Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

FedSTGCN: a novel federated spatiotemporal graph learning-based network intrusion detection for the Internet of Things

Yalu WANG[1], Jie LI[2], Zhijie HAN[†‡3], Pu CHENG[3], Roshan KUMAR[4]

[1]*School of Computer and Information Engineering, Henan University, Kaifeng 475004, China.*

[2]*School of Intelligent Engineering, Zhengzhou University of Aeronautics, Zhengzhou 450046, China*

[3]*School of Software, Henan University, Kaifeng 475004, China.*

[4]*Miami College, Henan University, Kaifeng 475004, China.*

[†]E-mail: hanzj@henu.edu.cn

Received Oct. 20, 2024; Revision accepted Mar. 25, 2025; Crosschecked

Unedited

Abstract: The rapid growth and increasing complexity of Internet of Things (IoT) devices have made network intrusion detection a critical challenge, especially in edge computing environments where data privacy is a primary concern. Machine learning-based intrusion detection techniques enhance IoT network security but often require centralized network data, posing significant risks to data privacy and security. Although federated learning-based network intrusion detection methods have emerged in recent years to address privacy concerns, they have not fully leveraged the advantages of graph neural networks (GNNs) for intrusion detection. To address this issue, we propose a Federated Spatiotemporal Graph Convolutional Network framework (FedSTGCN), which integrates the capabilities of spatiotemporal graph neural networks (STGNNs) and federated learning (FL). This framework enables collaborative model training across distributed IoT devices without requiring the sharing of raw data, thereby improving network intrusion detection accuracy while preserving data privacy. Extensive experiments were conducted on two widely used IoT intrusion detection datasets to evaluate the effectiveness of the proposed approach. The results demonstrate that FedSTGCN outperforms other methods in both binary and multi-class classification tasks, achieving over 97% accuracy in binary classification and 92% accuracy in multi-class classification.

Key words: Internet of Things; Network intrusion detection, Spatiotemporal graph neural network; Federated learning; Data privacy.

https://doi.org/10.1631/FITEE.2400932 **CLC number:**

1 Introduction

With the rapid growth of the number and complexity of Internet of Things (IoT) devices, network intrusion detection has become increasingly important, particularly for edge computing devices. Devices in edge networks—such as smart TVs, network cameras, smart curtains, and wireless sensors—while greatly convenient for both life and work, have also become potential targets for cyberattacks (Ghasempour et al., 2019) With the exponential increase in the number and frequency of attacks, ensuring the security of these smart devices has become a key issue driving the continuous development of the IoT.

To ensure smart device security, researchers have proposed various network intrusion detection methods. Among them, machine learning-based intrusion detection algorithms are particularly popular because they can automatically learn and identify potential threat patterns by analyzing data flows. In recent years, a sub-field of deep learning—graph neural networks

‡ Corresponding author

© Zhejiang University Press 2025

(GNNs)—has also been applied to network intrusion detection due to its advantages in learning from graph-structured data (Zhou X et al., 2021; Caville E et al., 2022; Wai Weng Lo et al., 2022; X. Hu et al., 2023). Although GNN-based intrusion detection methods address some challenges in network intrusion detection, they rely on centralized deep learning approaches. To train a generalized network intrusion detection model, raw data must be trans-mitted from edge devices to a central server for training, which exposes the data to potential attacks and raises privacy concerns. Therefore, this paper integrates spatiotemporal graph neural networks (STGNNs) with FL and proposes a federated spatiotemporal graph neural network (FSTGNN) model. When applied to network intrusion detection, this approach not only im-proves detection accuracy but also preserves data privacy.

Two key challenges must be overcome to apply FSTGNN to network intrusion detection systems. The first is how to construct local sub-graphs from the network intrusion detection datasets on each client and then build a global graph across all clients. The second challenge is guaranteeing that the aggregation process maintains data privacy, since GNNs are utilized as the training model for each client in the FSTGNN architecture, which necessitates collecting information from surrounding nodes for training GNNs. To tackle the first challenge, this paper utilizes the graph construction method proposed by N-STGAT (Wang Y et al., 2023) and presents an improved approach for constructing a global graph from subgraphs. For the second challenge, the paper employs the graph convolutional aggregation method mentioned in FedCog (Lei R et al., 2023) to prevent privacy leakage during the aggregation process.

This paper's contributions to the field of network intrusion detection are as follows:

• To the best of our knowledge, this is the first time a federated spatiotemporal graph learning model has been used to address the problem of network intrusion detection. This is also the first attempt to perform distributed training using a spatiotemporal graph model.

• A method for constructing edges between subgraphs on different clients is proposed, ensuring both proper edge construction and the protection of data privacy.

• Experiments were conducted on the latest datasets, comparing the proposed method with several recent FL-based approaches. The results demonstrate that the proposed method outperforms other methods in all respects.

The remaining sections of this paper are organized as follows:

Section 2 introduces related work. Section 3 briefly presents the relevant background of this study. Section 4 provides a detailed description of the proposed network intrusion detection method. Section 5 presents the experiments conducted. Section 6 concludes the paper.

2 Related works

In recent years, many scholars have con-tributed significantly to intrusion detection systems, proposing numerous methods for detecting network attacks. These methods can primarily be categorized as traditional machine learning, GNNs, and FGL (federated graph learning), among others.

2.1 Traditional machine learning

Li X et al. (2023) designed a tensor train support vector machine (TT-SVM) model based on dimensionality reduction data processing, which can comprehensively optimize the BDM platform and has been compared and evaluated against other models. Aburomman et al.(2016) utilized support vector machine (SVM) and k-nearest neighbors (KNN) to train their model but created an ensemble classifier with better accuracy using weights generated by particle swarm optimization for intrusion detection. Almogren A S et al. (2020) proposed a deep belief network (DBN) based on advanced intrusion detection methods, which addresses the issue of intrusion detection in distributed environments. In the early applications of machine learning to network intrusion detection, Sangkatsanee P et al. (2011) applied decision tree techniques, reporting good results with their designed RT-IDS. Dina A S et al. (2023) pro-posed a focal loss function to address the issue of data imbalance and applied it to two well-known machine learning methods (feed-forward neural networks [FNNs] and convolutional neural networks [CNNs]).

2.2 Graph neural networks (GNNs)

Since graph neural networks (GNNs) primarily learn from graph structures, they have a natural advantage when handling inherently graph-based network systems. Researchers have recently applied GNNs to intrusion detection systems. Lo et al.(2022)

Unedited

proposed the E-GraphSAGE architecture based on GraphSAGE, which captures edge features and topological information of graphs for IoT net-work intrusion detection. This approach has also provided a solid foundation for subsequent re-searchers to apply GNNs to network intrusion detection. Caville E et al. (2022) introduced a self-supervised network intrusion detection method called Anomal-E, building on E-GraphSAGE. The authors claim that this method is the first successful and practical network intrusion detection approach. Considering the challenges of limited and imbalanced attack data in existing intrusion detection methods, the authors of reference (Zhou X et al., 2021) proposed a novel hierarchical adversarial attack (HAA) generation method to realize the level-aware black-box adversarial attack strategy, targeting the GNNs-based intrusion detection in IoT systems with a limited budget. To address the complexity of existing graph-based intrusion detection methods, which require large amounts of training data, Hu et al. (2023) pro-posed an early and accurate network intrusion detection method based on graph embedding techniques, named Graph2vec + RF.

2.2 Federated graph learning (FGL)

Recently, researchers have integrated GNNs with federated learning (FL), applying FGL to network intrusion detection. To address issues such as label dependency, detection delay, privacy leakage, and imbalanced attack scenario distribution, Qinghua Mao et al. (2025) proposed a method called FeCo-Graph. It is a label-aware federated graph contrastive learning framework designed for intrusion detection in limited scenarios, and its performance has been validated through experiments. Jianping Wu et al. (2024) proposed an attention-based graph neural net-work for detecting cross-level and cross-department network attacks. This method not only supports collaborative model training but also protects data privacy on distributed devices. To tackle model dependency and privacy concerns, Mansour Bahar et al. (2024) integrated FL into a GNN-based intrusion detection system architecture. Compared to the state-of-the-art provenance-based IDS (PID), this approach has shown promising results in reducing false positive rates.

Although these FL-based approaches in-corporate GNNs, they only consider the spatial characteristics of network attacks. In reality, network attacks also exhibit temporal characteristics, as data flows within the same time period tend to have similar properties. Therefore, considering temporal characteristics in network intrusion detection is essential.

3 Background

Since this paper adopts federated spatiotemporal graph learning to address the network intrusion problem, it utilizes two core methods: N-STGAT and FedCog (an FL model based on graph convolutional networks [GCN]). There-fore, it is necessary to provide a brief introduction to N-STGAT and FL.

