Flow Topology-Based Graph Convolutional Network for Intrusion Detection in Label-Limited IoT Networks

Xiaoheng Deng, *Member, IEEE*, Jincai Zhu, Xinjun Pei, Lan Zhang, *Member, IEEE*, Zhen Ling, *Member, IEEE*, and Kaiping Xue, *Senior Member, IEEE*

***Abstract*—Given the distributed nature of the massively connected "Things" in IoT, IoT networks have been a primary target for cyberattacks. Although machine learning based network intrusion detection systems (NIDS) can effectively detect abnormal network traffic behaviors, most existing approaches are based on a large amount of labeled traffic flow data, which hinders their implementation in the highly dynamic IoT networks with limited labeling. In this paper, we develop a novel Flow Topology based Graph Convolutional Network (FT-GCN) approach for label-limited IoT network intrusion detection. Our main idea is to leverage the underlying traffic flow patterns, $i.e.$, the flow topological structure, to unlock the full potential of the traffic flow data with limited labeling, where the FT-GCN will be deployed at the edge servers in IoT networks to detect intrusions via software defined network technologies. Specifically, FT-GCN first takes the time correlation of traffic flows into account to construct an interval-constrained traffic graph (ICTG). Besides, a Node-Level Spatial (NLS) attention mechanism is designed to further enhance the key statistical features of traffic flows in ICTG. Finally, the combined representation of statistical flow features and flow topological structure are learned by the cost-effective Topology Adaptive Graph Convolutional Networks (TAGCN) for intrusion identification in IoT networks. Extensive experiments are conducted on three real-world datasets, which demonstrate the effectiveness of the proposed FT-GCN compared to state-of-the-art approaches.**

***Index Terms*—Network intrusion detection, graph convolutional networks, attention mechanism, label limitation, IoT.**

Manuscript received 27 May 2022; revised 22 August 2022; accepted 1 October 2022. Date of publication 14 October 2022; date of current version 7 March 2023. This work was supported by the National Natural Science Foundation of China Project (62172441, 62172449, 61772553), the local science and technology developing foundation guided by the central government (Free exploration project 2021Szvup166), the Opening Project of State Key Laboratory of Nickel and Cobalt Resources Comprehensive Utilization (GZSYS-KY-2020-033), and the Fundamental Research Funds for the Central Universities of Central South University (2021zzts0201). The associate editor coordinating the review of this article and approving it for publication was A. Mourad. *(Corresponding author: Xiaoheng Deng.)*

Xiaoheng Deng, Jincai Zhu, and Xinjun Pei are with the School of Computer Science and Engineering, Central South University, Changsha 410083, China (e-mail: dxh@csu.edu.cn; jincaizhu@csu.edu.cn; pei_xinjun@163.com).

Lan Zhang is with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI 49931 USA (e-mail: lanzhang@mtu.edu).

Zhen Ling is with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China (e-mail: zhenling@seu.edu.cn).

Kaiping Xue is with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China (e-mail: kpxue@ustc.edu.cn).

Digital Object Identifier 10.1109/TNSM.2022.3213807

I. Introduction

THE PAST decade witnessed an astounding increase in interest in Internet of Things (IoT). IoT connects distributed "Things" over the Internet, creating an integration that allows various applications to operate in physical and cyber worlds [1]. Unfortunately, the distributed nature of massively connected "Things" makes IoT network a primary cyberattack target [2], [3], [4], [5]. Network Intrusion Detection Systems (NIDS) have been one essential technique to detect and mitigate malicious network activities inside IoT network [6], [7]. By monitoring the traffic flows in and out of IoT devices, NIDS can alert IoT networks when an intrusion is observed [8]. Specifically, NIDS identifies the abnormal behaviors of network traffic by comparing them with the patterns of normal behaviors, where the traffic flow behavior that deviates from the normal pattern will be classified as an intrusion.

One major mechanism to extract the patterns of traffic flow behaviors in NIDS is to leverage machine learning techniques [9]. Although machine learning based mechanisms are well-recognized to learn discriminative features, they usually require a large amount of labeled data for learning in a supervised manner [10]. Given the highly dynamic IoT environments, it is non-trivial to obtain sufficient labeled traffic flow data in time. Labeling requires massive manual work and human interaction, which is time-consuming. Unfortunately, when a limited amount of labeled data is obtained, the detection performance can be seriously degraded [11]. This is mainly due to the ignorance of key underlying features, such as the topological structure of traffic flows. Hence, it is essential to fully exploit the underlying features, i.e., the topological structure of traffic flows, for deploying the machine learning based NIDS in label-limited IoT networks.

Due to the powerful capability of extracting graph semantic and topological structure, Graph Convolutional Networks (GCN) have been one potential approach [12]. In [13], Zheng and Li convert the traditional traffic trace graph to construct a new traffic graph and use the GCN model for representation learning. For ease of understanding, the process is shown in Fig. 1, where the nodes represent traffic flows and edges denote that nodes have common IP hosts. However, simply connecting the traffic flows with the common IP hosts makes the constructed graph have a huge number of

Fig. 1. (a) Nodes present IP hosts, and edges represent traffic flows. (b) Nodes represent traffic flows, and edges denote that nodes have common IP hosts. Colors represent different applications.

redundant edges, seriously degrading the performance of GCN. To address this issue, Sun et al. [14] build a traffic graph based on the K-Nearest Neighbor algorithm to describe the topology and learn the representation by leveraging GCN. In the process of graph construction, the similarity between nodes is used to find the K-most similar nodes as its neighbors. Although with a reduced edge number, the constructed graphs ignore the fact that the effectiveness of connections between two traffic flows is affected by the time intervals between their generation. When the time intervals become larger, the correlation (referring to the probability that they belong to the same application) between the two flows becomes weaker. Therefore, we take the time interval into account and construct an interval-constrained traffic graph (ICTG) to describe the topological structure of traffic flows.

On the other hand, since the feature dimensionality of large graphs is usually high, GCN-based models can easily suffer from performance degradation. Although feature engineering can reduce the feature dimension, it usually involves a lot of human interactions and expert knowledge. As a solution to avoid feature engineering, the attention technique can autonomously learn the relationship between features and help the model focus on the key features to improve classification performance. Enlighted by SENet attention [15], we design a Node-Level Spatial Attention mechanism, named NLS, to enhance the key features of nodes in the traffic graph. NLS can selectively enhance important features by utilizing global feature information, making the whole model focus more on the key features affecting the classification results. Moreover, to further improve the efficiency of the above designs, instead of leveraging the conventional spectral-domain GCN that usually suffers performance degradation due to the computational expensive graph convolution operations [16], [17], [18], [19], we adopt a novel vertex-domain GCN, Topology Adaptive Graph Convolutional Network (TAGCN) [20]. Specifically, TAGCN provides a systematic approach to designing a set of fixed-size learnable filters to perform convolutions on graphs without requiring convolution approximations. What's more, TAGCN inherits the local feature extraction property of classical CNNs with low computational complexity [21], which is more suitable than spectral GCN for highly dynamic IoT environments.

To put all these ideas together, this paper develops a novel Flow Topology based Graph Convolutional Network (FT-GCN) approach for intrusion detection in label-limited IoT networks, where the proposed FT-GCN can be deployed at edge servers to detect intrusions and mitigate them through software defined network technologies. To enable effective detection with limited labeling, we first construct an interval-constrained traffic graph (ICTG) to describe the topological structure of traffic flows and limit the time interval between traffic flow generation. Then, to reduce the effect brought by high-dimensional features and avoid manual operations brought by feature engineering, we design an NLS attention mechanism to enhance the representation ability of key features in ICTG. Finally, we put it and processed features into Topology Adaptive Graph Convolutional Network (TAGCN) for feature learning and classification. We transform the network intrusion detection in label-limited IoT networks into a node classification task, and use traffic topology information as complementary features to enhance the graph representation. First, we construct a unique traffic graph to represent the topological structure of traffic flows, and design an attention mechanism to enhance the key statistical features of traffic flows. After that, we use a graph learning model with low computational complexity to perform node classification. The performance of FT-GCN is validated on three real-world datasets which represent different network scales. Our major contributions are as follows:

- We propose an innovative NIDS approach, FT-GCN, to enable intrusion detection under limited labeling for IoT networks. The proposed FT-GCN converts the topological structure of the traffic flows to a graph representation and utilizes TAGCN for feature learning and classification.
- We take the time correlation of traffic flows into account and construct the interval-constrained traffic graph (ICTG). Furthermore, we design a Node-Level Spatial (NLS) attention mechanism to utilize key node features in ICTG, while also reducing the model burden.
- We conduct extensive experiments on real-world datasets to evaluate the effectiveness of FT-GCN in identifying network intrusion. The results demonstrate that FT-GCN achieves higher detection accuracy with limited labeling compared to other state-of-the-art methods.

