

# Applying Self-supervised Learning to Network Intrusion Detection for Network Flows with Graph Neural Network

Renjie Xu<sup>a</sup>, Guangwei Wu<sup>a,\*</sup>, Weiping Wang<sup>b</sup>, Xing Gao<sup>a,b</sup>, An He<sup>a</sup>, Zhengpeng Zhang<sup>a</sup>

<sup>a</sup>*College of Computer and Information Engineering, Central South University of Forestry and Technology, Changsha, 410004, P.R. China*

<sup>b</sup>*School of Computer Science and Engineering, Central South University, Changsha, 410083, P.R. China*

## Abstract

Graph Neural Networks (GNNs) have garnered intensive attention for Network Intrusion Detection System (NIDS) due to their suitability for representing the network traffic flows. However, most present GNN-based methods for NIDS are supervised or semi-supervised. Network flows need to be manually annotated as supervisory labels, a process that is time-consuming or even impossible, making NIDS difficult to adapt to potentially complex attacks, especially in large-scale real-world scenarios. The existing GNN-based self-supervised methods focus on the binary classification of network flow as benign or not, and thus fail to reveal the types of attack in practice. This paper studies the application of GNNs to identify the specific types of network flows in an unsupervised manner. We first design an encoder to obtain graph embedding, that introduces the graph attention mechanism and considers the edge information as the only essential factor. Then, a self-supervised method based on graph contrastive learning is proposed. The method samples center nodes, and for each center node, generates subgraph by it and its direct neighbor nodes, and corresponding contrastive subgraph from the interpolated graph, and finally constructs positive and negative samples from subgraphs. Furthermore, a structured contrastive loss function based on edge features and graph local topology is introduced. To the best of our knowledge, it is the first GNN-based self-supervised method for the multiclass classification of network flows in NIDS. Detailed experiments conducted on four real-world databases (NF-Bot-IoT, NF-Bot-IoT-v2, NF-CSE-CIC-IDS2018, and NF-CSE-CIC-IDS2018-v2) systematically compare our model with the state-of-the-art supervised and self-supervised models, illustrating the considerable potential of our method. Our code is accessible through <https://github.com/renj-xu/NEGSC>.

**Keywords:** Network intrusion detection system, Self-supervised learning, Graph neural network, Network flows, Graph attention mechanism.

## 1. Introduction

With the exponential growth in user hosts and network services, the frequency and the complexity of cyberattacks are increasing [1, 2]. There has been a rise in innovative cyberattacks on vital infrastructure of Internet of Things (IoT) networks, traditional Internet environments and research institutions [3, 4]. Protecting network security has become more challenging than ever. In the real world, networks are generally heterogeneous, where hundreds of millions of devices come from different manufacturers, and security protocols are difficult to align. The emergence of intelligent networking makes networks even more complex. These trends dramatically increase cybersecurity risks in networks, and pose a substantial challenge to cyber-attack detection technology.

In the highly liberal digital space, network devices are vulnerable to various cyberattacks, including data theft, denial of

service (DoS), distributed denial of service (DDoS), reconnaissance attacks and brute force attack, and so on [5]. These attacks threaten the normal operation of the device and the network security. Indeed, the individuals may suffer from property loss and privacy breaches, and companies may be at risk of intellectual property of high economic interest, customer data or disrupt normal operations. Remediation is costly or even irretrievable if the attacks cannot be accurately detected.

Network Intrusion Detection Systems (NIDSs) have proven their reliability in detecting and mitigating cyber-attacks, whose main tasks are to capture malicious network traffic flow and apply the identification outputs to achieve precise and prompt responses. To achieve this task, they require high-quality network flow datasets and data mining to identify potential intrusions. Nowadays, NetFlow is one of the main formats in NIDS for collecting and recording network traffic flows, which provides rich IP flow data and becomes an essential tool for monitoring and recording network flow in network security [6]. The flow data can effectively help us identify and locate anomalous behaviors in the network, and detect and prevent network attacks in a timely manner [7], which is of the importance in network security. As a consequence, NIDSs are regularly deployed at the edge of networks and crucial facilities to monitor and analyse the inbound and outbound network traffic [8].

\*Corresponding author

Email addresses: renjixu@csuft.edu.cn (Renjie Xu),  
guangweiwu@csuft.edu.cn (Guangwei Wu), wpwang@csu.edu.cn  
(Weiping Wang), 20202756@csuft.edu.cn (Xing Gao),  
an\_he@csuft.edu.cn (An He), zhengpeng@csuft.edu.cn (Zhengpeng  
Zhang)

---

NIDSs can be broadly classified into two main types: Signature-based Network Intrusion Detection (SNIDS) and Behavior-based Network Intrusion Detection (BNIDS). SNIDS apply predefined rules or signatures to detect known attack patterns, based on behavioral patterns or specific attack signatures, which trigger system alerts when there are network flows or system activities matching with the rules [9]. SNIDS are unable to flag newly identified and unknown suspicious behaviors [10]. Through monitoring and analyzing diverse behaviors and activities within computer systems and networks, BNIDS detect potential intrusions or threats by identifying newly found or unknown patterns of behavior. Their operations are now more efficient and effective thanks to the development of advanced machine learning (ML) algorithms. However, NIDS continue to encounter numerous obstacles due to the increasing frequency and complexity of attacks.

Extracting features from raw data of network flow samples is vital in the learning process. In traditional ML-based intrusion detection systems, hand-crafted data features are often designed [11]. Such an approach requires suitable functions or statistical metrics to extract the desired information from network flow, placing demands on human experts experience. Dealing with modelling cyber-attacks and building such large datasets can be expensive and time-consuming. Furthermore, these hand-crafted functional representations have relatively low capabilities and are hard to express deep semantic information. The massive increase in connected devices and more complex data interactions impose new demands on automated feature extraction. Deep learning (DL) technology can automatically extract high-level feature representations from raw data of network flows through a well-designed multi-layer neural network, thus eliminating manual feature engineering. DL has made significant progress in NIDS [4]. Through modelling network traffic detection task as a binary versus multiclass classification problem, it enables accurate identification and classification of malicious attacks. Recently, many researchers focus on the DL methods that are capable of dealing with graphical structure data, which is the main form of network traffic flows.

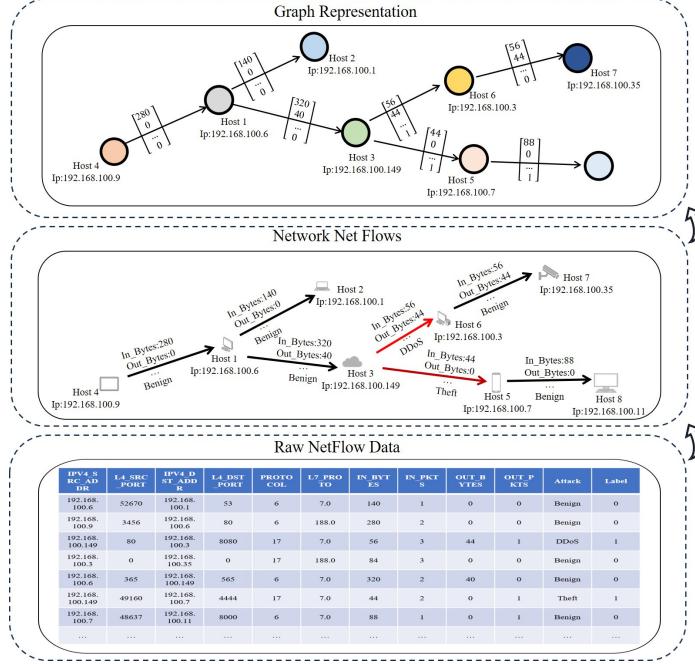
Graphs have a powerful ability to represent non-Euclidean data, and have a well-established theoretical foundation. Intuitively, network flows data naturally form a graph, where hosts and network flows between hosts in network can be viewed as nodes and edges between nodes in graph, respectively (See Fig. 1 for more details). An earlier research conducted by Staniford-Chen et al [12] has proposed the application of graph theory in modelling network behaviors to improve the effectiveness of NIDS in managing the collaborative attack capability of large networks. In recent studies [13], the Graph Neural Networks (GNNs), which are a powerful deep learning method for graph structured data, have been introduced into NIDS. Most DL-based NIDS directly apply GNN models [14, 15]. These models are node-centric, i.e., they rely heavily on node features but do not adequately utilize edge features. Node-centric representations assume that nodes are potentially related to their neighbors. It is not entirely suitable for extracting the interaction information of the graph constructed by network traffic flows, where the core information of the graph is concentrated in edges

rather than nodes in the graph. Some NIDS models extend the graph neural networks to more accurately uncover anomalous behavioral characteristics behind network flow [16]. Specifically, these revised models accept both node and edge features and integrate edge information into the hierarchical feature representation. But they are usually relatively straightforward, and do not fully take the advantages of GNN and the properties of network traffic flows.

Most deep learning-based NIDSs are supervised paradigm, which require numerous accurately labelled data samples [17]. These methods use supervised signals labelled by hand to establish mapping between data features and categories. However, in practice, only a fews part of network traffic can typically be labelled. With the varying and advancing patterns and strategies of cyber-attacks, it is unfeasible to encompass all possible forms of attacks even with a vast pool of labelled data. This requires network intrusion detection systems to be adaptive and flexible enough to effectively detect new and unknown attacks [18]. Labelled information is user-friendly for humans to comprehend and make decisions, but its assistance to models in generalizing knowledge may be limited, imprecise and even fragile. Because the intrinsic associations of the data may be not exactly precise to be simply divided by the labels, and imbalances in the fixed label categories of the samples may affect the model preferences, especially in networks with diverse attacks and noisy. Moreover, in real-time network environments, obtaining high-quality labels usually implies high costs or is not accessible. Duan et al. [19] enhance the information aggregation capability of the convolution process by using a linear structured graph neural network and introduce a semi-supervised training method to reduce the reliance on labels. The best current anomaly detection accuracy was achieved on multiple datasets, however, fully unsupervised learning remains to be achieved.