3.1 N-STGAT

N-STGAT is a centralized network intrusion detection method based on GNNs, which is part of our previous research (Wang Y et al., 2023). It compares the different states of nodes during abnormal and normal traffic, as well as the temporal relationships in detecting abnormal traffic. The method combines GAT (graph attention network—a derivative of GNNs) (Veličković P et al., 2017) with long short-term memory (LSTM) for model training. The model's

Unedited

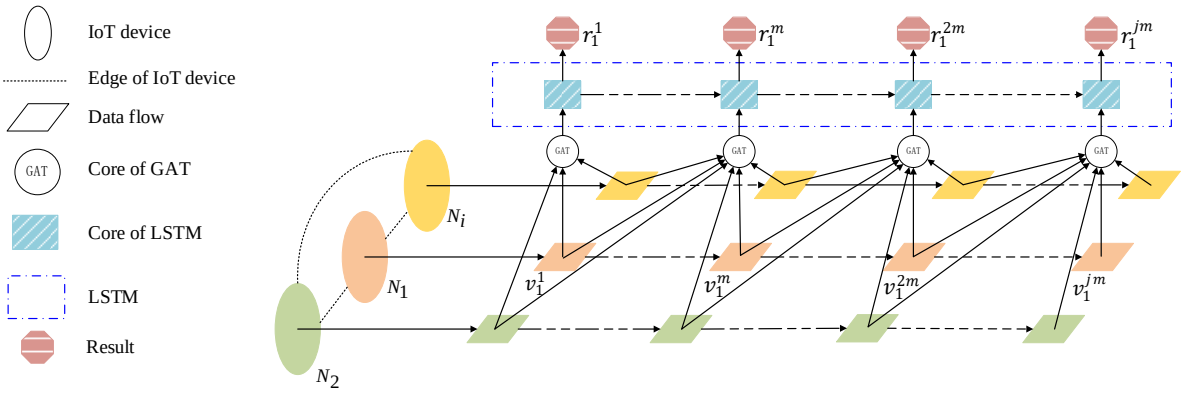

Fig. 1 The model structure of N-STGAT. N_1 represents the data sending node 1, v_1^m represents the m-th data flow of node 1, and r_1^m represents the attack type of the m-th data flow of node 1

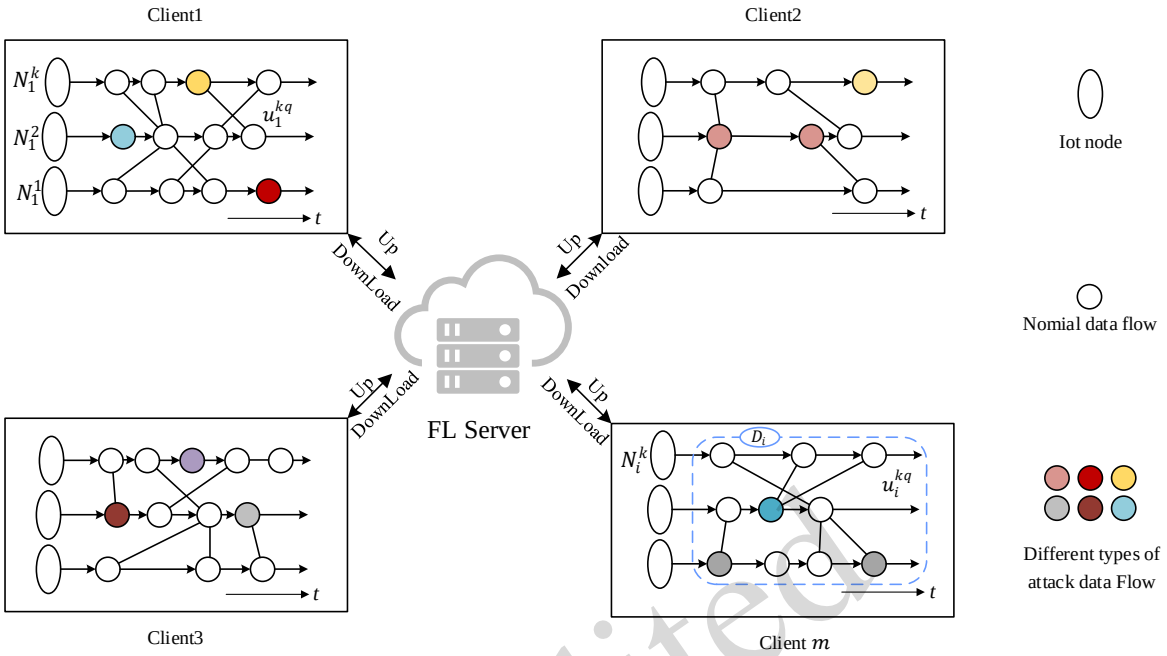

Fig. 3 Structure of the problem definition. Where, N_i^j represents the data sending node, v_i^{jq} represents data flow

structure is depicted in Fig. 1. The main process involves first using GAT to aggregate neighbor information for the current node on the generated graph, producing node embeddings of data flows incorporating temporal information. Then, LSTM is used to perform the final classification based on these node embeddings. This paper utilizes N-STGAT extensively, particularly in the graph construction process. Interested readers are encouraged to refer to the relevant literature for further details.

3.2 Federated learning

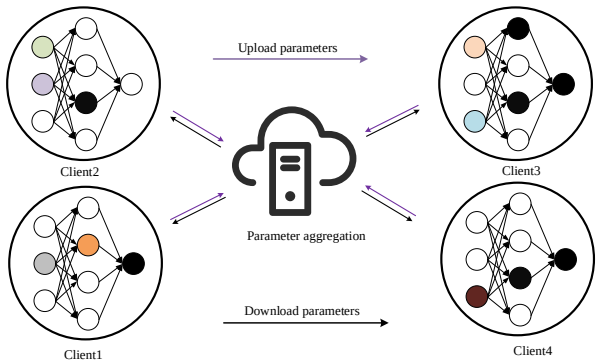

Fig. 2 FL architecture diagram

FL is a decentralized machine learning method designed to train models across multiple devices or servers without transferring the original data to a central location. This approach helps address data privacy and security concerns while reducing the need for data transmission. In recent years, FL has been predominantly used in fields such as the IoT (Nguyen D C et al., 2021), healthcare (Nguyen D C et al., 2022), finance (Chatterjee P et al., 2023), mobile devices, and edge computing (Lim W Y B et al., 2020). FL allows clients to collaboratively train a shared model while keeping personal data on their devices; its typical architecture is illustrated in Fig. 2. FL also has various categories, such as vertical FL and horizontal FL; interested readers can refer to the paper by Q Yang et al. (2019).

4 Method description

4.1 Problem definition

To protect data privacy while training a high-accuracy model that covers a wide range of attack types, it is necessary to abstractly define the real-world environment. The result is shown in Fig. 3. The problem studied in this paper can be defined from the structure shown in Fig. 3. Let there be a set of $C = \{C_1, C_2, C_3, \dots, C_m\}$ clients, each with n IoT nodes, where each client has a private dataset $d_i \in D$ related to network intrusion detection, which is not shared among the clients. Within each client, all network flow datasets are sorted according to nodes and time series, defined as $u_i^{jq}(i \in [1, m], q \in [1, \infty), j \in [1, n])$. The goal of this paper is to use deep learning methods to train a network intrusion detection model to

Unedited

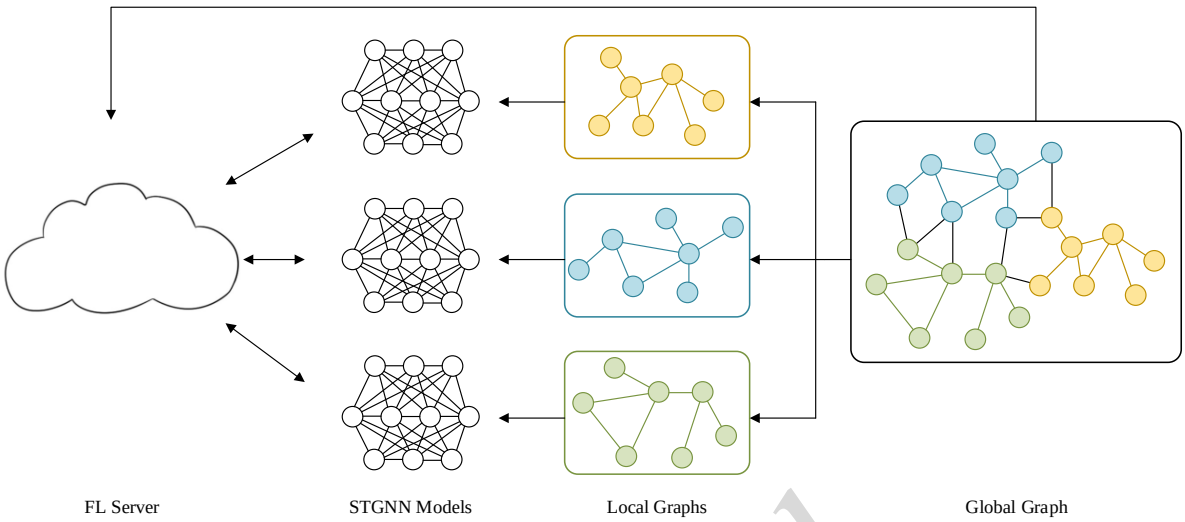

Fig. 4 The general structure of FL on distributed graph data. The FL server has a global graph, while each client has a local graph, which is a subgraph of the global graph

identify the attack types r_i^{jq} in the data flow u_i^{jq}. To protect the data privacy of each client, this paper employs an FGL approach to achieve this objective.

