FeCoGraph: Label-Aware Federated Graph Contrastive Learning for Few-Shot Network Intrusion Detection

Qinghua Mao, *Graduate Student Member, IEEE*, Xi Lin, *Member, IEEE*, Wenchao Xu, *Senior Member, IEEE*, Yuxin Qi, *Student Member, IEEE*, Xiu Su, *Member, IEEE*, Gaolei Li, *Member, IEEE*, and Jianhua Li, *Senior Member, IEEE*

***Abstract*—With increasing cyber attacks over the Internet, network intrusion detection systems (NIDS) have been an indispensable barrier to protecting network security. Taking advantage of automatically capturing topology connections, recent deep graph learning approaches have achieved remarkable performance in distinguishing different types of malicious flows. However, there remain some critical challenges. 1) previous supervised learning methods rely heavily on abundant and high-quality annotated samples, while label annotation requires abundant time and expert knowledge. 2) Centralized methods require all data to be uploaded to a server for learning behavior patterns, which results in high detection latency and critical privacy leakage. 3) Diverse attack scenarios exhibit highly imbalanced distribution, making it hard to characterize abnormal behaviors. To address these issues, we proposed FeCoGraph, a label-aware federated graph contrastive learning framework for intrusion detection in few-shot scenarios. The line graph is introduced to directly process flow embeddings, which are compatible with diverse GNNs. Furthermore, We formulate a graph contrastive learning task to effectively leverage label information, allowing intra-class embeddings more compact than inter-class embeddings. To improve the scalability of NIDS, we utilize federated learning to cover more attack scenarios while protecting data privacy. Experiment results show that FeCoGraph surpass E-graphSAGE with an average 8.36% accuracy on binary classification and 6.77% accuracy on multiclass classification, demonstrating the efficiency of our approach.**

***Index Terms*—Network intrusion detection, few-shot learning, graph contrastive learning, graph neural networks.**

Received 28 May 2024; revised 14 November 2024 and 30 December 2024; accepted 4 February 2025. Date of publication 13 February 2025; date of current version 26 February 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62202302 and Grant U21B2019. The associate editor coordinating the review of this article and approving it for publication was Dr. Weizhi Meng. *(Corresponding author: Xi Lin.)*

Qinghua Mao, Xi Lin, Yuxin Qi, Gaolei Li, and Jianhua Li are with Shanghai Key Laboratory of Integrated Administration Technologies for Information Security, School of Electronic Information and Electrical Engineering, Institute of Cyber Science and Technology, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: mmmm2018@sjtu.edu.cn; linxi234@sjtu.edu.cn; qiyuxin98@sjtu.edu.cn; gaolei_li@sjtu.edu.cn; ljh888@sjtu.edu.cn).

Wenchao Xu is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, and also with Shenzhen Research Institute, The Hong Kong Polytechnic University, Shenzhen 518057, China (e-mail: wenchao.xu@polyu.edu.hk).

Xiu Su is with the Big Data Institute, Central South University, Changsha 410075, China, and also with the School of Computer Science, Faculty of Engineering, The University of Sydney, Sydney, NSW 2006, Australia (e-mail: xisu5992@uni.sydney.edu.au).

Digital Object Identifier 10.1109/TIFS.2025.3541890

I. Introduction

THE last few decades have witnessed an increasing number of cyberattacks towards Internet infrastructures around the world. Consequently, network intrusion detection systems (NIDS) have been an essential component in rapidly detecting emerging security threats. There are two kinds of methods, including signature-based [1] and anomaly-based methods [2]. While signature-based methods suffer from low detection rates and poor scalability, anomaly-based approaches have broader prospects with the potential to detect emerging and diverse attacks.

Previous studies have introduced deep learning methods to solve NIDS problems. High-level discriminative features are extracted from statistical features for classification. Although they have achieved excellent performance, they don't consider the relation between network flows. Some advanced attacks launched by a cluster of computers, such as distributed denial-of-service (DDoS) attacks [3], can be depicted as host interactions. When we convert network traffic into a graph, these host interactions can be captured by graph neural networks (GNNs) to enhance the topology-aware expressive capabilities [4].

The last few years have seen an increased number of GNN-based solutions for network intrusion detection. Some studies proposed tailored GNN models for automatically detecting malicious botnet nodes [5], [6]. However, only the structural information is used. Regarding intrusion detection as an edge classification problem, E-GraphSAGE [7] reformulated GraphSAGE to aggregate neighboring edge features. Chang et al. [8] incorporated the residual connection into E-GraphSAGE to tackle the imbalance issue. Other works are based on time-spatial analysis, with the time correlation-aware graph [9] or the interaction history [10]. The above methods are used as supervised models, requiring a huge amount of labeled data. However, annotating numerous labels is a time-consuming and labor-intensive task, hindering adaptation to unknown attacks.

Recent studies integrate self-supervised learning techniques into GNN models in the absence of labeled data. In Anomal-E [11], deep graph infomax objective [12] is leveraged to increase the amount of shared information between local and global representation. TS-IDS [13] enhances the expressive

ability of edge embedding by predicting nodes' properties based on the volume of traffic traversing them. Compared with Anomal-E, TS-IDS provides an end-to-end training paradigm, which obviates the need for selecting the pooling function [14] and optimizing the encoder and classifier separately.

Although these two approaches have demonstrated considerable performance without requiring labeled data, they still encounter three critical challenges. First, recently attack behaviors usually exhibit intricate patterns, while limited network traffic samples are available. This might increase the difficulty of malicious network flow detection with restricted samples. Secondly, cyber attacks usually demonstrate imbalanced distribution and vague boundaries between different attack scenarios, making it hard to accurately characterize abnormal behavioral patterns. Considering long-term APT attacks as an example, the attackers can constantly adjust attack strategies until reaching their goal of extracting critical information, leading to the complexity and chronicity in extracting intrinsic features of such an attack. Finally, existing centralized NIDS approach will inevitably cause heavy computation overheads and high latency, thereby resulting in detection delay of abnormal behaviors and poor scalability in large-scale networks. Since network flow data are collected from different institutions, centralized data transmission and storage also heavily incurs privacy concerns [15]. Given the challenges brought by diverse emerging attacks, limited number of available samples and privacy concerns about distributed data, it is of great significance to develop an effective federated graph learning framework towards label-scarce NIDS scenario.

To address the above challenges, we propose FeCoGraph, a novel federated graph contrastive learning framework for network intrusion detection in few-shot scenarios. We construct a label-aware graph contrastive learning module to efficiently exploit scarce labeled samples. Intra-class similarities and inter-class differences are extracted via a semi-supervised learning strategy, thereby enhancing the discriminative ability of network flow embeddings. Furthermore, we incorporate the label-aware graph contrastive module into a federated NIDS framework to improve detection efficiency for abnormal activities, while encouraging knowledge sharing in a privacy-preserving way. The main contribution of our work can be summarized as follows:

- In this paper, we propose a label-aware federated graph contrastive learning framework to efficiently utilize scarce labels and non-IID data in federated NIDS. The proposed federated NIDS can maintain accurate and efficient performance even with few-shot labels.
- We propose a label-aware graph contrastive learning strategy. With the proposed method, both inter-view and intra-class similarities are extracted from network flows by minimizing label-aware positive pairs' distances, thereby obtaining robust and discriminative flow embeddings.
- Additionally, a personalized FL algorithm is employed to allow collaborative training between distributed devices in a privacy-preserving manner. The bi-level optimization strategy can alleviate data heterogeneity of different scenarios and boost detection capability of distributed NIDS.
- We conduct extensive experiments on three recently released NIDS public datasets. Experimental results show that FeCoGraph surpasses E-graphSAGE and Anomal-E with an average 5.30% accuracy.

The remaining sections of the paper follow this structure. Section II covers the review of related works. Section III introduces the system model and Section IV elaborates on the design rationale and essential components of FeCoGraph. Then Section V evaluates FeCoGraph and discusses the experimental results. Section VI concludes the whole paper.

II. Related Work

A. Machine Learning-Based Intrusion Detection

While signature-based methods cannot easily generalize to emerging attacks, anomaly-based schemes can identify unknown threats. Machine learning-based approaches have received much attention in anomaly-based NIDS. These approaches include both conventional machine learning and deep learning approaches.

For traditional machine learning solutions, feature engineering is first adopted to extract features from abundant network flows, along with a shallow classifier. Various machine learning methods have been utilized to build NIDS, including K-nearest neighbors (KNN), support vector machine (SVM), naive Bayes, and decision trees. The work by Wang et al. [16] proposed a two-stage IDS method, in which SVM and a density-based clustering algorithm are used for classification. Gu et al. [17] implemented a naive Bayes model to process original data for obtaining high-quality data with obvious feature categories. Ding et al. [18] investigated the explainable artificial intelligence issue by utilizing the decision tree model, thereby enhancing trust management in intrusion detection. The limitation of the above methods lies in feature engineering. Abundant time and domain knowledge are necessary to select and process appropriate features.

On the contrary, deep learning (DL) methods [19] automatically extract flow features from raw data. Current DL-based methods generally process network traffic in the form of packet sequences. For example, Vinayakumar et al. [20] extensively evaluated the performance of CNN and its variant architectures on intrusion detection and demonstrated the capability of extracting high-level abstract features. Jiang et al. [21] proposed a multi-channel LSTM-based detection method to explore the temporal correlations between network sequences. By merging the strengths of CNN and RNN, Hasson et al. [22] suggested a hybrid model to extract meaningful local features and retain long-term dependencies among them. Some researchers address network anomaly detection with unsupervised methods such as Deep Belief Network (DBN) [23] or various iterations of Auto Encoder (AE) [24]. Other works exploited advanced deep learning techniques, including meta-learning [25], reinforcement learning [26], to realize improved detection performance. Nevertheless, current deep learning-based techniques primarily concentrate on extracting statistical features from network flows in isolation, leading to a deficiency in modeling intricate topology patterns within multiple types of traffic flows.

TABLE I
COMPARISON OF GNN-BASED INTRUSION DETECTION METHODS

Reference	Data Format	Available Information	Training Process	Few-shot paradigm
[7]	Graph	Topological information, Network statistics	Supervised learning	None
[8]	Graph	Topological information, Network statistics	Supervised learning	Residual learning
[11]	Graph	Topological information, Network statistics	Unsupervised learning	Contrastive learning
[13]	Graph	Topological information, Network statistics	Unsupervised learning	Predictive learning
[9]	Interval-constrained traffic graph	Topological information, Network statistics, Temporal associations	Supervised learning	None
[10]	Spatiotemporal graph	Topological information, Network statistics, Temporal associations	Semi-supervised learning	Transductive training
Our work	Graph	Topological information, Network statistics	Semi-supervised learning	Label-aware contrastive learning

B. GNN-Based Network Intrusion Detection

Network traffic can be naturally formed as a graph, where entities (computers, routers) represent nodes and their interaction denotes edges. Since intrusion behaviors usually manifest as suspicious patterns underlying the interactions between network entities, complex structure patterns are essential for identifying specific attack types, such as advanced persistent threats (APT), whose signals are too weak to effectively detect [27].