The rest of the paper is organized as follows. Section II presents the related work of NIDS in IoT Section III introduces the system model. In Section IV, we detail the design rationale and key components of the proposed FT-GCN. And then we evaluate FT-GCN and analyze the experimental results in Section V. Finally, Section VI concludes the paper.

II. Related Work

A. Machine Learning Based NIDS in IoT

Recently, Machine Learning has been widely used in NIDS to identify normal and abnormal behaviors. Atay [22] Aloul et al. [23] utilized adversarial autoencoders with the K nearest neighbor algorithm to implement intrusion detection, which can be executed on small routers near the IoT edge. Kim et al. [24] proposed a DNN-based NIDS to identify malicious traffic, which achieves an average accuracy rate of 99% on the KDD dataset. Chen et al. [25] proposed a novel NIDS system, which uses a deep-learning-based detection model to extract statistical features from the original network traffic. Das et al. [26] implemented a supervised ensemble ML framework (SupEnML), which combines multiple ML

classifiers from various classification families and the ensemble feature selection (EnFS) technique for network intrusion detection. The work by [27] proposed a lightweight intrusion detection method, which utilizes a supervised machine learning model, i.e., support vector machine (SVM), to detect abnormal data in the IoT network. Wu et al. [28] proposed an intrusion detection method based on dynamic ensemble incremental learning relevance vector machine(DEIL-RVM), and implemented a dynamically adjusted ensemble intrusion detection model. Another work by [29] used feature selection and Random Forest to classify traffic flows, achieving high accuracy on the NSL-KDD dataset. Although Machine Learning based NIDSs have achieved high detection accuracy for network traffic classification in IoT networks, they usually require numerous correctly labeled data (ground truth) for model training. Due to the highly dynamic nature of the IoT network, obtaining the corrected labels is time-consuming, which involves massive human interactions. However, the performance of these existing deep learning-based NIDS will degrade when only a limited number of tags can be utilized during model training [30], [31]. The reason is that they usually ignore the key information underlying the topological structures of traffic flows [14]. Therefore, it is worth exploring how to use the traffic topology to improve the classification accuracy of NIDS in label-limited IoT networks.

B. NIDS With Limited Labeled Data

In existing NIDSs, there are several approaches to improve detection accuracy with limited labeled data. Liao et al. [11] designed a NIDS based on a Generative Adversarial Network (GAN) model, which enhances the original training set by continuously generating samples. In [32], Gallagher et al. introduced the concept of link homogeneity and designed a statistical relational learning algorithm to improve classification performance. By using only 5% labeled data, it can achieve an average accuracy rate of 90%. The work in [33] proposed a semi-supervised intrusion detection model optimized by a cloud grey wolf optimization (CGWO) algorithm, which solves the problem of low detection efficiency caused by the lack of sufficient training sets. Zheng and Li [13] converted the traffic trace graph to a new traffic graph with nodes representing traffic flows, and utilized a GCN model for feature learning to improve the detection accuracy under a low labeling rate. Sun et al. [14] built a traffic graph based on a KNN algorithm and combined the GCN and autoencoder for feature representation learning. Another work by [34] proposed a novel classifier called Energy-based Flow Classifier (EFC) for network flow classification, which is inspired by the inverse Potts model from quantum mechanics. This anomaly-based classifier uses inverse statistics to infer a statistical model with labeled benign examples. In this paper, we propose a novel network intrusion detection method, which can transform network traffic into a graph representation to exploit the network traffic topology. Moreover, we use a TAGCN model for feature representation learning and classification.

III. System Model, Threat Model and Overview of Our Scheme

This paper considers a typical IoT network, consisting of IoT nodes, edge servers, SDN controllers, and SDN switches. Specifically, IoT nodes refer to IoT devices with communication capabilities, such as mobile phones, computers, smart refrigerators, *etc.*

A. System Model

Each IoT node can transmit and forward traffic flows. The IoT nodes have limited computing resources, making them unable to perform heavy protection mechanisms [35]. Edge servers with sufficient computing power can deploy security mechanisms. SDN controllers are programmable software, which are deployed on the edge servers to obtain the global information of the IoT network by SDN switches. The SDN controller is responsible for the processing and calculation of the protocol, and sends the defined flow table to the SDN switch to control the forwarding rules in the IoT network. An SDN switch close to the IoT node can capture the traffic flows of IoT node for the edge server, and control them based on the installed flow tables.

Each node can communicate with other nodes through traffic flows. Each traffic flow has a source IP and a destination IP address. Most traffic flows cannot be sent directly from the source IP node to the destination IP node, and in most cases, they need to pass through one or more forwarding devices. In the IoT network, the SDN switches are the main forwarding devices that can forward or drop traffic flows based on flow tables. The transmission process of all the traffic flows can be formulated as

$$H_{src} \rightarrow (S_1, R_1) \rightarrow \cdots \rightarrow (S_n, R_n) \rightarrow H_{des}, \tag{1}$$

where H represents IP host, *src* and *des* represent source and destination, respectively, S_i and R_i denote the i-th network node and its traffic flow processing rule (i.e., the flow table of the switch), respectively. Each traffic flow captured by SDN switches contains D-fields statistical features which are extracted from the transport layer and network layer. Each traffic flow has a label $y \in \{0, 1\}$ indicating whether it is malicious or not. However, labeling a large amount of traffic flows is not trivial, resulting in most traffic flows being unlabeled.

B. Threat Model

This paper considers a general threat model in IoT networks. We consider the edge server and SDN switches are trustworthy due to their powerful computing capabilities for advanced defense. We assume that adversaries have no knowledge about the IoT network topology, but they can attack the IoT nodes from outside networks by launching malicious traffic flows, such as Backdoors, Fuzzers, Dos, *etc.* The malicious traffic flows can compromise and control vulnerable IoT nodes, rendering IoT nodes untrustworthy. Then, the infected IoT nodes will lose their original function and become malicious nodes, launching a large amount of malicious traffic to attack other normal nodes. The target of the adversaries is to compromise

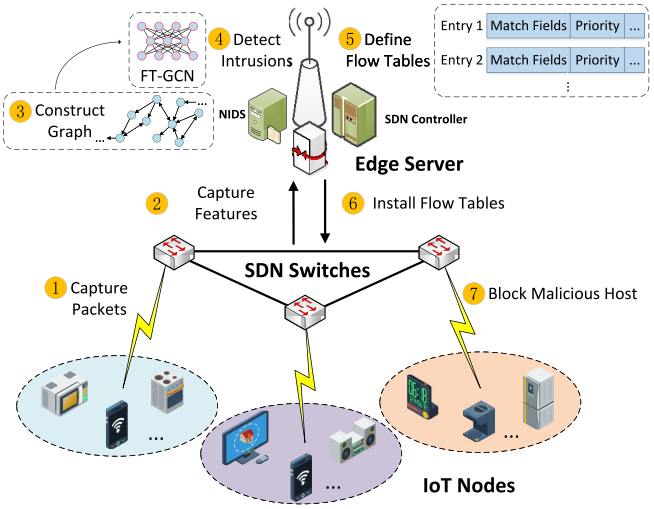


Fig. 2. Edge-assisted NIDS for IoT Systems.

as many IoT nodes as possible, eventually paralyzing the entire network.

Suppose there are N IoT nodes, and each node has two states: susceptible S, indicating that a node is vulnerable, and infected I, indicating that a node has been infected. Assuming β is the probability that a node in the IoT network is infected by malicious traffic, we can have:

$$S \xrightarrow{\beta} I \tag{2}$$

Then, we denote $i(t)$ and $s(t)$ as the ratio of infected nodes and susceptible nodes at a certain time t, respectively. The increase of $i(t)$ at each time can be expressed as

$$\frac{di(t)}{dt} = \beta s(t) i(t) = \beta i(t)(1 - i(t)). \tag{3}$$

By solving the differential equation (3), we can have:

$$i(t) = \frac{i(0)e^{\beta t}}{1 + i(0)\left(e^{\beta t} - 1\right)} = \frac{1}{1 + \left(\frac{1}{i(0)} - 1\right)e^{-\beta t}}. \tag{4}$$

It is worth noting that when $i(t) = 0.5$, $\frac{d\,i(t)}{d\,t}$ will reach the maximum value. This means that the number of infected nodes grows the fastest. As time t increases, $i(t)$ tends to 1. Eventually, almost all nodes in the network will be infected.

C. Edge-Assisted NIDS for IoT Networks

To against the above threat model, we propose an edge-assisted NIDS as shown in Fig. 2, which is composed of three main modules: data collection, intrusion detection, and intrusion mitigation.