4.2 Federated graph construction

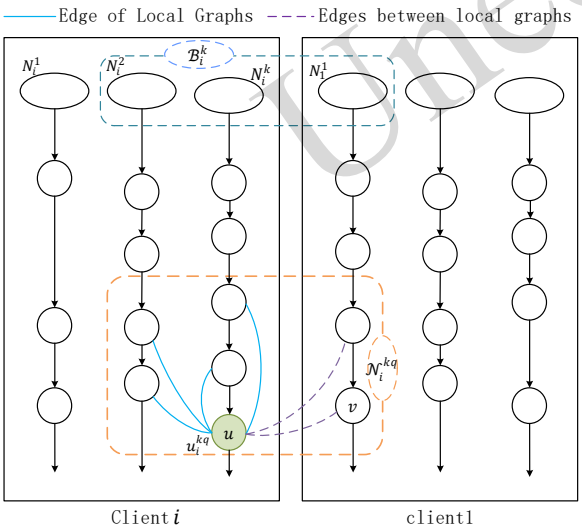

Fig. 5 Results of the graph construction. $\mathcal{B}_i^k$ represents the neighbors of the k-th node N_i^k in client i, $\mathcal{N}_i^{kq}$ represents the neighbors of the k-th data flow u_i^{kq} of node N_i^k, and u and v represent data flows

Using FSTGNN to train classification models, the structure is shown in Fig. 4 (Lei R et al., 2023), from which it can be seen that the first step involves generating a local graph G_i^l on client C_i and creating edges e between the G_i^l (represented by black edges in the global graph of Fig. 4). This forms the global graph G used for training, as illustrated in Fig. 5. Then, each client trains its own data using STGNN and aggregates the parameters from each client on the FL server. These two processes present two challenges: first, there is a risk of privacy leakage when generating edges e between local graphs; second, during STGNN training, the need to aggregate neighbor node information also poses privacy leakage concerns. The following section will discuss how to address the first issue.

Fig. 5 shows that constructing edges within a local graph is relatively straightforward and can be easily achieved using the methods provided by N-STGAT. However, when generating edges e between local graphs, using the methods provided by N-STGAT may lead to privacy leakage issues. This is because N-STGAT employs cosine similarity to compute $\mathcal{B}_i$; directly using cosine similarity calculations requires the other party to know the current node's information, which is not permissible in FL. Therefore, it is necessary to restructure the cosine similarity computation to ensure that the cosine value can still be calculated while preserving privacy.

The cosine similarity formula can be decomposed into three parts—A, B, and C—as shown in Equation (1). From the formula, it can be seen that as long as A, B, and C can be computed separately, the final cosine similarity can be determined.

$$\cos(\theta) = \frac{\sum_{i=1}^{n}(x_i \times y_i)}{\sqrt{\sum_{i=1}^{n}(x_i)^2} \times \sqrt{\sum_{i=1}^{n}(y_i)^2}} \tag{1}$$

Unedited

$$A = \sqrt{\sum_{i=1}^{n}(x_i)^2}$$

$$B = \sqrt{\sum_{i=1}^{n}(y_i)^2}$$

$$C = \sum_{i=1}^{n}(x_i \times y_i)$$

Assuming that two data flows belonging to different clients are X and Y, with feature values $\boldsymbol{X} = (x_1, x_2, \dots, x_n)$ and $\boldsymbol{Y} = (y_1, y_2, \dots, y_n)$, respectively, the process of calculating cosine similarity will be divided into the following six steps:

Step 1: Data flow X calculates A using formula $\sqrt{\sum_{i=1}^{n}(x_i)^2}$, while data flow Y calculates B using formula $\sqrt{\sum_{i=1}^{n}(y_i)^2}$. Then, data flow Y sends B to data flow X.

Step 2: Data flow X generates a random number $P = (p_1, p_2, \dots, p_n)$ and sends it to the FL server.

Step 3: Data flow X generates an intermediate state $XP = (xp_1, xp_2, \dots, xp_n)$ using formula $xp_i = x_i \times p_i$ and sends XP to data flow Y.

Step 4: Data flow Y generates an intermediate state $XPY = (xpy_1, xpy_2, \dots, xpy_n)$ using formula $xpy_i = xp_i \times y_i$ and sends XPY to the FL server.

Step 5: The FL server obtains C using formula $\sum_{i=1}^{n}(xpy_i \,/\, p_i)$ and sends C to data flow X.

Step 6: Data flow X calculates the cosine value $\cos(XY) = \frac{C}{A \times B}$.

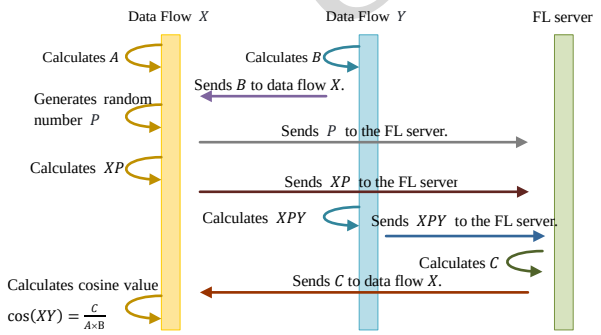

Fig. 6: The process steps for generating edges between subgraphs

To clarify the steps mentioned above, a process flow chart is provided in Fig. 6. Furthermore, to emphasize the details of privacy protection, all fields must be normalized before calculating cosine similarity using the afore-mentioned steps. This prevents extreme values from one party from allowing the other party to infer the range of its field values. The specific normalization method involves finding the maximum value X^{max} and minimum value X^{min} of the feature values for all data flows within the client of data flow X, as well as the maximum value Y^{max} and minimum value Y^{min} of the feature values for all data flows within the client of data flow Y. Their specific values are shown in Equation (2).

$$\begin{aligned} X^{max} &= (x_1^{max}, x_2^{max}, \dots, x_n^{max}) \\ X^{min} &= (x_1^{min}, x_1^{min}, \dots, x_n^{min}) \\ Y^{max} &= (y_1^{max}, y_2^{max}, \dots, y_n^{max}) \\ Y^{min} &= (y_1^{min}, y_1^{min}, \dots, y_n^{min}) \\ XY_i^{max} &= max(x_i^{max}, y_i^{max}) \quad i \in [1, n] \\ XY_i^{mix} &= min(x_i^{min}, y_i^{min}) \quad i \in [1, n] \end{aligned} \tag{2}$$

Here, x_i^{max} and x_i^{min} represent the maximum and minimum values of the iii-th dimension feature for all nodes in the client to which data stream X belongs, respectively. Similarly, y_i^{max} and y_i^{min} represent the maximum and minimum values of the iii-th dimension feature for all nodes in the client to which data stream Y belongs. With these parameters for normalization, we can easily normalize the nodes of each client without affecting the calculation of cosine similarity while also effectively preserving privacy.

In addition, to ensure the privacy of the cosine similarity calculation process, this paper introduces differential privacy by adding controlled noise to the random value P. Specifically, Laplace noise is added before sharing the intermediate states XP and XPY with the FL server. This ensures that even if an attacker gains access to these intermediate values, they cannot accurately infer the original data.

4.3 Federated spatiotemporal graph convolutional network framework (FedSTGCN) model

The global graph and local subgraphs are constructed using the above methods. The next step is model training. The main process is as follows: First, N-STGAT is used to perform local model training on the generated graphs, and then the trained models are sent to the federated server for aggregation. However, since the first step in N-STGAT model training is to use GAT to aggregate the neighbor information of the current data flow, it can be observed from Fig. 6 that the neighbors of the current data flow may include data flows from other clients. Therefore, directly using GAT to aggregate neighbor information may lead to privacy leakage.

FedCog primarily adopts an improved GCN (Kipf T N et al., 2016) to aggregate the neighbor information

Unedited

of nodes. Therefore, in this paper, N-STGAT is improved by replacing its GNN core (as shown in Fig. 1) with the improved GCN from FedCog. This training approach is referred to as FedSTGCN in this paper. The following sections will provide a more detailed explanation of the FedSTGCN method.

Since its core improvement is based on GCN, the GCN content is explained in detail here. GCN primarily converts a graph into an adjacency matrix and then uses convolution operations to aggregate node information. The core formula is as follows: suppose the graph is represented by $\mathcal{G} = (V, E, \boldsymbol{X})$, with V denoting all the nodes in the graph, E representing all the edges, and $\boldsymbol{X}$ denoting the features of all the nodes. In formula (3), u represents the current node being calculated, l represents the GNN layer number, $\mathcal{N}_u$ denotes all the neighboring nodes of node u, $\boldsymbol{h_u}$ represents the node embedding of u, $\boldsymbol{W}$ is the learnable weight matrix, σ represents the activation function, and $c_{uv} = 1/\sqrt{(1+d_u)(1+d_v)}$ represents the degree calculation between nodes u and v.

$$\boldsymbol{h}_u^{(l+1)} = \sigma\left(\sum_{v\in\mathcal{N}_u} c_{uv}\boldsymbol{h}_v^{(l)}\boldsymbol{W}^l\right), u \in V \quad (3)$$

In FedCog, the primary method for addressing privacy concerns in GNNs is graph decoupling, where the graph $\mathcal{G}$ is decoupled into an internal graph $\mathcal{G}^I$ and an external graph $\mathcal{G}^B$. As shown in Fig. 6, the nodes connected by blue lines form the internal graph $\mathcal{G}^I$, while those connected by purple dashed lines form the external graph $\mathcal{G}^B$. The aggregation of nodes is then divided into two processes: the aggregation of nodes in the internal graph $\mathcal{G}^I$ and the aggregation of nodes in the external graph $\mathcal{G}^B$. The external graph aggregation method is shown in formula (4), and the internal graph node aggregation method is shown in formula (5). After completing the aggregation of both external and internal graphs, formula (6) is used to merge the two results, thus computing the node embedding of u at layer $l+1$. The detailed proof is provided in FedCog, and interested readers can refer to the original paper for more information.

