Hindawi
Security and Communication Networks
Volume 2022, Article ID 4274139, 11 pages
https://doi.org/10.1155/2022/4274139

*Research Article*

# GCN-ETA: High-Efficiency Encrypted Malicious Traffic Detection

**Juan Zheng**,[1] **Zhiyong Zeng**,[2] **and Tao Feng**[2]

[1]*School of Statistics and Mathematics, Yunnan University of Finance and Economics, Kunming 650221, China*
[2]*School of Information Science, Yunnan University of Finance and Economics, Kunming 650221, China*

Correspondence should be addressed to Tao Feng; vonpower@ynufe.edu.cn

Received 2 November 2021; Revised 15 December 2021; Accepted 4 January 2022; Published 22 January 2022

Academic Editor: Leandros Maglaras



Encrypted network traffic is the principal foundation of secure network communication, and it can help ensure the privacy and integrity of confidential information. However, it hides the characteristics of the data, increases the difficulty of detecting malicious traffic, and protects such malicious behavior. Therefore, encryption alone cannot fundamentally guarantee information security. It is also necessary to monitor traffic to detect malicious actions. At present, the more commonly used traffic classification methods are the method based on statistical features and the method based on graphs. However, these two methods are not always reliable when they are applied to the problem of encrypted malicious traffic detection due to their limitations. The former only focuses on the internal information of the network flow itself and ignores the external connections between the network flows. The latter is just the opposite. This paper proposes an encrypted malicious traffic detection method based on a graph convolutional network (GCN) called GCN-ETA, which considers the statistical features (internal information) of network flows and the structural information (external connections) between them. GCN-ETA consists of two parts: a feature extractor that uses an improved GCN and a classifier that uses a decision tree. Improving on the traditional GCN, the effect and speed of encrypted malicious traffic detection can be effectively improved and the deployment of the detection model in the real environment is increased, which provides a reference for the application of GCN in similar scenarios. This method has achieved excellent performance in experiments using real-world encrypted network traffic data for malicious traffic detection, with the accuracy, AUC, and $F_1$-score exceeding 98% and more than 1,300 flows detected per second.

## 1. Introduction

Network traffic classification technology is receiving increasing attention because of the quality of service (QoS) and network security issues. Network traffic classification is the basic role of network management. It can identify distinct protocols and applications in a network and is widely used such as for QoS and anomaly detection. Traffic classification is the core component of emerging QoS support products and automated QoS architecture. However, because of continuous network expansion and innovation in communication technology, network traffic presents the characteristics of complexity and diversification. Information security is ensured through the encryption of data packets in network traffic, and over 90% of network traffic is now encrypted [1]. While encryption can ensure the confidentiality and integrity of information, it can hide the characteristics of data, increase the difficulty of detecting malicious traffic, and protect such behavior. An attacker cannot only guess a user's access trace with high likelihood but can exploit encryption to hide the attack and avoid detection [2]. Zscaler, a cloud security company, predicts that attacks on encrypted traffic that bypass traditional security controls will increase by 260% in the next five years [3]. Therefore, encryption alone cannot fundamentally guarantee information security. It is also necessary to monitor traffic to detect malicious actions.

Malicious traffic detection is essentially a traffic classification problem whose methods are based on ports, payloads, statistical features, and graphs. The traditional port-based method relies on checking standard ports used by applications. However, it is not always reliable because not all current applications use standard ports [4]. Deep packet inspection (DPI) methods based on payload have

become popular alternatives [5, 6]. DPI can identify an application by verifying the signature in the packet payload, thus avoiding the problem of a dynamic port. However, payload-based methods cannot identify encrypted traffic, and checking the payload can give rise to privacy issues. Methods based on statistical features calculate the features of data packets and input them to machine learning algorithms such as decision tree (DT) and support vector machine (SVM) for classification [7–9]. Because analysis based on ordinary machine learning algorithms depends on feature engineering, researchers have considered deep learning algorithms such as the convolutional neural network (CNN) [10, 11] and recurrent neural network (RNN) [12]. However, these methods focus on the internal information of network traffic, ignoring external connections between network flows. Graph-based methods classify network traffic based on the host behavior or a local structure in the traffic trajectory graph and can solve problems of dynamic ports and traffic encryption.

Much work has demonstrated that the graph structure can be used to classify traffic [13, 14], but the graph-based method only focuses on the external connections between network flows. The graph neural network has attracted much attention [15, 16], as it allows attention to be focused on information of the structure, nodes, and edges. As far as the author knows, there is currently little work on the classification of encrypted traffic with graph neural networks. Existing work only uses graph neural networks to classify normal encrypted traffic into different types, and its application in the detection of encrypted malicious traffic has not yet been made. In this work, we propose a high-efficiency encryption malicious traffic detection method based on a graph convolutional network (GCN). The powerful performance of GCN is that it can train node information and structural features between nodes at the same time. In the task of detecting encrypted malicious traffic, considering the structural characteristics will improve the detection effect, but it will also greatly reduce the detection speed. This will cause the detection model to fail to meet the deployment requirements in the real environment. We improved the GCN through a detailed analysis of the GCN model architecture, which reduced the training time of the GCN while ensuring the performance of the GCN. The improved GCN is used as a feature extractor to simultaneously train traffic statistical features and the graph structure of the traffic trajectory, and a DT is used as a classifier to detect encrypted malicious traffic. Experiments were carried out on the actual encrypted network traffic, and the results confirmed that our method effectively improves the effect and speed of the encrypted malicious traffic detection model.

The main contributions of this work are as follows:

(i) We propose an encrypted malicious traffic detection method based on GCN that considers both the statistical features of network flows and structural information between them, which strengthens people's understanding of the problem of encrypted malicious traffic detection

(ii) To improve the traditional GCN, the effect and speed of encrypted malicious traffic detection model are effectively improved, and the deployment of the detection model in the real environment is increased, which provides a reference for the application of GCN in similar scenarios

(iii) Our model achieves excellent results on the real-world network traffic data for encrypted malicious traffic detection, and it outperforms several state-of-the-art methods

The rest of this article is organized as follows: related work is reviewed in Section 2. Section 3 introduces the notation and problem definition, background, and the proposed method, named Encrypted Malicious Traffic Detection Based on Graph Convolution Network (GCN-ETA). Experiments are discussed in Section 4 to demonstrate that this method has an excellent performance in encrypted malicious traffic detection. Our conclusions are summarized in Section 5.