- *Data Collection:* SDN switches capture traffic flows and extract statistical features of the network layer from them, then transmit them to the edge server for detection. However, in reality, labeling large-scale traffic flows in the IoT network can be extremely costly. Therefore, a few traffic flows are labeled by traffic generation tools such as Argus and Bro-IDS.
- *Intrusion Detection:* Based on the received features of traffic flows, the edge server constructs the traffic graph and the feature matrix of traffic flows as the input of FT-GCN. FT-GCN learns the combination representation from the topology and statistical features of traffic flows and gives the classification results to determine whether a given traffic flow is malicious or not.
- *Intrusion Mitigation:* SDN controllers and SDN switches are responsible for intrusion mitigation. SDN controller deployed on the edge server generates flow tables according to the detection result of FT-GCN. These flow tables are sent and installed on SDN switches, which can control the forwarding of traffic flows. Once a malicious traffic flow is detected, its source node is regarded as malicious, and all the traffic flows from this node are blocked by the flow tables. In such a case, the SDN switch will discard the traffic flows sent by malicious host IP.

Compared with NIDS based on cloud computing, deploying the detection model on the edge server can effectively reduce the delay of data transmission, which meets the low-latency requirements in the IoT network. In the entire edge-assisted NIDS, the role of SDN technology is mainly divided into two aspects. The first function of SDN is to provide data for the detection model. The SDN switches collect the statistical features of traffic flows and transfer them to the edge server. The second function of SDN is to allow NIDS to customize some specific defense strategies to defend against intrusions. The SDN controller generates a flow table based on the results generated by the detection model and sends it to the SDN switches to control the traffic forwarding of the IoT network.

IV. Flow Topology-Based Graph Convolutional Networks

In this section, we introduce the proposed FT-GCN as shown in Figure 3. We start by presenting the overview of FT-GCN, followed by the key components of FT-GCN.

A. Overview of FT-GCN

The FT-GCN consists of four consecutive main stages. Specifically, Specifically, the traffic flows are first preprocessed (Section IV-B), and then input into the NLS attention mechanism to obtain the processed feature $\hat{X}$ (Section IV-D). Then, the interval-constrained traffic graph is constructed to describe the topological structure of traffic flows by leveraging their IP address and timestamp features (Section IV-C). Finally, the graph and the processed features are taken as the input of TAGCN for representation learning (Section IV-E).

B. Data Preprocessing

Generally, traffic flows have various statistical properties at the network or transport layer, such as IP address, port number, packet lengths and so on. These properties may imply potentially malicious behaviors, which is useful to distinguish malicious traffic from benign traffic. Table I presents some representative features of traffic flows. Instead of traditional solutions that require complex feature analysis, we filter the features used in the graph construction, i.e., Srcip, Dstip, and Stime, and drop some features containing illegal values

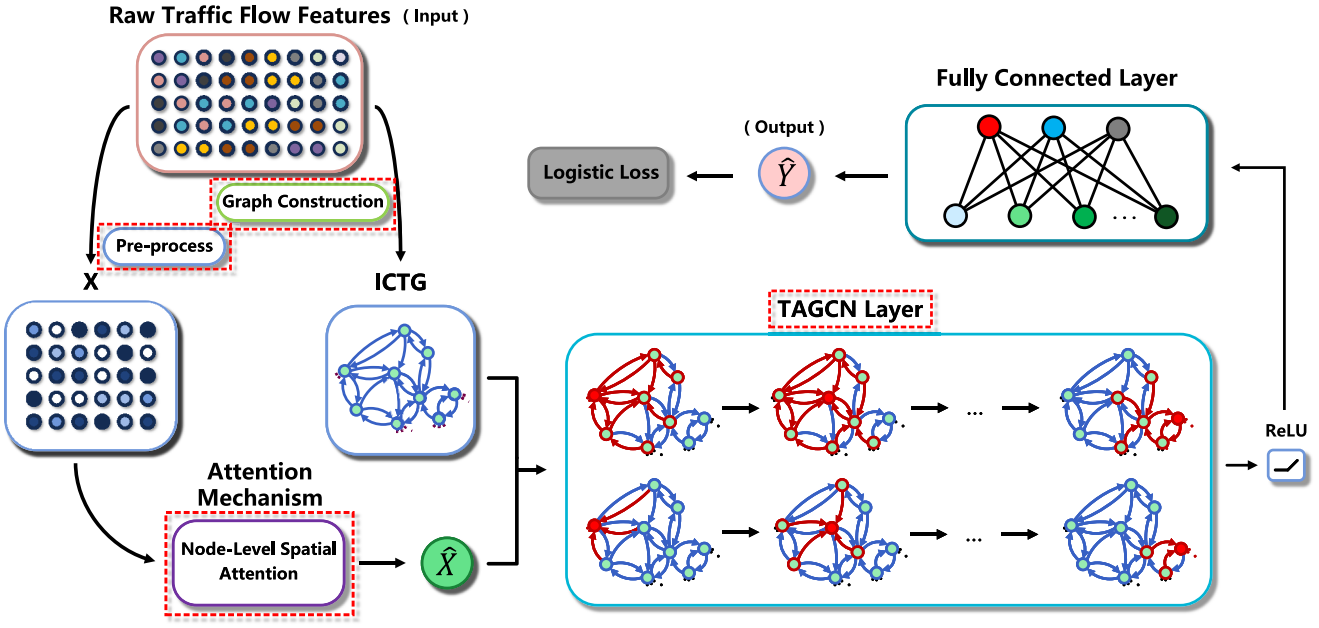


Fig. 3. Overview of FT-GCN.

TABLE I
SOME REPRESENTATIVE FEATURES OF TRAFFIC FLOWS

Name	Type	Description
Srcip	Nominal	Source IP address
Sport	Integer	Source port number
Dstip	Nominal	Destination IP address
Dsport	Integer	Destination port number
Proto	Nominal	Transaction protocol
Dur	Float	Traffic flow total duration
Stime	Timestamp	Traffic flow start time
Sbytes	Integer	Source to destination transaction bytes
Dbytes	Integer	Destination to source transaction bytes
Sload	Float	Source bits per second
Dload	Float	Destination bits per second
Spkts	Integer	Source to destination packet count
Dpkts	Integer	Destination to source packet count
Label	Binary	0 for normal and 1 for attack records

like INF and Nan. After that, we feed these features into an attention mechanism for feature enhancement.

C. Graph Construction

In this section, we create an interval-constrained traffic graph (ICTG) to describe the topological structure of traffic flows, i.e., the relationship between traffic flows, and transform the network intrusion detection into a node classification task. According to link homophily [32], traffic flows with common IP hosts have the same application trend. The main idea of ICTG construction is to extract the similarity relationships between traffic flows based on the link homophily through the generation time interval of traffic flows.

We design three rules to construct ICTG. The ICTG can be represented as $G = (V, E, A)$, where $V = \{v_1, v_2, \ldots, v_N\}$ denotes the set of nodes which are traffic flows, and $E = \{e_1, e_2, \ldots, e_M\}$ represents edges set indicating that nodes are more relevant than other nodes. Let $A \in \mathbb{R}^{N \times N}$ be the adjacency matrix of ICTG, when traffic flow i is relevant or similar to traffic flow j, $A_{ij} = 1$, otherwise $A_{ij} = 0$. We denote F_i as i-th traffic flow of dataset. The rules are given below:

Rule 1: If traffic flow F_i and F_j have a common source IP, $A_{ij} = A_{ji} = 1$.

When the source IP of F_i and F_j are the same, it indicates that they are highly correlated and tend to have similar activities or applications. For example, in P2P applications, the host usually needs to connect with multiple collaborator hosts to broadcast data and the traffic flows between them usually belong to the same application. If any of F_i and F_j is normal, the other sent by this IP host may also be normal. On the other hand, when a traffic flow sent by a source IP host is detected to be malicious, what can be inferred is that this source IP host is compromised and all other traffic flows sent by it are equally likely to be malicious. As shown in Fig. 4 (a), F_1, F_2 and F_3 have the common source IP host, therefore, in Figure 3 (b), these flows are interconnected. Similarly, the value of the corresponding position in the adjacency matrix A of ICTG is set to 1, i.e., $A_{12} = A_{21} = A_{13} = A_{31} = A_{23} = A_{32} = 1$.

Rule 2: If the destination IP of traffic flow F_i is the same as the source of another traffic flow F_j, $A_{ij} = 1$.

