have been fully verified in traffic classification tasks [1,23], and are currently widely used baseline models in traffic classification. Due to the specificity of graph classification methods, we concatenate all flows of a single graph sample into one flow sample in the dataset of the compared models. This setup ensures that the number of classification samples for the other methods and VT-GAT is identical, thus ensuring fairness in the comparison experiments. Besides, we also choose the recently proposed CLD-Net model [22] for comparative experiments with our model.

Hyperparameter Settings. In this experiment, the training and test sets are randomly divided in a ratio of 8:2. The sample sizes of the training and test sets are 174,534 and 43,634, respectively. We train our model and deep learning models using a maximum of 500 training epochs and an Adam optimizer with a learning rate of 10^{-4}. The activation function for all models is Relu. The DNN model consists of three fully connected layers of size 1024, each followed by a BatchNormalization layer. Finally, there is a fully connected layer of size 100. The LSTM model starts with an LSTM layer of size 256, followed by three fully connected layers of size 128, 64, and 100. The CNN model consists of two 1D convolutional layers of size (32, 7). There are two MaxPooling layers after the convolutional layer. Finally, there is a fully connected layer of size 100. The sliding window size is 300 s, and the sliding interval is 100 s. In order to ensure a fair comparison, all methods are parameterized to the best effect.

4.2 Analysis of Results

Comparison with Classical Machine Learning Algorithms. In this section, we compare the VT-GAT model with six classical machine learning algorithms.

In Table 2, we show the performance comparison results between VT-GAT and other algorithms. The results show that VT-GAT outperforms the other algorithms in all metrics. Due to the abnormal effect of SVM and NaiveBayes algorithms, it is evident that these two algorithms are not suitable for VPN traffic classification. So we do not include these two algorithms in comparison with

[3] https://github.com/pyg-team/pytorch_geometric.

Table 2. Performance comparison of VT-GAT and machine learning algorithms

Model	Accuracy	Precision	Recall	F1-Score
SVM	0.0443	0.1394	0.0468	0.0219
KNN	0.4357	0.4274	0.3620	0.3613
NaiveBayes	0.0855	0.1504	0.0691	0.0343
Decision Tree	0.9675	0.9478	0.9426	0.9448
Random Forest	0.9817	0.9747	0.9611	0.9666
Logistic Regression	0.7253	0.6634	0.6420	0.6506
VT-GAT	**0.9986**	**0.9977**	**0.9961**	**0.9968**

VT-GAT. Compared to the worst-performing algorithm (KNN), VT-GAT improved the different classification metrics by 56.29%–63.55%. Even compared to the best performing algorithm (Random Forest), VT-GAT improved all metrics by 1.69%–3.5%. These results indicate that the construction of traffic graphs can significantly help in VPN traffic classification. It is also clear from the results that tree-based models generally perform better than other machine learning models. This result is because flow-level traffic data is essentially tabular data. The tree-based models generally performed better in the task of classifying tabular data. The experimental results in this section also follow the findings of Shwartz-Ziv et al. [24]. It can be seen from the results that some algorithms (KNN, SVM, Naive-Bayes) perform very poorly. This result is due to the encrypted properties of VPN traffic and the only one flow problem that loses many effective features that can be used for classification. These algorithms cannot find practical features among the existing features that support their classification. These reasons lead to severe overfitting of these algorithms, resulting in poor classification results.

Comparison with Deep Learning Algorithms. In this section, we compare the VT-GAT with deep learning algorithms (DNN, CNN, LSTM), and the results are shown in Table 3.

Table 3. Performance comparison of VT-GAT and deep learning algorithms

Model	Accuracy	Precision	Recall	F1-Score
DNN	0.9380	0.9203	0.9109	0.9149
CNN	0.8490	0.8162	0.8055	0.8070
LSTM	0.8649	0.8292	0.8097	0.8147
VT-GAT	**0.9986**	**0.9977**	**0.9961**	**0.9968**

As can be seen from the results, VT-GAT improves by 6.06%–19.06% in all metrics compared to other deep learning algorithms. Deep learning algorithms use only the spatio-temporal features of the traffic. However, VT-GAT

combines graph behavioral features with spatio-temporal features. By comparison, we demonstrate the effectiveness of VT-GAT in introducing behavioral features of traffic graphs. Among the various deep learning algorithms, the DNN is more effective than the others because flow-level traffic features are closer to tabular data. There are partial spatio-temporal correlations between the different feature attributes, which can be learned by the time-series-based LSTM and the spatial-based CNN. These correlations, however, are not very close. Therefore, the performance of DNN that is more suitable for learning tabular data is the best. In addition, Random Forest outperforms all deep learning algorithms. This result is in line with the No Free Lunch Theorem [25]. We can conclude that the tree model (Random Forest, Decision tree) is the most suitable flow-based method for VPN traffic classification. However, among all types of VPN traffic classification methods, VT-GAT is the most superior method. It can be seen from the comparison of different indicators that the F1-Score of the DNN model decreases more than the accuracy. This is because the F1-Score is determined by the recall and precision, and its influencing factors are more than the accuracy, which leads to the instability of the DNN model. Decision trees and Random Forests have similar magnitudes of change. And VT-GAT has the smallest changes in different indicators, which also shows that VT-GAT is more robust than other baseline models.