## 2. Related Work

Much research has addressed the use of internal features (such as the port number) to classify network traffic. The wide use of encryption has diminished the effect of traditional network traffic classification methods [17]. In recent years, there have been frequent attacks on encrypted traffic. Therefore, studying how to classify encrypted traffic and strengthening network security protection is an urgent problem to be solved. In fact, although the encryption technology hides the payload of the data packet, side-channel data such as the size and direction of the packet can still be obtained from the encrypted connection. To solve this problem, some researchers have combined statistical features (side-channel data like packet size and direction) of traffic with machine learning to classify network traffic [18, 19]. Taylor et al. [20] used a new machine learning strategy to identify similar encrypted network traffic between applications. The general workflow of this type of method is as follows: first, manually design traffic statistical features, then extract and select these features from the original traffic, and finally, use a classifier (traditional machine learning classifier [21, 22] or deep learning classifier [23]) to classify the traffic. In addition to such methods, other researchers also use the powerful feature extraction capabilities of deep learning methods to directly classify the original traffic [24–26]. Liu et al. [25] applied RNN to the problem of traffic encryption classification and proposed the FS-Net model. However, FS-Net still needs to preprocess the original traffic. Bahaa et al. [26] used word embedding layer, convolutional neural network, and bidirectional recurrent neural network to directly classify the original traffic and proposed a new DPI method (nnDPI). nnDPI does not require experts to extract features related to network traffic, and its performance has been proved to be superior to other traffic classification methods of the same type [24].

The above methods only focus on the internal information of the network flow itself and ignore the external connection between the network flows. Some researchers have examined network flows based on their external links and classified them based on graphs (host behavior). Such methods need to build a traffic trajectory graph, where nodes represent Internet Protocol (IP) hosts, edges represent network traffic, and labeled edges are used to infer unlabeled edges. Graph-based methods have been successfully implemented for network traffic representation [13, 14]. Gallagher et al. [13] have found that the network traffic at the application layer has link homogeneity. A network trajectory graph is constructed, with nodes as IP hosts and edges as network flows. Labels are inferred for unlabeled flows based on the application labels of neighboring flows. Graph-based methods may be referred to as host behavior-based methods in many other pieces of research [27, 28]. Karagiannis et al. [28] classified network traffic by analyzing host behavior in the transport layer. By comparing behavior information with stored application signatures, it can identify the applications a host is running but cannot identify their subtypes.

In general, the most commonly used methods of the four types of traffic classification are methods based on statistical features and methods based on graphs. However, the detection of encrypted malicious traffic based only on statistical features or only based on graphics still has limitations. The former only focuses on the internal information of the network flow itself and ignores the external connections between the network flows. The latter only focuses on the external connections between the network flows and ignores the internal information of the network flows themselves. The proposal of methods such as GCN [29] has inspired researchers to consider a flow's information and the external connection between flows when classifying traffic. Zheng et al. [30] applied GCN to network traffic classification for the first time and proposed the model GCN-TC. Shuang et al. [31] classified encrypted traffic, using GCN to distinguish virtual private network (VPN) and conventional encrypted traffic, and classified traffic into categories such as files and emails. Researchers have applied GCN to common traffic classification tasks (such as distinguishing protocols used for traffic or VPN traffic) but have not yet considered encrypted malicious traffic detection.

## 3. Methods

In this section, we first introduce the symbols and problem definition, including the detailed description of the research problem and the definition of the traffic trajectory graph. Secondly, we introduce GCN and our proposed method, GCN-ETA, where GCN-ETA is composed of the improved GCN and the DT algorithm.

*3.1. Symbols and Problem Definition.* A network flow is a unidirectional flow sent from a source IP address to a destination IP address within a period of time. All packets have the same five-tuple of the source port number, destination port number, protocol number, and source and destination IP addresses. Graph-based methods use traffic trajectory graphs to classify network flows. Normally, nodes in these graphs are IP hosts, and edges are network flows between them. A trajectory graph is utilized to find link homogeneity. Flows with the same IP host may share a category, in which case network traffic classification becomes edge classification. To use the GCN framework, we use a new traffic trajectory graph [30], as shown on the right of Figure 1. In the new traffic trajectory graph, a node represents network flow, and an edge represents that nodes on both its ends share an IP address. Therefore, we convert the encrypted malicious traffic detection task into a malicious node detection problem by constructing this new traffic trajectory graph.

We define such a graph as $\mathcal{G} = (\mathcal{V}, A)$, where $\mathcal{V}$ is the vertex set consisting of nodes $\{v_1, \ldots, v_n\}$, and $A \in \mathbb{R}^{n\times n}$ is a symmetric (typically sparse) adjacency matrix. Element $a_{ij}$ of the adjacency matrix is the weight of the edge between nodes $v_i$ and $v_j$, and $a_{ij} = 0$ indicates that there is no edge connection between the two nodes. We define the degree matrix $D = \mathrm{diag}(d_1, \ldots, d_n)$ as a diagonal matrix whose each diagonal entry is the row-sum of the adjacency matrix $d_i = \sum_j a_{ij}$.

Each node $v_i$ in the graph has a corresponding $d$-dimensional feature vector $\mathbf{x}_i \in \mathbb{R}^d$. The eigenvectors of all $n$ nodes form a complete eigenmatrix $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^T \in \mathbb{R}^{n\times d}$. Each node belongs to one of $C$ the classes and can be labeled with a $C$-dimensional one-hot vector, $\mathbf{y}_i \in \{0, 1\}^C$.

According to the above definition, we assume $n$ network flows, each with $d$-dimensional features, flow feature matrix $X \in \mathbb{R}^{n\times d}$, and traffic adjacency matrix $A = (a_{ij})_{n\times n}$, where $a_{ij} = 1$ when there are flows $i$ and $j$ with shared IP hosts, and otherwise, $a_{ij} = 0$. As shown on the right of Figure 1, the neighbors of flow 1 include flows 2, 3, 4, and 5, so $a_{12} = a_{13} = a_{14} = a_{15} = 1$.