To address the aforementioned problems, a GNN-based self-supervised representation learning is proposed for NIDS, where our focus is on edge features, i.e., network flow information and graph topology. Compared to conventional DL-based intrusion detection approaches, our model performs self-supervised learning that automatically reveals the underlying structure of the data without the need for manual labeling or any prior knowledge. This reduces the labor and time costs while expanding the scope of intrusion detection for unknown attacks in networks. The main contributions of this paper are as follows:

- 1) We propose NetFlow-Edge Generative Subgraph Contrast (NEGSC), a self-supervised graph representation learning method for identifying malicious attacks and their specific types. The framework deeply incorporates the features of network flows in a self-supervised graph contrast learning framework Generative Subgraph Contrast (GSC), and pays attention to the local structure of graph, which generates contrastive subgraphs by the generation module based on the center nodes and their direct neighbor nodes. This is beneficial for revealing different types of malicious traffics more elaborately. Furthermore, a structured contrastive loss function is used to distinguish the contrastive subgraph samples by fully exploiting the local network flow structure and attributes.



**Fig. 1.** Converting Netflow-based data into graph representation. An arrow along with nodes indicates a network traffic flow from the source host to the destination host. Normal and attack flows are denoted by black and red arrows, respectively, where different shades of red arrows indicate different types of attacks.

2) To efficiently utilize the raw data of network flows for constructing graph embedding fed to NEGSC, we introduce an edge-featured self-attention mechanisms, named NetFlow-Edge Graph Attention Network (NEGAT), as the encoder of NEGSC. It makes better use of the information among adjacent network flows by weighted aggregation. Additionally, in NEGAT, the node acts as an auxiliary relay of network flow to obtain the encoding of adjacent edge features, thus avoiding significant computational cost for the attention weights.

Extensive experiments are conducted on four well-known modern intrusion detection datasets in both binary and multiclass classification scenarios. The datasets (NF-Bot-IoT, NF-Bot-IoT-v2, NF-CSE-CIC-IDS2018 and NF-CSE-CIC-IDS2018-v2) are all NetFlow-based. In binary classification scenario, our method significantly outperforms the existing self-supervised learning method in terms of Acc, Precision, Recall and F1. Due to that there is no unsupervised GNN-based model for multiclass classification, our method is compared to other latest supervised GNN-based model. The experiment indicates that our method, in a self-supervised manner, can match the supervised GNN-based model on the metrics Recall and F1.

The rest of the paper is organized as follows. Section 2 reviews the related works. The proposed methods in detail are introduced in section 3. Section 4 describes the experimental setup and results. In Section 5, we summarize our work and discuss the future research directions.

## 2. Related Work

The section mainly explores the trends of DL-based methods in NIDSs. These methods provide new perspectives and tools

to address the increasing frequency and complexity of cyberattacks, and have profoundly impacted cybersecurity.

### 2.1. DL-based NIDSs

DL methods exhibit superior capabilities in handling large-scale data when compared with traditional ML methods. It autonomously extracts feature representations from raw data, eliminating the need for manual feature selection and thereby enhancing the model's expressiveness [20].

Wang et al. [21] proposed a CNN-based NIDS, training a Deep Learning-based detection model using extracted features and raw network traffic. The study demonstrated superior accuracy when the model was trained on raw traffic compared to using extracted features. Gupta et al. [22] proposed an anomaly based network intrusion detection system LIO-IDS that uses Long Short Term Memory classifiers and improved One-vs-One technique to handle network intrusions. Jiang et al. [23] proposed a model that combines CNN and Bidirectional Long Short-Term Memory (BiLSTM) for Intrusion Detection Systems (IDS). Le et al. [24] proposed a model applying the RNN and its variants, and the detection error rate of their model is lower than the baseline on NSL-KDD and ISCX data.

Although DL-based NIDSs have achieved remarkable results especially in large-scale data scenario, these methods often overlook or are incapable of dealing with the inherent graph structure of network traffic flows, that is beneficial for identifying anomalous nodes and behaviors [25].

### 2.2. GNN-based NIDSs

GNNs are a powerful DL method for graph-structured data. In recent years, GNNs have attracted much attention for their

excellent performance in NIDS. The inherent graph formation of network flows makes GNNs particularly effective in this context. The main challenge of applying GNNs to NIDS is that the datasets in NIDS are usually edge-centered, that is, the core information is concentrated on edges (network flows) and the classification task is for edges (network flows).

Zhou et al. [26] proposed a GNN-based approach for automatic learning and detection of botnet strategies. Their end-to-end data-driven approach utilizes GNNs to capture specific botnet topologies, relying solely on graph structures for detection. Xiao et al. [27] gave a network anomaly detection method for automatically learning implicit features of network traffic. They transformed the network traffic into first- and second-order graphs, obtained low-dimensional vector representations of nodes, and trained classifiers for anomaly detection. Lo et al. [14] proposed an innovative GNN-based NIDS to detect malicious traffic in IoT networks. The approach, E-GraphSAGE, is an extension of the original GraphSAGE algorithm [16], that generates edge embeddings and implements edge classification to categorize network flows using edge features and topology information. These results further prove the potential and advantages of GNN in detecting network intrusion.

The aforementioned methods employ GNNs for network anomaly detection under a supervised learning manner, relying on pre-labeled data during training. However, labeled data represents only a tiny fraction of the total data under realistic network environment. On the other hand, self-supervised learning has shown great potential as an effective strategy to address the challenges of scarcity of labeled data, such as the limited availability of training data, domain-specific requirements, and vulnerability to labeling against attacks [28, 29].

Lo et al. [30] proposed a self-supervised GNN-based approach called Anomal-E for distinguishing between normal and malicious network traffics. Anomal-E employs E-GraphSAGE as an encoder to generate graph embedding from raw NetFlow data, and then feeds the embedding to a revised version of a self-supervised learning framework Deep Graph Infomax (DGI) [31]. The revised version leverages edge features and graph topological structure in original DGI framework. The positive and negative embeddings by Anomal-E are utilized to tune and optimize E-GraphSAGE, which is used in downstream anomaly detection algorithms. This is the first successful approach to network intrusion detection using a self-supervised GNN. Very recently, Nguyen et al. [15] proposed an intrusion detection method named TS-IDS with a self-supervised module. It converts the problem of edge classification in graph into predicting node attributes. The loss function is obtained by combining the self-supervised loss of the nodes and the supervised loss of the edges. Note that TS-IDS is still supervised, where the supervised loss is computed using edge labels, and the self-supervised module is used to enhance the graph representation.

Even though GNN-based NIDSs have received increasing interests, there is a lack of the self-supervised GNNs classification results for the problem. To the best of our knowledge, no GNN-based algorithm is capable for the multiclass classification task in NIDS under a self-supervised manner. Meanwhile,

many sophisticated deep learning techniques, such as attention mechanisms, have not yet been widely applied in NIDS.

### 2.3. Edge-Featured Approaches

As in NIDS, edge features play an essential role in many other fields of the real world. For example, in social networks, edges represent interpersonal interactions, the importance and complexity of which can be captured by information about the edges. Many studies have begun to focus on and utilize the properties or features of edges to enhance graph data processing.

The EGNN framework proposed by Gong and Cheng [32] can more fully utilize edge features in the graph, including undirected or multidimensional edge features. It uses the edge features to compute the weight matrix, which assists in propagating the node features. Jiang et al. [33] proposed a new graph neural network framework, CensNet, which can utilize graph structure, node, and edge features to learn both node and edge embeddings. Chen et al. [34] proposed an Edge-Featured Graph Attention Network (EGAT) that can handle both node and edge features, combining edge features and node features to compute message and attention weights so that edge features can effectively participate in learning graph feature representations.

It is worth mentioning that the above discussed methods may not be directly suitable for NIDSs. They consider both edge data and node data as essential factors during the aggregation, and focus on node classification tasks. But in NetFlow-based databases in NIDS, each entry (a network flow sample) represents the features of corresponding edge, and the objective is to achieve good results on edge classification tasks.

## 3. Methodology

In this section, we first present the notation used in the paper, and describe the data preprocessing in details. Then, an GNN-based unsupervised intrusion detection approach for NetFlow-based network traffic flows is introduced. The approach consists of two main sequential parts. The first part is an improved encoder named NEGAT based on GAT, which combines the attention mechanism with edges information to elaborately extract valuable features from inherent network topology and the differentiation between traffic flows. The second part is a self-supervised method named NEGSC based on graph contrastive learning [35], which uses NEGAT as encoder and focuses on local structure information of graph for enhancing sufficient intrusion classification performance. Our method achieves both binary and multiclass classification tasks for network flows via self-supervised learning. Finally, the time complexity analysis of our model is also provided.

### 3.1. Preliminary

Denote an undirected graph  $G$  by  $(V, E)$ , where  $V$  is the set of nodes and  $E$  is the set of edges in  $G$ . Let  $n$  and  $m$  be the number of nodes and the number of edges, respectively. We use  $\vec{h}_v$  to represent the features of node  $v$  for each  $v \in V$ , where

$\vec{h}_v \in \mathbb{R}^{F_V}$ , and  $\vec{e}_{vu}$  to represent the features of edge  $e_{vu}$  between nodes  $v$  and  $u$  for each  $e_{vu} \in E$  in  $G$ , where  $\vec{e}_{vu} \in \mathbb{R}^{F_E}$ .  $F_V$  (or  $F_E$ ) is the dimension of node (or edge) features. The set  $N_v$  represents the direct neighbor nodes of node  $v$ .

### 3.2. Data Preprocessing

Data preprocessing plays a crucial role in transforming raw NetFlow data into graph data for our training and testing. The following provides a detailed description of our data preprocessing procedure. Firstly, critical information for each flow is extracted from a large amount of NetFlow data in datasets. This information includes the source IP address, destination IP address, source port number, destination port number, TCP markers, byte counts and protocol type. Forming the basis for constructing the graph data, the source IP address and destination IP address are considered as nodes, while information such as TCP markers, protocol type, and byte counts are treated as edge features. Secondly, direct training of all data would consume too many resources due to the large size of some original datasets. As a result, we opt for the down-sampling method (grouping datasets by “Attack” attribute and then taking random samples from each group) to reduce the overall data size. The down-sampled data is then divided into a training set and a testing set, with the training set used to train our model and the testing set used to evaluate its performance. Attention is paid to maintaining the consistency of the data distribution during the dataset division. Thirdly, the data is coded to transform categorical features into numerical features. Subsequently, normalization using the StandardScaler function is applied, ensuring that feature values are uniformly distributed within a standard normal distribution with mean 0 and variance 1. This approach helps avoid dominance of the model by specific dimensions with excessively large eigenvalues. Finally, the data is converted into a graph representation compatible with GNN.