During the communication process, traffic flows are not always directly transmitted from the source node to the destination node, but will be forwarded many times through multiple intermediate nodes before reaching the destination node, as shown in formula (1). It means that the traffic flows received by a node are related to the traffic sent or forward by that node. When an adversary launches an attack on a destination node, the intermediate forwarding nodes will also be affected, such as a DoS attack. The other consideration is that when F_i is malicious, the destination IP address of F_i will become the target of the network attack. If the destination IP host is compromised by F_i, the other traffic flows sent or forwarded by it will be affected and become malicious. As shown

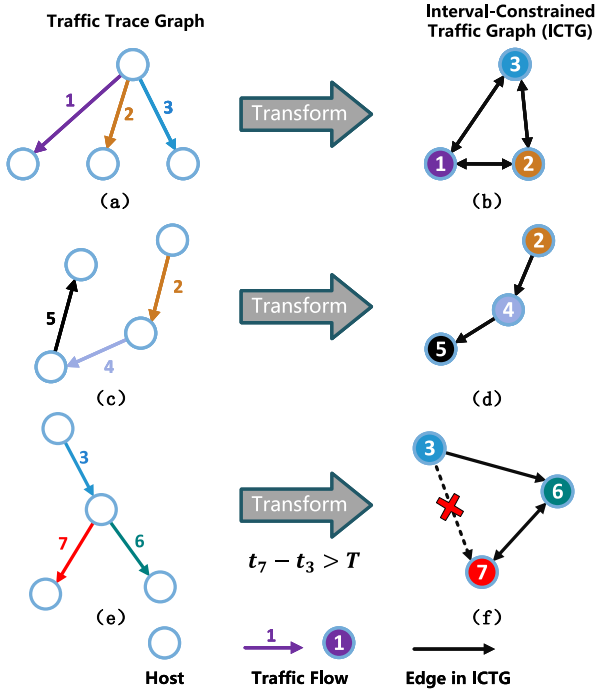


Fig. 4. Transformation of rule 1, 2, 3. t_i represents the generation time of i-th traffic flow and T represent the maximum generation time interval of two traffic flows. In the traffic trace graph, nodes represent IP hosts and edges represent traffic flows. In the ICTG, nodes represent traffic flows and edges denote that the two nodes are more relevant than the other nodes.

in Fig. 4 (c), the destination IP of F_2 and the source IP of F_4 are the same. In our case, the two flows are connected by a directed edge, as shown in Fig. 3 (d). Then, the value of the corresponding position in the adjacency matrix A of ICTG is set to 1, $i.e.$, $A_{24} = 1$. Similarly for the traffic flow F_4 and F_5, $A_{45} = 1$.

Rule 3: Let t_i and t_j be the generation time of traffic flow F_i and F_j, respectively. We define T as the maximum time interval for any two connected traffic flows. If $|t_i - t_j| < T$, $A_{ij} = 1$, otherwise, $A_{ij} = 0$

In practice, the transmission speed of traffic flows is very fast, and the traffic flows sent by the same host in a short period of time are highly similar in their applications. Therefore, there is a strong correlation between these traffic flows in short time intervals. For example, in a video streaming service, the subscribers need to continuously request the video resources from the server until the connections are broken. However, this correlation weakens when the generation time interval becomes larger, which means that these traffic flows are less similar to each other. More importantly, to compromise some targeted IoT nodes, the attackers need to initiate a lot of malicious traffic in a short period of time. Therefore, as the generation time interval decreases, the correlation between malicious traffic flows become closer. In addition, when the generation time interval T is not limited, the number of neighbors of a given node will be extremely large, which may lead to the densely constructed ICTG and affect the subsequent TAGCN model learning. The reason may be that edge information is too redundant, so that TAGCN cannot focus on learning the key edge information, resulting in performance degradation.

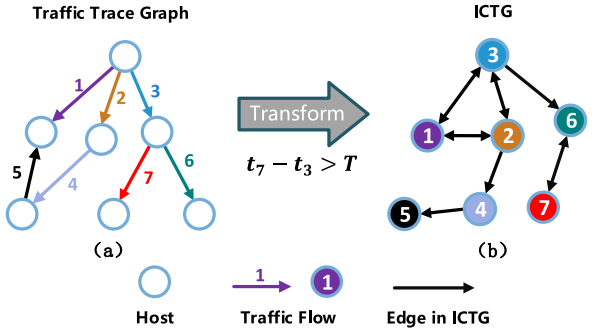


Fig. 5. The overall transformation of ICTG construction.

Notably, most existing graph learning methods perform better classification on sparse graphs than on dense graphs. As shown in Fig. 4 (e), traffic flow F_6 and F_7 have the same source IP address while $t_7 - t_6 < T$. Thus, by rule 1, we can get $A_{67} = A_{76} = 1$. The destination IP of F_3 is the same as the source IP of F_6, F_7. However, considering that $t_7 - t_3 > T$, we set A_{36} and A_{37} to 1 and 0, respectively. By doing so, the interval-constrained traffic graph G can be constructed to describe the topology of traffic flows. Fig. 5 shows the transformation process. By leveraging rules 1, 2 and 3, we can extract the topological structure of traffic flows.

D. Node-Level Spatial Attention

Inspired by the squeeze and excitation network [15], we design a node-level spatial attention mechanism (NLS) to extract the key node features, which takes the raw feature matrix X as input, and then outputs a new feature matrix $\hat{X}$ for the following TAGCN model. The execution process of the NLS attention mechanism is shown in Fig. 6.

The raw input feature is denoted as $X \in \mathbb{R}^{N \times D}$ where N and D represent the total number and the feature dimensions of the traffic flows, respectively. It can be denoted as

$$X = \begin{bmatrix} X_0 \\ \vdots \\ X_{N-1} \end{bmatrix}, \tag{5}$$

where X_i is the D-dimension feature vector. Then, the output feature map $\hat{X} \in \mathbb{R}^{N \times D}$ is expressed as

$$\hat{X} = \begin{bmatrix} \hat{X}_0 \\ \vdots \\ \hat{X}_{N-1} \end{bmatrix}, \tag{6}$$

where $\hat{X}_i$ denotes the i-th processed node feature.

First, the global average pooling is performed to compress matrix X to a $1 \times D$ vector, which can express the information of the global receptive field i.e., squeeze map Z. The calculation process is shown in the formula (7)

$$Z = \frac{1}{N} \sum_{i=0}^{N-1} X_i. \tag{7}$$

Then, we use a fully connected layer to reduce dimension, i.e., $W_1 Z$. Next, we upgrade the dimension through the

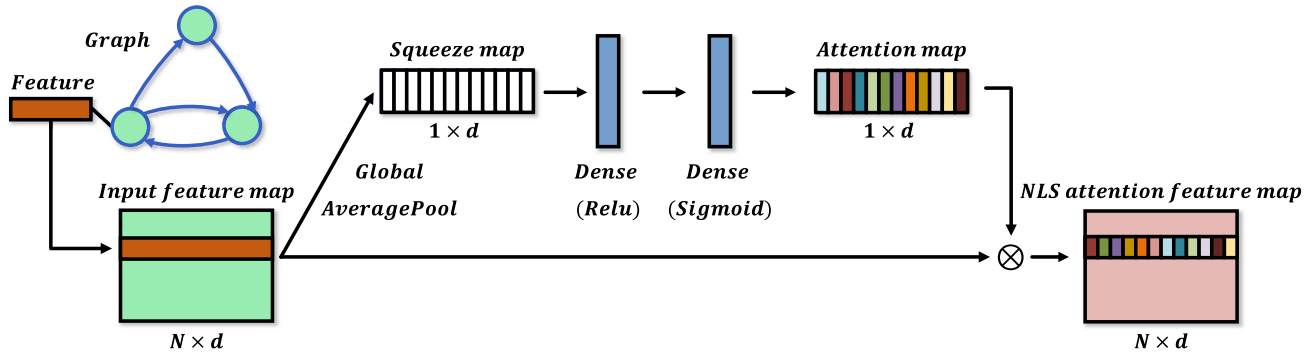


Fig. 6. The procedure of NLS attention mechanism.

full connection i.e., $\sigma(W_2\delta(W_1Z))$ and utilize the sigmoid function to excite it.

$$S = \sigma(W_2\delta(W_1Z)), \tag{8}$$

We regard the weight of the attention map S as the degree of attention to different feature locations after feature dimension selection. Finally, we multiply the weights of the attention map S with the original feature matrix X to get an output of NLS attention.