With the ubiquitous application of graph neural networks (GNN), the last few years have witnessed an increasing number of graph-based NIDS approaches. Zhou et al. [5] propose a tailored GNN to automatically detect botnets by capturing the intrinsic properties of botnet structure. Similarly, XG-BoT [6] integrates a reversible residual connection into a graph isomorphism network for detecting malicious botnet nodes in large-scale networks. Automatic network forensics is performed with a GNNExplainer to help identify abnormal subgraphs and corresponding botnet nodes. However, only the topological structure is considered, leaving node or edge attributes unused.

GCN-TC [28] is One of the initial studies to integrate both flow statistical features and graph structure simultaneously in the network traffic classification. It can enhance classification performance even with a limited amount of labeled data. The work by Lo et al. [7] proposed E-graphSAGE, which formulates NIDS into an edge classification problem. In E-graphSAGE, node representations are acquired by aggregating edge attributes from the sampled neighborhood nodes. Although topology information and network flow features are utilized simultaneously, the processing of mapping the original source IP into random IP addresses would inevitably disturb the spatial distribution of network flows. Chang et al. [8] proposed to incorporate residual learning into the original E-graphSAGE architecture, to retain the original graph feature and improve the performance on minority class samples. Meeting the challenge of limited labeled data in large-scale IoT networks, some recent studies have leveraged the temporal correlation between network flows. Deng et al. [9] constructed an interval-constrained traffic graph and enhanced statistical and structural features through a topology adaptive GCN. Duan et al. [10] proposed a dynamic line graph neural network, which is utilized to extract structural information from spatial-temporal graphs and the interaction history of IP addresses from successive snapshots. However, the above supervised learning approaches require extensive labeled data, which limits their ability to detect unfamiliar attacks and scale effectively in large-scale IoT environments.

Recent works have investigated how to build effective GNN-based NIDS given few-shot labeled samples. Hu et al. [29] proposed to extract flow graph features using subgraph topology from a small number of initial interactive packets, achieving fast and accurate intrusion detection. Anomal-E [11] adapts E-graphSAGE to self-supervised learning with Deep Graph Infomax (DGI) [12]. TS-IDS proposed by Nguyen et al. [13] incorporates a predictive-based self-supervised learning module to enrich the node embedding by categorizing endpoint nodes based on the volume of traversing traffic. Unlike the previous Graph-based network intrusion detection methods, we suggest a label-aware graph contrastive learning framework to tackle the challenges of few-shot learning and imbalanced data in NIDS.

FeCoGraph differs from existing methods in three aspects. Firstly, compared with supervised GNN approaches [7], [8] and spatial-temporal GNN methods [9], [10], we consider a more generic and realistic scenario, *e.g.*, few-shot network intrusion detection based on static attack flow graph analysis. Our method is built on semi-supervised learning scenario with only a few labeled samples. Secondly, compared with self-supervised GNN methods [11], [13], the proposed label-aware graph contrastive learning strategy could learn robust and discriminative flow embeddings by exploiting inter-class differences and intra-class similarities. Our method facilitates the effective utilization of both labeled and unlabeled data. Finally, we particularly investigate the impact of graph-specific non-IIDness in federated learning and propose an effectively personalized federated learning strategy. A comprehensive comparison of GNN-based intrusion detection methods is shown in Table I.

C. Contrastive Learning on Graphs

Contrastive learning is a self-supervised learning approach that focuses on acquiring discriminative representations. It achieves this by reducing the embedding distance between positive pairs while increasing the embedding distance from negative samples. Augmentation techniques to generate negative samples and contrastive learning objectives have been extensively studied in the vision domain [30], [31], [32], [33], while there are a relatively limited number of researches on graph contrastive learning. Deep Graph Infomax (DGI) [12] focused on maximizing the mutual information between local node-level embedding and global graph-level embedding. GRACE [34] performed node-level contrastive learning on augmented graphs by randomly dropping edges and masking node attributes. Based on GRACE, GCA [35] further explored adaptive augmentation schemes to preserve important edges

and feature dimensions. GraphCL [36] performs data augmentation by subgraph sampling, randomly disturbing nodes or edges, and feature masking. Apart from contrasting positive and negative pairs without any label information, some recent studies used labels to improve self-supervised learning performance. Inspired by supervised contrastive learning, Wan et al. [37] contrasted the output of GCN and hierarchy GCN through a combination of supervised contrastive loss and generative loss. Akkas et al. [38] proposed a joint contrastive learning strategy to exploit the benefits of both unlabeled data and limited labeled data.

D. Federated Learning Over Graphs

The integration of federated learning and graph neural networks has received unprecedented attention in recent years, with an extensive courage including recommender systems [39], [40], molecular graphs [41] and fraud detection [42]. Existing literature can be divided into two categories, *i.e.*, intra-graph FL where each client owns a set of graphs and intra-graph FL where each client owns a subgraph from an entire graph. Data heterogeneity is a critical challenge in FL, while graph-specific data heterogeneity is less explored. Fedgraphnn [43] and Spreadgnn [41] simulate graph-level non-IIDness in distributing graph datasets to multiple clients. GraphFL [44] addresses the issue of graph-level non-IIDness and novel domain occurrence with model-agnostic meta-learning. FedStar [45] enables the structural information sharing in FL via feature-structure disentanglement. In the NIDS scenario, related works have been proposed to enable intelligent intrusion detection via federated graph learning. Given the distributed and heterogeneous nature of network data and the need for privacy protection, it is reasonable to model the distributed network environment using subgraph federated learning. Zhang et al. [46] propose a GNN-based CAN bus NIDS to simultaneously detect multiple CAN attacks, where a two-stage classification is deployed to handle high skewed data. Additionally, federated learning is utilized to cover diverse driving scenarios while protecting privacy. Krishnan et al. [47] propose a federated GNN-based framework to facilitate low probability of detection in the wireless network. FedADSN [48] builds up a decentralized FL framework for anomaly detection in social networks. However, none of previous works consider effective federated GNN countermeasure against the realistic but challenging few-shot scenario.

III. System Model and Threat Model

A. System Model

We consider a typical IoT environment consisting of terminal devices, edge servers, and gateway routers. The edge servers are responsible for real-time data storage and intrusion model construction, alleviating the computational overload on cloud servers. Gateway routers play a crucial role in connecting heterogeneous devices, data forwarding, and protocol conversion. Each IoT device is capable of transforming sensory data into network traffic packets, which will be transmitted via wireless communication protocols. A sequence of packets between a source and a destination node during an interval constitutes a network flow, which is characterized by the source IP address, source port, destination IP address, destination port, and other statistical features, such as the incoming number of bytes. Most network flows cannot be directly transmitted from the source IP to the destination IP. There are usually multiple forwarding devices on the path. All the network flows that pass through them are captured by forwarding devices, whether to be allowed or dropped depends on the flow table.

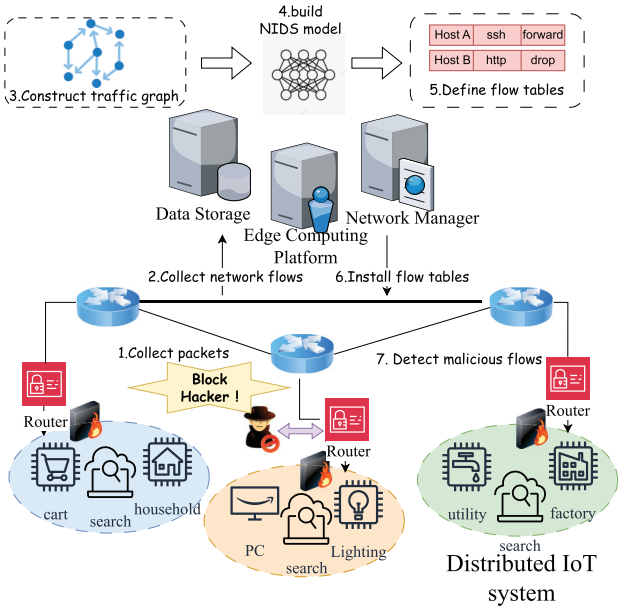
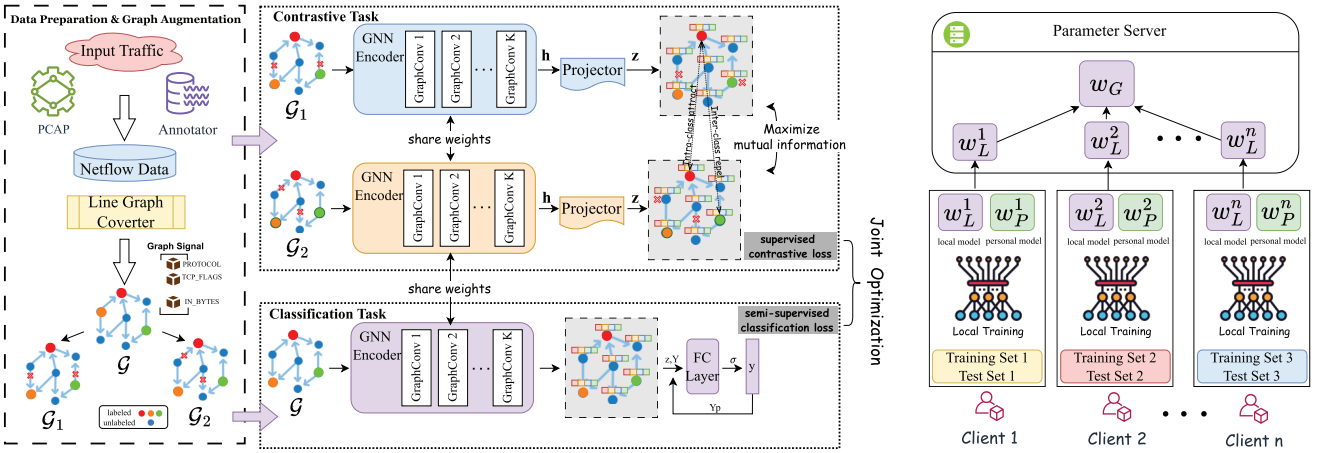


Fig. 1. The overview of threat model. A central server aggregates NIDS models and develop traffic blocking rules. Each local gateway manages IDS for collecting network flows and detecting abnormal IoT devices.