### 3.3. Edge-featureed Encoder

Since in NIDS, data is usually organized by NetFlow representing traffic flows between hosts in network, critical information predominantly resides in edge features rather than node features. Notably, in numerous GNNs, including models like GraphSAGE, the process of propagating and aggregation is primarily centered around nodes, with edges playing a supporting role. The risk of overlooking crucial edge information arises if attention is solely directed towards node features. Furthermore, the advanced GNN techniques are rarely used in NIDS, such as attention mechanisms. Despite the relative maturity of applying attention mechanisms to nodes, exploration into their application on edges remains underexplored. Hence, there is a need for an encoder for graph embedding, that imperatively leverages the significance of edge features to facilitate a more comprehensive capture of graph structure information, thereby enhancing the overall performance of our proposed model.

#### 3.3.1. Graph Attention Network

For a host in realistic network environment, the impacts of its related traffic flows are different. Attention mechanism can help

calculate different attention coefficients for each edge during the aggregation process, and thus should enable a more accurate capture and utilization of the importance of edges.

Velickovic et al. [36] introduced the Graph Attention Network (GAT), which currently stands as the predominant approach for incorporating attention in GNN [37]. It employs an attentional mechanism during the processing of input data, facilitating the automatic learning of relationships between nodes. Specifically, GAT assigns different weights to edges connecting a node with its neighboring nodes. Motivated by real-world applications, an Edge-Featured Graph Attention Network (EGAT) is proposed, which is an extension of the GAT and can handle graph learning tasks with node and edge features [34]. In EGAT, both node and edge features play important roles when computing information and attention weights. EGAT proposes an edge feature update mechanism that updates edge features by integrating features of connected nodes, and then uses a multi-scale merge strategy to connect the features at each level to construct a hierarchical representation. Essentially, EGAT is still node-based, with edge information as an adjunct.

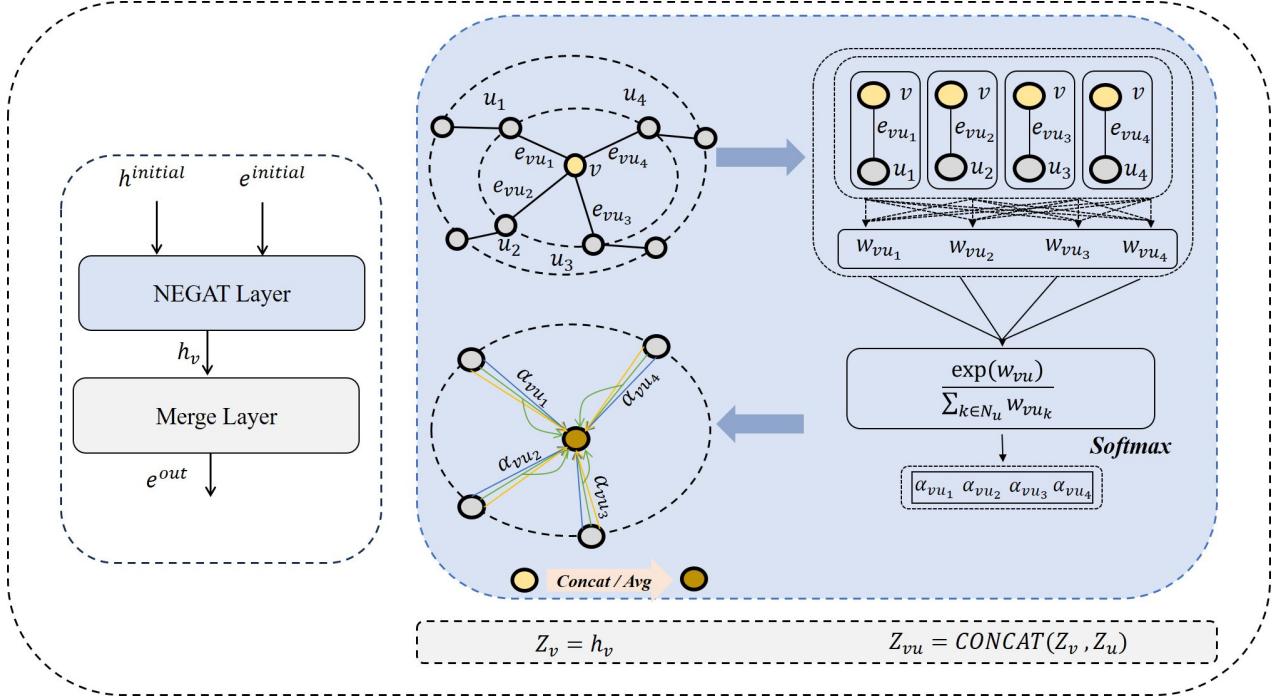
Both GAT and EGAT cannot be directly applied to NIDSs. In NIDSs, the critical information is on network flows between network hosts, and intrusion detection tasks also focus on network flows, that is the edges in graph. This suggests that we further explore and develop an encoder to introduce attention mechanism to graph neural network models, that is able to use the edge information fully and accurately, and thus can better handle edge-dominated tasks in NIDSs.

#### 3.3.2. The Proposed Encoder NEGAT

Due to that the existing graph attention networks do not align with the network flow structure in NIDSs, we propose an NetFlow-Edge Graph Attention Network (NEGAT). Compared with traditional GAT and EGAT, NEGAT focuses on extracting edge information from NetFlow-based data through attention mechanism. It works as an edge-featured encoder that generates edge embeddings for the following self-supervised framework NEGSC. Specifically, the proposed NEGAT first incorporates an attention mechanism into the edges to capture and utilize the importance of the edges more accurately. Then in the process of propagating, NEGAT considers edge data as an essential factor, and node data as transit port for its adjacent edges data, thus avoiding the lift of computational cost by the aggregation of node features. This approach distinguishes NEGAT from existing models and is suitable for the complex network flow structure in NIDS. The details are given in Algorithm 1 and Fig.2.

The input of NEGAT consists of a graph  $G = (V, E)$  with  $n$  nodes and  $m$  edges, and node feature vectors  $\vec{h}_v$  for all  $v \in V$  and edge feature vectors  $\vec{e}_{uv}$  for all  $e_{uv} \in E$ . Note that the attributes in an  $\vec{e}_{uv}$  are mapping from the properties of an entry in database, correspondingly, after preprocessing, and all  $\vec{h}_v$  are initialized as  $[1, \dots, 1]$  (i.e., there is no initial information on nodes).  $K$  is a hyperparameter representing the number of NEGAT layers.

Line 1 of Algorithm 1 sets the edge feature vectors as the



**Fig. 2.** The architecture of NEGAT.  $h$  and  $e$  represent node features and edge features, respectively. The outputs of the NEGAT layer,  $h_v$ , are fed to the merged layer to generate the final representation of  $e^{out}$ . Different colored arrows indicate different attention heads, and the bottom two nodes represent the operation used in our model to concatenate attention heads.

### Algorithm 1 The NEGAT algorithm

**Input:**

A graph  $G = (V, E)$ ;  
edge features  $\{e_{vu}, \forall e_{vu} \in E\}$ ;  
node features  $\{\vec{h}_v = [1, \dots, 1], \forall v \in V\}$ ;  
Layer  $K$ ;  
Neighborhood function  $N_v$ ;  
Weight matrix  $W^k, \forall k \in \{1, \dots, K\}$ ;  
Non-linearity  $\sigma$ ;

**Output:**

Embeddings  $\vec{z}_{vu}$ ;  
1:  $\vec{h}_v^0 \leftarrow \vec{h}_v, \forall v \in V$   
2: **for**  $k = 1$  to  $K$  **do**  
3:   **for**  $v \in V$  **do**  
4:     **for**  $u \in N_v$  **do**  
5:        $w_{vu}^k = a(W^k \vec{h}_v^{k-1} \| W^k \vec{e}_{vu}^{k-1} \| W^k \vec{h}_u^{k-1})$   
6:     **end for**  
7:      $\alpha_{vu}^k = softmax_w(w_{vu}^k)$   
8:      $\vec{h}_v^k \leftarrow \sigma(\sum_{u \in N_v} \alpha_{vu}^k w_1^k (\vec{h}_v^{k-1} \| \vec{e}_{vu}^{k-1} \| \vec{h}_u^{k-1}))$   
9:   **end for**  
10: **end for**  
11: **for**  $v \in V$  **do**  
12:    $\vec{z}_v = \vec{h}_v^K$   
13: **end for**  
14: **for**  $e_{vu} \in E$  **do**  
15:    $\vec{z}_{vu} \leftarrow CONCAT(\vec{z}_v, \vec{z}_u)$   
16: **end for**

initial embedding.

Lines 3-9 describe an NEGAT layer. We have improved GAT to incorporate edge features into the attention mechanism. This improvement can be seen in Line 5 of the algorithm. Unlike the standard GAT method that does not utilize the features of edges for computing the attention coefficient, the computation of attention coefficients for each edge of a node in NEGAT involves concatenating the previous layer embeddings of the source node, target node, and the edges between them. This concrete concatenated function is shown as follows:

$$w_{vu}^k = a(W^k \vec{h}_v^{k-1} \| W^k \vec{e}_{vu}^{k-1} \| W^k \vec{h}_u^{k-1}) \quad (1)$$

, where  $k$  represents the current layer, and  $k - 1$  denotes the previous layer, and  $\|$  represents the feature concatenation operation. Attention mechanism  $a$  is a single-layer feed-forward neural network, where  $W^k$  represents the weight matrix of current layer  $k$  for the input linear transformation.  $\vec{h}_v^{k-1}$  and  $\vec{h}_u^{k-1}$  denote the previous layer embedding of the central node  $v$  and of node  $u$ , one of the neighboring nodes of  $v$ , respectively.  $\vec{e}_{vu}^{k-1}$  refers to the previous layer embedding of edge between the node  $v$  of the node  $u$ . In Line 7, the normalized attention coefficients  $\alpha$  of the node  $v$  are calculated similarly to GAT, using the softmax function on  $w_{vu}^k$  of all adjacent edges of  $v$ . Once  $\alpha$  is obtained, Line 8 executes the aggregation process for the node  $v$ . The aggregation formula is given as follows:

$$\vec{h}_v^k \leftarrow \sigma(\sum_{u \in N_v} \alpha_{vu}^k w_1^k (\vec{h}_v^{k-1} \| \vec{e}_{vu}^{k-1} \| \vec{h}_u^{k-1})) \quad (2)$$

, where  $h_v^k$  denotes the aggregated messages, incorporating the non-linearity  $\sigma$  and the weight matrix  $w_1^k$ . The normalized at-

tention coefficients  $\alpha_{vu}$  are utilized to calculate a linear combination of the embeddings of the node  $v$  and its neighbor nodes  $u$  along with their corresponding edges.