$$\hat{X} = S \cdot X. \tag{9}$$

E. Topology Adaptive Graph Convolution Network

In this section, we introduce Topology Adaptive Graph Convolution Network (TAGCN) [20] for feature learning, which can directly operate on the graph structure data. TAGCN takes processed feature matrix $\hat{X}$ and adjacency matrix of ICTG as initial input and then learns the node representations. Each node has D-dimension features. The d-th feature of all nodes is presented as vector $\hat{x}_d \in \mathbb{R}^N$, where N denotes the total number of nodes. There are multiple graph filters used in this layer. Let $G_{d,f} \in R^{N\times N}$ be the f-th graph filter. The graph convolution operation can be calculated by multiplying matrix $G_{d,f}$ and vector $\hat{x}_d$, i.e., $G_{d,f}\hat{x}_d$. Then, the f-th output feature map can be calculated by

$$y_f = \sum_{d=1}^{D} G_{d,f}\hat{x}_d + b_f 1_N, \tag{10}$$

where b_f denotes a learnable bias, 1_N denotes an N-dimension vector of all ones [20], [36]. Then we have

$$G_{d,f} = \sum_{k=0}^{K} g_{d,f,k}A^k \tag{11}$$

where $g_{d,f,k}$ denotes the learnable polynomial coefficients of the graph filter; The quantity A^k is a calculated matrix, where $A^k_{i,j}$, denotes the sum of weights of all paths from node j to i with length k. For a node sequence $(v_j, v_{j+1}, \ldots, v_i)$, we define a path of length k from node j to i, and each step of it corresponds to a weighted directed edge in ICTG. Then, the weight of this path can be obtained by multiplying the weights of the k directed edges, i.e., $\phi(p_{j,i}) = \prod_{m=j}^{i} \mathbf{A}_{v_m,v_{m+1}}$. Finally, we sum up the weights of all paths to get $A^k_{i,j}$

$$A^k_{i,j} = \omega\left(p^k_{j,i}\right) = \sum_{j\in\{\tilde{j}|\tilde{j} \text{ is } k \text{ paths to } i\}} \phi(p_{j,i}). \tag{12}$$

Therefore, the final feature map can be expressed as

$$y_f(i) = \sum_{k=1}^{K}\sum_{d=1}^{D} \sum_{j\in\{\tilde{j}|\tilde{j} \text{ is } k \text{ paths to } i\}} g_{d,f,k}\phi(p_{j,i})\hat{x}_d(j) + b_f \mathbf{1}_N. \tag{13}$$

After the graph convolution operation, we feed the output features into a dense layer and use a nonlinear operation $\delta(\cdot)$ to activate it.

$$X_f = \delta(Wy_f), \tag{14}$$

where W is a layer-specific trainable weight matrix and $\delta(\cdot)$ can be a ReLU activation function.

The entire procedures of FT-GCN are briefly summarized as Algorithm 1. The loss function of FT-GCN $\mathcal{L}$ is Log loss and the parameters are updated by Adam optimizer. The loss function is defined as follows.

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)) \tag{15}$$

where y_i and $\hat{y}_i$ represent the true and predicted class of i-th traffic flow, respectively.

V. Evaluation

In this section, we introduce the experimental setting. In our experiments, we use three real-world datasets. Then, we introduce the comparison method and illustrate the implementation of FT-GCN, as well as the performance evaluation metrics. After that, we show the classification performance and training efficiency of FT-GCN with finite labeling rate. Finally, we explore the effect of label rate and time interval on the performance of FT-GCN.

Algorithm 1 Flow Topology Based Graph Convolutional Network

Input: Original traffic flow feature X.
Output: Predicted category $\hat{Y}$ of traffic flows.

```
Initialize all the parameters of TAGCN layers, NLS attention layer and fully connected layers.
Set iteration number I and time interval T.
Construct interval-constrained traffic graph G and obtain adjacent matrix A utilizing rule 1,2,3.
for i = 1, 2, ..., I do
    Input feature matrix X into NLS attention layer to get processed feature matrix X̂.
    Obtain predicted class Ŷ through two TAGCN layers and two fully connected layers.
    Calculate and minimize loss function ℒ.
    Update all the parameters.
end for
return Ŷ.
```

TABLE II
MAIN DATASETS USED ON THE EXPERIMENTS

Datasets	UNSW-NB15	CIC-Darknet2020	ISCXTor2016
Nodes of ICTG	440,044	141,530	66,615
Edges of ICTG	4,177,549	2,453,910	1,543,362
Features	39	75	24

A. Experimental Setting

1) Datasets: We use three datasets representing different network scales. They are UNSW-NB15 [37], DIDarknet [38], and Tor-nonTor [39] datasets which are widely used in network intrusion detection as illustrated in Table II.

- *UNSW-NB15:* This dataset was generated by IXIA tools, which has a hybrid of the real modern normal and the contemporary synthesized attack activities of the network traffic. It contains normal traffic flows and nine prominent attack families like Backdoors, Dos, Worms, *etc.*
- *CIC-Darknet2020:* This dataset is an open-source repository provided by [38]. It includes darknet traffic and corresponding normal traffic from Audio-Stream, Browsing, Chat, Email, *etc.*, which are implemented over Tor and VPN infrastructure.
- *ISCXTor2016:* The current pervasive use of encryption tools makes traffic classification an open challenge. Tor [40] is one of the most popular privacy-enhancing tools. The ISCXTor2016 dataset is a labeled Tor traffic dataset published by UNB (University of New Brunswick) [39], which contains eight types of traffic flows from more than 18 representative applications.

2) Comparative Methods: We compare FT-GCN with 6 popular state-of-the-art methods:

- AdaBoosting [41] is a kind of Ensemble Learning, which trains a series of weak learnable classifiers and combines them into a strong classifier. The maximum number of training iterations for each weak classifier is set to 100.
- K-Nearest Neighbor (KNN [42]) classifies the traffic flow using the Neighbor information. In our experiment, the value of K is set to 5.
- RF (Random Forest [43]) constructs multiple decision trees for model training and prediction to classify traffic flows.
- GCN-TC [13] combines traffic trace graphs and the statistical features of flows and utilizes the GCN model for representation learning. The neighborhood order of traffic flow is set to 2, and the maximum neighborhood edge is 50.
- CNN (Convolutional Neural Network [44]) models network traffic events as time-series data with a million benign and malware connections and takes them as input for classification.
- DNN (Deep Neural Network [45]) is a typical type of Feed Forward Neural Network which has 2 hidden layers.

3) Implementation: We leverage Pytorch and DGL frameworks for the implementation of FT-GCN and GCN-TC. To optimize hyper-parameters of FT-GCN, the number of hidden units for all layers is set to 64, while k is set to 2. Then, we train FT-GCN with 300 epochs. We set the learning rate to 0.01, and take dropout with a 0.5 ratio for the first TAGCN layer to avoid overfitting. Other comparative methods are implemented by the Tensorflow 2.0 and scikit-learn library. The hidden units of CNN, DNN, and GCN-TC are set the same as FT-GCN. And, we add the same dropout layer as used in FT-GCN to these three models. The kernel size of CNN is set to 3. All methods are run 10 times. All experiments were performed on a PC with a 10-core CPU, NVIDIA 3080, and 32GB RAM.

4) Evaluation Metrics: We use 4 standard metrics to evaluate the performance of FT-GCN.

- Accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}, \tag{16}$$

- Recall:

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{17}$$

- Precision:

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{18}$$

- F1-Score:

$$\text{F1-Score} = \frac{2Precision \cdot Recall}{Precision + Recall}, \tag{19}$$

where TP, FP, TN, and FN denote true positive, false positive, true negative, and false negative, respectively.

B. Performance Results

1) Classification Performance: In this experiment, we compared FT-GCN with six state-of-art methods. All the comparison methods have achieved good performance in network intrusion detection. To highlight the performance of each method at low label rates, we set the labeling rate to 5% on all three datasets. In other words, this means that only 5% of traffic flows are labeled. The parameter time interval T of FT-GCN is set to 3 seconds. Fig. 7 illustrates the accuracy of all methods. Moreover, we give more evaluation metrics in Table III. FT-GCN achieves the classification accuracy

TABLE III
PERFORMANCE COMPARED WITH THE STATE-OF-ART METHODS

Methods	UNSW-NB15			CIC-Darknet2020			ISCXTor2016		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
AdaBoosting	0.9389	0.9394	0.9371	0.905	0.908	0.8984	0.9293	0.9327	0.9305
KNN	0.9630	0.9633	0.9631	0.9281	0.9306	0.9286	0.9221	0.9249	0.9232
RF	0.9493	0.947	0.9441	0.9256	0.9274	0.9222	0.9584	0.9594	0.9587
CNN	0.9655	0.9651	0.9653	0.8951	0.9006	0.8954	0.9221	0.9249	0.9232
DNN	0.9713	0.97	0.9704	0.919	0.9213	0.9198	0.9399	0.9421	0.9407
GCN-TC	0.9695	0.9687	0.9689	0.9697	0.9696	0.9697	0.9394	0.9423	0.9400
FT-GCN	0.9866	0.9865	0.9865	0.9839	0.9838	0.9839	0.9681	0.9688	0.9682

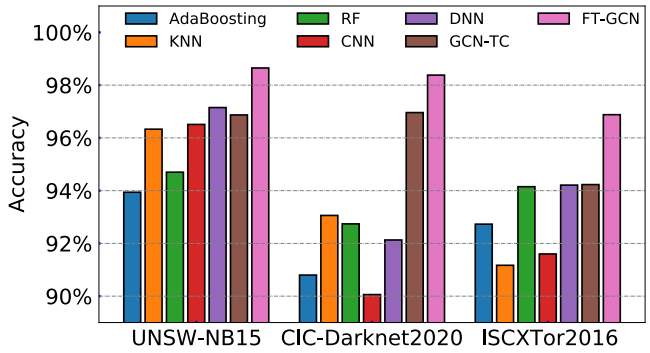


Fig. 7. Accuracy of all methods on three datasets.