Performance with Different Number of Categories. The number of websites and applications is exploding with Internet technology's development. To verify the robustness of the VT-GAT, we design experiments with different numbers of categories (10, 30, 50, 80, and 100, respectively). This experiment compares the best-performing deep learning algorithm (DNN) and the two best-performing machine learning algorithms (Random Forest and Decision Tree). We randomly select a fixed number of categories for the experiments, and the results are shown in Fig. 6 and Fig. 7.

Experimental results show that most models perform well when the number of categories is small. As the number of categories gradually increases, the performance of Random Forest, Decision Tree, and DNN show a significant decrease by more than 1.5%. This phenomenon is expected as the increase in categories increases the similarity between samples, thus making the classification task more difficult. In contrast, the classification performance of VT-GAT remains almost unchanged. The overall variation of our model is less than 0.5% and remains at a high level, demonstrating the better robustness of VT-GAT. Furthermore, VT-GAT has the potential to be applied to real-world classification tasks for massive VPN traffic categories.

Performance Comparison of VT-GAT and CLD-Net. In order to further verify the application effect of VT-GAT in practical scenarios, we selected the recently proposed CLD-Net model [22] to verify it on the ISCX VPN-nonVPN Dataset 2016 [17]. The selected task is a binary classification task, that is, to

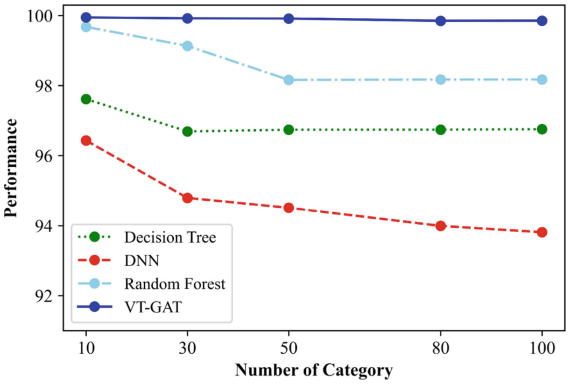

Fig. 6. Accuracy with different number of categories

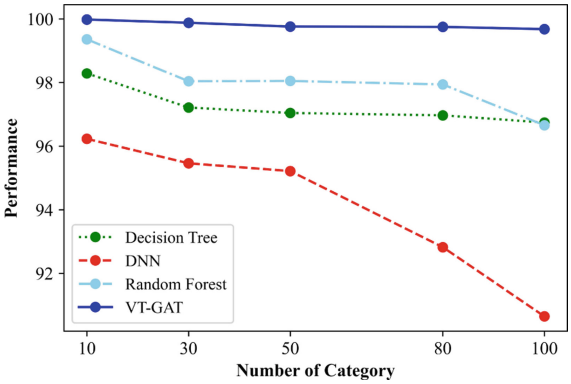

Fig. 7. F1-Score with different number of categories

determine whether a given network flow is traffic generated by a VPN application. The experimental results are shown in Fig. 8.

The experimental results show that the VT-GAT model is 1.38%–2.79% better than the CLD-Net model (Accuracy: 2.79%, Precision: 1.38%, Recall: 2.11%, F1-Score: 2.01%). It can be seen from the above results that the VT-GAT model has good applicability in both the VPN traffic identification task and the VPN application identification task. By relying on the solid identification ability of VT-GAT, it helps to help trace the type of VPN application that hackers use during the intrusion process, which is of great significance in identifying the attacker's identity.