The final embedding of graph  $G$  will be used as input of the NEGSC model, where the embeddings of nodes are essentially their embeddings at the final layer  $K$  (at Lines 11-13), and the embeddings of edges are the concatenation of that of their two end nodes (at Lines 14-16).

### 3.4. The Proposed NEGSC Model

Due to the increasing frequency of attack updates, numerous new or unknown attacks have been arising and need to be responded to in time. To augment the adaptability of NIDS to the network environment, we introduce a self-supervised GNN framework NEGSC based on a graph contrastive learning method GSC, for distinguishing normal network traffic flows and different types of malicious network flows. In this subsection, we first give a brief description of the original GSC, and then present NEGSC, especially its improvements on GSC, that enables our model for multiclass classification task in NetFlow-based NIDSs databases under unsupervised manner.

#### 3.4.1. GSC

Generative Subgraph Comparison (GSC) is a self-supervised method for graph representation learning that captures the local structure of graph by generating contrastive samples [35]. The method utilizes a contrastive learning framework for adaptive subgraph generation to capture the intrinsic graph structure, and employs optimal transmission distance as a similarity measure between subgraphs. Given a graph and its embedding, GSC first samples center nodes, and constructs subgraphs around center nodes by the breadth first search (BFS). Notice that the path between two nodes in subgraph may cross multiple hops. Then a generation module is proposed. For each sampled subgraph, the module adaptively interpolates new nodes by learning the relationship weights between nodes in subgraph, and automatically generate the edges between the interpolated nodes according to the similarities between nodes. It next pairs the sampled and generated subgraph with the same center node as the positive samples, and that with different center node as the negative samples. Finally, two optimal transmission distances, Wasserstein distance and Gromov-Wasserstein distance, are used to construct a structured comparison loss. GSC is a general learning method, that mainly focuses on node representation in the graph. It lacks sufficient consideration for application in NIDS, where the core information is on edges and edges represent real network flows rather than relationships between hosts.

#### 3.4.2. NEGSC

We now introduce NetFlow-Edge Generative Subgraph Contrast (NEGSC), a self-supervised graph representation learning method for identification of malicious attacks and their specific attack types. As shown in Fig.3, NEGSC follows a similar framework as that of GSC. We make numerous improvements by fully exploiting the properties of network traffic flows and

their topological structure, in order to enable NEGSC for accurately and effectively characterizing the difference between network flows in NIDS.

After converting NetFlow-based data to graph representation, the proposed NEGAT, that directly works on edge features, is employed by NEGSC as encoder to get the graph embedding, where each node embedding  $\vec{z}_v$  is aggregated from the adjacent edges features by attention mechanism, and each edge embedding  $\vec{z}_{vu}$  is concatenated by two end nodes embeddings  $\vec{z}_v \parallel \vec{z}_u$ . NEGAT helps NEGSC in extracting important information from raw features of network flows.

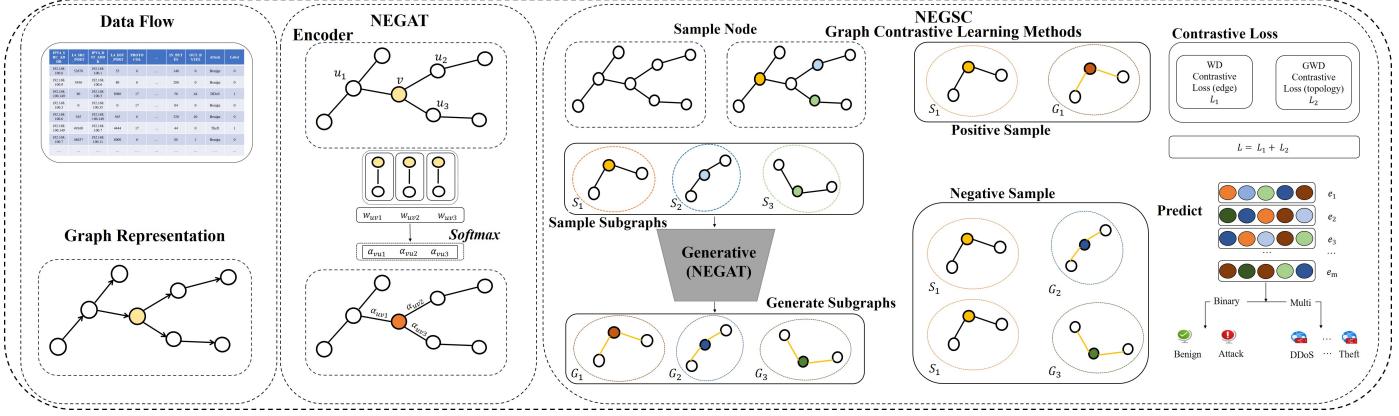
At the graph sampling step, NEGSC samples center nodes and for each center node, constructs subgraph by it and the same number of its direct neighbor nodes. Therefore, in the sample subgraphs, the distance of each path between node and center node is 1, and the longest distance between nodes is no larger than 2. Compared with the BFS method used by GSC, our sampling method is simpler and pays more attention to local structure of graph. It is because that in a network attack scenario, malicious hosts typically launch attacks on adjacent host, and multi-hop attacks only play a small part of all network attacks, especially when the number of hops is larger than 2. And in some specific attacks, such as DoS and DDoS attacks, source address forgery is fairly common, and makes it useless or even impossible to trace network flows far from the host.

At the contrastive subgraph generating step, NEGSC uses a learnable subgraph generation method to adaptively generate contrastive subgraph for each sampled graph, and then pairs the sampled and generated subgraphs with the same center node as the positive samples, and that with different center node as the negative samples. Firstly, for each node  $v$  in any sampled subgraph  $S$ , NEGSC conducts local structure information interpolation to generate a new node  $v'$  for the corresponding generated contrastive subgraph  $G$ . Instead of the formula  $\hat{z}_{v'} = \sum_{u \in N_v} \alpha_{vu} \vec{z}_u$  in GSC, we use

$$\hat{z}_{v'} \leftarrow \sigma \left( \sum_{u \in N_v} \alpha_{vu} \mathbf{w}(\vec{z}_v \parallel \vec{z}_{vu} \parallel \vec{z}_u) \right) \quad (3)$$

in NEGSC as interpolation formula, that is similar to Equation (2), and can utilize the embeddings of both edges and nodes more accurately. Here,  $\hat{z}_{v'}$  is the interpolated embedding of node  $v'$  in generated subgraph  $G$ ,  $\alpha_{vu}$  is the learned attention coefficient between nodes  $v$  and  $u$ , and  $\vec{z}_v$ ,  $\vec{z}_u$  and  $\vec{z}_{vu}$  are the input embeddings of nodes  $u$ ,  $v$  and edge  $e_{vu}$ , respectively. Secondly, GSC directly generates edges between new nodes in generated graph if and only if there exist edges between the corresponding nodes in sampled graph, i.e.,  $e_{v'u'}$  exists in  $G$  IFF  $e_{vu}$  exists in  $S$ . The embedding of each edge  $\hat{z}_{v'u'}$  is the concatenation of its two end nodes representations  $\hat{z}_{v'} \parallel \hat{z}_{u'}$ . NEGSC avoids the time-consuming method in GSC, that generates edges according to the cosine similarity between new nodes. The reason is that in NIDS, edge represents network traffic flow between nodes, and the representation similarity between two nodes does not imply they communicate with each other.

Then, we come to the loss function in NEGSC. A new loss function based on the optimal transport distance is proposed



**Fig. 3.** The architecture of our method. We first employ the encoder to obtain the graph embedding. Then, we obtain the subgraphs for every sampled center nodes. Next, we use the generation module to generate the contrastive sample of each sampled subgraph, and pair the sampled and generated subgraphs with the same center node as the positive samples while the subgraphs with the different center nodes as the negative samples. Finally, we use the proposed loss function to measure the distance between the topological structures of subgraph pairs in the positive and negative samples.

in order to accurately characterize the geometric difference between the sampled subgraph and the generated subgraph in the positive and negative samples, in terms of the edge and graph topological structure. Specifically, we utilize the Wasserstein Distance to measure the contrastive loss on edge features, and utilize the Gromov-Wasserstein Distance to measure the contrastive loss on the topological structure of graph.

Wasserstein Distance (WD) [38] is a common method used for matching two discrete distributions, e.g., two sets of node (or edge) embeddings. Here, we use WD to evaluate the similarity between the edges in the sampled and generated subgraph of each positive or negative sample. Let  $\mu = \{\mu_1, \dots, \mu_m\}$  and  $\nu = \{\nu_1, \dots, \nu_m\}$  be discrete distributions of two subgraphs  $S$  and  $G$ , respectively, where  $\sum_{i=1}^m \mu_i = 1$  and  $\sum_{j=1}^m \nu_j = 1$ , and  $m$  is the number of edges in subgraph (Recall that  $S$  and  $G$  have the same number of edges and nodes due to the graph sampling step and the contrastive subgraph generating step in NEGSC). The WD between two distributions  $\mu$  and  $\nu$  is defined as:

$$WD(S, G) = \min_{T \in \pi(\mu, \nu)} \sum_{i=1}^m \sum_{j=1}^m T_{ij} c(\vec{z}_{e_i}, \hat{z}_{e_j}). \quad (4)$$

$\pi(\mu, \nu)$  denotes all the joint distributions of the edges in subgraphs  $G$  and  $S$ .  $c(\vec{z}_{e_i}, \hat{z}_{e_j})$  is the cost function evaluating the distance between edge  $e_i$  in subgraph  $S$  and edge  $e_j$  in subgraph  $G$ , where  $\vec{z}_{e_i}$  and  $\hat{z}_{e_j}$  represent the embeddings of  $e_i$  and  $e_j$ , respectively. The matrix  $T$  is the transport plan, and  $T_{ij}$  denotes the amount of mass shifted from  $e_i$  and  $e_j$ . Using the same cost function in GSC for  $c(\vec{z}_{e_i}, \hat{z}_{e_j})$ , the WD between  $\mu$  and  $\nu$  can be efficiently achieved by the methods proposed in [39].