$$\widehat{\boldsymbol{h}}_u^{(l+1)} = \sigma\left(\sum_{v\in\mathcal{N}_u^B} \gamma_{uv}\widehat{\boldsymbol{h}}_v^{\left(l+\frac{1}{2}\right)}\right) \quad (4)$$

$$\widehat{\boldsymbol{h}}_{u_j}^{(l+\frac{1}{2})} = \sigma\left(\sum_{v\in\mathcal{N}_{u_j}^I} \beta_{u_j v}\widehat{\boldsymbol{h}}_v^{(l)}\boldsymbol{W}^{(l)}\right) \quad (5)$$

$$\widehat{\boldsymbol{h}}_u^{(l+1)} = \sigma\left(\gamma_{uu}\widehat{\boldsymbol{h}}_u^{\left(l+\frac{1}{2}\right)} + \sum_{j\in B_u}\gamma_{uu_j}\widehat{\boldsymbol{h}}_{u_j}^{\left(l+\frac{1}{2}\right)}\right) \quad (6)$$

$$= \sigma\left(\sum_{j\in B_u}\sum_{v\in\mathcal{N}_{u_j}^I} c_{uv}\widehat{\boldsymbol{h}}_v^{(l)}\boldsymbol{W}^l\right)$$

$\beta_{uv} = 1/\sqrt{(1+d_v)}$, $\gamma_{uv} = 1/\sqrt{(1+d_u)}$, $\mathcal{N}_u^I$ represents the neighbors of node u in the internal graph $\mathcal{G}^I$, and $\mathcal{N}_u^B$ represents the neighbors of node u in the external graph $\mathcal{G}^B$.

The FedSTGCN model training process has been clearly explained through the above methods. The model's pseudocode is as follows:

Algorithm 1: Pseudocode for FedSTGCN

Input: Local graph $\{\mathcal{G}_i(V, E, \boldsymbol{X})\}_{i=1}^n$ for n FL clients.
L-layer GCN model $f(\mathcal{G};\ \theta)$, $\theta = \{\boldsymbol{W}^{(0)}, \dots, \boldsymbol{W}^{(L-1)}\}$
$LSTMcore(u, \boldsymbol{W_{lstm}})$

1 ➢ Graph Decoupling
2 **for** $i = 1$ to n **do** ◇Execute on client i
3 $\mathcal{G}_i^I, \mathcal{G}_i^B$ ←GraphDecoupling($\mathcal{G}_i$)
4 ➢ Internal and Border Converges
5 **for** $l = 0$ to $L-1$ **do**
6 **for** $i = 1$ to n **do** ◇Execute on client i
7 $\widehat{\boldsymbol{h}}_u^{(l+\frac{1}{2})} = \sum_{v\in\mathcal{N}_u^I}\beta_{uv}\widehat{\boldsymbol{h}}_v^{(l)}\boldsymbol{W}^{(l)}$
8 Receive $\left\{\widehat{\boldsymbol{h}}_{uv}^{(l+\frac{1}{2})}, v \in V_j\right\}$ from client j
9 $\widehat{\boldsymbol{h}}_u^{(l+1)} \leftarrow \sigma\left(\sum_{j\in B_u}\sum_{v\in\mathcal{N}_{u_j}^I} c_{uv}\widehat{\boldsymbol{h}}_v^{(l)}\boldsymbol{W}^l\right)$
10 $r_u \leftarrow LSTMcore(\widehat{\boldsymbol{h}}_u^{(l+1)}, \boldsymbol{W_{lstm}})$
11 ➢ Federated Training
12 **for** $t = 1$ to k **do**
13 Sample a set of parties $\boldsymbol{C_t}$
14 **for** $i \in \boldsymbol{C_t}$ **do** ◇Execute on client i
15 $\theta_i \leftarrow \theta$
16 $\mathcal{G}_i$ ←GradientDescent(θ_i, r_u)
17 $\bar{g} \leftarrow \sum_{i\in\boldsymbol{C_t}} p_i^{(t)} g_i$ ◇Execute on FL Server
18 $\theta \leftarrow \theta - \eta\bar{g}$

In **Algorithm 1**, lines 2–3 describe the FedCog graph decoupling process. Each client generates an internal graph $\mathcal{G}^I$ and an external graph $\mathcal{G}^B$. Lines 5–9 explain how GCN performs node aggregation between the internal and external graphs, a key step in addressing the privacy issue. Since STGNNs with temporal correlations are required to train the nodes, an LSTM is introduced in line 10 to process the node embeddings generated by the GCN. Lines 12–18 detail the FL training method, where FedAvg (McMahan B et al., 2017) is used to aggregate the parameters uploaded by each client.

5 Experimental evaluation

5.1 Experimental setup

1) Experimental environment: To evaluate the

Unedited

proposed method, a lightweight Python framework, PyTorch, was used to build the FL framework on an Ubuntu 22.04.4 LTS system. The hardware used in the experiment include a dual Intel Xeon E5-2696 v4 CPU with an NVIDIA Tesla A10 GPU (24GB), 256GB of RAM, and 21TB of storage.

2) Datasets: The two datasets used in the experiments are those mentioned in N-STGAT, and are derived from the original datasets NF-BoT-IoT-v2 and NF-ToN-IoT-v2 (Sarhan M et al., 2022), BoT-IoT (Koroniotis N et al., 2022), and ToN-IoT (Moustafa N et al., 2022). These datasets contain status information about nodes during data flow transmission and include 14 NI fields and 43 FI fields. Sample instances of the dataset can be found in the original N-STGAT paper. They are widely applied in the field of network intrusion detection, as referenced in the literature (Z Zeng et al., 2022; Nuaimi M et al., 2023; A N et al., 2023). Due to the large size of the experimental data, 20% of each dataset was utilized for the experiments. The experimental dataset was then divided into two parts, with 70% used for training and 30% for testing. In addition, this paper adopts the dataset preprocessing process from the original N-STGAT paper.

3) Baseline studies: This paper compares the proposed method with the FL-based network intrusion detection methods introduced by Li B et al. (2020), Huang X et al. (2022), and Chen Z et al. (2020) These methods claim to achieve satisfactory results in their respective studies.

4) Experimental setting: The datasets, were normalized, and fields that were not useful for training the model, such as IPV4_SRC_ADDR and L4_SRC_PORT, were removed. Invalid data were set to zero. The number of clients was configured to $m = 20$, and the number of rounds for aggregating FL parameters using the FedAvg method was set to $R = 100$.

5) Metrics for measuring performance: To evaluate the performance of the proposed method, the performance metrics shown in Table 1 were used. These metrics are commonly employed to assess deep learning models. A substantial amount of literature (Khan N W et al., 2023; Aljuhani A et al., 2022; Ma Z et al., 2023; Altaf T et al., 2023) uses these metrics for performance comparison, including the baseline methods compared in this paper.

Table 1 Metrics for measuring performance

Metric	Definition
Recall	$\frac{TP}{TP+FN}$
Precision	$\frac{TP}{TP+FP}$
F1-score	$2\times\frac{Recall\times Precision}{Recall+Precision}$
Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$

5.2 Result

This paper will present the experimental results of various methods in both binary and multiclass classifications. The training dataset will be evenly divided into m parts, denoted as $TRD_i, i\in[1,m]$, and m represents the number of clients. The testing dataset will also be evenly divided into m parts, denoted as $TED_i, i\in[1,m]$. All testing datasets will be represented as GD.

5.2.1 Local testing dataset

In this section, TRD_i and TED_i are provided to client i to simulate a scenario in which each client has its own training and testing datasets. In this experiment, the number of clients is set to 5, i.e., $m = 5$. The experimental results for both binary and multi-class classifications are presented below.

Table 2 Results of each method on binary classification of NF-BoT-IoT-v2

R	DeePFed				EEFED				FedAGRU				FedSTGCN			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
10	0.8758	0.9048	0.8547	0.8941	0.9147	0.9214	0.9014	0.8941	0.9084	0.8749	0.8874	0.9064	0.9294	0.9014	0.9218	0.9314
20	0.9020	0.9177	0.8925	0.9020	0.9095	0.9334	0.9065	0.9026	0.9219	0.9213	0.8603	0.9062	0.9268	0.9232	0.9381	0.9582
30	0.9324	0.9229	0.9184	0.9021	0.9301	0.9392	0.9084	0.9178	0.9226	0.9329	0.8566	0.9072	0.9305	0.9189	0.9409	0.9699
40	0.9401	0.9220	0.9259	0.9126	0.9299	0.9414	0.9125	0.9193	0.9219	0.9344	0.9134	0.9021	0.9464	0.8890	0.9477	0.9732
50	0.9400	0.9327	0.9261	0.9216	0.9306	0.9416	0.9124	0.9205	0.9256	0.9344	0.9212	0.9121	0.9619	0.9422	0.9494	0.9732
60	0.9412	0.9328	0.9261	0.9394	0.9316	0.9422	0.9115	0.9202	0.9267	0.9344	0.9231	0.9118	0.9646	0.9584	0.9518	0.9726
70	0.9415	0.9328	0.9262	0.9387	0.9316	0.9422	0.9093	0.9193	0.9265	0.9345	0.9219	0.9120	0.9650	0.9467	0.9522	0.9725
80	0.9415	0.9345	0.9262	0.9482	0.9316	0.9424	0.9126	0.9181	0.9269	0.9345	0.9222	0.9127	0.9681	0.9616	0.9533	0.9731
90	0.9415	0.9345	0.9262	0.9491	0.9316	0.9424	0.9128	0.9201	0.9268	0.9345	0.9232	0.9131	0.9682	0.9623	0.9528	0.9732
100	0.9417	0.9347	0.9264	0.9496	0.9318	0.9426	0.9147	0.9214	0.9274	0.9347	0.9245	0.9142	**0.9694**	**0.9712**	**0.9547**	**0.9749**