Assessment of Execution Time. In order to better evaluate the performance of the model, we conduct a time performance comparison experiment. Since the training part of the supervised model can be performed in advance, we do not compare the model training time. In addition, most of the time cost of

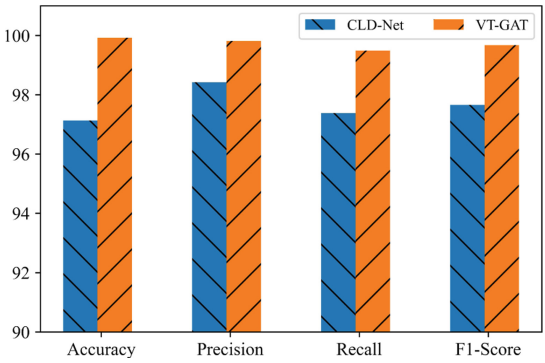

Fig. 8. Comparison of CLD-Net and VT-GAT

the model inference process is in the data feature preprocessing part, and the actual inference time (excluding the preprocessing process) of all methods can be completed within 30 s. The total inference time for flow-based methods is similar. Therefore, we only select the graph-based model VT-GAT for comparison with the flow-based model Random Forest. The experimental results are shown in Fig. 9.

Results show that VT-GAT can complete the prediction of thousand-level flow data within 3 min. The statistics show that the inference time consumption of VT-GAT in all orders of magnitude is about 3.4–7.4 times that of the Random Forest. Since VT-GAT needs to construct the traffic graph data, it has more time overhead than flow-based methods. Although the classification accuracy of Random Forest is not as good as that of VT-GAT, it runs very efficiently. As the size of the data increases, the time overhead for the feature extraction part also

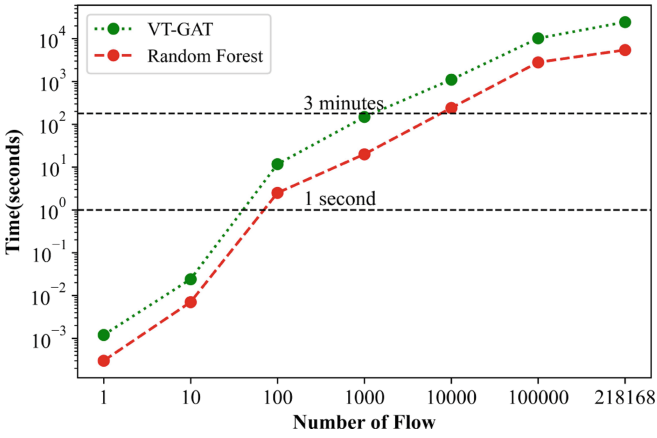

Fig. 9. Inference time comparison of different algorithms

reaches hourly levels. It can be considered to reduce the time cost of the data preprocessing part by parallelizing. Random Forest and VT-GAT have taken approximately 89 min and 406 min to predict the entire dataset (about 3.7 GB, including training set and test set).

5 Discussion

VT-GAT solves the problem that the existing VPN traffic classification techniques are generally ineffective due to only one flow problem. However, there are some shortcomings. First, our model can only identify the categories in the training data due to the supervised algorithm. It is almost impossible to collect the traffic of all websites or applications in a realistic environment. Some semi-supervised methods (e.g., FlowPrint [12]) can identify unknown categories, although their accuracy cannot match that of VT-GAT.

Secondly, during data collection, only one category of traffic is collected at the same time. In the real world, there may be multiple types of traffic generated at the same time. In the graph classification, the same graph may contain traffic data generated by different applications. To alleviate this problem, we use sliding windows to find the splitting points of traffic between different categories. However, the performance of VT-GAT may be degraded when there are no obvious traffic splitting points.

Our model will be further enhanced to address the above problem in future work. Graph contrast learning [18] is a self-supervised learning algorithm for graph data, which can solve the problem of graph data with missing labels or complex labeling. By introducing graph contrast learning, we can solve the problem that VT-GAT cannot identify unseen categories. In addition, during the traffic interaction, a flow is intuitively seen as an edge connecting two nodes. VT-GAT transforms edge features to node features while building the graph and then transforms the edge classification problem into a graph classification problem. We can modify the structure of the graph neural network to implement edge classification directly. In the edge classification task, an edge has a label. It is reasonable to include different labels within a single graph. So there is no need to find traffic splitting points.

6 Conclusion

There is a problem of only one flow in VPN traffic, which seriously affects the classification effectiveness of existing techniques. We propose a graph-based VPN traffic classification model VT-GAT to solve this problem. Moreover, we develop a traffic graph construction method to handle VPN traffic. The traffic graph data generated by VT-GAT contains the interaction behavior relationships between nodes, which is crucial for VPN traffic classification. Compared to existing techniques, our model supplements the traffic interaction behavior features based on the original spatio-temporal features and automatically learns the traffic graph differences among different categories via GAT. In addition, we release a traffic

graph dataset that is useful for classifying VPN traffic. The results of comparative experiments on the Datacon21 dataset show that VT-GAT significantly improves all metrics compared to existing machine learning and deep learning methods (Accuracy: 1.69%–56.29%, Precision: 2.3%–57.03%, Recall: 3.5%–63.41% and F1-Score: 3.02%–63.55%). These results illustrate the great potential of GNN-based traffic classification methods for the VPN traffic classification.