Gromov-Wasserstein Distance (GWD) [40] is commonly used to evaluate the distance between pairs of nodes within graph, and further to measure the differences in these distances across the subgraphs. Here, we use GWD to measure the similarity between topological structures of two subgraphs in each positive or negative sample. Similarly,  $\mu = \{\mu_1, \dots, \mu_n\}$  and  $\nu = \{\nu_1, \dots, \nu_n\}$  denote distributions of two subgraphs  $S$  and  $G$ , respectively, where  $\sum_{i=1}^n \mu_i = 1$  and  $\sum_{j=1}^n \nu_j = 1$ , and  $n$  is

the number of nodes in subgraph. The GWD of two discrete distributions  $\mu$  and  $\nu$  is defined as:

$$GWD(S, G) = \min_{T \in \pi(\mu, \nu)} \sum_{v, u, v', u'} T_{vv'} T_{uu'} \hat{c}(\vec{z}_v, \hat{z}_{v'}, \vec{z}_u, \hat{z}_{u'}). \quad (5)$$

Edges  $e_{vu}$  and  $e_{v'u'}$  respectively exist in the subgraphs  $S$  and  $G$ .  $\hat{c}(\vec{z}_v, \hat{z}_{v'}, \vec{z}_u, \hat{z}_{u'}) = \|c(z_v, z_u) - c(z_{v'}, z_{u'})\|_2$  is the cost function measuring the topological structure distance between two pairs of nodes in different subgraphs, where we used the same function  $c(\cdot, \cdot)$  in GSC to denote the distance between the two nodes within a subgraph.

WD-Based Contrastive Loss is:

$$\mathcal{L}_{edges} = \frac{-1}{N(M+1)} \sum_{i=1}^N [\log(\exp(-WD(S_i, G_i)/\tau)) + \sum_{ni \in \{1, \dots, M\} - \{i\}} \log(1 - \exp(-WD(S_i, G_{ni})/\tau))] \quad (6)$$

for measuring the similarity of edges, and GWD-Based Contrastive Loss is:

$$\mathcal{L}_{topology} = \frac{-1}{N(M+1)} \sum_{i=1}^N [\log(\exp(-GWD(S_i, G_i)/\tau)) + \sum_{ni \in \{1, \dots, M\} - \{i\}} \log(1 - \exp(-GWD(S_i, G_{ni})/\tau))] \quad (7)$$

for measuring the similarity of graph topological structure.  $N$  is the number of sampled subgraphs,  $M$  is the number of negative samples of each subgraph, and  $\tau$  is a temperature parameter.  $(S_i, G_i)$  denotes the positive sample subgraphs pair, and  $(S_i, G_{ni})$  denotes the negative sample subgraphs pair. Essentially, the GWD-based contrastive loss in NEGSC is the same as that in GSC, although GSC calls it edge loss while NEGSC calls it topological loss.

Finally, the loss function of NEGSC is defined as follow:

$$\mathcal{L} = \mathcal{L}_{edges} + \mathcal{L}_{topology}. \quad (8)$$

Since the critical information is on network flows in netflow-based NIDS, compared with the function in GSC, our function focuses on the edge embeddings, i.e., the properties of network flows, and the relationships among the adjacent edges, while ignoring the node embeddings.

### 3.5. The Time Complexity Analysis

The running time of our model is determined by its two modules, NEGAT and NEGSC. We first consider the time complexity of NEGAT. In a single attention head, each node need to consider its adjacent edges along with their nodes for computing attention coefficients, and hence each edge is counted twice. Therefore, the attention coefficient computing step takes time  $O(2m)$ , where  $m$  is the number of edge in the graph. Similarly, the aggregating step also takes time  $O(2m)$ . Let  $I$  and  $K$  be the number of attention heads and of the layers, respectively, the time complexity of NEGAT is  $O(4KIm)$ . In the NEGSC module, the first sampling step takes time  $O(n)$ , where  $n$  is the number of nodes in the graph, because the sampled subgraphs have no common nodes. The generation module interpolates new node for each node in sampled subgraphs, which can be regarded as an aggregating step in NEGAT and thus takes time  $O(2Im)$ , and constructs edges between node, which takes time  $O(n)$ . The final step gets the value of the loss function by a time-consuming operation of graph matching. However in our setting of method, the sampled subgraph and the generated subgraph consist of a fixed number of nodes and edges, and the number  $N$  and  $M$  of positive and negative samples are also fixed. Hence the matching operation between subgraphs can be done in constant time. Therefore in all, the time complexity of the proposed model is  $O(4KIm + 2Im + 2n)$ , which is an efficient linear time on the number of nodes and edges in the graph.

### 3.6. Training

Each dataset is divided into a training set and a testing set in the same proportion. Before the training of our self-supervised model NEGSC, all data are converted into graph representation, and are fed into the encoder NEGAT to get a powerful embedding. Note that we use Layer  $K = 1$  in NEGAT, due to consideration of the complexity and computational efficiency of our model. Detailed information on hyperparameters is provided in Table I. In the loss function, we use the binary cross-entropy (BCE) function and the cross-entropy (CE) function. These are commonly used loss functions that can effectively measure the gap between the predicted and true values of the model. In order to optimize the model parameters, back-propagation gradient descent is used. In addition, we use the Adam optimizer and set the learning rate in the training set to 0.001.

## 4. Experiments and Results

In this section, we conduct extensive experiments to evaluate performance of our method under a self-supervised manner for

**Table I**  
Hyperparameter.

Hyperparameter	Value
Layer $K$	1
Attention heads $I$	3
Learning Rate	[2e-2,1e-3]
Activation Func	RELU
Loss Func	BCE/CE
Optimizer	Adam

both binary and multiclass classification tasks. Besides, ablation experiments are provided to separately illustrate the effectiveness of our proposed encoder NEGAT and self-supervised framework NEGSC.

### 4.1. Setup

Our model is implemented using Python, Pytorch [41] and DGL [42]. The experiments are conducted on a single NVIDIA GeForce RTX 4090 with 24GB of GPU memory. We use the hold-out method [43] to divide the datasets. First randomly choose data from datasets in a fixed proportion. Then in the training phase, 70% of the data are sampled for training. This data is preprocessed and fed into the model, which is iteratively optimized so that the model can learn from it and extract useful information and knowledge. The remaining 30% of the data is used for testing and performance evaluation. Code to reproduce our experiments is available at <https://github.com/renj-xu/NEGSC>.

### 4.2. Datasets

In the study, we select four public NetFlow-based benchmark NIDS datasets: NF-BoT-IoT [6], NF-BoT-IoT-v2 [44], NF-CSE-CIC-IDS2018 [6] and NF-CSE-CIC-IDS2018-v2 [44]. The NF-BoT-IoT and the NF-BoT-IoT-v2 are datasets on network traffic of IoT. The NF-CSE-CIC-IDS2018 and the NF-CSE-CIC-IDS2018-v2 are network intrusion detection datasets collected on server and host. Table II specifies the details of the four datasets, including attack categories, predictable attributes, as well as the amount of data we used. All these datasets support binary and multiclass classification scenarios.

The NF-BoT-IoT was generated from the original dataset BoT-IoT [45] and made up of 8 basic NetFlow-based features. It contains 600,100 network flows (entries) in total, out of which 13,859 are benign samples, and 586,241 are attack samples, unevenly distributed among 4 attack types. The NF-BoT-IoT-v2 was generated by the NF-BoT-IoT, that expands the number of network flow features from 8 to 39, and increases the number of network flows. The number of network flows after increasing is 37,763,497, of which 135,037 are benign samples, and 37,628,460 are attack samples.

The NF-CSE-CIC-IDS2018 was generated from the original dataset CSE-CIC-IDS2018[46] and made up of 8 basic NetFlow-based features. It contains 8,392,401 total network flows, out of which 7,373,198 are benign, and 1,019,203 are attack samples, unevenly distributed among 14 attack types. The

**Table II**

Statistics of datasets used in our experiments.

Dataset	Attribute	Classe	Sample	
			version1	version2
NF-BoT-IoT [6] NF-BoT-IoT-v2 [44]	Label	Normal	13859	4051
		Attack	586241	1128853
		Benign	13859	4051
		Theft	1909	73
		DoS	56833	500195
	Attack	DDoS	56844	549955
		Reconnaissance	470655	78630
		Normal	737320	831778
		Attack	101920	112908
		Benign	737320	831778
NF-CSE-CIC-IDS2018 [6] NF-CSE-CIC-IDS2018-v2 [44]	Attack	DDOS attack-HOIC	23	54043
		DoS attacks-Hulk	10814	21632
		DDoS attacks-LOIC-HTTP	37820	15365
		Bot	1568	7155
		Infiltration	6207	5818
		SSH-Bruteforce	5654	4749
		DoS attacks-GoldenEye	3285	1386
		FTP-BruteForce	11602	1297
		DoS attacks-SlowHTTPTest	10555	706
		DoS attacks-Slowloris	2282	476
		Brute Force -Web	261	107
		DDOS attack-LOIC-UDP	167	106
		Brute Force -XSS	174	46
		SQL Injection	4	22

NF-CSE-CIC-IDS2018-v2 was generated by the NF-CSE-CIC-IDS2018, that also expands the number of network flow features from 8 to 39, and increases the number of network flows. The number of network flows after increasing is 18,893,708, out of which 16,635,567 are benign, and 2,258,141 are attack samples.

The NF-CSE-CIC-IDS2018 and the NF-CSE-CIC-IDS2018-v2 datasets show a significantly smaller number of attack samples than that of benign samples. This situation reflects a common phenomenon in the real world, where anomalous behaviors (e.g., attacks) are usually less than benign behaviors (e.g., normal network flow) in a network environment.

Considering that the NF-BoT-IoT-v2, NF-CSE-CIC-IDS2018 and NF-CSE-CIC-IDS2018-v2 datasets are too large to process completely in our experiment environment, we randomly selected 3%, 10%, and 5% of the original data, respectively. Each dataset has two attributes, providing information and diversity of network flows. The attribute “Label” helps us evaluate the performance of the model in binary classification tasks, where 0 represents benign behavior and 1 represents anomalous behavior; the other attribute “Attack” helps us evaluate our model performance in multiclass classification tasks, which is assigned difference values representing a specific type of attack. See table II for more details.

#### 4.3. Evaluation Metrics

To evaluate the performance of our proposed NIDS model, we select four metrics, namely, Accuracy (Acc) rate, Recall rate, Precision rate, and F1 Score. These metrics have been widely utilized in numerous studies [47, 48], that can be used to evaluate the performance of model in both multiclass classification and binary classification.

The calculation of these metrics is based on four counts: true positives ( $TP$ ), the model correctly predicts the positive class; true negatives ( $TN$ ), the model correctly predicts the negative class; false positives ( $FP$ ) The model incorrectly predicts the positive class, and false negatives ( $FN$ ) The model incorrectly predicts the negative class.