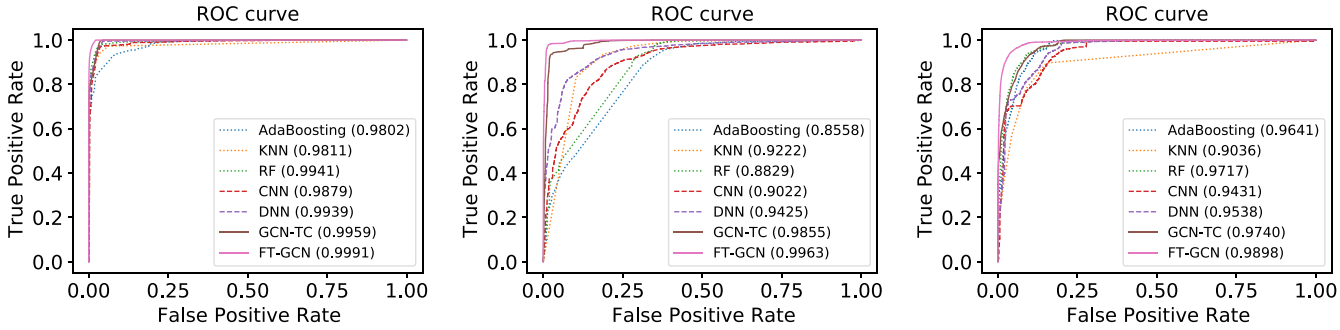
of 98.7%, 98.5% and 96.8% on the UNSW-NB15, CIC-Darknet2020, and ISCXTor2016 datasets, respectively. The accuracy of FT-GCN on the three datasets is at least 2%, 6% and 4% higher than that of shallow machine learn-based methods, i.e., AdaBoosting, RF and KNN. This indicates that the proposed FT-GCN successfully learns a combined representation of the topological and statistical features of the traffic flow. In addition, FT-GCN outperforms other deep-learning methods, such as CNN and DNN on all metrics, which is particularly evident on the CIC-Darknet2020 dataset. The reason may be that the feature dimension of CIC-Darknet2020 is too high, reaching 75. These methods lack the guidance of attention mechanism and can not focus on the key features, resulting in insufficient performance in classification.

In addition, FT-GCN has a better performance than the best remaining baseline GCN-TC. The comparison result demonstrates that the constructed ICTG contains more meaningful information than the traffic graph in GCN-TC. Moreover, we provide the ROC and AUC results of all methods on three datasets in Fig. 8. Obviously, the AUC scores of FT-GCN and GCN-TC are higher than the other methods, reaching above 0.97 on all three datasets. This result indicates that these two methods have better classification performance than other methods under a limited labeling rate. FT-GCN and GCN-TC convert the topological structure of traffic flows into learnable graph data and utilize graph learning technology for classification. The excellent performance of FT-GCN and GCN-TC also verifies that in the case of a limited labeling rate, the topological structure of traffic flows can be used as a supplementary feature to compensate for the lack of learnable information.

2) Efficiency of FT-GCN: To validate the effectiveness of FT-GCN at a limited labeling rate, we conduct experiments using labeling 5% rate data. The parameter T is set to 3 seconds. From Fig. 9, we can observe that the loss and accuracy of the FT-GCN change quickly until about 170 epochs on the three datasets. This means that after 170 epochs, FT-GCN has converged, and its performance has reached the best. It is worth mentioning that the curves of the training set and the validation set are very similar, almost overlapping on the UNSWV-NB15 and CIC-Darknet2020 datasets, while the differences on the ISCXTor2016 datasets are very small. The reason may be that the FT-GCN can quickly learn the combined representation of the topology and statistical features of traffic flows, thereby effectively classifying normal and abnormal behaviors. We can also find that the accuracy of FT-GCN on the ISCXTor2016 dataset is lower than that of the other two datasets. This may be because the feature dimension of this dataset is smaller than that of other datasets. It is difficult for NLS to learn enough knowledge from insufficient feature information. To further demonstrate the effectiveness of FT-GCN, Fig. 10 presents the evolution of precision, recall, and F1-score. The three metrics curves ascend to a plateau quickly, exceeding 0.95 after 100 epochs, which indicates that FT-GCN can converge quickly on different datasets. In summary, FT-GCN can perform well when network scale changes, with fast convergence and the ability to deal with limited labeled data.

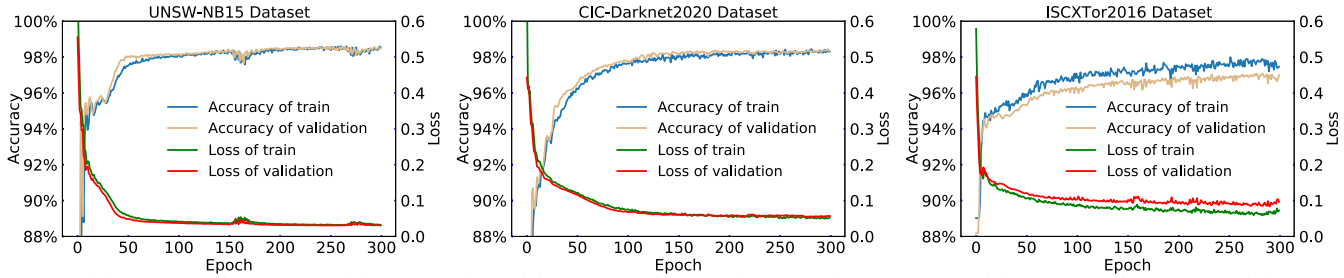
3) Effect of Labeling Rate: In this experiment, we are curious about how FT-GCN performs when the label rate varies. Therefore, we measure the accuracy of FT-GCN and other baselines with various labeling rates in the range of 1%, 5%, 10%, 20%, 50% and 80%. As can be seen from Fig. 11, with the increase of the labeling rate, the accuracy of most methods is improved, which indicates that the proportion of labeled data used for training is an essential factor affecting the accuracy. From Fig. 11, we can observe that the accuracy of FT-GCN is always higher than the other baselines, and the change is significantly reduced when the labeling rate declines from 80% to 1%. It is proved that the attention module in FT-GCN is helpful for classification and makes the model focus on key features. In addition, the experimental results also show that combining the topology and statistical features of traffic flow makes it easier for FT-GCN to learn the representation hidden in the traffic flow relationships, which improves the classification performance under different labeling rates.

From this figure, we can also find that when the labeling rate decreases, the accuracy of GCN-TC changes not significantly, which is similar to that of FT-GCN. This may be because GCN-TC constructs a traffic graph that combines the topological structure and statistical features of traffic flows. In addition, we can see that FT-GCN consistently outperforms



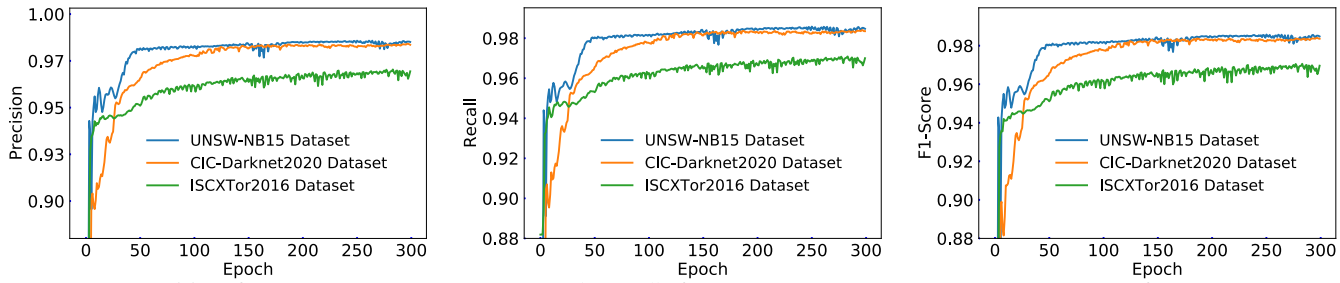
(a) ROC curve on UNSW-NB15 dataset. (b) ROC curve on CIC-Darknet2020 dataset. (c) ROC curve on ISCXTor2016 dataset.

Fig. 8. ROC curve of FT-GCN on the three datasets.



(a) Training process of UNSW-NB15 dataset. (b) Training process of CIC-Darknet2020 dataset. (c) Training process of ISCXTor2016 dataset.