Unedited

Table 3 Results of each method on binary classification of NF-ToN-IoT-v2

R	DeePFed				EEFED				FedAGRU				FedSTGCN			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
10	0.7418	0.8417	0.8048	0.7914	0.6708	0.7297	0.7419	0.6917	0.8715	0.8268	0.8493	0.8269	0.7448	0.7548	0.7818	0.7158
20	0.7909	0.9066	0.8280	0.8784	0.7161	0.8411	0.9242	0.8458	0.8835	0.8716	0.8287	0.8438	0.8859	0.8801	0.8304	0.8136
30	0.8020	0.9122	0.8790	0.9076	0.8584	0.9096	0.9326	0.8484	0.9020	0.8726	0.8685	0.8579	0.9563	0.8982	0.9470	0.9572
40	0.8816	0.9142	0.8810	0.9116	0.9144	0.9117	0.9317	0.8082	0.9017	0.8763	0.8747	0.8629	0.9588	0.9351	0.9390	0.9585
50	0.8990	0.9142	0.8613	0.9146	0.9223	0.9077	0.9316	0.8947	0.9020	0.8768	0.8783	0.8629	0.9699	0.9408	0.9243	0.9543
60	0.9048	0.9144	0.8773	0.9153	0.9236	0.9052	0.9335	0.9147	0.9021	0.8781	0.8784	0.8651	0.9666	0.9461	0.9416	0.9653
70	0.9244	0.9143	0.8972	0.9153	0.9293	0.9158	0.9339	0.9194	0.9045	0.8779	0.8829	0.8639	0.9602	0.9545	0.9487	0.9657
80	0.9250	0.9145	0.9017	0.9153	0.9299	0.9171	0.9339	0.9187	0.9045	0.8779	0.8824	0.8658	0.9707	0.9554	0.9717	0.9652
90	0.9247	0.9145	0.9020	0.9155	0.9304	0.9171	0.9341	0.9211	0.9045	0.8781	0.8842	0.8666	0.9703	0.9554	0.9734	0.9659
100	0.9267	0.9145	0.9039	0.9155	0.9311	0.9175	0.9343	0.9217	0.9047	0.8785	0.8847	0.8671	**0.9748**	**0.9556**	**0.9738**	**0.9671**

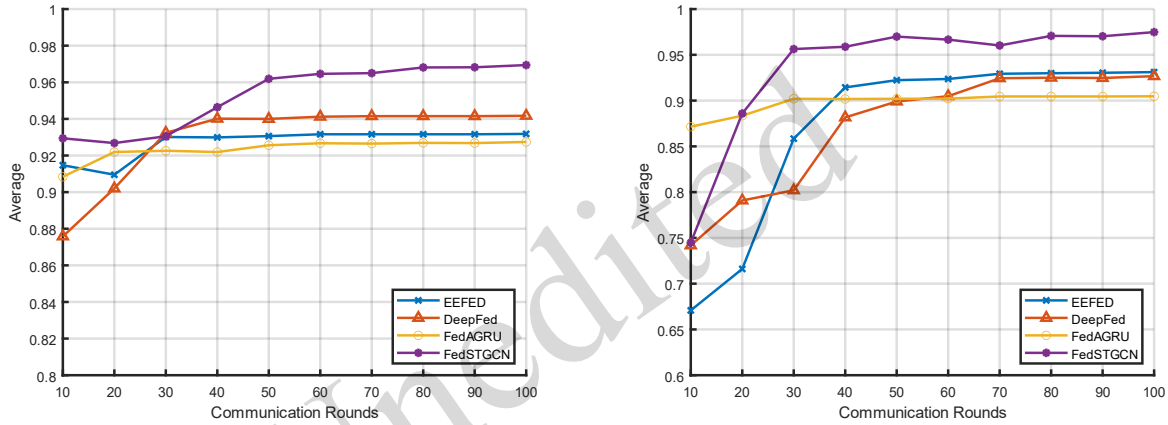

Fig. 7 The changes in binary classification Accuracy for each method across both datasets as the number of rounds increases. Left: Dataset NF-BoT-IoT-v2. Right: Dataset NF-ToN-IoT-v2

Unedited

1) Binary classification results

We conducted 100 rounds of training and communication on two datasets, calculating accuracy, precision, recall, and F1-score every 10 rounds. The results are shown in Tables 2 and 3. The results show that after 100 rounds, the proposed method achieved accuracies of 0.9694 and 0.9748, significantly outperforming other methods. Specifically, accuracy improved by 2.77%–7.01%, precision by 2.86%–7.71%, recall by 2.83%–8.91%, and F-score by 2.53%–10.00%. In addition, accuracy, precision, recall, and F-score all demonstrated significant improvements compared to other methods, proving the strong advantages of this approach. Fig. 7 shows that after 40 rounds, the accuracy stabilized above 95%, further validating its effectiveness in real-world applications. These results indicate that the FL-based intrusion detection method achieves over a 10% performance improvement in binary classification tasks, demonstrating its potential and superiority in network intrusion detection.

Table 4 Results of each method on Multiclass classification on NF-BoT-IoT-v2

R	DeePFed		EEFED		FedAGRU		FedSTGCN	
	Weighted Recall	Weighted F1-score	Weighted Recall	Weighted F1-score	Weighted Recall	Weighted F1-score	Weighted Recall	Weighted F1-score
10	0.4859	0.4348	0.5716	0.6257	0.4612	0.4586	0.6812	0.7214
20	0.5476	0.6240	0.6462	0.7490	0.5967	0.4748	0.8462	0.9205
30	0.5853	0.6583	0.6650	0.7703	0.6294	0.6428	0.8661	0.9205
40	0.5061	0.6825	0.7095	0.7788	0.6838	0.6471	0.8812	0.9199
50	0.6002	0.7317	0.7801	0.7577	0.6893	0.6305	0.8831	0.9201
60	0.7192	0.7384	0.8003	0.8220	0.6802	0.6828	0.9000	0.9210
70	0.7153	0.7422	0.8001	0.8221	0.6990	0.6882	0.8991	0.9210
80	0.7247	0.7419	0.8012	0.8255	0.7021	0.6883	0.9011	0.9208
90	0.7248	0.7448	0.8012	0.8249	0.7024	0.6847	0.9013	0.9210
100	0.7249	0.7458	0.8014	0.8268	0.7025	0.6921	**0.9015**	**0.9214**

Table 5 Results of each method on Multiclass classification on NF-ToN-IoT-v2

R	DeePFed		EEFED		FedAGRU		FedSTGCN	
	Weighted Recall	Weighted F1-score	Weighted Recall	Weighted F1-score	Weighted Recall	Weighted F1-score	Weighted Recall	Weighted F1-score
10	0.6819	0.6175	0.7513	0.7025	0.6217	0.5914	0.6817	0.7249
20	0.7678	0.6872	0.7553	0.7760	0.619	0.6734	0.7056	0.5700
30	0.7790	0.7324	0.7605	0.7787	0.6390	0.7647	0.7834	0.6748
40	0.7884	0.7361	0.8144	0.8009	0.663	0.7792	0.8399	0.9101
50	0.7914	0.7420	0.8214	0.8010	0.6993	0.7796	0.8767	0.9125
60	0.7917	0.7419	0.8214	0.8011	0.7028	0.7800	0.8644	0.9147
70	0.7919	0.7479	0.8214	0.8011	0.7226	0.7806	0.9081	0.9189
80	0.7919	0.7494	0.8215	0.8011	0.7498	0.7809	0.9106	0.9235
90	0.7924	0.7494	0.8215	0.8011	0.7503	0.7809	0.9079	0.9235
100	0.7926	0.7517	0.8217	0.8014	0.7514	0.7812	**0.9159**	**0.9237**

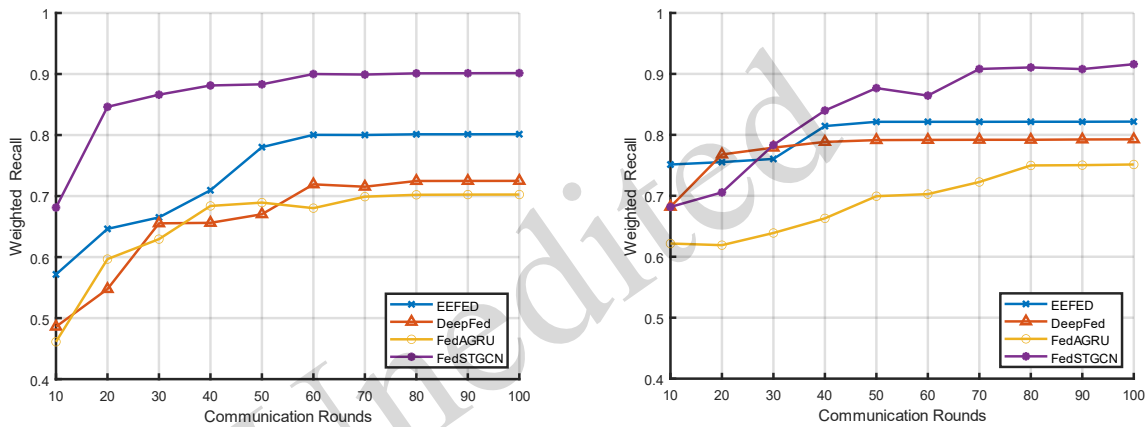

Fig. 8 The changes in Multiclass classification Accuracy for each method across both datasets as the number of rounds increases. Left: Dataset NF-BoT-IoT-v2. Right: Dataset NF-ToN-IoT-v2

Unedited

2) Multiclass classification results

We conducted 100 rounds of training and communication on two datasets, calculating weighted recall and weighted F-score every 10 rounds. The results are presented in Tables 4 and 5 and show that after 100 rounds, the proposed method achieved weighted recalls of 0.9015 and 0.9159, and weighted F-scores of 0.9214 and 0.9237, significantly outperforming other methods. Specifically, weighted recall improved by 9.42%–19.90%, and weighted F-score by 9.46%–22.93%. Fig. 8 illustrates that the proposed method surpasses the others after just 30 rounds, with weighted recall exceeding 90% on both datasets. These results indicate a significant performance improvement in multiclass classification tasks.