*3.2. GCN.* The GCN [29] is a multilayer neural network that can run directly on a graph. It guides the embedding vector of a node according to the attributes of neighbor nodes. It learns the new feature representation of each node, and uses this as the input of a linear classifier for node classification. The GCN can capture the information of first-order neighbors through a layer of convolution operations. When multiple GCN layers are stacked, information about a larger area can be integrated. For the $k$-th graph convolutional layer, the matrices $H^{(k-1)}$ and $H^{(k)}$ represent the input node and output node representation, respectively, of all nodes. Naturally, the initial node representation is the original input feature $H^{(0)} = X$, which represents the input of the first graph convolutional layer. For a single-layer GCN, the new $d'$-dimensional node feature matrix $H^{(1)} \in \mathbb{R}^{n\times d'}$ is as follows:

$$H^{(1)} = \mathrm{ReLU}\left(SX\theta^{(1)}\right). \tag{1}$$

The "normalized" adjacency matrix with added self-loop is as follows:

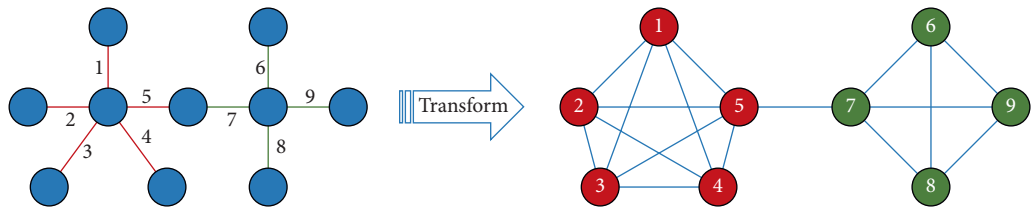


Figure 1: Traffic trajectory graph (index indicates network flow). (a): the node is the IP address, and the edge represents the network flow. (b): the node represents the network flow, and the edge represents that nodes on both ends have a common IP.

$$S = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}, \tag{2}$$

where $\tilde{A} = A + I$, with degree matrix $\tilde{D}$. $\theta^{(1)} \in \mathbb{R}^{d\times d'}$ is a weight matrix used for linear transformation of the smoothed hidden feature representation. As mentioned earlier, multiple GCN layers can be stacked to capture higher-order domain information:

$$H^{(k)} \leftarrow \mathrm{ReLU}\left(SH^{(k-1)}\theta^{(k)}\right). \tag{3}$$

For node classification, the last layer of the GCN uses the classifier to predict the label. Define the predicted category of $n$ nodes as $\hat{Y} = (\hat{y}_{ic})_{n\times C} \in \mathbb{R}^{n\times C}$, where $\hat{y}_{ic}$ represents the probability that node $i$ belongs to category $c$. The class prediction $\hat{Y}$ of a $K$-layer GCN can be written as follows:

$$\hat{Y}_{\mathrm{GCN}} = \mathrm{softmax}\left(SH^{(K-1)}\theta^{(K)}\right), \tag{4}$$

where $\mathrm{softmax}(\mathbf{x}) = \exp(\mathbf{x})/\sum_{c=1}^{C}\exp(x_c)$ acts as a normalizer across all classes.

Suppose we construct a two-layer ($K = 2$) GCN, the overall forward propagation formula is as follows:

$$\hat{Y} = f(X, A) = \mathrm{softmax}\left(S\mathrm{ReLU}\left(SX\theta^{(1)}\right)\theta^{(2)}\right), \tag{5}$$

where $\theta^{(2)} \in \mathbb{R}^{d'\times f}$. Finally, we calculate the cross entropy loss function for all labeled nodes:

$$\mathscr{L} = -\sum_{l\in\mathscr{Y}_L}\sum_{m=1}^{f} Y_{lm}\ln\hat{Y}_{lm}, \tag{6}$$

where $\mathscr{Y}_L$ is the set of node indices that have labels and $\theta^{(1)}$ and $\theta^{(2)}$ can be obtained by training using gradient descent.

*3.3. GCN-ETA.* GCN-ETA consists of a feature extractor and a classifier (see Figure 2). Firstly, the improved GCN is used to represent and learn the structural information and attribute information contained in the flow trajectory graph and to obtain the representation vector of each node in the graph. Then take the representation vector as input and use the DT classification algorithm to identify malicious traffic nodes and normal traffic nodes in the graph. Figure 3 shows the schematic layout of malicious traffic detection using GCN and GCN-ETA, respectively.

*3.3.1. Feature Extractor.* A GCN can act as a feature extractor to find a better feature representation of a sample, similar to CNN and RNN. Using a CNN or a similar model as a feature extractor trains the model to a certain state and takes its intermediate output as the input of downstream tasks [32, 33]. This takes much time. Considering the time spent on feature extraction, certain simplifications will minimize the time of this process when using a GCN model as a feature extractor.

The analysis of GCN reveals that the linear transformation weight matrix $\theta^{(k)}$ to be trained between layers is one of the main factors increasing the complexity and redundancy of the model. We assume the linear transformation between the GCN layers is not important, and most of the benefits of the model are explained by the local smoothing of the nodes. Therefore, we set the output size of each layer to be compatible with the input size and fix the linear transformation weight matrix $\theta^{(k)}$ to be trained between the layers as the identity matrix.

Similar to GCN, at the beginning of each layer of our feature extractor, the feature $h_i$ of each node $v_i$ must be averaged with the feature vector of its neighbors in the local domain:

$$\overline{\mathbf{h}}_i^{(k)} \leftarrow \frac{1}{d_i+1}\mathbf{h}_i^{(k-1)} + \sum_{j=1}^{n}\frac{a_{ij}}{\sqrt{(d_i+1)(d_j+1)}}\mathbf{h}_j^{(k-1)}. \tag{7}$$

We can update the entire graph as a simple matrix operation. The simultaneous update of all nodes in equation (7) can be summarized as a simple sparse matrix multiplication:

$$\overline{H}^{(k)} \leftarrow SH^{(k-1)}. \tag{8}$$

Intuitively, this step locally smooths the hidden representation of each node along the edge of the graph.