The NIDS is usually achieved by the collaboration between edge servers and gateway routers. The workflow of NIDS is composed of three main steps, including data collection, intrusion detection, and alarm response:

- Data collection: numerous network packets are captured by data collectors deployed on gateway routers. Then network packets are transformed into network flows and subsequently uploaded into an edge-level database. Note that each edge device is served as a local client in federated learning, which has the ability to continuously collect network flows from interactions with other devices. Network flows owned by different devices are heterogeneous and class imbalanced.
- Intrusion detection: In federated learning scenario, each edge device keeps local network flows private and only uploads model parameters to the edge server. The edge server aggregates those local parameters to construct a global GNN model for network flow classification and intrusion detection. In this way, edge server can conduct comprehensive intrusion threat analysis with sufficient data in a privacy-preserving manner.
- Alarm Response: The edge server generates flow tables on prediction results with global GNN model. Then flow tables are distributed to edge devices and installed on gateway routers to determine whether to allow or discard



(a) Local NIDS Model Training (b) Federated Learning Process

Fig. 2. The NIDS framework of label-aware federated graph contrastive learning. A label-aware graph contrastive learning module is incorporated into a personalized FL process.

network flows. When a traffic flow is flagged as abnormal, its source host is marked as malicious. Subsequently, all traffic flows originating from this host are blocked by the flow tables. Consequently, these gateway routers will discard traffic flows sent by a malicious device.

B. Threat Model

We consider threat model in a typical IoT environment. Both the edge server and IoT devices are semi-honest and non-colluded. They are honest in adhering to agreements. Although they are curious about the private data of other entities, they cannot infer private data from the specific individuals. The primary threats come from external sources, such as hackers who target the IoT network. The objective of an attacker is to gain access to sensitive data or inflict significant damage towards local devices or the whole network, thereby compromising data integrity and service availability. The capabilities of attackers include sending manipulated flows or launching a series of malicious activities, such as sniffing, DoS, backdoor, and brute-force attacks. The hostile operations are conducted to take control of benign nodes, and the infected nodes will send a larger amount of malicious flows to the remaining normal nodes, ultimately leading to the disorders and collapse of the entire network system.

IV. Label-Aware Federated Graph Contrastive Learning

We propose a novel federated graph contrastive learning method to tackle the challenges of label deficiency and imbalance in NIDS. Informative self-supervised signals and discriminative class label-aware signals are collaboratively extracted from the clients' network, under a federated learning framework. This section systematically presents FeCoGraph, the proposed method.

A. An Overview of FeCoGraph

An overview of the proposed federated NIDS framework is shown in Fig. 2. The framework is composed of the following key components:

- Graph Construction. The initial traffic graph is converted into its line graph structure for generating node-level flow embeddings. Network flows in the original traffic graph are transformed into nodes in the line graph, while an edge is correspondingly created between two nodes in the line graph if two flows share a common host IP address.
- Model Architecture. The backbone model is divided into two branches for supervised classification and contrastive learning, respectively. Three parts are included in the overall architecture Encoder, projector, and classifier. The encoder is shared by both branches, each of which extracts node-level representations from line graph data. The projector then maps these representations to a space where the contrastive loss is calculated. The classifier is used to discriminate benign and malicious flows.
- Adaptive Graph Augmentation. Graph augmentation is a critical step in learning robust representations that are invariant to perturbations. We adopt an adaptive augmentation strategy to obtain two graph views, which consider the impact of nodes and edges and encourage the model to learn intrinsic patterns underneath the input graph.
- Label-aware Graph Contrastive Learning. To reduce the distance between embeddings from the same class, contrastive pairs are formulated based on sample selection and corresponding labels. Positive pairs are sampled with data from the same class, while data from different classes are regarded as negative pairs. We propose to optimize the model with a joint learning strategy, where the loss function consists of cross-entropy loss and label-aware contrastive loss function.
- Federated Learning. We utilize personalized federated learning to enable collaborative knowledge sharing in a privacy-preserving manner. Each local client owns a subgraph consisting of network flows, which usually follow not independent and identically distributed (non-IID) distribution. Each local client first learns a local model with its local traffic graph and only uploads the local models to the server for parameter aggregation.

B. The Workflow of Federated Graph Contrastive-Based NIDS

1) Graph Construction: Network flows can be converted into a global traffic graph to model the interactions between different hosts. The source and destination IP are identified as nodes and network flows are constructed as edges, with edge attributes containing statistical fields like protocol type, incoming number of bytes, *etc*. In this way, we formulate network intrusion detection as an edge classification problem.

However, while graph convolution layers have shown excellent capabilities in generating node embeddings, they cannot directly extract sufficient edge features for network intrusion detection. Motivated by the line graph structure in graph theory, we convert the original graph into the corresponding line graph structure. To be specific, each edge in the original graph $\mathcal{G}$ corresponds to a node in the line graph $\mathcal{L}(\mathcal{G})$; for every two edges sharing a common host IP, an edge is created between the corresponding two nodes in the line graph $\mathcal{L}(\mathcal{G})$.

Compared to the original graph $\mathcal{G}$, line graph structure $\mathcal{L}(\mathcal{G})$ offers several advantages. First, every single node v_i corresponds to $d_i(d_i-1)/2$ links in the line graph $\mathcal{L}(\mathcal{G})$, which suggests that the line graph will pay more attention to the nodes frequently communicating with others. The increased connection density encourages graph convolution to aggregate more information from neighbors. Additionally, the line graph reduces the quantity of large eigenvalues in the Laplacian matrix of $\mathcal{L}(\mathcal{G})$ with a redundant spectrum, which enhances the numerical stability of the graph convolution process.

2) Adaptive Graph Augmentation: To preserve the input graph's inherent topological patterns and node attributes, we adopt an adaptive graph augmentation strategy to generate corrupted graph views. Previous graph augmentation schemes usually suffered from not capturing the impact of influential nodes and edges. The behavior patterns extracted from influential flows are conducive to analysis.

To generate positive and negative samples for contrastive learning, we generate two different but correlated graph views with stochastic augmentation function $\check{\mathcal{G}}_1 = t_1(\mathcal{G})$ and $\check{\mathcal{G}}_2 = t_2(\mathcal{G})$, where t_1 and t_2 are sampled from augmentation function set $\mathcal{T}$. In the GCA model, augmentation schemes are designed to preserve influential attributes and structures, while perturbing less important edges or features. Specifically, the probabilities of dropping edges and masking features are higher for less important feature dimensions and edges, and lower for important counterparts.

For topology-level perturbation, the augmented edge set is formulated with probability

$$P\{(u,v) \in \tilde{\mathbb{E}}\} = 1 - p^e_{uv}, \tag{1}$$

where $(u,v) \in \mathbb{E}$ and p^e_{uv} is the probability of removing (u,v) and $\tilde{\mathbb{E}}$ is the augmented edge set. We leverage widely-used node centrality metrics to connect p^e_{uv} with the importance of the edge (u,v). Specifically, edge centrality is defined based on the centrality of two correlated nodes.To simplify, this can be expressed as the average centrality value between two nodes in an undirected graph. $w^e_{uv} = (\varphi_c(u) + \varphi_c(v))/2$, or the centrality of the destination node on a directed graph as $w^e_{uv} = \varphi_c(v)$. Then we calculate the probabilities of dropping edges based on edge centrality values. We take the logarithm of edge centrality to mitigate the issue caused by nodes with extremely dense connections, *i.e.*, $s^e_{uv} = \log w^e_{uv}$. The drop probability of edge (u,v) can be obtained with a normalization step

$$p^e_{uv} = \min\left(\frac{s^e_{max} - s^e_{uv}}{s^e_{max} - \mu^e_s} \cdot p_e, p_\tau\right), \tag{2}$$

where p_e is a scaling hyper-parameter that controls the scale of the probability of removing edges, s^e_{max} and μ^e_s denotes the maximum and average of s^e_{uv}, $p_\tau < 1$ is a cut-off probability.

For attribute-level augmentation, node attributes are perturbed with a random mask vector $\tilde{\mathbf{m}} \in \{0,1\}^F$

$$\tilde{X} = [\mathbf{x}_1 \circ \tilde{\mathbf{m}}; \mathbf{x}_2 \circ \tilde{\mathbf{m}}; \ldots; \mathbf{x}_N \circ \tilde{\mathbf{m}}], \tag{3}$$

where F is the feature dimension. Each dimension of the mask vector is independently drawn from a Bernoulli distribution as $\tilde{m}_i \sim \text{Bern}(1 - p^f_i)$.

Similar to topology-level augmentation, the mask probability p^f_i is designed to gauge the significance of the i-th dimension of node features. Our assumption is that crucial feature dimensions tend to be more prevalent in influential nodes. It is reasonable because each dimension of node features corresponds to a statistical feature of each network flow. Then, statistical features that occur frequently in influential network flows should be regarded as important features. Specifically, for any node u with continuous features and the corresponding feature dimension i, the dimension weights can be written as

$$w^f_i = \sum_{u \in \mathcal{V}} |x_{ui}| \cdot \varphi_c(u), \tag{4}$$

where $\varphi_c(u)$ is a node centrality value. Next, we calculate the probability of masking dimension i based on the feature dimension weights. First, we take the logarithm of feature dimension weight to mitigate the overflow issue caused by nodes with extremely dense connections. Then, the feature mask probability is obtained by performing normalization on the weights s^f_i

$$p^f_i = \min\left(\frac{s^f_{max} - s^f_i}{s^f_{max} - \mu^f_s} \cdot p_f, p_\tau\right), \tag{5}$$

where $s^f_i = \log w^f_i$, p_f is a hyper-parameter that controls the scale of the mask probability, s^f_{max} and μ^f_s denotes the maximum and average of s^f_i, $p_\tau < 1$ is a cut-off probability.

3) Model Architecture: We propose a label-aware graph contrastive module with two branches to enhance the capability to discriminate benign and malicious flows. One branch of the model is used for classification via cross-entropy loss, while another branch of the model is used for contrastive learning via supervised contrastive loss. Since a line graph has transformed network flows into nodes, the node-level graph neural networks, e.g. GCN, can be adapted as backbone models. This eliminates the need for complex neighbor aggregation on edge features. The model comprises an encoder, a projector, and a classifier. Here is a detailed description of each component:

1) Encoder. The encoder is leveraged to learn node representations by aggregating information from neighborhood nodes. The aggregation process at the k-th layer can be expressed as

$$\mathbf{h}_{\mathcal{N}(v)}^{k} = AGG_k(\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)), \tag{6}$$

$$\mathbf{h}_v^k = (\mathbf{W}_k \cdot CONCAT(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)), \tag{7}$$

where $\mathbf{h}_{\mathcal{N}(v)}$is the l-layer representation of node v, and $h_v^0 = x_v$ is the node feature. $\mathcal{N}(v)$ denotes neighbors of node v, AGG$(\cdot)$ denotes the aggregation function which can vary for different GNN models and W_k denote learnable weights at the k-th layer. Given the raw node feature $X \in \mathbb{R}^d$, the node embedding generated by encoder layers can be formulated as $h = Encoder(X) \in \mathbb{R}^h$. Then node representations are subsequently shared by projection layers and classification layers.