Additionally, the recognition results for each category from both datasets were collected, as shown in

Table 6 Recall results per class on the two datasets.

Dataset	Method	Per class Recall									
		Benign	Reconnaissance	DDos	Dos	Theft					
NF-BoT-IoT-v2	DeePFed	0.7914	0.8136	0.7049	0.6901	0.8492					
	EEFED	0.8419	0.8103	0.7915	0.7496	0.8617					
	FedAGRU	0.8016	0.7219	0.6483	0.7814	0.8049					
	FedSTGCN	**0.9516**	**0.9743**	**0.8671**	**0.8819**	**0.9416**					
		Benign	Reconnaissance	DDos	Dos	Backdoor	Injection	MITM	Password	Scanning	XSS
NF-ToN-IoT-v2	DeePFed	0.8216	0.7946	0.7297	0.7525	0.8216	0.6785	0.7509	0.9016	0.8116	0.8052
	EEFED	0.8318	0.8613	0.7619	0.7945	0.8016	0.7916	0.7148	0.8043	0.7319	0.8419
	FedAGRU	0.7913	0.7419	0.7743	0.7581	0.6913	0.7459	0.8216	0.8015	0.6649	0.8216
	FedSTGCN	**0.9314**	**0.9813**	**0.8719**	**0.8934**	**0.9916**	**0.9763**	**0.9417**	**0.9518**	**0.9743**	**0.9997**

Table 6. It can be observed that the proposed method achieves a high recognition rates in most categories, with the recognition results for all categories surpassing those of the other methods, directly demonstrating the effectiveness of the proposed method. Notably, while some categories achieve an accuracy as high as 99.97%, the accuracy for DDoS attacks only reaches 86.71%. One primary reason for this discrepancy is the high similarity between certain types of attacks, such as DoS and DDoS. ©This paper argues that better distinguishing between these similar attacks is crucial for im-proving recognition accuracy.

Since the results of both binary and multiclass classification tests were very good, multiple tests were conducted to verify the proposed method's stability. The confidence intervals for the method in binary and multiclass classification were statistically analyzed, and the results can be found in the supplementary materials.

5.2.2 Multi-Participant

In the previous section, model training was conducted with five clients; however, the number of clients in FL is variable. For instance, in a smart community within a city, the number of smart communities may increase over time, necessitating that the method retains its advantages with different client counts. Therefore, this section conducts experiments with varying numbers of clients m (where $m = 1,2,\ldots,20$) and records the recognition results for both multiclass and binary classifications, as shown in Fig. 9, Fig. 10 , Table 7, and Table 8. It is evident from the figures that even with changes in the number of clients, as the number of communication rounds increases, the final binary classification accuracy and multiclass weighted recall results consistently exceed those of the other methods. Tt can be observed from Tables 7 and 8 that as the number of clients increases, the classification accuracy gradually decreases and

Unedited

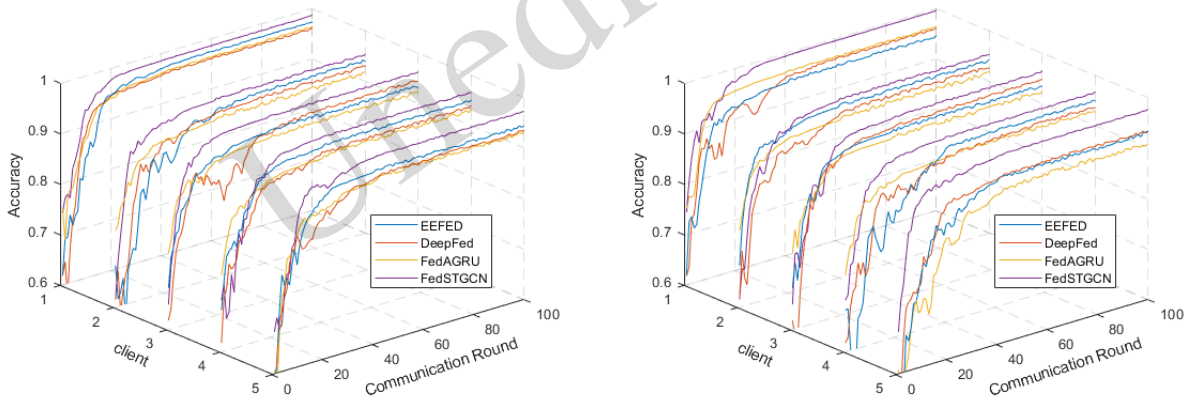

Fig. 9 The changes in binary classification Accuracy for each method across different numbers of clients $m = 1, 2, 3, 4, 5$) as the number of rounds increases. Left: Dataset NF-BoT-IoT-v2. Right: Dataset NF-ToN-IoT-v2

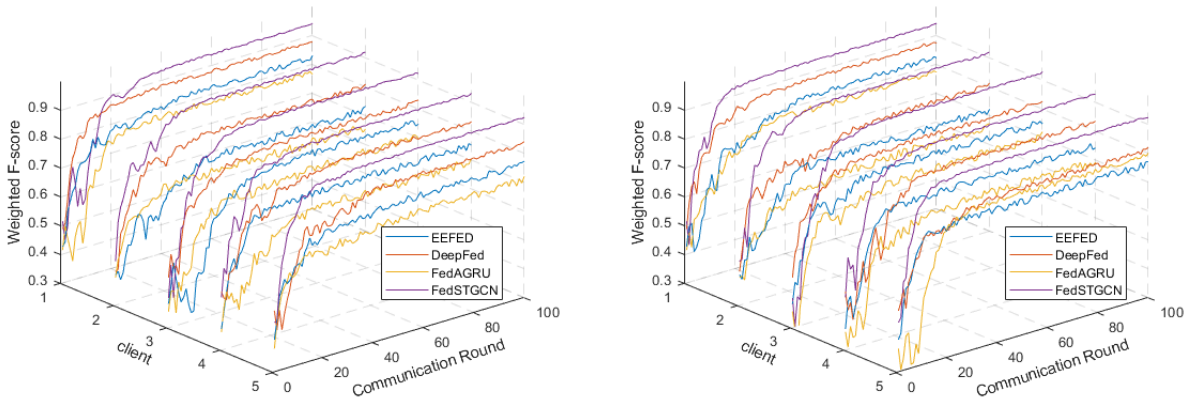

Fig. 10 The changes in Multiclass classification Weighted Recall for each method across different numbers $m = 1, 2, 3, 4, 5$) of clients as the number of rounds increases. Left: Dataset NF-BoT-IoT-v2. Right: Dataset NF-ToN-IoT-v2

Table 7 Training Results (Accuracy) for Binary Classification with Different Numbers of Clients

	NF-BoT-IoT-v2				NF-ToN-IoT-v2			
Clients	DeePFed	EEFED	FedAGRU	FedSTGCN	DeePFed	EEFED	FedAGRU	FedSTGCN
1	0.9512	0.9586	0.9523	0.9788	0.9521	0.9434	0.9535	0.9794
2	0.9501	0.9574	0.9513	0.9781	0.9501	0.9422	0.9493	0.9801
3	0.9461	0.9515	0.9412	0.9726	0.9322	0.9386	0.9301	0.9787
4	0.9432	0.9422	0.9355	0.9702	0.9358	0.9312	0.9152	0.9763
5	0.9317	0.9418	0.9274	0.9694	0.9267	0.9311	0.9047	0.9748
8	0.9345	0.9395	0.9276	0.9699	0.9218	0.9299	0.9085	0.9732
11	0.9351	0.9387	0.9257	0.9664	0.9201	0.9286	0.9043	0.9754
15	0.9318	0.9391	0.9247	0.9685	0.9216	0.9257	0.9068	0.9736
18	0.9322	0.9402	0.9288	0.9697	0.9224	0.9266	0.9062	0.9749
20	0.9341	0.9398	0.9274	0.9675	0.9234	0.9247	0.9054	0.9752

Table 8 Training Results (Weighted Recall) for Multiclass Classification with Different Numbers of Clients

	NF-BoT-IoT-v2				NF-ToN-IoT-v2			
Clients	DeePFed	EEFED	FedAGRU	FedSTGCN	DeePFed	EEFED	FedAGRU	FedSTGCN
1	0.7513	0.8714	0.7217	0.9324	0.8264	0.8512	0.7842	0.9288
2	0.7321	0.8526	0.7189	0.9301	0.8154	0.8394	0.7715	0.9213
3	0.7258	0.8114	0.7113	0.9152	0.8112	0.8315	0.7599	0.9199
4	0.7294	0.8094	0.7088	0.9082	0.8039	0.8254	0.7536	0.9164
5	0.7249	0.8143	0.7025	0.9045	0.7996	0.8217	0.7514	0.9159
8	0.7215	0.8122	0.7064	0.9072	0.8001	0.8263	0.7548	0.9167
11	0.7264	0.8099	0.7059	0.9077	0.7981	0.8178	0.7532	0.9188
15	0.7252	0.8135	0.7034	0.9068	0.8023	0.8247	0.7513	0.9157
18	0.7269	0.8101	0.7052	0.9052	0.8137	0.8211	0.7495	0.9149
20	0.7233	0.8164	0.7039	0.9032	0.7986	0.8197	0.7515	0.9166

Unedited

eventually stabilizes. The initial gradual decrease is due to the distribution of training data across different clients, which reduces the amount of training data available on each client and affects the results. As the variation in the training data across clients becomes smaller, the training performance stabilizes. This indicates that changes in the number of clients have a smaller impact on the results. Additionally, it can be seen that the advantages of the method proposed in this paper over the other methods do not change with the number of clients. It is noteworthy that the recognition results are the best for both binary and multiclass classifications when m=1. This is because, in this scenario, all data from the FL system are trained on a single client, essentially reflecting the ideal model of the entire system.