After local smoothing, the smoothed hidden feature representation for each GCN layer is linearly transformed by learning the weight matrix $\theta^{(k)}$. A nonlinear activation function such as ReLU is applied pointwise before outputting feature representation $H^{(k)}$. According to our assumption ($\theta^{(k)}$ is the identity matrix), to reduce the complexity of the model, the feature representation output of the $k$-th layer of our feature extractor is as follows:

$$H^{(k)} \leftarrow \mathrm{ReLU}\left(\overline{H}^{(k)}I\right) \leftarrow \mathrm{ReLU}\left(SH^{(k-1)}I\right), \tag{9}$$

where $H^{(0)} = X$, $S$ represents the "normalized" adjacency matrix with added self-loop, and $I$ is the identity matrix. The number of layers $K$ of the feature extractor is a self-defined hyperparameter, so the final output of the original node feature after the $K$-layer feature extractor is as follows:

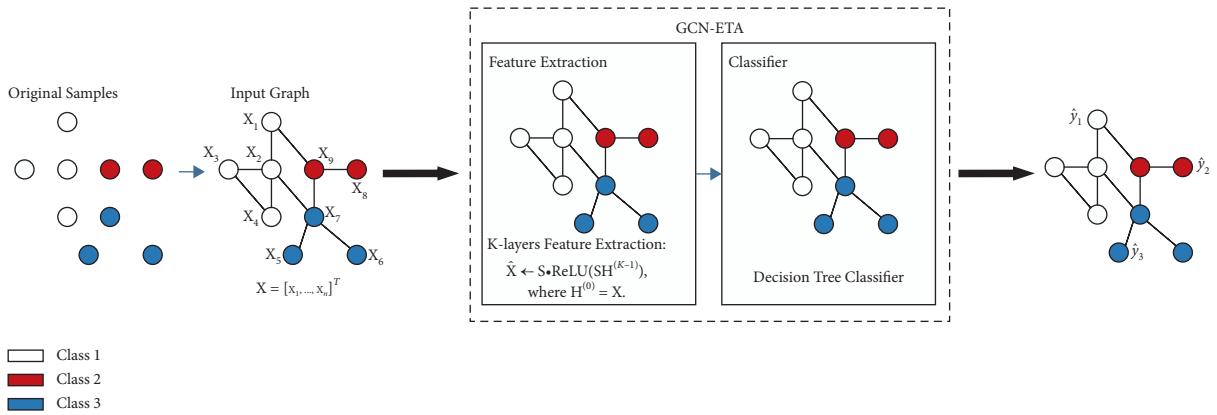


Figure 2: Model architecture.

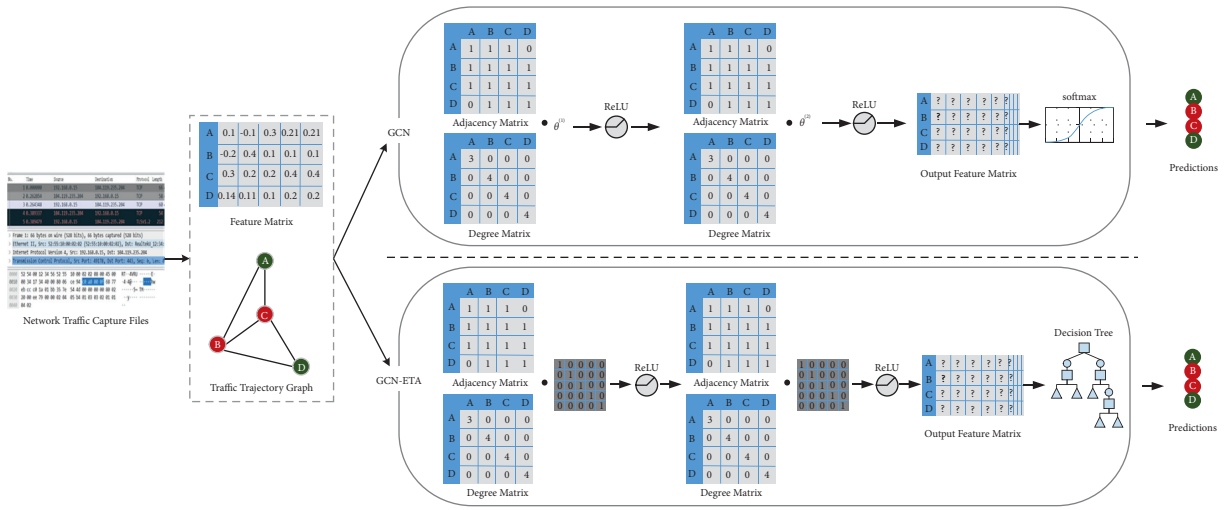


Figure 3: Schematic layout of malicious traffic detection using GCN and GCN-ETA, respectively (a two-layer structure is used as an example).

$$\hat{\mathbf{X}} = S \cdot \mathrm{ReLU}\left(SH^{(K-1)}I\right)I = S \cdot \mathrm{ReLU}\left(SH^{(K-1)}\right). \tag{10}$$

*3.3.2. Classifier.* Because the DT algorithm has the advantages of high efficiency, easy interpretation, and less time complexity, combined with our goal of achieving high-efficiency encrypted malicious traffic detection, the DT is selected as the classifier. Nodes processed by the feature extraction module are handed over to the classifier for identification.

The classification module uses a DT, a tree-structured inductive learning classifier, to create a classification model from unordered training data. Different from deep learning, a DT is a white-box model that can quickly identify and classify data. In order to generate a classification tree, the decision tree needs to find the best node and the best branching method. The measure of this "best" index is called impurity. There are two options for the impurity index, namely Entropy and Gini coefficient. In actual use, the effects of Entropy and Gini coefficient are basically the same, but calculating the Gini coefficient is faster than calculating Entropy. Therefore, we use the Gini coefficient as the impurity index to select the optimal partition feature. For a dataset $\mathbf{T}$, this is as follows:

$$\mathrm{Gini}(\mathbf{T}) = \sum_{c=1}^{C} p_c(1 - p_c) = 1 - \sum_{c=1}^{C} p_c^2, \tag{11}$$

where $C$ is the number of data categories, and $p_c$ is the probability that a sample point belongs to the $c$-th category. For binary classification, if the probability that a sample point belongs to the first class is $p$, then its Gini coefficient is as follows:

$$\mathrm{Gini}(\mathbf{T}) = 2p(1 - p). \tag{12}$$

Intuitively, $\mathrm{Gini}(\mathbf{T})$ reflects the probability that two samples with inconsistent categories are randomly selected from $\mathbf{T}$. Therefore, a smaller value indicates higher purity of $\mathbf{T}$, and a greater probability that the data belong to the same category.

## 4. Experimental Results and Analysis

We evaluate GCN-ETA (our code is published at https://github.com/zhengjuan1996/GCN-ETA) ($K = 2$) through experiments. We must determine whether our model can achieve a better, faster detection effect of encrypted malicious traffic. We use Pytorch to implement GCN-ETA, and its experimental platform configuration is shown in Table 1.

*4.1. Dataset.* We used a dataset from malware and normal software collected from February to June 2020 [34]. The Qianxin Technology Research Institute's Tianqiong Sandbox collected the traffic generated by running these applications. Black and white samples were encrypted traffic generated by malicious and normal software, respectively (all exe types). The traffic content consisted of TLS and SSL data packets generated on port 443.