Accuracy measures the proportion of correct predictions for all samples.

$$\text{Acc} = \frac{TP + TN}{TP + FP + TN + FN} \times 100\% \quad (9)$$

Precision measures the proportion of samples predicted as attacks by the model that are actual attacks.

$$\text{Precision} = \frac{TP}{TP + FP} \times 100\% \quad (10)$$

Recall reflects the ability of the model to identify attacks and measures the sensitivity of the model to attack behavior.

$$\text{Recall} = \frac{TP}{TP + FN} \times 100\% \quad (11)$$

The F1 score is the harmonic mean of Precision and Recall.

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \times 100\% \quad (12)$$

#### 4.4. Experimental Results

The subsection provides extensive experiment results on our proposed method for classifying network flows in NIDS under self-supervised manner. Binary classification experiments are first provided to illustrate the ability of distinguishing between benign and malicious network flows. Then, more complex multiclass classification experiments are conducted to evaluate the

**Table III**  
On dataset NF-CSE-CIC-IDS2018 binary classification results.

Model	Acc	Precision	Recall	F1
Anomal-E (0%)-average	82.65%	89.12%	82.65%	84.73%
Anomal-E (0%)-max	87.32%	89.35%	87.32%	88.12%
Anomal-E (4%)-average	79.60%	81.95%	79.60%	80.54%
Anomal-E (4%)-max	85.46%	88.65%	85.46%	86.68%
<b>Ours</b>	<b>97.66%</b>	<b>96.75%</b>	<b>96.77%</b>	<b>96.76%</b>

**Table V**  
On dataset NF-CSE-CIC-IDS2018-v2 binary classification results.

Model	Acc	Precision	Recall	F1
Anomal-E (0%)-average	97.83%	97.86%	97.83%	97.75%
Anomal-E (0%)-max	<b>97.87%</b>	<b>97.90%</b>	<b>97.87%</b>	<b>97.79%</b>
Anomal-E (4%)-average	96.14%	95.66%	94.45%	94.74%
Anomal-E (4%)-max	97.02%	96.98%	97.02%	96.90%
<b>Ours</b>	97.76%	97.78%	97.76%	97.67%

ability of identifying different attack types. Additionally, ablation experiments are given to evaluate the effectiveness and contribution of each component (NEGAT and NEGSC) in our method. We report that the results by excerpting from the corresponding papers or by running the code when available.

#### 4.4.1. Binary Classification Results

In the binary classification experiments, we compare the proposed method with Anomal-E [30] and TS-IDS [15]. Anomal-E is the only GNN-based self-supervised approach for binary classification in NIDS. TS-IDS is a latest GNN-based supervised approach, that incorporates a self-supervised module.

The Anomal-E method uses four traditional anomaly detection algorithms to get the results: principal component analysis-based anomaly detection, isolation forest-based anomaly detection, clustering-based local outlier factor and histogram-based outlier score. For the sake of simplicity, our method is compared with the max and average value of the four anomaly detection algorithms results in Anomal-E. Anomal-E (0%) indicates that all sampled network flows in the training sets are benign, and Anomal-E (4%) indicates that 4% of sampled network flows are attacks. It is worth noting that in our method, the sampled data in the training and testing sets follow the same distribution in the corresponding dataset.

Table III and Table IV show the binary classification results of our method and Anomal-E on the NF-CSE-IDS2018 and NF-BoT-IoT datasets, respectively, in terms of Acc, Precision, Recall and F1. Compared with the Anomal-E, our method

**Table IV**  
On dataset NF-BoT-IoT binary classification results.

Model	Acc	Precision	Recall	F1
Anomal-E (0%)-average	20.67%	63.80%	20.67%	28.05%
Anomal-E (0%)-max	42.34%	96.58%	42.34%	57.22%
Anomal-E (4%)-average	22.10%	91.88%	22.10%	34.40%
Anomal-E (4%)-max	23.28%	92.20%	23.28%	35.97%
<b>Ours</b>	<b>98.77%</b>	<b>98.70%</b>	<b>98.77%</b>	<b>98.59%</b>

**Table VI**  
On dataset NF-BoT-IoT-v2 binary classification results.

Model	Acc	Precision	Recall	F1
Anomal-E (0%)-average	38.75%	99.42%	38.75%	55.37%
Anomal-E (0%)-max	43.51%	<b>99.44%</b>	43.51%	60.28%
Anomal-E (4%)-average	29.68%	99.26%	29.68%	44.82%
Anomal-E (4%)-max	39.82%	99.21%	39.82%	56.64%
<b>Ours</b>	<b>99.64%</b>	99.29%	<b>99.64%</b>	<b>99.46%</b>

achieves the best values in all four metrics, and significant outperforms on the NF-BoT-IoT dataset. Table V and Table VI report the comparison results on the NF-CSE-IDS2018-v2 and NF-BoT-IoT-v2 datasets, respectively. On the NF-CSE-IDS2018-v2 dataset, our method achieves Acc of 97.76%, Precision of 97.78%, Recall of 97.76% and F1 of 97.67%, which are very close to those of Anomal-E. On the NF-BoT-IoT-v2 dataset, our method outperforms Anomal-E in all the metrics except Precision, whose value is also very close. It can be found that our method achieves the better predictive performance than Anomal-E, especially on the datasets in which the number of benign flows is much smaller than that of attack flows. It is probably because Anomal-E needs sufficient benign flows for training, while our method, based on Generative Subgraph Comparison, is more adapted to the situation.

To further validate the performance of our model, a comparison with the supervised GNN-based method TS-IDS is conducted. As shown in Table VII, on the NF-BoT-IoT, NF-BoT-IoT-v2, and NF-CSE-CIC-IDS2018 datasets, our method is better than TS-IDS in terms of the four metrics, and even has the gain of 9% improvement in Acc and Recall on NF-BoT-IoT-v2. On the NF-CSE-CIC-IDS2018-v2 dataset, TS-IDS slightly exceeds our method of less than 2% in all metrics.

Consequently, the results show that our method has an excellent capability of binary classification compared with the latest methods, which can effectively distinguish normal and malicious network flows without labeled samples.

**Table VII**

Comparison of our method and TS-IDS in binary classification results.

Model	Datasets	Metric			
		Acc	Precision	Recall	F1
TS-IDS	NF-BoT-IoT	92.67%	98.16%	92.67%	94.67%
Ours	NF-BoT-IoT	<b>98.77%</b>	<b>98.70%</b>	<b>98.77%</b>	<b>98.59%</b>
TS-IDS	NF-BoT-IoT	90.50%	98.12%	90.50%	93.45%
Ours	-v2	<b>99.64%</b>	<b>99.29%</b>	<b>99.64%</b>	<b>99.46%</b>
TS-IDS	NF-CSE-CIC-	89.58%	93.36%	89.58%	90.66%
Ours	IDS2018	<b>96.77%</b>	<b>96.75%</b>	<b>96.77%</b>	<b>96.76%</b>
TS-IDS	NF-CSE-CIC-	<b>99.95%</b>	<b>99.02%</b>	<b>98.85%</b>	<b>99.95%</b>
Ours	IDS2018-v2	97.76%	97.78%	97.76%	97.67%

#### 4.4.2. Multiclass Classification Results

Multiclass classification experiments are conducted to demonstrate the ability of the proposed method for distinguishing different types of network flows without labels.

Table VIII presents the detailed values of Recall and F1 achieved on two datasets, NF-CSE-CIC-IDS2018 and NF-CSE-CIC-IDS2018-v2. Recall reaches 95.79% and 97.85% for two datasets respectively. Especially for the Dos attacks-Hulk, SSH-Bruteforce, and FTP-Bruteforce in the NF-CSE-CIC-IDS2018-v2 dataset, the high Recall values of 99.60%, 99.93% and 99.49%, respectively, are achieved. The detection rates for DoS attacks-SlowHTTPTest, Brute Force-XSS and SQL Injection are 0 on both datasets. It may be caused by lack of sufficient corresponding flow samples due to the data unbalance in datasets (refer to Table II). Table IX reports the result of Recall and F1 on the NF-BoT-IoT and NF-BoT-IoT-v2 datasets, respectively, where Recall achieves 83.06% and 95.16%, and F1 reaches 82.70% and 95.15%. The difference between our method performances on two datasets may rely on that NF-BoT-IoT-v2 provides more attribute features than NF-BoT-IoT.

**Table VIII**

Multiclass classification results of our method on NF-CSE-CIC-IDS2018 and NF-CSE-CIC-IDS2018-v2.

Class Name	NF-CSE-CIC-IDS2018		NF-CSE-CIC-IDS2018-v2	
	Recall	F1	Recall	F1
Benign	98.95%	98.36%	99.72%	98.89%
DDOS attack-HOIC	00.00%	00.00%	90.17%	94.00%
DoS attacks-Hulk	87.11%	86.65%	99.60%	98.72%
DDoS attacksLOICHTTP	99.83%	99.77%	87.00%	91.25%
Bot	00.00%	00.00%	53.91%	64.55%
Infiltration	00.00%	00.00%	00.17%	00.33%
SSH-Bruteforce	88.15%	87.53%	99.93%	98.82%
DoS attacks-GoldenEye	82.76%	64.97%	91.35%	89.10%
FTP-BruteForce	82.61%	65.66%	99.49%	78.50%
DoS attacksSlowHTTPTest	00.00%	00.00%	00.00%	00.00%
DoS attacks-Slowloris	01.17%	02.27%	00.00%	00.00%
Brute Force -Web	01.28%	02.38%	00.00%	00.00%
DDOS attack-LOIC-UDP	74.00%	69.81%	65.62%	79.25%
Brute Force -XSS	00.00%	00.00%	00.00%	00.00%
SQL Injection	00.00%	00.00%	00.00%	00.00%
<b>Weighted Average</b>	<b>95.79%</b>	<b>94.79%</b>	<b>97.85%</b>	<b>97.43%</b>

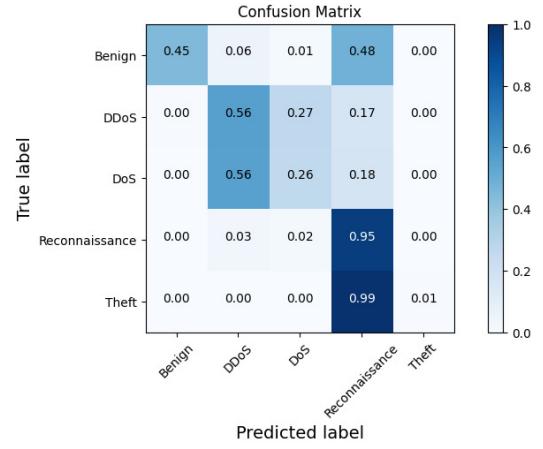
As seen from Table VIII and Table IX, our proposed method performs well on the NF-CSE-CIC-IDS2018, NF-CSE-CIC-IDS2018-v2 and NF-BoT-IoT-v2 datasets, with Recall and F1 around 95%. A relatively poor experimental results are ob-

**Table IX**

Multiclass classification results of our method on NF-BoT-IoT and NF-BoT-IoT-v2.