5.2.3 Robustness

In real-world complex scenarios, the variety of attack types and the speed of sample collection and labeling often do not meet the demands of the situation. Each client's network flow data may not contain the same types of attacks, making it difficult to identify unknown attacks locally. Therefore, the ability to recognize unknown attacks is crucial for such methods. In addition to the aforementioned experiments, this paper also conducted binary classification experiments for recognizing unknown attacks. Each type of attack was labeled with a numerical identifier, and datasets containing different attack types were distributed among the clients m (where $m = 5$). The binary classification training was then performed, and the recognition results were recorded, as illustrated on the right side of Fig. 11. For instance, during the model training with the NF-BoT-IoT-v2 dataset, benign flows were labeled 0, while the four types of attack flows were labeled 1–4. The training dataset for Client 1 only included classifications labeled 0–3, excluding the class labeled 4. Client 2's training dataset included

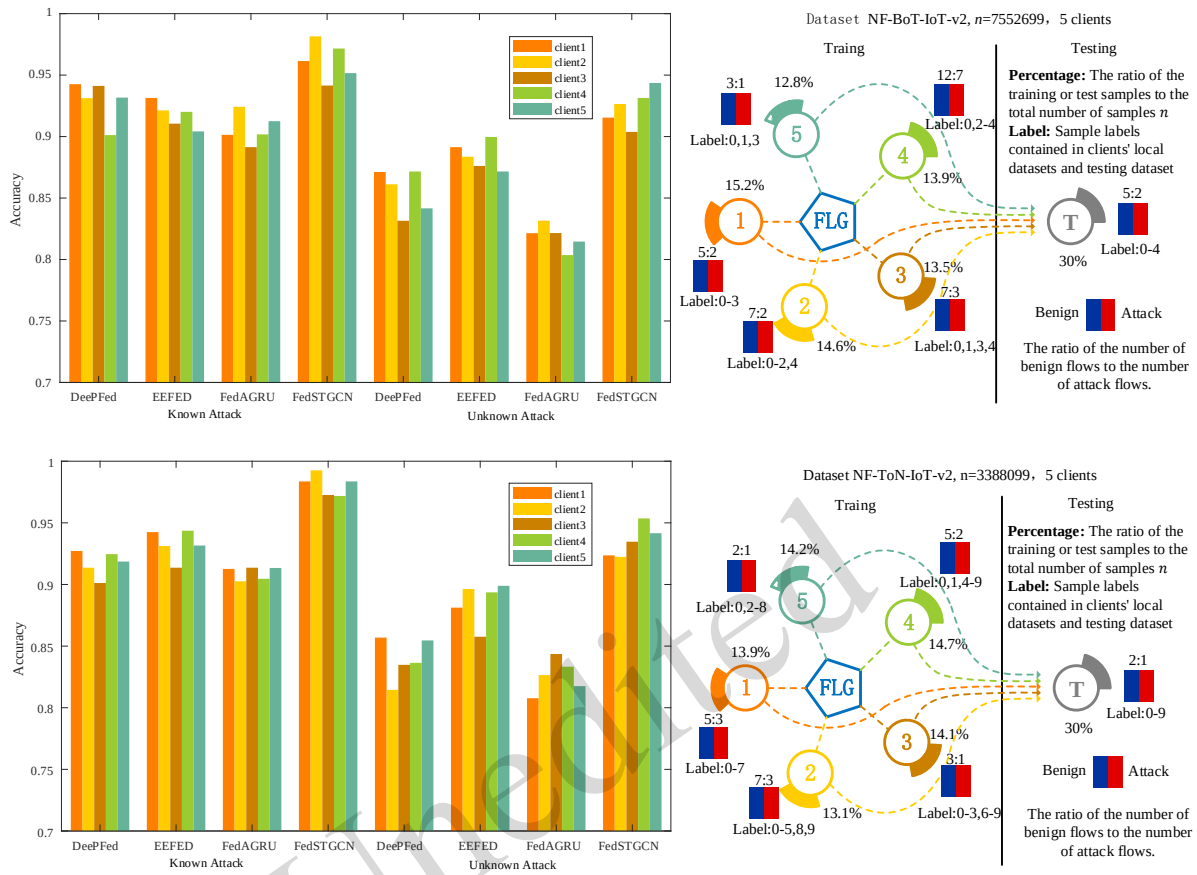

Fig 11 The recognition results for unknown attacks in binary classification for each method across the two datasets. Top: Dataset NF-BoT-IoT-v2. Bottom: Dataset NF-ToN-IoT-v2

labels 0–2 and 4, omitting class 3. Similarly, other clients excluded their respective classifications. After model training, a testing set containing all labels was used to validate each client, which comprised 30% of the entire dataset. The recognition results for known and unknown attacks for each client across all methods are presented on the left side of Fig. 11.

As shown in Fig. 11, on the NF-BoT-IoT-v2 dataset, the proposed method achieves an iden-tification accuracy of 95% for unknown attacks, representing an improvement of over 14% compared to the other methods. On the NF-ToN-IoT-v2 dataset, the proposed method reaches an accuracy of 94% for unknown attack detection, surpassing the other methods by more than 12%. These results strongly indicate that the method proposed in this paper is the most ef-fective for identifying unknown attacks, with each client demonstrating a higher responsive-ness to unknown attacks compared to other methods. This further proves that the proposed method exhibits greater robustness compared to other approaches.

In addition to the above experiments, this paper also analyzes each method's resource consumption during the training process. The results can be found in the supplementary materials.

6. Conclusions

To address the issue of training distributed network intrusion detection models, this paper proposes an advanced model called FedSTGCN, which combines spatiotemporal GNNs with FL. Solutions to the privacy issues arising during the graph construction and node information aggregation processes are also proposed. In graph construction, the cosine similarity function is split into several parts, effectively utilizing its relationships to compute the cosine similarity between two nodes, generating a graph for training the GNN. In the node information aggregation process, this paper combines N-STGAT with FedCog to address privacy protection issues. Finally, classification accuracy experiments, client number variation experiments, unknown attack recognition experiments, and computational cost experiments were conducted on two recent datasets. The experimental results all confirm the effectiveness of the proposed method.

Unedited

It is worth noting that the methods and experiments presented in this paper are based on independent and identically distributed (IID) datasets, where FL generally achieves very good training results. However, in complex network scenarios, the datasets collected from different clients are often non-IID, which poses significant challenges for FGL. Therefore, future research will focus on the challenge of distributed network intrusion detection under non-IID environments.

Contributors

Yalu WANG and Zhijie HAN designed the research. Yalu WANG, Jie LI and Zhijie HAN processed the data. Yalu WANG drafted the paper. Pu CHENG helped organize the paper. Yalu WANG and Roshan KUMAR revised and finalized the paper.

Conflict of interest

Yalu WANG, Jie LI, Zhijie HAN, Pu CHENG and Roshan KUMAR declare that they have no conflict of interest.