The dataset contained 10,000 packet capture (pcap) data files, consisting of equal volumes of black and white traffic. Wireshark was used to split the file into the unidirectional flow (the flow with the same source port number, destination port number, protocol number, and source and destination IP addresses), as shown in Table 2. When dealing with imbalanced datasets, classification models will be restricted by the data distribution itself to learn more about the majority class and disregard the minority class. As the imbalance of data category distribution restricts models' recognition performance for a few categories of targets, we adjusted the ratio of black and white samples to 1 : 1 to avoid data imbalance.

We chose the five most basic single flow statistical features to represent network flows, as shown in Table 3.

*4.2. Baselines*

(i) LR: based on the statistical features of the network flow, logistic regression was used directly as the classifier

(ii) BernoulliNB: based on the statistical features of the network flow, BernoulliNB was used directly as the classifier

(iii) DT: based on the statistical features of the network flow, DT was used directly as the classifier

(iv) XGBoost: based on the statistical features of the network flow, XGBoost was used directly as the classifier

(v) nnDPI [26]: the classification is based on the byte information of the data packet of the original encrypted traffic

(vi) GCN-TC [30]: based on the statistical features of the network flow and the traffic trajectory graph, a 2-layer GCN was used as the classifier

(vii) GCN + DT: a 2-layer GCN was used to extract features of the network flow, and DT as the classifier

(viii) GCN + RF: a 2-layer GCN was used to extract features of the network flow and random forest as the classifier

(ix) GCN + XGBoost: a 2-layer GCN was used to extract features of the network flow, and XGBoost as the classifier

(x) GGCN + KNN: a 2-layer GCN was used to extract features of the network flow and k-nearest neighbors as the classifier

The machine learning classification algorithms used grid search to adjust parameters, and the deep learning models involved were trained to the optimal state.

*4.3. Experimental Evaluation Indices.* We evaluate our proposed method GCN-ETA from two perspectives, namely detection effect, and detection speed.

*4.3.1. Detection Effect.* Treating black samples (malicious traffic) as positive instances and white samples (normal traffic) as negative instances, the model detection effect was evaluated according to accuracy and the $F_1 -$ score:

$$\begin{aligned} \text{accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \\ \text{precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ \text{recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ F_1 - \text{score} &= 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \end{aligned} \tag{13}$$

Among them, TP represents the number of positive classes that are correctly predicted, TN represents the number of negative classes that are correctly predicted, FP represents the number of positive classes that are incorrectly predicted, and FN represents the number of negative classes that are incorrectly predicted. In addition, we also use the ROC curve and AUC to evaluate the effect of the model. The ROC curve can reflect the classification ability of the model. Its abscissa is the false positive rate (FPR), and its ordinate is the true positive rate (TPR). The AUC is the area under the ROC curve enclosed by the coordinate axis.

*4.3.2. Detection Speed.* To evaluate the model detection speed, we used the number of network flows detected per second:

$$v_{\text{flow}} = \frac{N_{\text{test}}}{t_{\text{total}}}, \tag{14}$$

where $N_{\text{test}}$ is the number of samples in the model test set and $t_{\text{total}}$ is the total time (in seconds) to complete model training and output the prediction results.

Table 1: Experimental platform configuration.

| Lab environment | Parameter configuration |
|---|---|
| Operating system | Windows 10 |
| CPU | Intel(R) Core(TM) i7-10750H CPU |
| Programming language | Python3.8 |
| Deep learning framework | Pytorch1.6 |
| Statistical feature extraction tool | Wireshark |

Table 2: Dataset distribution.

| Samples | Pcap files | Flows | Experimental samples |
|---|---|---|---|
| White | 5000 | 26501 | 26501 |
| Black | 5000 | 89205 | 26501 |
| Total | 10000 | 115706 | 53002 |

Table 3: Statistical features.

| Feature name | Feature description |
|---|---|
| Duration | Duration of flow |
| Number of in-packets | Number of packets transferred from server to client |
| Number of out-packets | Number of packets transferred from client to server |
| Size of in-packets | Total volume of bytes transferred from server to client |
| Size of out-packets | Total volume of bytes transferred from client to server |

*4.4. Analysis of Results.* We compared our method with multiple benchmark models in terms of detection effect and speed. Fivefold cross-validation is used to prevent models from overfitting and verify their robustness. Table 4 shows the distribution of the experimental data. Based on the results in Table 5, we conclude that GCN-ETA is competitive and that our model can achieve a relatively better and faster detection effect when detecting encrypted malicious traffic.

We performed classification based on the statistical features of the network flow when training ordinary machine learning classification models. Since only simple processing (such as normalization) of statistical features was performed, ordinary machine learning classification models could have certain advantages in detection speed. As shown in Table 6, the common machine learning classification model with the fastest detection speed was BernoulliNB, but its detection effect was not satisfactory. Compared with BernoulliNB, the accuracy and the AUC of GCN-ETA were about 19% higher, and the $F_1 -$ score was about 18% higher. The general machine learning classification model with the best detection effect was XGBoost, but its detection effect was still not as good as GCN-ETA. Because the encrypted traffic conceals the characteristics of the data, the detection effect of the nnDPI model that directly classifies the traffic based on the packet byte information is not good (the accuracy, the AUC, and the $F_1 -$ score were both lower than 70%). In addition, nnDPI only performs simple processing on traffic data packets (such as removing the part including the source IP and destination IP), so the redundant data packet byte information makes the detection speed of nnDPI slower than other models (less than 1).

The feature dimension of nodes in our dataset was small (only five features), so we guessed that the detection effect of GCN-TC might not be good. Because GCN-TC is essentially a two-layer GCN model, and it only used softmax as a classifier in the last layer. The experimental results confirmed our guess. It can be seen from Table 5 that the detection effect of GCN-TC was even worse than that of XGBoost, the accuracy and the AUC were about 3% lower, and the $F_1 -$ score was about 4% lower. Because GCN can train node information and the connections between nodes at the same time, we propose using GCN as the feature extractor, and some machine learning algorithms as the classifier. It can be seen from Table 7 that this is feasible. In our experiments, the implementation of this idea improved the detection of malicious traffic. The accuracy and $F_1 -$ score of GCN + XGBoost, GCN + RF, GCN + KNN, and GCN + DT were all above 97%, and their AUC were over 96%. Both indicators were higher than GCN-TC by about 14%, both were higher than XGBoost by more than 10%, and the detection speeds of the four models were similar. Of the four models, GCN + RF had the best detection effect. It can be seen from Figure 4 that the detection effect of GCN-ETA and GCN + RF is not much different. But the detection speed $v_{\text{flow}}$ of GCN-ETA was nearly 19 times that of GCN + RF.