Class Name	NF-BoT-IoT		NF-BoT-IoT-v2	
	Recall	F1	Recall	F1
Benign	44.76%	58.44%	61.40%	71.83%
DDoS	55.92%	46.42%	97.46%	96.58%
DoS	26.41%	31.35%	93.56%	94.71%
Reconnaissance	94.64%	94.33%	91.11%	89.26 %
Theft	00.52%	01.04%	31.82%	24.35%
<b>Weighted Average</b>	<b>83.06%</b>	<b>82.70%</b>	<b>95.16%</b>	<b>95.15%</b>

tained on the NF-BoT-IoT dataset. Therefore, we illustrate the confusion matrix for the NF-BoT-IoT dataset in Fig.4. From the Confusion matrix, it can be found that most theft and benign samples in the NF-BoT-IoT datasets are misclassified as reconnaissance samples. This may be related to the similar characteristics of these three types of samples. Compared to reconnaissance samples that are the most numerous, far more than other categories, the theft and benign samples only occupy a minor fraction of the data in the dataset. Our method may need more corresponding data for training. DoS attacks are misclassified as DDoS attacks, probably because DoS samples are most similar to DDoS samples when extracting features.

**Fig. 4.** Confusion matrix of multiclass classification results on NF-BoT-IoT.

The experimental result in Table X shows the comparison between the supervised method TS-IDS and our method in Recall and F1 metrics. On the NF-BoT-IoT dataset, our method obtains the gain of at least 2% improvement on both metrics, and achieves Recall of 95.16% and F1 of 95.15% on NF-BoT-IoT-v2, closing to that of TS-IDS. Note that we do not obtain experimental data for TS-IDS on the NF-CSE-CIC-IDS2018 and NF-CSE-CIC-IDS2018-v2 datasets, because the provided code can not satisfy the requirements when performing data processing. So we directly excerpt the result for the NF-CSE-CIC-IDS2018-v2 dataset from the paper of TS-IDS [15], which is better than that of our method within 2% in the two metrics.

The Receiver Operating Characteristic (ROC) curve comprehensively shows the performance of the model under different thresholds, including the true positive rate and the false positive rate, and the ROC curve reflects the model performance

stably even if the distribution of positive and negative samples in the testing set varies. Due to the failure of completing the experiments of TS-IDS on the NF-CSE-CIC-IDS2018 and NF-CSE-CIC-IDS2018-v2 datasets, only the ROC curves of the NF-BoT-IoT and NF-BoT-IoT-v2 datasets are shown. Fig. 5 and Fig. 6 describe the ROC curves of our method and TS-IDS for the NF-BoT-IoT and NF-BoT-IoT-v2 datasets, respectively. The micro-average of the ROC curve by our method in the NF-BoT-IoT dataset is 89%, that is better than 86% of the TS-IDS method. Specifically, compare to the TS-IDS method, the areas of the ROC curves of our method for the DoS and DDoS samples are significant larger, while the areas for the Reconnaissance and Theft samples are smaller.

Fig. 7 and Fig. 8 report the ROC curve results of the TS-IDS method and our method on the NF-BoT-IoT-v2 dataset, respectively. The micro-average of the ROC curve for our method is 97%, which is less than 99% achieved by TS-IDS. That implies that our method slightly underperforms the TS-IDS method on NF-BoT-IoT-v2. In fact, the areas of the ROC curves for different types all match those of TS-IDS, except the Benign and Theft samples that only account for a small portion of the data. It may be because our self-supervised method needs more corresponding data for training than that of supervised method.

**Table X**  
Comparison of the result of our method with TS-IDS method for multiclass classification.

Model	Dataset	Recall	F1
TS-IDS	NF-BoT-IoT	77.29%	78.20%
Ours	NF-BoT-IoT	<b>83.06%</b>	<b>82.70%</b>
TS-IDS	NF-BoT-IoT-v2	<b>97.89%</b>	<b>97.84%</b>
Ours	NF-BoT-IoT-v2	95.16%	95.15%
TS-IDS	NF-CSE-CIC-IDS2018	-	-
Ours	NF-CSE-CIC-IDS2018	95.79%	94.79%
TS-IDS	NF-CSE-CIC-IDS2018-v2	<b>99.03%</b>	<b>98.08%</b>
Ours	NF-CSE-CIC-IDS2018-v2	97.85%	97.43%

#### 4.4.3. Ablation Experiments

To verify the effectiveness of the encoder NEGAT and self-supervised GNN framework NEGSC used in our method, we perform a series of ablation experiments, and list the detailed results of these experiments in Table XI. In the first experiment, to demonstrate the effectiveness of our proposed self-supervised framework NEGSC, we keep using the proposed NEGAT as encoder and replace NEGSC with original GSC [35]. As can be seen from Table XI, on the four datasets NF-BoT-IoT, NF-BoT-IoT-v2, NF-CSE-CIC-IDS2018 and NF-CSE-CIC-IDS2018-v2, our method (NEGAT + NEGSC) significantly outperform NEGAT+GSC in Rcall and F1 Metrics. Compared to GSC, our proposed NEGSC is capable of handling both edge and node information, and thus is more suitable for NIDS where edge information plays a crucial role. In the second experiment, to verify the effectiveness of the proposed encoder NEGAT, we use the state-of-art method E-GraphSage [14] to replace NEGAT as encoder and feed the graph embedding output of E-GraphSage to NEGSC. From Table XI, it can be found that the value of Recall and F1 decrease after replacing the encoder by E-GraphSage on all four databases, which

shows that our encoder outperforms E-GraphSAGE in network intrusion detection tasks. The reason is that the attention mechanism introduced by NEGAT can help our model better understand the local structure of graph, and more accurately utilize the different impacts of related traffic flows.

**Table XI**  
Results of ablation experiments.

Experimental Seting	Dataset	Recall	F1
NEGAT+NEGSC (ours)	NF-BoT-IoT	<b>83.06%</b>	<b>82.70%</b>
NEGAT+GSC	NF-BoT-IoT	79.27%	72.14%
E-GraphSAGE+NEGSC	NF-BoT-IoT	82.71%	82.19%
NEGAT+NEGSC (ours)	NF-BoT-IoT-v2	<b>95.16%</b>	<b>95.15%</b>
NEGAT+GSC	NF-BoT-IoT-v2	90.21%	90.13%
E-GraphSAGE+NEGSC	NF-BoT-IoT-v2	94.33%	94.27%
NEGAT+NEGSC (ours)	NF-CSE-CIC-IDS2018	<b>95.79%</b>	<b>94.79%</b>
NEGAT+GSC	NF-CSE-CIC-IDS2018	85.03%	87.52%
E-GraphSAGE+NEGSC	NF-CSE-CIC-IDS2018	87.81%	83.93%
NEGAT+NEGSC (ours)	NF-CSE-CIC-IDS2018-v2	<b>97.85%</b>	<b>97.43%</b>
NEGAT+GSC	NF-CSE-CIC-IDS2018-v2	88.36%	88.23%
E-GraphSAGE+NEGSC	NF-CSE-CIC-IDS2018-v2	97.42%	96.69%

## 5. Conclusion

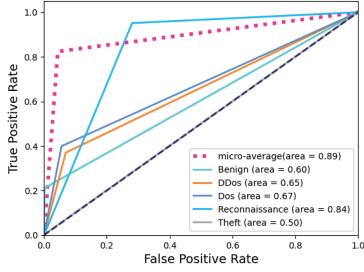
This paper presented a self-supervised graph neural network for network intrusion detection systems, that is designed to efficiently and elaborately differentiate the normal network flows and the malicious flows with different attack types. To the best of our knowledge, this is the first GNN-based method for multiclass classification tasks in NIDS in an unsupervised manner. Extensive experimental evaluations on multiple netflow-based datasets demonstrate the preliminary generalization capability and application potential of our method in real-world traffic detection scenarios. In addition, although here the proposed NEGSC is used for the edge-centered field NIDS, it is actually capable of simultaneously handling node and edge information, and thus may find applications in other scenarios. Our future work for NIDS will focus on solving the problem of data imbalance in datasets and consider the timing dependency of network flows in real network environments.

## Declaration of competing interest

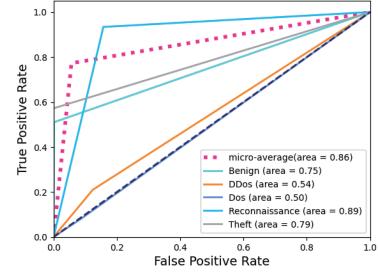
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Author Statements

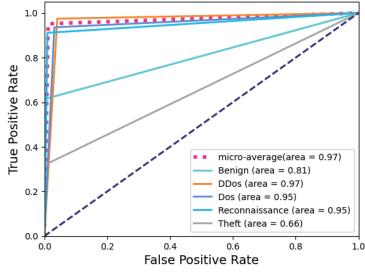
Conceptualization: R. Xu, G. Wu, W. Wang and A. He; Investigation: R. Xu, X. Gao and Z. Zhang; Writing - original



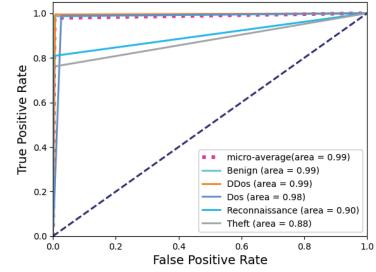
**Fig. 5.** ROC curve of our method on NF-BoT-IoT.