2) Projector. The role of projection layers is to map embedding into a representation space with fixed dimension. The projector is implemented as a two-layer multilayer perceptron (MLP), thereby enhancing the quality of representations generated by the encoder. The output embeddings of projection layers are further normalized to be situated on the unit hypersphere, which allows us to measure the distance by employing a dot product in the projection space. Note that the projector is expected to be discarded during the inference phase

3) Classifier. The classifier is implemented with a multilayer fully connected neural network, which makes the prediction probabilities of network flows corresponding to each category. The final prediction probabilities are utilized to classify the attack type of test samples.

4) Label-Aware Graph Contrastive Learning Task: Existing contrastive learning approaches have improved the generalization of graph representation learning in a self-supervised manner, but they lack the exploitation of limited but valuable label information. Considering the challenges of obvious class imbalance and ambiguous network flow features in an ever-changing network environment, label-aware contrastive learning is adopted to further condense intra-class traffic samples in the embedding space and further clarify the decision boundary between normal and abnormal flows.

Motivated by visual supervised contrastive learning [33], suppose the original graph contains N points, there are $2N$ sample points in two augmented views. Given an anchor embedding z_i, more than one samples are known to belong to the same class. Samples from the same class in two views are considered as positive samples, while samples from different classes in two graph views constitute negative samples. The label-aware graph contrastive loss can be defined as:

$$\mathcal{L}_{supcon} = \frac{1}{2N}\sum_{i=1}^{2N}\mathcal{L}_i^{supcon}, \tag{8}$$

$$\mathcal{L}_i^{supcon} = \frac{1}{2N}\sum_{i=1}^{2N}\frac{-1}{|P(i)|}\sum_{p\in P(i)}\log\frac{e^{(z_i\cdot z_p/\tau)}}{\sum_{i=1}^{2N}\mathbb{1}[k\neq i]e^{(z_i\cdot z_a/\tau)}}. \tag{9}$$

Note that $P(i)$ denotes the set of positive samples which belong to the same class and $|P(i)|$ denotes the number of positive samples for anchor embedding z_i. The optimization of supervised contrastive loss shortens the distance of embeddings from the same class and enlarges the distance of embeddings from other classes. The most computationally intensive part is to calculate the loss function based on cosine similarity scores of positive and negative pairs. The computation complexity of contrastive loss is $O(N \cdot M)$, where N is the batch size and M is the number of negative samples per positive sample. The computation complexity of cosine similarity is $O(D)$, where D is the dimension of the feature vectors. Consequently, the overall computation complexity is $O(N \cdot M \cdot D)$.

For another branch, the cross-entropy loss is deployed for flow representation classification, which is defined as:

$$\hat{y} = f(\mathbf{x}) = softmax(W \cdot Enc(\mathbf{x}) + b), \tag{10}$$

$$\mathcal{L}_{ce} = -\sum_{i=1}^{n}\sum_{j=1}^{c} y_{ij} log(\hat{y}_{ij}). \tag{11}$$

Note n and c denote the number of training samples and classes, y and $\hat{y}$ denote true labels and predictions. The final loss is a combination of supervised cross-entropy loss and supervised contrastive loss, which simultaneously encourages correctly classifying the labeled network flows and obtaining a more discriminative representation:

$$\mathcal{L} = (1 - \lambda_{ce}) \cdot \mathcal{L}_{supcon} + \lambda_{ce} \cdot \mathcal{L}_{ce}. \tag{12}$$

5) Federated Learning: In this paper, we adopt two classic federated learning schemes, *i.e.*, FedAvg [49] and Ditto [50], which represent general FL and personalized FL approaches. The model update process of FedAvg is described as follows. Each participant owns his private local dataset, which cannot be shared with other participants. The overall training process can be achieved between the cloud server and clients for a number of communication rounds. In each communication round, local training and parameter aggregation are executed alternately. Each client $\mathcal{C}_k$ first trains a local GNN model W_k with the client's private network traffic graph $\mathcal{G}_k$. Specifically, the derived loss, denoted as $f(W_k, \mathcal{G}_k)$, is computed following the supervised contrastive loss function in Eq 12. During the local model update stage, the mini-batch gradient $G_\tau(W_k)$ at each local iteration τ can be derived with the partial derivatives of $f(W_k; \mathcal{G}_k)$. Then, the local model parameter of the i-th client would be updated as follows:

$$[W_k]_{\tau+1} \leftarrow [W_k]_\tau - \eta G_\tau(W_k), \tag{13}$$

where η represents the learning rate. All the local participants will upload their updated local models to the cloud server for parameter aggregation:

$$W_k^{t+1} = \frac{1}{N}\sum_{k=1}^{N} W_k^t. \tag{14}$$

where W_k^{t+1} is the globally learned model, which will be distributed to clients as the initial model in round t+1.

While FedAvg provides a vanilla formulation of federated learning, it may significantly suffer from client drift caused by non-IID data distribution. Since network flow statistics of various clients follow non-IID distribution, FedAvg aggregation

could degrade global-level and client-level performance due to the issue of client drift. To address this limitation, we introduce Ditto, a personalized FL scheme that seeks to simultaneously optimize a local model and a personalized model via bi-level problem optimization. The local model is updated to the cloud server for global model aggregation, while the personalized model is kept private to fit non-IID flow data distribution. The mini-batch optimization loss function of the i-th client can be modified into the following form:

$$\min_{\theta_k} h(\theta_k, W^*; \mathcal{G}_k) = f(\theta_k; \mathcal{G}_k) + \frac{\mu}{2}\|\theta_k - W^*\|^2, \tag{15}$$

$$s.t. \quad W^* \in \arg\min \frac{1}{N}\sum_{k=1}^{N} F(W_k; \mathcal{G}_k). \tag{16}$$

Ditto formulates a federated multi-task learning framework to better handle flow data non-IIDness, including the global task $F(W_k; \mathcal{G}_k)$ and the local task $f(\theta_k; \mathcal{G}_k)$, which trains a personalized model using only local data. A regularization term $\frac{\mu}{2}\|\theta_k - W^*\|^2$ is introduced to encourage the personalized model as close to the aggregated model as possible. λ controls the trade-off between global generalization and local personalization. When we set λ to 0, it is reduced to local training. When λ increases, it is focused more on global model generalization.

Algorithm 1 The Workflow of FeCoGraph

Input: client set $\mathcal{I}$, line graph $\mathcal{G}_k$ for each client $\mathcal{C}_k$, communication rounds T, local epochs r, personalized epochs s, local learning rate η_l, personalized learning rate η_p, interpolation factor μ, initialized global model w^0, initialized personalized model $\{v_k^0\}_{k\in[K]}$

for $t = 0, \ldots, T-1$ **do**
  Server randomly selects a subset of clients S_t from $\mathbb{I}$, and sends w_t to them
  for *client* $\mathcal{C}_k \in S_t$ *in parallel* **do**
    Perform attribute-level and topology-level adaptive graph augmentation on $\mathcal{G}_k$ to get $\mathcal{G}_k^1$ and $\mathcal{G}_k^2$.
    Calculate supervised contrastive loss F_k with $\mathcal{G}_k^1$ and $\mathcal{G}_k^2$ according to Eq (17)
    Set w_k^t to w_t and update w_k^t for r local iterations on F_k:
    $w_k^t \leftarrow w_k^t - \eta_l \nabla F_k(w_k^t)$
    Calculate cross-entropy loss f_k with $\mathcal{G}_k$ according to Eq (18)
    Update θ_k for s personalized iterations on f_k:
    $\theta_k \leftarrow \theta_k - \eta_p(\nabla f_k(\theta_k) + \mu(\theta_k - w^t))$
    Send $\Delta_k^t := w_k^t - w^t$ back
  end for
  Server updates w^{t+1} as
  $w^{t+1} \leftarrow w^t + \frac{1}{|S_t|}\sum_{k\in S_t} \Delta_k^t$
end for
return $\{v_k\}_{k\in[K]}(personalized), w^T(global)$

The detailed workflow of FeCoGraph in the personalized FL setting is shown in Algorithm 1. According to the above equations, each client contributes to the shared global model while training a personalized model locally. To accomplish this, we employ the supervised contrastive loss function as the global objective to encourage sharing more fundamental network behavior patterns among clients. Simultaneously, we leverage a supervised cross-entropy loss to capture specific knowledge of the local data distribution. The two objectives are formulated as follows respectively:

$$F(W_k; \mathcal{G}_k) = \lambda_{ce} \cdot \mathcal{L}_{ce}(W_k; \mathcal{G}_k) + (1 - \lambda_{ce}) \cdot \mathcal{L}_{supcon}(W_k; \mathcal{G}_k), \tag{17}$$

$$f(\theta_k; \mathcal{G}_k) = \mathcal{L}_{ce}(\theta_k; \mathcal{G}_k). \tag{18}$$

V. Experiments

In this section, we conduct extensive experiments to evaluate the proposed network intrusion detection methods on real-world NIDS datasets. First, we present the necessary information on the experimental setup. Secondly, we report the experiment results attached with performance analysis.

A. Experimental Setup

1) Datasets: We adopt three widely used benchmark datasets to evaluate our method, including NF-BoT-IoT-v2, NF-ToN-IoT-v2, and NF-CSE-CIC-IDS2018-v2 [51]. All of these datasets are netflow datasets. Each data sample is a network traffic flow representing a series of packets between two edge devices. There are total 43 features that show information on general flow statistics and some specific protocols. All flow-based features are extracted from packet headers rather than payload information. The detailed descriptions of each dataset are as follows. The v2 version is an extended version of the corresponding v1 dataset, which contains more samples and statistical features. The detailed statistics of three datasets are shown in Table II

- **NF-BoT-IoT-v2** is a netflow-based IoT dataset from a network environment in 2018. It contains 37,763,497 flows out of which 37,628,460(99.64%) are malicious and 135,037(0.36%) are benign.
- **NF-ToN-IoT-v2** is a netflow-based IoT dataset realized in 2019. It contains 16,940,496 data flows out of which 10,841,027(63.99%) are malicious and 6,099,469(36.01%) are benign.
- **NF-CSE-CIC-IDS2018-v2** is a dataset collected in 2018. It contains 18,893,708 network flows, out of which 2,258,141(11.95%) are malicious and 16,635,567(88.05%) are benign.