Fig. 9. Loss and accuracy curves of FT-GCN during training.



(a) Precision of FT-GCN. (b) Recall of FT-GCN. (c) F1-score of FT-GCN.

Fig. 10. Precision, recall, F1-socre of FT-GCN during training.

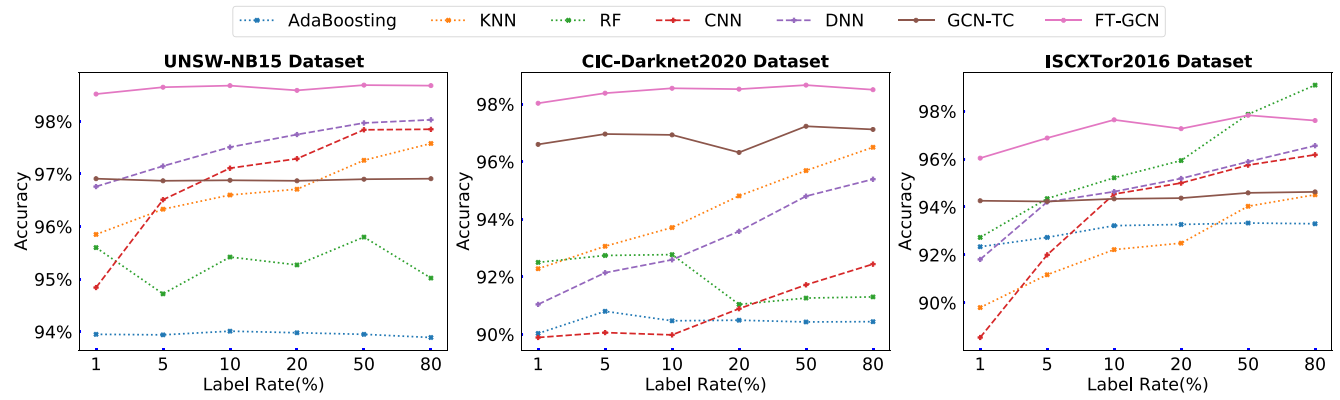


Fig. 11. Accuracy of FT-GCN on three datasets with different labeling rate.

GCN-TC by an average of 2% on accuracy. The reason is that GCN-TC sets the maximum of neighbors of each traffic flow, resulting in the reduction of information in the constructed traffic graph. Instead of setting the maximum number of neighbors per traffic flow, FT-GCN sets different values of T to dynamically control the sparsity of the traffic graph. A larger T makes a node have more neighbors and the ICTG becomes denser. The comparison results also reveal that our

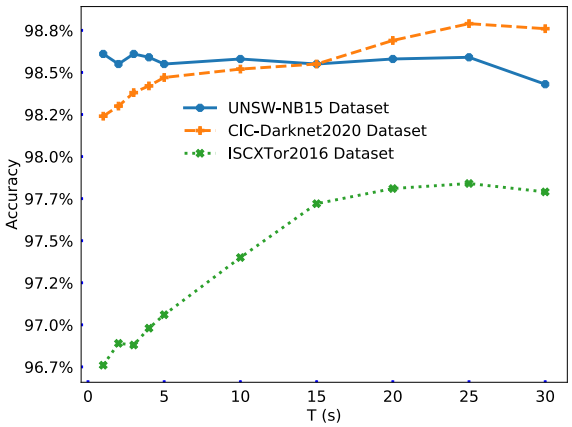


Fig. 12. The impact of time interval T on accuracy.

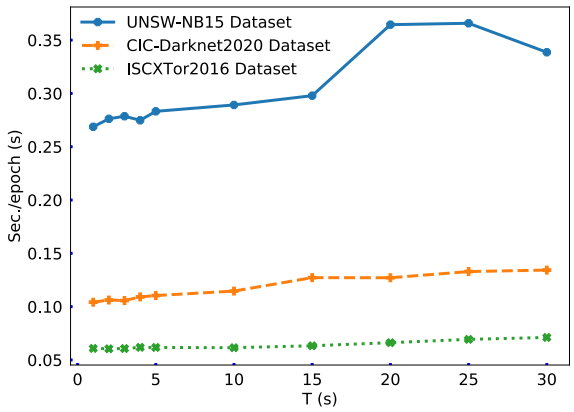


Fig. 13. The impact of time interval T on training time per epoch.

constructed ICTG contains more structural information than the traffic graph used in GCN-TC, which verifies the validity of the rules we designed.

4) Effects of Time Interval T: To describe the topological structures of the traffic flows, we develop an Interval constrained Traffic Graph (ICTG), which limits the generation time interval between traffic flows.The validity of the connection between two traffic flows is affected by the time interval T. As the time interval T becomes larger, the correlation becomes weaker. The time interval T also determines whether the constructed graph G is sparse. A larger value of T will result in more neighbors in the graph G, making the graph denser. Therefore, the time interval T is an important factor affecting the performance of FTGCN. To evaluate the effect of T, we measure the accuracy of FT-GCN and the training time per epoch, with time intervals T chosen from the range of 1, 2, 3, 4, 5, 10, 15, 20, 25, 30. Fig. 12 shows the change of the accuracy of FT-GCN when T increases from 1 second to 30 seconds. It can be observed that when the time interval T ranges from 1 second to 25 seconds, the accuracy of FT-GCN on CIC-Darknet2020 and ISCXTor2016 increases from 96 to 97.84% and from 97 to 98.79%, respectively. This is because a larger time interval T will cause more traffic to be contained in the traffic graph, which helps FT-GCN learn more useful information. Also, we can find that the changes of accuracy of FT-GCN on the UNSW-NB15 dataset are not noticeable. Specifically, when T is set to 30 seconds, the accuracy of FT-GCN on the three datasets decreases slightly. The reason may be that the training graph G contains too many edges, which brings unnecessary redundant information and increases the burden of training FT-GCN, leading to performance degradation. In addition, too many edges will also increase the training time of FT-GCN per epoch and occupy more computing resources. Fig. 13 shows the average training time per epoch at different time intervals T. The results show that with the increase of T, the training time of each epoch increases almost linearly. However, the accuracy of the model is not improved significantly. It also validates the importance of Rule 3 designed in Section IV-C.

VI. Conclusion

In this paper, we proposed a network intrusion detection approach (FT-GCN) for label-limited IoT networks, which converts the intrusion detection into a special node classification task. Specifically, FT-GCN constructs interval constrained traffic graph (ICTG) to exploit the topology of traffic flow, and uses a node-level space (NLS) attention mechanism to enhance the representation of nodes in the traffic graph. In topologically adaptive convolutional networks, the generated ICTG and the processed feature matrix are used for representation learning and classification. The performance of FT-GCN is evaluated on three real datasets representing different network sizes. Experimental results show that FT-GCN can achieve high detection accuracy even under a limited labeling rate, which outperformed other state-of-the-art methods. In our future work, we will optimize the FT-GCN and construct an improved traffic graph where its edges are weighted with different values to represent the strength of correlation between traffic flows. More importantly, the multi-classification task of malicious traffic flows is also the focus of our next research.

Acknowledgment

The authors would like to thank the Editor-in-Chief, the Associate Editor, and the reviewers for their insightful comments and suggestions.

References

[1] A. Hameed and A. Alomary, "Security issues in IoT: A survey," in *Proc. Int. Conf. Innov. Intell. Inform. Comput. Technol. (3ICT)*, 2019, pp. 1–5.
[2] K. Yang, J. Ren, Y. Zhu, and W. Zhang, "Active learning for wireless IoT intrusion detection," *IEEE Wireless Commun.*, vol. 25, no. 6, pp. 19–25, Dec. 2018.
[3] R. Lu, L. Zhang, J. Ni, and Y. Fang, "5G vehicle-to-everything services: Gearing up for security and privacy," *Proc. IEEE*, vol. 108, no. 2, pp. 373–389, Feb. 2020.
[4] P. Schulz et al., "Latency critical IoT applications in 5G: Perspective on the design of radio interface and network architecture," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 70–78, Feb. 2017.
[5] A. Mourad, H. Tout, O. A. Wahab, H. Otrok, and T. Dbouk, "Ad hoc vehicular fog enabling cooperative low-latency intrusion detection," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 829–843, Jan. 2021.
[6] S. S. S. Sugi and S. R. Ratna, "Investigation of machine learning techniques in intrusion detection system for IoT network," in *Proc. 3rd Int. Conf. Intell. Sustain. Syst. (ICISS)*, 2020, pp. 1164–1167.