**Fig. 6.** ROC curve of TS-IDS method on NF-BoT-IoT.



**Fig. 7.** ROC curve of our method on NF-BoT-IoT-v2.



**Fig. 8.** ROC curve of TS-IDS method on NF-BoT-IoT-v2.

draft: R. Xu, G. Wu and X. Gao; Methodology: R. Xu G. Wu and Z. Zhang; Software: R. Xu; Validation: G. Wu; Visualization: R. Xu and Z. Zhang; Resources: G. Wu, W. Wang and A. He; Funding acquisition: G. Wu, W. Wang and A. He; Writing - review and editing: G. Wu and X. Gao; Supervision: G. Wu. All authors have read and agreed to the version of the manuscript.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant 62272486, the Natural Science Foundation of Hunan Province, China, under Grant 2020JJ4949, the Key Project of Teaching Research of Hunan Province under Grant HNJG-2022-0124.

## References

- [1] J. Piet, A. Sharma, V. Paxson, D. Wagner, Network Detection of Interactive SSH Impostors Using Deep Learning, in: USENIX Security Symposium, (USENIX Security), 2023, pp. 4283-4300.
- [2] Y. Qin, W. Wang, S. Zhang, K. Chen, An exploit kits detection approach based on HTTP message graph, IEEE Transactions on Information Forensics and Security, 2021, pp. 3387-3400.
- [3] A.M. Mandalari, H. Haddadi, D.J. Dubois, D. Choffnes, Protected or Porous: A Comparative Analysis of Threat Detection Capability of IoT Safeguards, in: IEEE Symposium on Security and Privacy, (SP), 2023, pp.3061-3078.
- [4] H. Jmila, M.I. Khedher, Adversarial machine learning for network intrusion detection: A comparative study, Computer Networks, 2022, pp. 109073.
- [5] S.I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh, O. Jognunola, Federated Deep Learning for Zero-Day Botnet Attack Detection in IoT-Edge Devices, IEEE Internet of Things Journal, 2021, pp. 3930-3944.dataset1
- [6] M. Sarhan, S. Layeghy, N. Moustafa, M. Portmann, Netflow datasets for machine learning-based network intrusion detection systems, Big Data Technologies and Applications, 2020, pp. 117-135.
- [7] F. Yang, J. Xu, C. Xiong, Z. Li, K. Zhang, PROGRAPHER: An Anomaly Detection System based on Provenance Graph Embedding, in: USENIX Security Symposium, (USENIX Security), 2023, pp. 4355-4372.
- [8] S. Zhang, K. Xiao, J. Yu, X. Liu, W. Wang, Accurate IoT Device Identification based on A Few Network Traffic, in: IEEE/ACM International Symposium on Quality of Service, (IWQoS), 2023, pp. 1-10.
- [9] A.H. Qureshi, H. Larijani, J. Ahmad, N. Mtetwa, A Heuristic Intrusion Detection System for Internet-of-Things (IoT), in: Intelligent Computing Proceedings of Computing Conference, 2019, pp. 86-98.
- [10] H. Holm, Signature based intrusion detection for zero-day attacks:(not) a closed chapter?, in: Hawaii International Conference on System Sciences, (HICSS), 2014, pp. 4895-4904.
- [11] G. Engelen, V. Rimmer, W. Joosen, Troubleshooting an Intrusion Detection Dataset: the CICIDS2017 Case Study, 2021 IEEE Security and Privacy Workshops, 2021, pp. 7-12.
- [12] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R.W. Yip, D. Zerkle, GrIDS-a graph based intrusion detection system for large networks, in: national information systems security conference, (NISS), 1996, pp. 361-370.
- [13] I.A. Chikwendu, X. Zhang, I.A. Mensah, C.C. Ukwuoma, C.J. Ejiji, A Comprehensive Survey on Deep Graph Representation Learning Methods, Journal of Artificial Intelligence Research, 2023, pp. 287-356.
- [14] W.W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, M. Portmann, E-GraphSAGE: A Graph Neural Network based Intrusion Detection System for IoT, in: IEEE/IFIP Network Operations and Management Symposium, (NOMS), 2022, pp. 1-9.
- [15] H. Nguyen, R. Kashef, TS-IDS: Traffic-aware self-supervised learning for IoT Network Intrusion Detection, Knowledge-Based Systems, 2023, pp. 110966.
- [16] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Advances in neural information processing systems, 2017, pp. 1024-1034.
- [17] F. Wei, H. Li, Z. Zhao, H. Hu, xNIDS: Explaining Deep Learning-based Network Intrusion Detection Systems for Active Intrusion Responses, in: USENIX Security Symposium, (USENIX Security 23), 2023, pp. 4337-4354.
- [18] P. Rieger, M. Chilese, R. Mohamed, M. Miettinen, H. Fereidooni, A.R. Sadeghi, ARGUS: Context-Based Detection of Stealthy IoT Infiltration Attacks, in: USENIX Security Symposium, (USENIX Security 23), 2023, pp. 4301-4318.
- [19] G. Duan, H. Lv, H. Wang, G. Feng, Application of a Dynamic Line Graph Neural Network for Intrusion Detection With Semisupervised Learning,

- IEEE Transactions on Information Forensics and Security, 2022, pp. 699-714.
- [20] C. Janiesch, P. Zschech, K. Heinrich, Machine learning and deep learning, Electronic Markets, 2021, pp. 685-695.
- [21] W. Wang, M. Zhu, X. Zeng, X. Ye, Y. Sheng, Malware traffic classification using convolutional neural network for representation learning, in: International Conference on Information Networking, (ICOIN), 2017, pp. 712-717.
- [22] N. Gupta, V. Jindal, P. Bedi, LIO-IDS: Handling class imbalance using LSTM and improved one-vs-one technique in intrusion detection system, Computer Networks, 2021, pp. 108076.
- [23] K. Jiang, W. Wang, A. Wang, H. Wu, Network Intrusion Detection Combined Hybrid Sampling With Deep Hierarchical Network, IEEE Access, 2020, pp. 32464-32476.
- [24] T.T.H. Le, Y. Kim, H. Kim, Network Intrusion Detection Based on Novel Feature Selection Model and Various Recurrent Neural Networks, Applied Sciences, 2019, pp.1392.
- [25] D. Zügner, A. Akbarnejad, S. Günnemann, Adversarial attacks on neural networks for graph data, in: International Joint Conference on Artificial Intelligence, (IJCAI), 2018, pp. 6246-6250.
- [26] J. Zhou, Z. Xu, A.M. Rush, M. Yu, Automating Botnet Detection with Graph Neural Networks, arXiv preprint arXiv:2003.06344, 2020.
- [27] Q. Xiao, J. Liu, Q. Wang, Z. Jiang, X. Wang, Y. Yao, Towards Network Anomaly Detection Using Graph Embedding, in: International Conference on Computational Science, (ICCS), 2020, pp. 156-169.
- [28] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, S.Y. Philip, Graph self-supervised learning: A survey, IEEE Transactions on Knowledge and Data Engineering, 2022, pp. 5879-5900.
- [29] Y. Xie, Z. Xu, J. Zhang, Z. Wang, S. Ji, Self-Supervised Learning of Graph Neural Networks: A Unified Review, IEEE transactions on pattern analysis and machine intelligence, 2022, pp. 2412-2429.
- [30] E. Caville, W.W. Lo, S. Layeghy, M. Portmann, Anomal-E: A self-supervised network intrusion detection system based on graph neural networks, Knowledge-Based Systems, 2022, pp. 110030
- [31] P. Veličković, W. Fedus, W.L. Hamilton, P. Liò, Y. Bengio, R.D. Hjelm, Deep graph infomax, in: International Conference on Learning Representations, (ICLR), 2019.
- [32] L. Gong, Q. Cheng, Exploiting edge features for graph neural networks, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, (CVPR), 2019, pp. 9211-9219.
- [33] X. Jiang, P. Ji, S. Li, CensNet: Convolution with Edge-Node Switching in Graph Neural Networks, in: International Joint Conference on Artificial Intelligence, (IJCAI), 2019, pp. 2656-2662.
- [34] J. Chen, H. Chen, EGAT: Edge-Featured Graph Attention Network, in: 2021 International Conference on Artificial Neural Networks, 2021, pp. 253-264.
- [35] Y. Han, L. Hui, H. Jiang, J. Qian, J. Xie, Generative Subgraph Contrast for Self-Supervised Graph Representation Learning, in: European Conference on Computer Vision, (ECCV), 2022, pp. 91-107.
- [36] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph Attention Networks, in: International Conference on Learning Representations, (ICLR), 2018.
- [37] S. Brody, U. Alon, E. Yahav, How Attentive are Graph Attention Networks?, in: International Conference on Learning Representations, (ICLR), 2022.
- [38] G. Peyré, M. Cuturi, Computational optimal transport, Foundations and Trends® in Machine Learning, 2019, pp. 355-607.
- [39] L. Chen, Z. Gan, Y. Cheng, L. Li, L. Carin, J. Liu, Graph optimal transport for cross-domain alignment, in: International Conference on Machine Learning, (ICML), 2020, pp. 1542-1553.
- [40] G. Peyré, M. Cuturi, J. Solomon, Gromov-Wasserstein Averaging of Kernel and Distance Matrices, in: International Conference on Machine Learning, (ICML), 2016, pp. 2664-2672.
- [41] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An Imperative Style, High-Performance Deep Learning Library, in: Advances in Neural Information Processing Systems, (NIPS), 2019, pp. 8024-8035.
- [42] M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma, Z. Huang, Q. Guo, H. Zhang, H. Lin, J. Zhao, J. Li, A.J. Smola, Z. Zhang, Deep Graph Library: Towards Efficient and Scalable Deep Learning on Graphs, arXiv preprint arXiv:1909.01315v2, 2020.
- [43] C. Bishop, Pattern Recognition and Machine Learning, Information science and statistics, 5th Edition, Springer, New York, 2006.
- [44] M. Sarhan, S. Layeghy, M. Portmann, Towards a Standard Feature Set for Network Intrusion Detection System Datasets, Mobile Networks and Applications, 2021, pp. 259-370.
- [45] N. Koroniots, N. Moustafa, E. Sitnikova, B. Turnbull, Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset, Future Generation Computer Systems, 2019, pp. 779-796.
- [46] I. Sharafaldin, A.H. Lashkari, A.A. Ghorbani, Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization., in: Proceedings of the International Conference on Information Systems Security and Privacy, (ICISSP), 2018, pp. 108-116.
- [47] M. Sarhan, S. Layeghy, M. Portmann, Evaluating standard feature sets towards increased generalisability and explainability of ML-based network intrusion detection, Big Data Research, 2022, pp. 100359.
- [48] D.M.W. Powers, Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation, arXiv preprint arXiv:2010.16061, 2020.