2) Data Preparation: We perform four pre-processing steps before converting network flows into network graphs. For each flow data, we merge the IP address and port number into one attribute by concatenating two attributes. Since Certain types of attacks are associated with specific port numbers, only IP addresses are not sufficient to distinguish them. The merged attribute are expected to more precisely capture the behavioral patterns associated with different attacks. Secondly, we fill the missing and infinite values with zero. Thirdly, we perform target encoding to convert categorical features into numerical values. The labels indicating benign/malicious and attack type are also transformed into numerical values. Finally, a standard scaler is adopted to obtain the normalized features.

TABLE II
ATTACK CATEGORY STATISTICS OF THREE NIDS DATASETS

Dataset	Class Distribution (%)										Total Size
NF-BoT-IoT-v2	Benign	DDoS	DoS	Reconnaissance	Theft						37,763,497
	0.36	48.54	44.15	6.94	0.0064						
NF-ToN-IoT-v2	Benign	Backdoor	DDoS	DoS	Injection	MITM	Password	Ransomware	Scanning	XSS	16,940,496
	36.01	0.099	11.96	4.21	4.04	0.046	6.81	0.020	22.32	14.49	
NF-CSE-CIC-IDS2018-v2	Benign	BruteForce	Bot	DoS	DDoS	Infiltration	Web Attack				18,893,708
	88.05	0.64	0.76	2.56	7.36	0.62	0.019				

3) Baselines: To provide a comprehensive evaluation, We compare our methods with two types of baselines, including machine learning methods and GNN methods.

- The machine learning baseline algorithms include AdaBoost, k Nearest Neighbors (KNN), Decision Tree, and tree-based ensemble methods (XGBoost, Random Forest, Extra Trees). We directly feed network flow-based features into the machine learning classifier.
- We primarily compare our methods with E-graphSAGE [7] to showcase the improvement brought by contrastive learning under few-shot scenarios. It is a supervised approach that adjusts the message propagation mechanism by aggregating edge features instead of node features.
- Anomal-E [11] is included to demonstrate the effect of label information in contrastive learning. It is built upon an E-graphSAGE encoder, incorporates a modified DGI (Deep Graph Infomax) objective, and integrates four traditional anomaly detection algorithms.
- we additionally compare our methods with E-ResGAT [8]. It incorporates residual learning into the original GAT architecture to improve the performance of minority classes and stabilize model training.

4) Evaluation Metrics: Four metrics are leveraged to evaluate the performance of FeCoGraph, including accuracy, precision, recall, and f1-score. These metrics have been extensively used in numerous previous works. These metrics are calculated with the number of true positive (*TP*), false positive (*FP*), true negative (*TN*) and false negative (*FN*), which can be formulated as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}, \tag{19}$$

the accuracy is calculated as the ratio of correctly predicted samples to all samples in the dataset;

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{20}$$

Precision is defined as the ratio of all correctly predicted positive samples to all samples predicted as positive.;

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{21}$$

Recall is defined as the ratio of all correctly predicted positive samples to all actual positive samples in the dataset;

$$\text{F1} - \text{Score} = \frac{2 \times Precision \times Recall}{Precision + Recall}, \tag{22}$$

The F1-score is defined as the harmonic mean of precision and recall. It achieves a balance between two metrics in imbalanced datasets. Note that we use a macro version of the above metrics, which are calculated and equally averaged for each class. Since class distributions are generally imbalanced in network intrusion detection, macro values are appropriate for treating each class equally.

5) Experimental Settings: The proposed algorithm is implemented with PyTorch, DGL and PyTorch Geometric. The backbone encoder is composed of a two-layer graph convolutional network (GCN). The input feature dimension is 39, which corresponds to the number of network flow features. The number of hidden units in the two layers is set to 64 and 32 respectively. The dimension of projector hidden layers in tuned from the parameter set $\{64, 128, 256\}$, in which 256 is selected as the optimal value based on validation performance. Datasets were uniformly downsampled by label proportion due to the heavy computation burden of using full dataset, with a downsampling ratio of 2%. Since the volume of network flows is relatively large, we calculate supervised contrastive loss in batches to avoid the issue of CUDA out of memory. All experiments were performed on Ubuntu 22.04 OS equipped with 2 NVIDIA GeForce RTX 3090 GPUs. Relevant software libraries include Python 3.8.8, PyTorch 1.13.0, DGL 1.2, and PyTorch Geometric 2.3.0, etc.

For machine learning approaches, we directly utilize algorithms to fit training samples and provide prediction results for test samples. For GNN-based approaches, batch gradient descent is leveraged with an Adam optimizer. The models are uniformly trained in 2000 epochs with a learning rate of 0.001. Following the setting in E-graphSAGE with additional simulation for few-shot NIDS, we split the overall dataset into train and test graphs. 30% of the data is used for training, and the remaining 70% for test. Note that the anomaly detector of Anomal-E is finetuned in an unsupervised manner, it is unfair to compare it with our method. In line with TS-IDS, we exclusively utilize the E-graphSAGE encoder from Anomal-E to produce edge embeddings. These embeddings are then utilized to fine-tune an XGBoost classifier.

For FL experiments, it is a common practice in FL experiments to split a centralized dataset into multiple partitions and distribute them across different clients. The whole network traffic graph is partitioned into 10 subgraphs based on the Latent Dirichlet Allocation (LDA) Strategy [52]. Specifically, we partition nodes in each class k into J shards following a symmetric Dirichlet distribution $\mathbf{p}_k \sim Dir_J(\alpha)$, where client

TABLE III

BINARY AND MULTICLASS PERFORMANCE COMPARISON WITH THE STATE-OF-THE-ART ALGORITHMS. (NF-BOT-IOT-V2 IS ABBREVIATED AS BOT, NF-TON-IOT-V2 IS ABBREVIATED AS TON, NF-CSE-CIC-IDS2018-V2 IS ABBREVIATED AS IDS2018)

Dataset	Binary Classification					Multiclass Classification				
	Method	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Method**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
BoT	KNN	99.64	49.82	50.00	49.91	KNN	46.26	19.97	19.85	19.11
	AdaBoost	99.88	95.95	86.30	90.56	AdaBoost	97.87	60.37	59.31	59.64
	Decision Tree	99.82	95.53	77.36	84.17	Decision Tree	97.84	74.18	73.14	73.65
	XGBoost	99.89	97.84	85.78	90.94	XGBoost	98.46	76.88	75.23	76.01
	E-graphSAGE [7]	99.65	**99.82**	51.05	51.97	E-graphSAGE [7]	96.81	71.28	72.37	71.81
	Anomal-E [11]	99.87	97.37	83.94	89.54	Anomal-E [11]	96.79	76.07	71.95	73.74
	E-ResGAT [8]	99.74	66.58	40.12	44.57	E-ResGAT [8]	97.95	77.70	64.63	68.26
	Ours	**99.89**	96.92	**86.61**	**91.12**	**Ours**	**98.48**	**98.03**	**79.92**	**86.86**
ToN	KNN	93.08	93.14	91.76	92.38	KNN	85.35	55.14	51.73	52.7
	AdaBoost	86.64	85.93	88.85	86.24	AdaBoost	44.84	14.23	23.57	15.3
	Decision Tree	96.63	96.56	96.10	96.32	Decision Tree	89.92	70.09	63.66	63.79
	XGBoost	95.60	96.26	94.25	95.12	XGBoost	90.77	73.86	64.52	65.94
	E-graphSAGE [7]	78.47	79.8	82.15	78.25	E-graphSAGE [7]	83.05	69.74	**76.88**	71.39
	Anomal-E [11]	95.77	95.88	94.89	95.36	Anomal-E [11]	87.26	68.06	61.90	62.12
	E-ResGAT [8]	93.84	94.23	92.46	93.23	E-ResGAT [8]	84.40	63.17	56.68	58.36
	Ours	**96.90**	**96.81**	**96.44**	**96.62**	**Ours**	**94.32**	**74.28**	72.46	**73.31**
IDS2018	KNN	87.15	50.36	50.04	47.71	KNN	87.83	13.09	14.26	13.39
	AdaBoost	99.26	98.9	97.56	98.22	AdaBoost	95.3	38.17	41.88	39.71
	Decision Tree	99.07	97.85	97.72	97.79	Decision Tree	98.87	73.65	73.36	72.96
	XGBoost	99.24	98.9	97.74	98.31	XGBoost	99.3	83.83	73.41	74.75
	E-graphSAGE [7]	93.20	92.49	73.40	79.36	E-graphSAGE [7]	92.16	67.44	62.72	61.26
	Anomal-E [11]	91.57	83.03	72.22	76.17	Anomal-E [11]	96.79	76.07	71.95	73.74
	E-ResGAT [8]	98.45	97.34	95.30	96.29	E-ResGAT [8]	98.36	80.62	67.80	68.38
	Ours	**99.63**	**99.38**	**98.85**	**99.11**	**Ours**	**99.52**	**84.41**	**79.07**	**81.38**

j will be assigned with a shard of $\mathbf{p}_{k,j}$ proportion. The total nodes of client j are obtained by gathering each corresponding $\mathbf{p}_{k,j}$ partition of class k. Then subgraphs are generated by recovering the original links between some of nodes. For the consideration of few-shot detection, we subsequently keep 30% of the entire nodes labeled. We uniformly set the number of communication rounds and local epochs as 100 and 5, respectively. Evaluation metrics of each individual client are averaged in FL scenarios. We use *best mean testing accuracy* (BMTA) [53] as the primary evaluation metric.

B. Result Analysis

In this part, we first provide binary classification results to validate the ability of FeCoGraph to differentiate benign flows from malicious flows. Secondly, the multiclass classification results are presented to demonstrate FeCoGraph's capability to identify different attack types. We also investigate the impact of label proportion, supervised contrastive loss, and key hyperparameters. Finally, we illustrate the convergence performance of FeCoGraph under the non-IID data scenario.

1) Binary Classification Performance: To evaluate the ability of our method to discriminate malicious flows from benign flows, we conduct the binary classification experiment under the 30% proportion of labeled samples. The results compared with other representative GNN-based methods are reported in Table III, including accuracy, precision, recall and F1-score. As shown in the table, our solution performs well on three datasets. On the NF-CSE-CIC-IDS2018-v2 dataset, our method achieves the best F1 score and accuracy. On NF-ToN-IoT-v2 dataset, our method obtains an accuracy of 96.9% and an F1-score of 96.62%. On NF-BoT-IoT dataset, our method obtains an accuracy of 99.89% and 91.12% of F1-score, achieving the pinnacle of performance on these two datasets. Furthermore, our method exhibits commendable results when compared with graph-based methods, which demonstrate the potential utility of supervised contrastive loss to learn compact intra-class representations. Finally, tree-based ensemble learning methods obtain competitive results on all datasets, which indicates that ensemble learning has led to significant performance improvement than individual classifiers alone.