From the whole experiment, we can also see the effect of each step of GCN-ETA improvement. Firstly, replace the classification layer on the basis of GCN-TC (essentially a two-layer GCN model) to obtain GCN + XGBoost, GCN + RF, GCN + KNN, and GCN + DT, which significantly improves the detection effect. Secondly, on the basis of GCN + DT, the weight matrix to be trained between the GCN layers is fixed to the identity matrix to obtain the GCN-ETA, which significantly improves the detection speed while maintaining the detection effect. Compared with ordinary machine learning classification models with a faster

Table 4: Experimental dataset distribution.

| Nodes | Edges | Classes | Features | Train/test |
|---|---|---|---|---|
| 53002 | 92694142 | 2 | 5 | 42402/10600 |

Table 5: Test accuracy (%), $F_1$ − score (%), AUC (%), and $\nu_{\text{flow}}$ after 5-fold cross-validation.

| Model | Accuracy | $F_1$ − score | AUC | $\nu_{\text{flow}}$ |
|---|---|---|---|---|
| LR | 73.55 ± 0.39 | 76.18 ± 0.22 | 73.55 ± 0.26 | 58994 |
| BernoulliNB | 79.86 ± 0.44 | 80.31 ± 0.43 | 79.85 ± 0.47 | 129224 |
| DT | 83.26 ± 0.67 | 83.81 ± 0.68 | 83.24 ± 0.50 | 2061 |
| XGBoost | 86.39 ± 0.26 | 87.10 ± 0.24 | 86.39 ± 0.85 | 111 |
| nnDPI | 65.31 ± 0.35 | 69.03 ± 0.33 | 65.80 ± 0.16 | <1 |
| GCN-TC | 83.12 ± 0.70 | 83.34 ± 0.46 | 83.55 ± 0.97 | 28 |
| GCN + XGBoost | 97.65 ± 0.33 | 97.63 ± 0.33 | 97.52 ± 0.80 | 68 |
| GCN + RF | 98.61 ± 0.11 | 98.59 ± 0.11 | 98.34 ± 0.85 | 70 |
| GCN + KNN | 97.37 ± 0.29 | 97.36 ± 0.29 | 96.94 ± 0.83 | 71 |
| GCN + DT | 98.38 ± 0.07 | 98.37 ± 0.17 | 98.02 ± 0.94 | 72 |
| GCN-ETA | 98.65 ± 0.09 | 98.63 ± 0.09 | 98.59 ± 0.65 | 1333 |

Table 6: Test accuracy (%), $F_1$ − score (%), AUC (%), and $\nu_{\text{flow}}$ after 5-fold cross-validation.

| Model | Accuracy ↑ | $F_1$ − score ↑ | AUC ↑ | $\nu_{\text{flow}}$ |
|---|---|---|---|---|
| LR | 73.55 ± 0.39 | 76.18 ± 0.22 | 73.55 ± 0.26 | 58994 |
| BernoulliNB | 79.86 ± 0.44 | 80.31 ± 0.43 | 79.85 ± 0.47 | **129224** |
| DT | 83.26 ± 0.67 | 83.81 ± 0.68 | 83.24 ± 0.50 | 2061 |
| XGBoost | **86.39 ± 0.26** | **87.10 ± 0.24** | **86.39 ± 0.85** | 111 |
| nnDPI | 65.31 ± 0.35 | 69.03 ± 0.33 | 65.80 ± 0.16 | <1 |
| GCN-ETA | **98.65 ± 0.09** | **98.63 ± 0.09** | **98.59 ± 0.65** | **1333** |

Sorted in ascending order of accuracy, $F_1$ − score, and AUC.

Table 7: Test accuracy (%), $F_1$ − score (%), AUC (%), and $\nu_{flow}$ after 5-fold cross-validation.

| Model | Accuracy | $F_1$ − score | AUC | $\nu_{flow}$ ↑ |
|---|---|---|---|---|
| GCN + XGBoost | 97.65 ± 0.33 | 97.63 ± 0.33 | 97.52 ± 0.80 | 68 |
| GCN + RF | **98.61 ± 0.11** | **98.59 ± 0.11** | **98.34 ± 0.85** | **70** |
| GCN + KNN | 97.37 ± 0.29 | 97.36 ± 0.29 | 96.94 ± 0.83 | 71 |
| GCN + DT | 98.38 ± 0.07 | 98.37 ± 0.17 | 98.02 ± 0.94 | 72 |
| GCN-TC | 83.12 ± 0.70 | 83.34 ± 0.46 | 83.55 ± 0.97 | 28 |
| GCN-ETA | **98.65 ± 0.09** | **98.63 ± 0.09** | **98.59 ± 0.65** | **1333** |

Sorted in ascending order of $\nu_{flow}$.

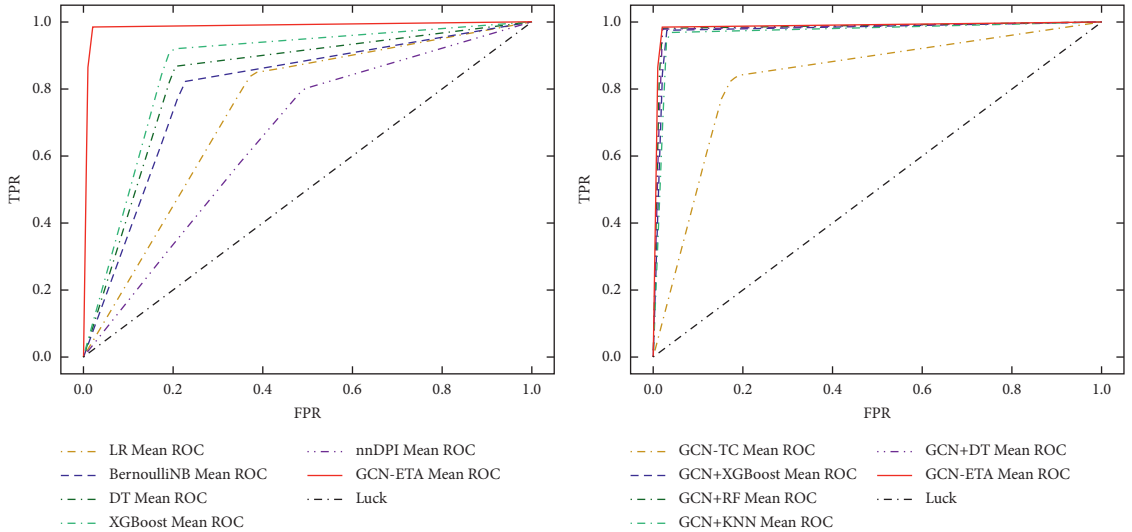


Figure 4: Model mean ROC curve after 5-fold cross-validation.