References

Aburomman AA, Reaz MBI, 2016. A novel SVM-kNN-PSO ensemble method for intrusion detection system. *Appl Soft Comput*, 38:360-372. https://doi.org/10.1016/j.asoc.2015.10.011

Aljuhani A, Kumar P, Kumar R, et al., 2023. Fog intelligence for secure smart villages: architecture and future challenges. *IEEE Consum Electron Mag*, 12(5):12-21. https://doi.org/10.1109/MCE.2022.3193268

Almogren AS, 2020. Intrusion detection in Edge-of-Things computing. *J Parallel Distrib Comput*, 137:259-265. https://doi.org/10.1016/j.jpdc.2019.12.008

Altaf T, Wang X, Ni W, et al., 2023. A new concatenated Multigraph Neural Network for IoT intrusion detection. *Internet Things*, 22:100818. https://doi.org/10.1016/j.iot.2023.100818

Caville E, Lo WW, Layeghy S, et al., 2022. Anomal-E: a self-supervised network intrusion detection system based on graph neural networks. *Knowl-Based Syst*, 258:110030. https://doi.org/10.1016/j.knosys.2022.110030

Chatterjee P, Das D, Rawat DB, 2024. Federated learning empowered recommendation model for financial consumer services. *IEEE Trans Consum Electron*, 70(1):2508-2516. https://doi.org/10.1109/TCE.2023.3339702

Chen Z, Lv N, Liu PF, et al., 2020. Intrusion detection for wireless edge networks based on federated learning. *IEEE Access*, 8:217463-217472. https://doi.org/10.1109/ACCESS.2020.3041793

Dina AS, Siddique AB, Manivannan D, 2023. A deep learning approach for intrusion detection in Internet of Things using focal loss function. *Int Things*, 22:100699. https://doi.org/10.1016/j.iot.2023.100699

Ghasempour A, 2019. Internet of things in smart grid: architecture, applications, services, key technologies, and challenges. *Inventions*, 4(1):22. https://doi.org/10.3390/inventions4010022

Hu XY, Gao WJ, Cheng G, et al., 2023. Toward early and accurate network intrusion detection using graph embedding. *IEEE Trans Inf Forensics Secur*, 18:5817-5831. https://doi.org/10.1109/TIFS.2023.3318960

Huang XT, Liu J, Lai YX, et al., 2023. EEFED: personalized federated learning of execution & evaluation dual network for CPS intrusion detection. *IEEE Trans Inf Forensics Secur*, 18:41-56. https://doi.org/10.1109/TIFS.2022.3214723

Khan NW, Alshehri MS, Khan MA, et al., 2023. A hybrid deep learning-based intrusion detection system for IoT networks. *Math Biosci Eng*, 20(8):13491-13520. https://doi.org/10.3934/mbe.2023602

Kipf TN, Welling M, 2017. Semi-supervised classification with graph convolutional networks. 5th Int Conf on Learning Representations, p.1-14.

Koroniotis N, Moustafa N, Sitnikova E, et al., 2019. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: bot-IoT dataset. *Future Gener Comput Syst*, 100:779-796. https://doi.org/10.1016/j.future.2019.05.041

Lei RZ, Wang PH, Zhao JZ, et al., 2023. Federated learning over coupled graphs. *IEEE Trans Parallel Distrib Syst*, 34(4):1159-1172. https://doi.org/10.1109/TPDS.2023.3240527

Li BB, Wu YH, Song JR, et al., 2021. DeepFed: federated deep learning for intrusion detection in industrial cyber–physical systems. *IEEE Trans Industr Inform*, 17(8):5615-5624. https://doi.org/10.1109/TII.2020.3023430

Li XM, Zhang D, Zheng Y, et al., 2023. Evolutionary computation-based machine learning for smart city high-dimensional big data analytics. *Appl Soft Comput*, 133:109955. https://doi.org/10.1016/j.asoc.2022.109955

Lim WYB, Luong NC, Hoang DT, et al., 2020. Federated learning in mobile edge networks: a comprehensive survey. *IEEE Commun Surv Tutorials*, 22(3):2031-2063. https://doi.org/10.1109/COMST.2020.2986024

Lo WW, Layeghy S, Sarhan M, et al., 2022. E-GraphSAGE: a graph neural network based intrusion detection system for IoT. IEEE/IFIP Network Operations and Management Symp, p.1-9. https://doi.org/10.1109/NOMS54207.2022.9789878

Ma ZC, Liu L, Meng WZ, et al., 2023. ADCL: toward an adaptive network intrusion detection system using collaborative learning in IoT networks. *IEEE Int Things J*, 10(14):12521-12536. https://doi.org/10.1109/JIOT.2023.3248259

Mansour Bahar AA, Ferrahi KS, Messai ML, et al., 2024. FedHE-Graph: federated learning with hybrid encryption on graph neural networks for advanced persistent threat detection. Proc 19th Int Conf on Availability, Reliability and Security, article 119.

Unedited

https://doi.org/10.1145/3664476.3670466

Mao QH, Lin X, Xu WC, et al., 2025. FeCoGraph: label-aware federated graph contrastive learning for few-shot network intrusion detection. *IEEE Trans Inform Forensics Secur*, 20:2266-2280. https://doi.org/10.1109/TIFS.2025.3541890

McMahan B, Moore E, Ramage D, et al., 2017. Communication-efficient learning of deep networks from decentralized data. Proc 20th Int Conf on Artificial Intelligence and Statistics, p.1273-1282.

Moustafa N, 2021. A new distributed architecture for evaluating AI-based security systems at the edge: network TON_IoT datasets. *Sustain Cities Soc*, 72:102994. https://doi.org/10.1016/j.scs.2021.102994

Nguyen DC, Ding M, Pathirana PN, et al., 2021. Federated learning for internet of things: a comprehensive survey. *IEEE Commun Surv Tutorials*, 23(3):1622-1658. https://doi.org/10.1109/COMST.2021.3075439

Nguyen DC, Pham QV, Pathirana PN, et al., 2022. Federated learning for smart healthcare: a survey. *ACM Comput Surv*, 55(3):60. https://doi.org/10.1145/3501296

Nitish A, Hanumanthappa J, Shiva Prakash SP, et al., 2023. On-device context-aware misuse detection framework for heterogeneous IoT edge. *Appl Intell*, 53(12):14792-14818. https://doi.org/10.1007/s10489-022-04039-5

Nuaimi M, Fourati LC, Hamed BB, 2023. Intelligent approaches toward intrusion detection systems for Industrial Internet of Things: a systematic comprehensive review. *J Netw Comput Appl*, 215:103637. https://doi.org/10.1016/J.JNCA.2023.103637

Sangkatsanee P, Wattanapongsakorn N, Charnsripinyo C, 2011. Practical real-time intrusion detection using machine learning approaches. *Comput Commun*, 34(18):2227-2235. https://doi.org/10.1016/j.comcom.2011.07.001

Sarhan M, Layeghy S, Portmann M, 2022. Towards a standard feature set for network intrusion detection system datasets. *Mobile Netw Appl*, 27(1):357-370. https://doi.org/10.1007/s11036-021-01843-0

Veličković P, Cucurull G, Casanova A, et al., 2018. Graph attention networks. 6th Int Conf on Learning Representations, p.1-12.

Wang YL, Li J, Zhao W, et al., 2023. N-STGAT: spatio-temporal graph neural network based network intrusion detection for near-earth remote sensing. *Remote Sens*, 15(14):3611. https://doi.org/10.3390/rs15143611

Wu JP, Qiu GQ, Wu CM, et al., 2024. Federated learning for network attack detection using attention-based graph neural networks. *Sci Rep*, 14(1):19088. https://doi.org/10.1038/s41598-024-70032-2

Yang Q, Liu Y, Chen TJ, et al., 2019. Federated machine learning: concept and applications. *ACM Trans Intell Syst Technol*, 10(2):12. https://doi.org/10.1145/3298981

Zeng ZR, Peng W, Zeng DT, 2022. Improving the stability of intrusion detection with causal deep learning. *IEEE Trans Netw Serv Manage*, 19(4):4750-4763. https://doi.org/10.1109/TNSM.2022.3193099

Zhou XK, Liang W, Li WM, et al., 2022. Hierarchical adversarial attacks against graph-neural-network-based IoT network intrusion detection system. *IEEE Int Things J*, 9(12):9310-9319. https://doi.org/10.1109/JIOT.2021.3130434

Learning," in IEEE Transactions on Network and Service Management, vol. 19, no. 4, pp. 4750-4763, Dec. 2022. https://doi: 10.1109/TNSM.2022.3193099.

Nuaimi M, Fourati L C, Hamed B B. I ntelligent approaches toward intrusion detection systems for Industrial Internet of Things: A systematic comprehensive review[J]. Journal of Network and Computer Applications, 2023: 103637.

A, N., J, H., P, S.P.S. et al. On-device context-aware misuse detection framework for heterogeneous IoT edge. Appl Intell 53, 14792–14818 (2023). https://doi.org/10.1007/s10489-022-04039-5

Li B, Wu Y, Song J, et al. DeepFed: Federated deep learning for intrusion detection in industrial cyber–physical systems[J]. IEEE Transactions on Industrial Informatics, 2020, 17(8): 5615-5624.

Huang X, Liu J, Lai Y, et al. EEFED: Personalized federated learning of execution&evaluation dual network for CPS intrusion detection[J]. IEEE Transactions on Information Forensics and Security, 2022, 18: 41-56.

Chen Z, Lv N, Liu P, et al. Intrusion detection for wireless edge networks based on federated learning[J]. IEEE Access, 2020, 8: 217463-217472.

Khan N W, Alshehri M S, Khan M A, et al. A hybrid deep learning-based intrusion detection system for IoT networks[J]. Math. Biosci. Eng, 2023, 20(8): 13491-13520. Available Link: https://www.aimspress.com/aimspress-data/mbe/2023/8/PDF/mbe-20-08-602.pdf

Aljuhani A, Kumar P, Kumar R, et al. Fog intelligence for secure smart villages: Architecture and future challenges[J]. IEEE Consumer Electronics Magazine, 2022, 12(5): 12-21. https://doi: 10.1109/MCE.2022.3193268

Ma Z, Liu L, Meng W, et al. ADCL: toward an adaptive network intrusion detection system using collaborative learning in IoT networks[J]. IEEE Internet of Things Journal, 2023, 10(14): 12521-12536. https://doi: 10.1109/JIOT.2023.3248259

Altaf T, Wang X, Ni W, et al. A new concatenated multigraph neural network for iot intrusion detection[J]. Internet of Things, 2023, 22: 100818. doi: 10.1016/j.iot.2023.100818

Unedited