2) Multiclass Classification Performance:

a) Comparison with State-of-the-Art Algorithms: To evaluate the ability to accurately discriminate various attack categories, the multiclass classification experiment is conducted with 30% proportions of labeled samples, the same setting as in binary classification. On NF-CSE-CIC-IDS-2018v2 dataset, our method achieves the best accuracy of 99.52% and the best F1-score of 81.38%. Moreover, the F1-score achieved by our methods is substantially ahead of other methods, surpassing XGBoost and obtaining second-best results by 6.61%. Results on both datasets suggest our method has advanced capabilities to detect genuine threats while minimizing the risk of false alarms. Similarly, On NF-BoT-IoT-v2 dataset, our method outperforms other methods across four metrics. On NF-ToN-IoT-v2 dataset, our method achieves the best detection performance, with an accuracy of 94.32% and a F1-score of 73.31%.

An interesting phenomenon is that tree-based ensemble learning methods have showcased a universally high performance on NF-ToN-IoT-v2 dataset for both binary and multiclass classification, in contrast to the performance on NF-CSE-CIC-IDS2018-v2 dataset. It can be inferred that these methods are sufficient to capture complex patterns and relationships within the data. Besides, the gap between the performance of our method and the best performance achieved by the Random Forest algorithm is smaller than that in

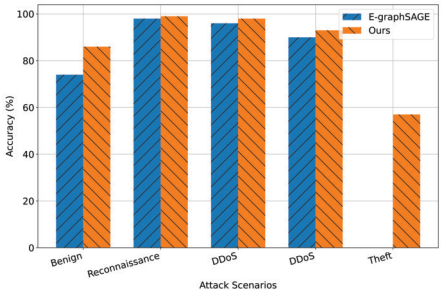


Fig. 3. F1-score for each attack scenario on NF-BoT-IoT-v2 dataset.

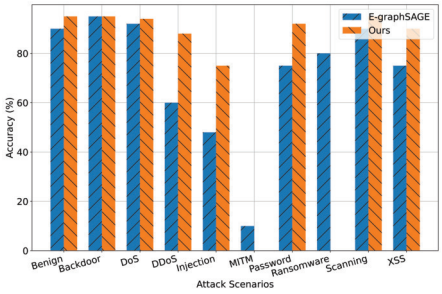


Fig. 4. F1-score for each attack scenario on NF-ToN-IoT-v2 dataset.

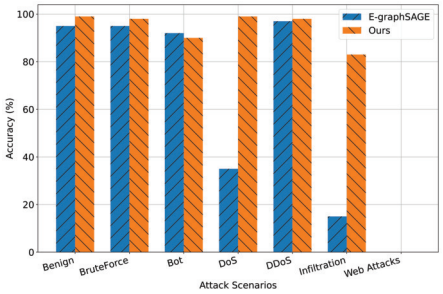


Fig. 5. F1-score for each attack scenario on NF-CSE-CIC-IDS2018-v2 dataset.

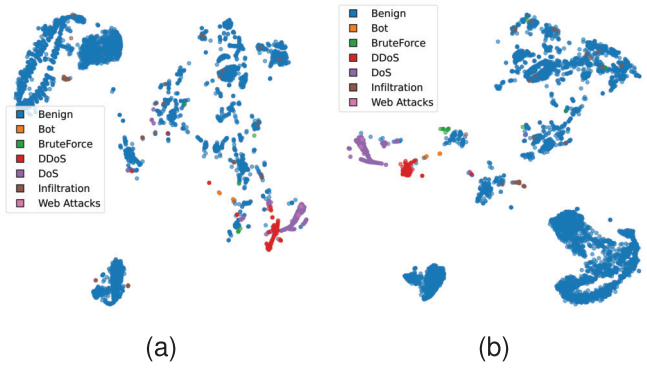


Fig. 6. Visualization of reduction (a) raw edge features from NF-CSE-CIC2018-v2 test set. (b) edge embeddings generated by FeCoGraph encoder.

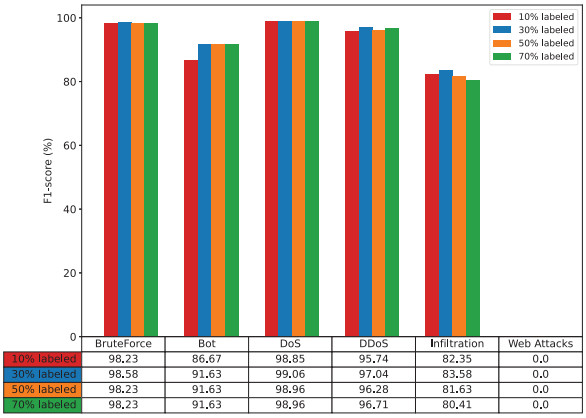
a binary classification task, showing the superiority of our methods in distinguishing different network flows via line graph construction.

b) Detection Performance on Attack Categories: With the presence of numerous emerging and complicated cyber attacks, it's critical to differentiate between malicious flows and benign flows and further distinguish different types of abnormal behaviors. Moreover, different types of malicious flows exhibit a skewed distribution, where certain types of attacks occur more frequently than others. We compare our method with E-graphSAGE, one of the state-of-the-art GNN-based approaches, in terms of F1-score values for each category on NF-BoT-IoT-V2, NF-ToN-IoT-v2 and NF-CSE-CIC-IDS2018-v2. The proportion of labeled samples is set to 30% on both datasets.

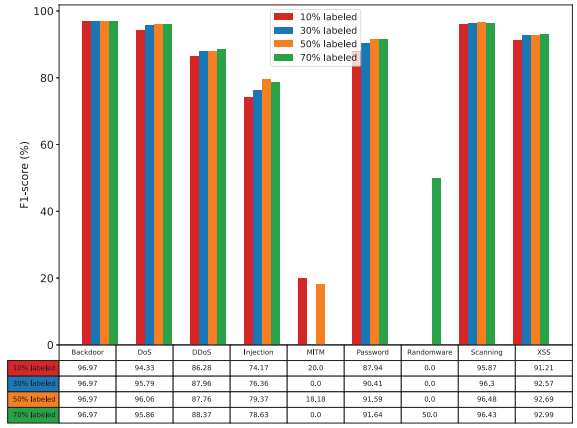
Figure 3 provides the results on NF-BoT-IoT-v2. It includes 4 attack scenarios: Reconnaissance, DDoS, DoS, and Theft. As shown in the Table, our approach consistently outperforms E-graphSAGE for all attack scenarios. Especially, E-graphSAGE cannot effectively identify Theft attacks due to scarce samples, while our approach achieves an F1-score of 57.14%. Theft aims to obtain sensitive data via some disguised ways, such as social engineering. The results demonstrate the ability of our approach to detect rare and covert attacks

Figure 4 provides the results on NF-ToN-IoT-v2. It includes 9 attack scenarios: backdoor, DoS, DDoS, injection, MITM, password, ransomware, scanning and XSS. As seen in the table, the F1-score of E-graphsage for injection, DDoS and XSS is 48.36%, 59.37% and 73.56%, respectively. On the one hand, the number of samples for these three scenarios is relatively small, so it's difficult for the model to capture the behavior patterns of these categories of attacks. On the other hand, these attacks are more easily misclassified as normal network flows. Injection involves manipulating or injecting malicious code into data streams to alter the execution path, Cross-site Scripting (XSS) is also a type of injection operated in web applications. DDoS is a distributed DoS attack launched by many hosts, attackers can lower the attack rate of DoS to evade detection. Compared with E-graphSAGE, our method introduces a supervised contrastive objective along with a refined graph convolution process via line graph construction. After these operations, the F1-score against DDoS attacks has improved from 59.37% to 87.96%. The F1-score against Injection attack has improved from 48.36% to 76.36%. The F1 score against XSS attack has increased from 73.56% to 92.57%. Note that E-graphSAGE performs better than our methods against Man-In-The-Middle (MITM) and ransomware attacks, which are two extremely rare attack scenarios.

The NF-CSE-CIC-IDS2018-v2 dataset covers 6 attack scenarios, including BruteForce, bot, DoS, DDoS, infiltration, and Web Attacks. As seen in Figure 5, the F1-score of E-graphSAGE against DoS and infiltration attack is only 34.04% and 14.59%, respectively. In addition to the difficulty of detecting DoS mentioned above, infiltration is another type of intricate attack. Infiltration may involve insiders with legitimate access to the system and employ stealth techniques to avoid detection. After employing a supervised contrastive objective, the F1-score against DoS attack has improved from 34.04% to 99.14%. Our method also brings a huge increase to the F1-score against infiltration attack, from 14.59% to 82.76%. The considerable performance improvement on two datasets has demonstrated the effect of supervised contrastive learning on classifying different types of malicious flows.



(a)



(b)

Fig. 7. Performance of FeCoGraph on specific categories with different label proportions. (a) NF-CSE-CIC-IDS2018-v2 dataset; (b) NF-ToN-IoT-v2 dataset.

To intuitively demonstrate the discrimination ability for different attacks, we visualize the raw edge features and edge embeddings generated by the GCN encoder of FeCoGraph. The graph data are sampled from NF-CSE-CIC-IDS2018-v2 dataset. We apply the t-distributed Stochastic Neighbor Embedding (t-SNE) [54] algorithm to map the high-level representation into two dimensions. The result is shown in Fig. 6. In Figure 6a, a large number of malicious flows are mixed with dominant normal flows. Samples of DDoS and DoS attacks cannot be easily differentiated. On the contrary, We can observe clear boundaries between different attacks(*e.g.*, Dos, DDoS) and more compact distribution(*e.g.*, Benign) in Figure 6b. This illustrates the effect of the supervised contrastive objective of separating different traffic flows in the embedding space, which in turn promotes the detection performance.

c) The Impact of Different Label Proportions: To further investigate the performance of our method for varied amounts of training data, we conduct a series of experiments with 10%, 30%, 50% and 70% labeled samples on NF-ToN-IoT-v2 and NF-CSE-CIC-IDS2018-v2 datasets. As seen in Figure 7a, the performances achieved when label proportion is 10% and 30% on NF-CSE-CIC-IDS2018-v2 dataset don't exhibit a large gap compared to those achieved when utilizing 50% and 70% labeled samples. It can be inferred that our method is capable of learning discriminate flow representations to distinguish different attack categories with critically limited training data.

As shown in Figure 7b, There is a little different tendency in detection results on NF-ToN-IoT-v2 dataset, where the f1-score against DDoS, injection, password and XSS attacks increases with the larger proportion of labeled data. It can be observed that these types of attacks are harder to detect, therefore more labeled data is required for our model to enhance the expressive ability of network flow representations for different attack categories. Note that our method always almost fails to detect some particular attack categories, such as web attacks, MITM attack and ransomware attack since there are few samples.