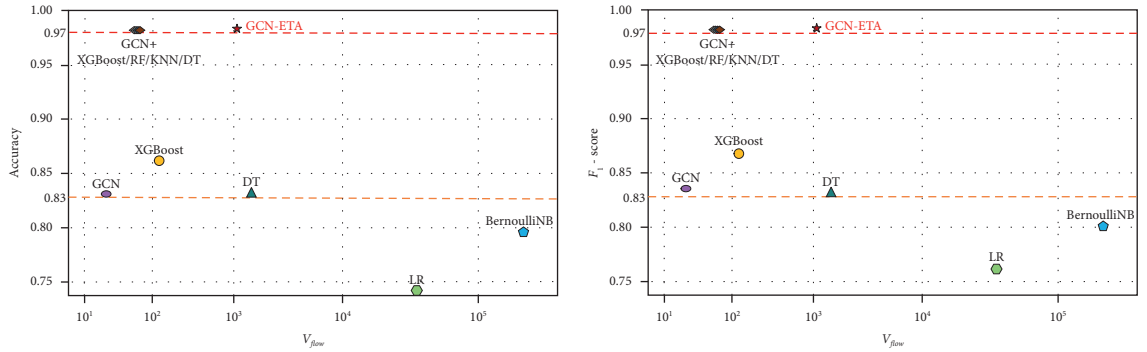


Figure 5: Overall performance of model testing effect (accuracy and $F_1$ − score) and speed.

detection speed, GCN-ETA, had a better detection effect. Among similar models with a good detection effect, GCN-ETA had the fastest detection speed, as shown in Figure 5. We believe this has three main reasons: (1) using GCN as the feature extractor to train network flow statistical features and traffic trajectory graphs at the same time; (2) replacing the GCN classifier to better capture differences between classes; (3) increasing the speed due to the simplified model having no parameters to be trained.

## 5. Conclusion

We proposed the GCN-ETA encrypted malicious traffic detection method, which is better and faster due to the use of GCN to simultaneously train the graph structure and statistical features. We simplified GCN to a graph-based preprocessing module for feature extraction and then connected the DT classifier. Results of experiments on actual network traffic confirmed that GCN-ETA has a better and faster detection effect than the ordinary machine learning classification model. GCN-ETA achieved excellent performance in experiments, with both the accuracy, the AUC, and $F_1$-score exceeding 98%, and more than 1,300 flows detected per second, and its detection speed is nearly 19 times that of the same detection effect model. The following conclusions can be drawn from the analysis of the experimental results: The powerful representation learning ability of GCN may mainly come from feature smoothing, and its interlayer linear conversion (which GCN-ETA does not preserve) is the main factor that increases complexity and redundancy. This is also the main reason for the excellent performance of GCN-ETA. However, two issues remain to be resolved and will be addressed in our future work: (1) We use a combination of neural network (GCN) and nonneural network (DT) methods for encrypted malicious traffic detection. We will try to replace the latter with a suitable artificial neural network and explore a unified neural network method; (2) The interpretability of the model is also important, and we will try to improve the neural network from this perspective [35, 36].

## Abbreviations

QoS: Quality of service
IP: Internet protocol
VPN: Virtual private network
TLS: Transport layer security
SSL: Secure sockets layer
pcap: Packet capture file storage format
DPI: Deep packet inspection
DT: Decision tree
LR: Logistic regression
SVM: Support vector machine
CNN: Convolutional neural network
RNN: Recurrent neural network
GCN: Graph convolutional network
FS-Net: Liu et al. [25] applied RNN to the problem of traffic encryption classification and proposed the FS-Net model
nnDPI: Bahaa et al. [26] used word embedding layer, convolutional neural network, and bidirectional recurrent neural network to directly classify the original traffic and proposed a new DPI method (nnDPI)
GCN-TC: Zheng et al. [30] applied GCN to network traffic classification for the first time and proposed the model GCN-TC
GCN + DT: A 2-layer GCN was used to extract features of the network flow, and DT as the classifier
GCN + RF: A 2-layer GCN was used to extract features of the network flow and random forest as the classifier
GCN + XGBoost: A 2-layer GCN was used to extract features of the network flow, and XGBoost as the classifier
GGCN + KNN: A 2-layer GCN was used to extract features of the network flow, and k-nearest neighbors as the classifier
GCN-ETA: Encrypted Malicious Traffic Detection Based on Graph Convolution Network (our model).

## Data Availability

This research used a dataset from malware and normal software collected from February to June 2020. The Qianxin Technology Research Institute's Tianqiong Sandbox collected the traffic generated by running these applications.

Black and white samples were encrypted traffic generated by malicious and normal software, respectively (all exe types). This dataset is a public dataset, and the link to the dataset is https://datacon.qianxin.com/opendata/maliciousstream.

## Conflicts of Interest

The authors declare no conflicts of interest.

## Authors' Contributions

Conceptualization was performed by Juan Zheng, Zhiyong Zeng, and Tao Feng; methodology was carried out by Juan Zheng and Tao Feng; software was provided by Juan Zheng; validation was conducted by Juan Zheng; formal analysis was done by Juan Zheng and Tao Feng; resources were provided by Zhiyong Zeng; data curation and visualization were done by Juan Zheng; the original draft was prepared by Juan Zheng; review and editing and supervision were carried out by Tao Feng; project administration and funding acquisition were performed by Tao Feng and Zhiyong Zeng. All authors have read and agreed to the published version of the manuscript.

## Acknowledgments

The authors would like to thank Qianxin Technology Research Institute for providing the data for this research. This research was funded by the Yunnan Provincial Department of Education (grant number 2021J0570), the National Natural Science Foundation of China (grant number 62166045), and the Yunnan Provincial Universities Engineering Research Center Construction Plan.

## References

[1] "The proportion of internet encrypted traffic has exceeded 90%," 2014, https://zhuanlan.zhihu.com/p/90443153.