In future work, we plan to use self-supervised learning methods to enhance our model.

Acknowledgements. We thank the anonymous reviewers for their insightful comments. This work was supported in part by the National Key Research and Development Program of China under Grant No. 2019YFB1005205.

References

1. Xie, J., Li, S., Yun, X., Zhang, Y., Chang, P.: HSTF-model: an http-based trojan detection model via the hierarchical spatio-temporal features of traffics. Comput. Secur. **96**, 101923 (2020)
2. Chen, H.Y., Lin, T.N.: The challenge of only one flow problem for traffic classification in identity obfuscation environments. IEEE Access **9**, 84110–84121 (2021)
3. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
4. Papadogiannaki, E., Ioannidis, S.: A survey on encrypted network traffic analysis applications, techniques, and countermeasures. ACM Comput. Surv. (CSUR) **54**(6), 1–35 (2021)
5. Finsterbusch, M., Richter, C., Rocha, E., Muller, J.A., Hanssgen, K.: A survey of payload-based traffic classification approaches. IEEE Commun. Surv. Tutor. **16**(2), 1135–1156 (2013)
6. Deri, L., Martinelli, M., Bujlow, T., Cardigliano, A.: NDPI: open-source high-speed deep packet inspection. In: 2014 International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 617–622. IEEE (2014)
7. Papadogiannaki, E., Halevidis, C., Akritidis, P., Koromilas, L.: OTTer: a scalable high-resolution encrypted traffic identification engine. In: Bailey, M., Holz, T., Stamatogiannakis, M., Ioannidis, S. (eds.) RAID 2018. LNCS, vol. 11050, pp. 315–334. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00470-5_15
8. Ren, Q., Yang, C., Ma, J.: App identification based on encrypted multi-smartphone sources traffic fingerprints. Comput. Netw. **201**, 108590 (2021)
9. Taylor, V.F., Spolaor, R., Conti, M., Martinovic, I.: Robust smartphone app identification via encrypted network traffic analysis. IEEE Trans. Inf. Forensics Secur. **13**(1), 63–78 (2017)
10. Taylor, V.F., Spolaor, R., Conti, M., Martinovic, I.: AppScanner: automatic fingerprinting of smartphone apps from encrypted network traffic. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 439–454. IEEE (2016)
11. Wang, S., Yang, C., Guo, G., Chen, M., Ma, J.: SSAPPIDENTIFY: a robust system identifies application over shadowsocks's traffic. Comput. Netw. **203**, 108659 (2022)
12. van Ede, T., et al.: Flowprint: semi-supervised mobile-app fingerprinting on encrypted network traffic. In: Network and Distributed System Security Symposium (NDSS), vol. 27 (2020)

13. Rezaei, S., Liu, X.: Deep learning for encrypted traffic classification: an overview. IEEE Commun. Mag. **57**(5), 76–81 (2019)
14. Xie, G., Li, Q., Jiang, Y.: Self-attentive deep learning method for online traffic classification and its interpretability. Comput. Netw. **196**, 108267 (2021)
15. Guo, L., Wu, Q., Liu, S., Duan, M., Li, H., Sun, J.: Deep learning-based real-time VPN encrypted traffic identification methods. J. Real-Time Image Proc. **17**(1), 103–114 (2020)
16. Lashkari, A.H., Draper-Gil, G., Mamun, M.S.I., Ghorbani, A.A.: Characterization of tor traffic using time based features. In: ICISSp, pp. 253–262 (2017)
17. Draper-Gil, G., Lashkari, A.H., Mamun, M.S.I., Ghorbani, A.A.: Characterization of encrypted and VPN traffic using time-related. In: Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP), pp. 407–414 (2016)
18. Zeng, J., Xie, P.: Contrastive self-supervised learning for graph classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 10824–10832 (2021)
19. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
20. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
21. DataCon-Community: Datacon open dataset - datacon2021 - encrypted proxy traffic dataset track open dataset, 24 December 2021. [Online; Accessed 14 June 2022]
22. Hu, X., Gu, C., Wei, F.: CLD-net: a network combining CNN and LSTM for internet encrypted traffic classification. In: Security and Communication Networks 2021 (2021)
23. Lotfollahi, M., Jafari Siavoshani, M., Shirali Hossein Zade, R., Saberian, M.: Deep packet: a novel approach for encrypted traffic classification using deep learning. Soft Comput. **24**(3), 1999–2012 (2020)
24. Shwartz-Ziv, R., Armon, A.: Tabular data: deep learning is not all you need. Inf. Fusion **81**, 84–90 (2022)
25. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)