[7] L. Zhang, G. Feng, and S. Qin, "Intrusion detection system for RPL from routing choice intrusion," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, 2015, pp. 2652–2658.
[8] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. 9th EAI Int. Conf. Bio-Inspired Inf. Commun. Technol. (BICT)*, vol. 3, 2016, pp. 21–26.
[9] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.
[10] M. R. Oliveira, J. Neves, R. Valadas, and P. Salvador, "Do we need a perfect ground-truth for benchmarking Internet traffic classifiers," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2015, pp. 2452–2460.
[11] D. Liao, S. Huang, Y. Tan, and G. Bai, "Network intrusion detection method based on GAN model," in *Proc. Int. Conf. Comput. Commun. Netw. Security (CCNS)*, 2020, pp. 153–156.
[12] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–14.
[13] J. Zheng and D. Li, "GCN-TC: Combining trace graph with statistical features for network traffic classification," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–6.
[14] B. Sun, W. Yang, M. Yan, D. Wu, Y. Zhu, and Z. Bai, "An encrypted traffic classification method combining graph convolutional network and autoencoder," in *Proc. 39th IEEE Int. Perform. Comput. Commun. Conf. (IPCCC)*, 2020, pp. 1–8.
[15] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, vol. 42, 2018, pp. 2011–2023.
[16] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2014, pp. 1–14.
[17] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast Localized spectral filtering," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.
[18] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters," 2017, *arXiv:1705.07664*.
[19] H. Sami, J. Bentahar, A. Mourad, H. Otrok, and E. Damiani, "Graph convolutional recurrent networks for reward shaping in reinforcement learning," *Inf. Sci.*, vol. 608, pp. 63–80, Aug. 2022.
[20] J. Du, S. Zhang, G. Wu, J. M. F. Moura, and S. Kar, "Topology adaptive graph convolutional networks," 2017, *arXiv:1710.10370*.
[21] J. Du, J. Shi, S. Kar, and J. M. F. Moura, "On graph convolution for graph CNNs," in *Proc. IEEE Data Sci. Workshop (DSW)*, 2018, pp. 1–5.
[22] B. Atay, "Intrusion detection with probabilistic neural network: Comparative analysis," in *Proc. Int. Conf. Adv. Technol. Comput. Eng. Sci. (ICATCES)*, 2018, pp. 1–4.
[23] F. Aloul, I. Zualkernan, N. Abdalgawad, L. Hussain, and D. Sakhnini, "Network intrusion detection on the IoT edge using adversarial autoencoders," in *Proc. Int. Conf. Inf. Technol. (ICIT)*, 2021, pp. 120–125.
[24] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, "Method of intrusion detection using deep neural network," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, 2017, pp. 313–316.
[25] L. Chen, X. Kuang, A. Xu, S. Suo, and Y. Yang, "A novel network intrusion detection system based on CNN," in *Proc. 8th Int. Conf. Adv. Cloud Big Data (CBD)*, 2020, pp. 243–247.
[26] S. Das et al., "Network intrusion detection and comparative analysis using ensemble machine learning and feature selection," *IEEE Trans. Netw. Service Manag.*, early access, Dec. 27, 2021, doi: 10.1109/TNSM.2021.3138457.
[27] S. U. Jan, S. Ahmed, V. Shakhov, and I. Koo, "Toward a lightweight intrusion detection system for the Internet of Things," *IEEE Access*, vol. 7, pp. 42450–42471, 2019.
[28] Z. Wu, P. Gao, L. Cui, and J. Chen, "An incremental learning method based on dynamic ensemble RVM for intrusion detection," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 1, pp. 671–685, Mar. 2022.
[29] N. Farnaaz and M. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Comput. Sci.*, vol. 89, pp. 213–217, Jan. 2016.
[30] L. Li-Zhong, L. Zhi-Guo, and D. Xian-Hui, "Network intrusion detection by a hybrid method of rough set and RBF neural network," in *Proc. 2nd Int. Conf. Educ. Technol. Comput.*, 2010, pp. 317–320.
[31] X. H. Yao, "A network intrusion detection approach combined with genetic algorithm and back propagation neural network," in *Proc. Int. Conf. E-Health Netw.*, 2010, pp. 402–405.
[32] B. Gallagher, M. Iliofotou, T. Eliassi-Rad, and M. Faloutsos, "Link homophily in the application layer and its usage in traffic classification," in *Proc. IEEE INFOCOM*, 2010, pp. 221–225.
[33] H. Yang and Z. Zhou, "A novel intrusion detection scheme using cloud Grey wolf optimizer," in *Proc. 37th Chin. Control Conf. (CCC)*, 2018, pp. 8297–8302.
[34] C. F. T. Pontes, M. M. C. de Souza, J. J. C. Gondim, M. Bishop, and M. A. Marotta, "A new method for flow-based network intrusion detection using the inverse Potts model," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1125–1136, Jun. 2021.
[35] M. Chehab and A. Mourad, "LP-SBA-XACML: Lightweight semantics based scheme enabling intelligent Behavior-aware privacy for IoT," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 1, pp. 161–175, Jan./Feb. 2022.
[36] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Graph Fourier transform," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2013, pp. 6167–6170.
[37] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, 2015, pp. 1–6.
[38] A. H. Lashkari, G. Kaur, and A. Rahali, "DIDarknet: A contemporary approach to detect and characterize the darknet traffic using deep image learning," in *Proc. 10th Int. Conf. Commun. Netw. Security*, 2020, pp. 1–13.
[39] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor traffic using time based features," in *Proc. 3rd Int. Conf. Inf. Syst. Security Privacy*, 2017, pp. 253–262.
[40] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proc. 13th Conf. USENIX Security Symp. Vol. 13 (SSYM)*, 2004, p. 21.
[41] W. Hu, W. Hu, and S. Maybank, "AdaBoost-based algorithm for network intrusion detection," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 2, pp. 577–583, Apr. 2008.
[42] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Evaluating effectiveness of shallow and deep networks to intrusion detection system," in *Proc. Int. Conf. Adv. Comput. Commun. Inform. (ICACCI)*, 2017, pp. 1282–1289.
[43] A. Montieri, D. Ciuonzo, G. Bovenzi, V. Persico, and A. Pescapé, "A dive into the dark Web: Hierarchical traffic classification of anonymity tools," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 3, pp. 1043–1054, Jul.–Sep. 2020.
[44] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *Proc. Int. Conf. Adv. Comput. Commun. Inform. (ICACCI)*, 2017, pp. 1222–1228.
[45] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

Xiaoheng Deng (Member, IEEE) received the Ph.D. degree in computer science from Central South University, Changsha, Hunan, China, in 2005. Since 2006, he has been an Associate Professor and then a Full Professor with the Department of Electrical and Communication Engineering, Central South University. He is a Joint Researcher of the Shenzhen Research Institute, Central South University. He is a Senior Member of CCF, and a Member of CCF Pervasive Computing Council, and ACM. He was a Chair of CCF YOCSEF CHANGSHA from 2009 to 2010. His research interests include edge computing, Internet of Things, online social network analysis, data mining, network security, and pattern recognization.

Jincai Zhu was born in Wenzhou, Zhejiang, China, in 1998. He received the B.Sc. degree in software engineering from Xinjiang University, Xinjiang, China, in 2020. He is currently pursuing the master's degree with the School of Software Engineering, Central South University, Changsha, China. Since 2020, he has been engaged in the direction of information security. His major research interests are IoT security and edge computing.

Xinjun Pei is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Central South University, Changsha, China. Since 2017, he has been engaged in the direction of information security. His research interests include wireless communications and networking, mobile security, edge computing, and the Internet of Things.

Lan Zhang (Member, IEEE) received the B.Eng. and M.S. degrees in telecommunication engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2013 and 2016, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, FL, USA, in 2020. She is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Michigan Technological University. Her research interests include wireless communications, vehicular systems, big data analysis, and security and privacy issues for various cyber–physical system application.

Zhen Ling (Member, IEEE) received the B.S. degree in computer science from the Nanjing Institute of Technology, China, in 2005, and the Ph.D. degree in computer science from Southeast University, China, 2014. He is a Full Professor with the School of Computer Science and Engineering, Southeast University, Nanjing, China. His research interests include artificial intelligence of things, mobile system security, network security and privacy, and trusted computing. He won the ACM China Doctoral Dissertation Award and the China Computer Federation Doctoral Dissertation Award in 2014 and 2015, respectively.

Kaiping Xue (Senior Member, IEEE) received the bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC) in 2003, and the Ph.D. degree from the Department of Electronic Engineering and Information Science, USTC, in 2007. From May 2012 to May 2013, he was a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, University of Florida. He is currently a Professor with the School of Cyber Security, USTC. His research interests include next-generation Internet architecture design, transmission optimization, and network security. He serves on the editorial board of several journals, including the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, and the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT. He has also served as a (Lead) Guest Editor for many reputed journals/magazines, including the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, the *IEEE Communications Magazine*, and the IEEE NETWORK. He is an IET Fellow.