[2] S. Sandra, M. Juarez, C. Diaz, N. Vallina-Rodriguez, and C. Troncoso, "Encrypted DNS--> Privacy? A traffic analysis perspective," 2019, https://arxiv.org/abs/1906.09682#:~:text=version%2C%20v3)%5D-,Encrypted%20DNS%20%2D%2D%3E%20Privacy,A%20Traffic%20Analysis%20Perspective&text=Virtually%20every%20connection%20to%20an,redirection%2C%20surveillance%2C%20and%20censorship.

[3] "Encryption-based threats grow by 260% in," 2020, https://www.helpnetsecurity.com/2020/11/11/encryption-based-threats-grow-2020/.

[4] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust network traffic classification," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1257–1270, 2015.

[5] Q. Lu, J. Lin, Z. Su, and F. Liu, "System design of network data classification based on deep packet inspection," *Journal of Physics: Conference Series*, vol. 1738, no. 1, Article ID 012118, 2021.

[6] W. Song, M. Beshley, K. Przystupa et al., "A software deep packet inspection system for network traffic analysis and anomaly detection," *Sensors*, vol. 20, no. 6, 2020.

[7] R. Alshammari and A. N. Z. Heywood, "Machine learning based encrypted traffic classification: identifying SSH and Skype," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–8, Ottawa, ON, Canada, July 2009.

[8] M. Dusi, A. Este, F. Gringoli, and L. Salgarelli, "Using GMM and SVM-based techniques for the classification of SSH-encrypted traffic," in *Proceedings of the 2009 IEEE International Conference on Communications*, pp. 1–6, Dresden, Germany, July 2009.

[9] R. Alshammari and N. Z. Heywood, "Investigating two different approaches for encrypted traffic classification," in *Proceedings of the 2008 Sixth Annual Conference on Privacy, Security and Trust*, pp. 156–166, Fredericton, Canada, October 2008.

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

[11] Z. Chen, K. He, J. Li, and Y. Geng, "Seq2Img: a sequence-to-image based approach towards IP traffic classification using convolutional neural networks," in *Proceedings of the 2017 IEEE International Conference on Big Data (Big Data)*, pp. 1271–1276, Boston, MA, USA, December 2017.

[12] H. Yao, C. Liu, P. Zhang, S. Wu, C. Jiang, and S. Yu, "Identification of encrypted traffic through attention mechanism based long short term memory," *IEEE Transactions on Big Data*, 2019.

[13] B. Gallagher, M. Iliofotou, T. Eliassi-Rad, and M. Faloutsos, "Link homophily in the application layer and its usage in traffic classification," in *Proceedings of the 2010 Proceedings IEEE INFOCOM*, pp. 1–5, San Diego, CA, USA, March 2010.

[14] M. Iiiofotou, H. Kim, M. Faloutsos, M. Mitzenmacher, P. Pappu, and G. Varghese, "Graption:A graph-based P2P traffic classification framework for the internet backbone," *Computer Networks*, vol. 55, pp. 1909–1920, 2011.

[15] P. W. Battaglia, J. B. Hamrick, V. Bapst, and A. Sanchez-Gonzalez, "Relational inductive biases, deep learning, and graph networks," 2018, https://arxiv.org/abs/1806.01261.

[16] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[17] Z. Cao, G. Xiong, Y. Zhao, Z. Li, and L. Guo, *A Survey on Encrypted Traffic Classification. International Conference on Applications and Techniques in Information Security*, Springer, Berlin, Germany, 2014.

[18] T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning. in IEEE Communications Surveys & Tutorials,," *Four Quarters*, vol. 10, pp. 56–76, 2008.

[19] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning," in *Proceedings of the IEEE/ACM Network Traffic Measurement and Analysis Conference (TMA'18) ACM*, pp. 445–458, Vienna, Austria, June 2019.

[20] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 63–78, 2017.

[21] S. E. Coull and K. P. Dyer, "Traffic analysis of encrypted messaging services," *ACM SIGCOMM - Computer Communication Review*, vol. 44, no. 5, pp. 5–11, 2014.

[22] M. Di Mauro and M. Longo, "Revealing encrypted WebRTC traffic via machine learning tools. 2015 12th International

Joint Conference on e-Business and Telecommunications (ICETE)," *IEEE*, vol. 4, pp. 259–266, 2015.

[23] T. Yu, F. Zou, L. Li, and P. Yi, "An encrypted malicious traffic detection system based on neural network," in *Proceedings of the 2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 62–70, IEEE, Guilin, China, October 2019.

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

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

[26] M. Bahaa, A. Aboulmagd, K. Adel, H. Fawzy, and N. Abdelbaki, "nnDPI: a novel deep packet inspection technique using word embedding convolutional and recurrent neural networks," in *Proceedings of the 2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, pp. 165–170, IEEE, Giza, Egypt, October 2020.

[27] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese, "Network monitoring using traffic dispersion graphs (tdgs)," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pp. 315–320, San Diego, CA, USA, October 2007.

[28] T. Karagiannis, K. Papagiannaki, M. Faloutsos, and B. L. I. N. C. Michalis Faloutsos, "Blinc," *ACM SIGCOMM - Computer Communication Review*, vol. 35, no. 4, pp. 229–240, 2005.

[29] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the Int. Conf. Learn. Representations*, pp. 1–14, Vancouver, Canada, April 2016.

[30] J. Zheng and D. Li, "GCN-TC: combining trace graph with statistical features for network traffic classification," in *Proceedings of the ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, Shanghai, China, May 2019.

[31] S. Mo, Y. Wang, D. Xiao, W. Wu, S. Fan, and C. Shi, "Encrypted traffic classification using graph convolutional networks," in *Proceedings of the International Conference on Advanced Data Mining and Applications*, pp. 207–219, Springer, Virtual Event USA, August 2020.

[32] Z. Hui-huang and H. Liu, "Multiple classifiers fusion and CNN feature extraction for handwritten digits recognition," *Granular Computing*, vol. 5, no. 3, pp. 411–418, 2019.

[33] K. Jun, D.-W. Lee, K. Lee, S. Lee, and M. S. Kim, "Feature extraction using an RNN autoencoder for skeleton-based abnormal gait recognition," *IEEE Access*, vol. 8, pp. 19196–19207, 2020.

[34] "DATACON: open data for multi domain large-scale competition for security research [EB/OL], ," Qianxin Technology Research Institute, 2020, https://datacon.qianxin.com/opendata.

[35] Z. C. Lipton, "The mythos of model interpretability," *ACM Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[36] Q. Zhang and S. C. Zhu, "Visual interpretability for deep learning: a survey," 2018, https://arxiv.org/abs/1802.00614.