TABLE IV
MULTICLASS CLASSIFICATION PERFORMANCE WITH λ

Dataset	λ	Accuracy	Precision	Recall	F1-Score
NF-BoT-IoT-v2	0.03	**98.48**	84.01	**95.03**	86.94
	0.05	98.46	83.95	94.71	86.73
	0.07	98.47	82.53	94.67	85.25
	0.3	98.43	**85.81**	94.22	**88.31**
	0.5	98.46	82.60	94.71	85.30
	0.7	98.47	82.21	94.73	85.13
NF-ToN-IoT-v2	0.03	92.10	72.22	67.67	69.25
	0.05	93.34	73.06	71.07	71.99
	0.07	94.07	74.12	72.23	73.10
	0.3	94.24	74.19	72.40	73.22
	0.5	**94.32**	**74.28**	**72.46**	**73.31**
	0.7	94.21	74.13	72.35	73.18
NF-CSE-CIC-IDS2018-v2	0.03	98.86	70.14	68.26	69.13
	0.05	99.38	**84.43**	76.42	79.40
	0.07	99.50	84.39	78.95	**81.29**
	0.3	**99.52**	84.03	**79.13**	81.27
	0.5	99.50	84.15	78.83	81.13
	0.7	99.37	84.41	76.23	79.28

TABLE V
MULTICLASS CLASSIFICATION PERFORMANCE WITH τ

Dataset	τ	Accuracy	Precision	Recall	F1-Score
NF-BoT-IoT-v2	0.1	**98.48**	84.01	95.03	86.94
	0.3	98.46	83.97	94.87	86.83
	0.5	98.46	83.97	94.87	86.83
	0.7	98.47	**89.17**	**95.06**	**91.44**
NF-ToN-IoT-v2	0.1	94.17	73.94	72.33	73.07
	0.3	**94.32**	**74.28**	**72.46**	**73.31**
	0.5	94.25	74.11	72.41	73.20
	0.7	92.78	73.74	70.12	71.52
NF-CSE-CIC-IDS2018-v2	0.1	**99.52**	84.03	**79.13**	**81.27**
	0.3	99.51	84.30	78.95	81.25
	0.5	99.38	84.44	76.35	79.37
	0.7	99.39	**84.45**	76.36	79.37

d) Detection Performance with Different λ and τ: λ and τ are two important factors in constructing a supervised contrastive loss function. Consequently, we perform a sensitivity analysis to examine how different values affect detection performance. The values of λ include 0.03, 0.05, 0.07, 0.3, 0.5 and 0.7. The values of τ include 0.1, 0.3, 0.5, 0.7. λ weighs the significance of label-aware contrastive loss over supervised learning loss. The smaller λ is, the more weight is given to minimize the distances between samples belonging to the same attack category. As shown in Table IV, the four metrics on NF-ToN-IoT-v2 consistently continue to increase and achieve the best results when λ is 0.5. The best F1-score is achieved when λ is 0.03 on NF-BoT-IoT-v2 and 0.07 on

TABLE VI

MULTICLASS CLASSIFICATION RESULTS OF ABLATION STUDY ON CONTRASTIVE LOSS. "SUPCON" DENOTES LABEL-AWARE GRAPH CONTRASTIVE LEARNING MODULE, AND "SSLCON" DENOTES SELF-SUPERVISED GRAPH CONTRASTIVE LEARNING MODULE

Dataset	Task	Setting	Accuracy	Precision	Recall	F1-Score
NF-ToN-IoT-v2	Binary	SupCon	**96.90**	**96.81**	**96.44**	**96.62**
		SSLcon	92.02	91.01	92.03	91.46
	Multiclass	SupCon	**94.32**	**74.28**	**72.46**	**73.31**
		SSLCon	91.59	72.41	66.89	68.78
NF-CSE-CIC-IDS2018-v2	Binary	SupCon	**99.63**	**99.38**	**98.85**	**99.11**
		SSLCon	95.26	87.12	92.34	89.48
	Multiclass	SupCon	**99.52**	**84.41**	**79.07**	**81.38**
		SSLCon	93.69	23.84	25.43	24.60

NF-CSE-CIC-IDS2018-v2, implying that learning class-aware compact representation is more encouraged.

τ is associated with the temperature parameter, which affects the sharpness of the probability distribution used in the contrastive loss function. A higher τ (e.g., larger temperature) results in a more uniform distribution over similarities, potentially improving generalization. On the other hand, a lower τ (e.g., smaller temperature) sharpens the distribution, focusing on maximizing the similarity with positive instances. As shown in Table V, the best results are obtained when τ is 0.3 on NF-ToN-IoT-v2 and 0.1 on NF-CSE-CIC-IDS2018-v2, showing that supervised contrastive loss tends to impose more penalties on hard negative examples. On the contrary, the best results are obtained when τ is 0.7 on NF-BoT-IoT-v2, indicating the tendency to encourage global uniformity of flow representation.

e) Ablation Study: Compared with self-supervised contrastive learning that seeks to reduce the distance of multi-view embeddings, label-aware contrastive learning leverages label information to learn compact intra-class representation, we conduct an ablation study to compare the impact of these two objectives in both binary and multiclass classification scenarios. As shown in Table VI, the label-aware contrastive loss function consistently outperforms the counterpart that incorporates self-supervised contrastive learning. On the one hand, the F1-score improves by 5.16% and 4.53% for binary and multiclass classification on NF-ToN-IoT-v2 dataset. On the other hand, the F1-score improves by 9.63% and 56.78% for binary and multiclass classification on NF-CSE-CIC-IDS2018-v2 dataset. The improvements indicate that supervised contrastive learning enables more effective use of unlabeled data, thereby more accurately distinguishing different attacks.

3) The Convergence Performance of FeCoGraph: As discussed before, federated learning is an effective strategy for building distributed intrusion detection systems. To this end, experiments are designed to evaluate the performance of our intrusion detection model in the federated learning scenario. The NetFlow data owned by each client follows non-IID distribution for different attack categories. The traffic data are then converted into a line graph following the paradigm in Section IV-B1. During training, each client trains a local GNN model with a supervised contrastive learning objective. The local parameters are subsequently uploaded into a central server for weight aggregation (e.g., weight-averaging rule).

To ensure a fair evaluation, We compare the performance of three federated learning algorithms. 1) we adopt a vanilla GCN-based FL method to detect malicious flows. 2) we utilize the proposed FeCoGraph intrusion detection model with a supervised contrastive objective in FedAvg. 3) We incorporate the FeCoGraph model with supervised contrastive objective into a personalized FL framework. As shown in Figure, our model with supervised contrastive loss outperforms the counterpart without supervised contrastive loss in terms of detection accuracy, which exhibits similar trends as non-federated experiments. The green curve shows Personalized FL converges faster and achieves better convergence performance compared with other FL settings, demonstrating the capability of supervised contrasting along with personalization strategy in non-IID federated learning scenarios.

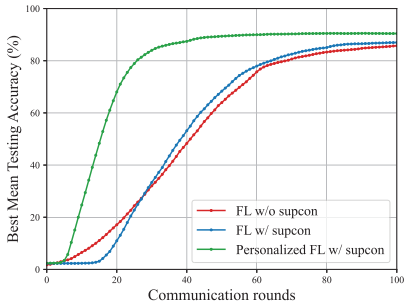


Fig. 8. The performance comparison of three FL algorithms.

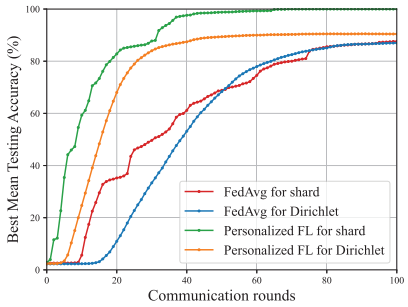


Fig. 9. The performance comparison between FedAvg and personalized FL under shard and dirichlet distribution.

We further investigate the impact of non-IID factors, including subgraph partition strategy and imbalance level of the Dirichlet distribution. As shown in Figure 9, the personalized FL method (abbreviated as perFL) exhibits a marginal improvement compared with FedAvg. It implies a bi-level optimization strategy can effectively avoid model degradation caused by the issue of client drift. Moreover, the increase in accuracy is more pronounced with the shard partition strategy (12.41% vs 3.52%). Since perFL aims to maintain local data distribution, it benefits more when training and test data share a consistent class label distribution. Moreover, Fig 10

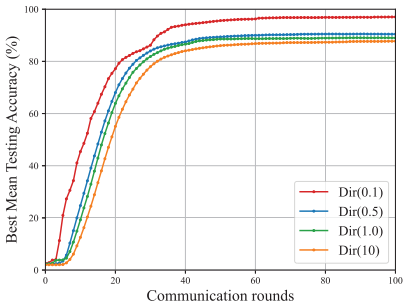


Fig. 10. The performance comparison under different α of the dirichlet distribution.

demonstrates a decreased tendency of model performance when the Dirichlet distribution tends to be more balanced, where the best result is achieved under the most imbalanced setting. It demonstrates that personalized FL is more compatible with federated NIDS on heterogeneous networks.

VI. Conclusion

In this paper, we propose FeCoGraph, a label-aware federated graph contrastive learning framework for few-shot intrusion detection. FeCoGraph constructs a label-aware graph contrastive module, promoting the discriminative and expressive capabilities of network flow representations. Furthermore, We incorporate graph contrastive learning module into a personalized FL algorithm to support distributed IDS in edge IoT. Experiment Results on three public datasets show the superiority of FeCoGraph with an average accuracy of 98.27% on binary classification and 96.92% on multiclass classification. In particular, our methods can effectively identify rare and stealthy types of attacks, including DDoS, injection, and infiltration attacks.

References

[1] V. Kumar and O. P. Sangwan, "Signature based intrusion detection system using SNORT," *Int. J. Comput. Appl. Inf. Technol.*, vol. 1, no. 3, pp. 35–41, Nov. 2012.

[2] M. Eskandari, Z. H. Janjua, M. Vecchio, and F. Antonelli, "Passban IDS: An intelligent anomaly-based intrusion detection system for IoT edge devices," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6882–6897, Aug. 2020.

[3] Y. Qing, X. Liu, and Y. Du, "Mitigating data imbalance to improve the generalizability in IoT DDoS detection tasks," *J. Supercomput.*, vol. 80, no. 7, pp. 9935–9960, May 2024.

[4] X. Ma et al., "A comprehensive survey on graph anomaly detection with deep learning," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 12, pp. 12012–12038, Dec. 2023.

[5] J. Zhou, Z. Xu, A. M. Rush, and M. Yu, "Automating botnet detection with graph neural networks," 2020, *arXiv:2003.06344*.

[6] W. W. Lo, G. Kulatilleke, M. Sarhan, S. Layeghy, and M. Portmann, "XG-BoT: An explainable deep graph neural network for botnet detection and forensics," *Internet Things*, vol. 22, Jul. 2023, Art. no. 100747.

[7] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, "E-GraphSAGE: A graph neural network based intrusion detection system for IoT," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, Budapest, Hungary, Apr. 2022, pp. 1–9.

[8] L. Chang and P. Branco, "Graph-based solutions with residuals for intrusion detection: The modified E-GraphSAGE and E-ResGAT algorithms," 2021, *arXiv:2111.13597*.

[9] X. Deng, J. Zhu, X. Pei, L. Zhang, Z. Ling, and K. Xue, "Flow topology-based graph convolutional network for intrusion detection in label-limited IoT networks," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 1, pp. 684–696, Mar. 2023.

[10] G. Duan, H. Lv, H. Wang, and G. Feng, "Application of a dynamic line graph neural network for intrusion detection with semisupervised learning," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 699–714, 2023.

[11] E. Caville, W. W. Lo, S. Layeghy, and M. Portmann, "Anomal-E: A self-supervised network intrusion detection system based on graph neural networks," *Knowl.-Based Syst.*, vol. 258, Dec. 2022, Art. no. 110030.

[12] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," 2018, *arXiv:1809.10341*.

[13] H. Nguyen and R. Kashef, "TS-IDS: Traffic-aware self-supervised learning for IoT network intrusion detection," *Knowl.-Based Syst.*, vol. 279, Nov. 2023, Art. no. 110966.

[14] Q. Sun et al., "SUGAR: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism," in *Proc. Web Conf.*, Apr. 2021, pp. 2081–2091.

[15] J. Cui et al., "Collaborative intrusion detection system for SDVN: A fairness federated deep learning approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 9, pp. 2512–2528, Sep. 2023.

[16] W. Sheng and J. Zhigang, "IDS classification algorithm based on fuzzy SVM model," *Appl. Res. Comput.*, vol. 37, no. 2, pp. 187–190, 2018.

[17] J. Gu and S. Lu, "An effective intrusion detection approach using SVM with naïve Bayes feature embedding," *Comput. & Secur.*, vol. 103, Apr. 2021, Art. no. 102158.

[18] B. Mahbooba, M. Timilsina, R. Sahal, and M. Serrano, "Explainable artificial intelligence (XAI) to enhance trust management in intrusion detection systems using decision tree model," *Complexity*, vol. 2021, pp. 1–11, Jan. 2021.

[19] P. Wu and H. Guo, "LuNet: A deep neural network for network intrusion detection," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2019, pp. 617–624.

[20] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 1222–1228.

[21] F. Jiang et al., "Deep learning based multi-channel intelligent attack detection for data security," *IEEE Trans. Sustain. Comput.*, vol. 5, no. 2, pp. 204–212, Apr. 2020.

[22] M. M. Hassan, A. Gumaei, A. Alsanad, M. Alrubaian, and G. Fortino, "A hybrid deep learning model for efficient intrusion detection in big data environment," *Inf. Sci.*, vol. 513, pp. 386–396, Mar. 2020.

[23] N. Gao, L. Gao, Q. Gao, and H. Wang, "An intrusion detection model based on deep belief networks," in *Proc. 2nd Int. Conf. Adv. Cloud Big Data*, Nov. 2014, pp. 247–252.

[24] A. Z. Alalmaie, P. Nanda, and X. He, "Zero trust-NIDS: Extended multi-view approach for network trace anonymization and auto-encoder CNN for network intrusion detection," in *Proc. IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Dec. 2022, pp. 449–456.

[25] C. Xu, J. Shen, and X. Du, "A method of few-shot network intrusion detection based on meta-learning framework," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3540–3552, 2020.

[26] R. Heartfield, G. Loukas, A. Bezemskij, and E. Panaousis, "Self-configurable cyber-physical intrusion detection for smart homes using reinforcement learning," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1720–1735, 2020.

[27] C. Do Xuan, M. H. Dao, and H. D. Nguyen, "APT attack detection based on flow network analysis techniques using deep learning," *J. Intell. Fuzzy Syst.*, vol. 39, no. 3, pp. 4785–4801, Oct. 2020.

[28] J. Zheng and D. Li, "GCN-TC: Combining trace graph with statistical features for network traffic classification," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[29] X. Hu, W. Gao, G. Cheng, R. Li, Y. Zhou, and H. Wu, "Toward early and accurate network intrusion detection using graph embedding," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 5817–5831, 2023.

[30] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, H. D. III and A. Singh, Eds., Jul. 2020, pp. 1597–1607.

[31] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9729–9738.

[32] R. D. Hjelm et al., "Learning deep representations by mutual information estimation and maximization," 2018, *arXiv:1808.06670*.

[33] P. Khosla et al., "Supervised contrastive learning," in *Proc. NIPS*, 2020, pp. 18661–18673.

[34] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Deep graph contrastive representation learning," 2020, *arXiv:2006.04131*.

[35] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *Proc. Web Conf.*, Apr. 2021, pp. 2069–2080.
[36] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Proc. NIPS*, vol. 33, 2020, pp. 5812–5823.
[37] S. Wan, S. Pan, J. Yang, and C. Gong, "Contrastive and generative graph convolutional networks for graph-based semi-supervised learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 11, 2021, pp. 10049–10057.
[38] S. Akkas and A. Azad, "JGCL: Joint self-supervised and supervised graph contrastive learning," in *Proc. Companion Web Conf.*, Apr. 2022, pp. 1099–1105.
[39] Z. Liu, L. Yang, Z. Fan, H. Peng, and P. S. Yu, "Federated social recommendation with graph neural network," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 4, pp. 1–24, Aug. 2022.
[40] C. Wu, F. Wu, Y. Cao, Y. Huang, and X. Xie, "FedGNN: Federated graph neural network for privacy-preserving recommendation," 2021, *arXiv:2102.04925*.
[41] C. He, E. Ceyani, K. Balasubramanian, M. Annavaram, and S. Avestimehr, "SpreadGNN: Decentralized multi-task federated learning for graph neural networks on molecular data," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, Jun. 2022, pp. 6865–6873.
[42] T. Suzumura et al., "Towards federated graph learning for collaborative financial crimes detection," 2019, *arXiv:1909.12946*.
[43] C. He et al., "FedGraphNN: A federated learning system and benchmark for graph neural networks," 2021, *arXiv:2104.07145*.
[44] B. Wang, A. Li, M. Pang, H. Li, and Y. Chen, "GraphFL: A federated learning framework for semi-supervised node classification on graphs," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2022, pp. 498–507.
[45] Y. Tan, Y. Liu, G. Long, J. Jiang, Q. Lu, and C. Zhang, "Federated learning on non-IID graphs via structural knowledge sharing," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2023, vol. 37, no. 8, pp. 9953–9961.
[46] H. Zhang, K. Zeng, and S. Lin, "Federated graph neural network for fast anomaly detection in controller area networks," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1566–1579, 2023.
[47] S. Krishnan, J. Park, S. Sagar, G. Sherman, B. Campbell, and J. Choi, "Federated graph learning for low probability of detection in wireless ad-hoc networks," in *Proc. IEEE Stat. Signal Process. Workshop (SSP)*, Jul. 2023, pp. 66–70.
[48] Y. Chen, L. Liang, and W. Gao, "FedADSN: Anomaly detection for social networks under decentralized federated learning," in *Proc. Int. Conf. Commun., Comput., Cybersecurity, Informat. (CCCI)*, Oct. 2022, pp. 111–117.
[49] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, vol. 54, A. Singh and J.Zhu., Eds., Fort Lauderdale, FL, USA, 2017, pp. 1273–1282.
[50] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 6357–6368.
[51] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets," *Mobile Netw. Appl.*, vol. 27, no. 1, pp. 357–370, Feb. 2022.
[52] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–23.
[53] Y. Huang et al., "Personalized cross-silo federated learning on non-IID data," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 7865–7873.
[54] L. v. d. Maaten and G. E. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, Jan. 2008.

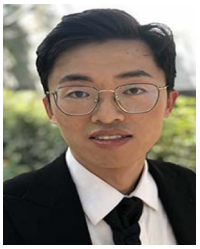
Qinghua Mao (Graduate Student Member, IEEE) received the B.S. degree in information security from the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China, where he is currently pursuing the Ph.D. degree in cyber security. His current research interests include federated learning, graph neural networks, trustworthy machine learning, and cyber security.

Xi Lin (Member, IEEE) received the B.S. degree from the School of Precision Instrument and Optoelectronics Engineering, Tianjin University, Tianjin, China, in 2016, and the Ph.D. degree in cyber security from Shanghai Jiao Tong University, Shanghai, China, in 2021. He is currently an Assistant Professor with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University. His research interests include blockchain, multi-access edge computing, and the Internet of Things.

Wenchao Xu (Senior Member, IEEE) received the B.E. and M.E. degrees from Zhejiang University, Hangzhou, China, in 2008 and 2011, respectively, and the Ph.D. degree from the University of Waterloo, Canada, in 2018. He is currently a Research Assistant Professor with The Hong Kong Polytechnic University. In 2011, he joined Alcatel Lucent Shanghai Bell Company Ltd., where he was a Software Engineer for telecom virtualization. He has been an Assistant Professor with the School of Computing and Information Sciences, Caritas Institute of Higher Education, Hong Kong. His research interests include wireless communication, the Internet of Things, distributed computing, and AI enabled networking.

Yuxin Qi (Student Member, IEEE) received the B.S. degree from the College of Computer Science and Technology, Jilin University, Changchun, China, in 2020. She is currently pursuing the Ph.D. degree with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. Her research interests are focusing on intelligent network and security.

Xiu Su (Member, IEEE) received the B.Sc. and master's degrees from Tianjin University and the Ph.D. degree in computer science from The University of Sydney. Currently, he is a Professor with the Big Data Institute, Central South University. His research interests include pattern recognition and machine learning fundamentals, with a focus on neural architecture search, channel number search, detection, and transformer.



Gaolei Li (Member, IEEE) received the B.S. degree in electronic information engineering from Sichuan University, Chengdu, China, and the Ph.D. degree in cyber security from Shanghai Jiao Tong University, Shanghai, China. From October 2018 to September 2019, he visited the Muroran Institution of Technology, Muroran, Japan, supported by China Scholarship Council Program. He is currently an Associate Professor with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University. His research interests include network security, adversarial machine learning, and privacy computing.

Jianhua Li (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Shanghai Jiao Tong University, Shanghai, China, in 1986, 1991, and 1998, respectively. He is currently a Professor and the Dean of the School of Cyber Security, Shanghai Jiao Tong University. He is also the Director of the National Engineering Laboratory for Information Content Analysis Technology, Engineering Research Center for Network Information Security Management and Service of Chinese Ministry of Education, and Shanghai Key Laboratory of Integrated Administration Technologies for Information Security, China.