A Hybrid Approach to Network Intrusion Detection Based On Graph Neural Networks and Transformer Architectures

1st Hongrun Zhang
College of Computer Technology and Applications
Qinghai University
QingHai, China
ys220854040277@qhu.edu.cn

2nd Tengfei Cao
College of Computer Technology and Applications
Qinghai University
QingHai, China
caotf@qhu.edu.cn

***Abstract*—In this paper, we propose a model of a Network Intrusion Detection System (NIDS) named E-T-GraphSAGE (ETG), which fuses Graph Neural Network (GNN) and Transformer techniques. With the widespread adoption of the Internet of Things (IoT) and cloud computing, network structures have become complex and vulnerable. The efficacy of traditional intrusion detection systems is limited in the context of novel and unconventional cyber-attacks. This paper proposes a novel approach to address this challenge. GNN is used to capture the complex relationships between network nodes and edges, analyze network traffic graphs, and identify anomalous behaviors. By introducing the Transformer, the model enhances its ability to handle long-range dependencies in network streaming data and to understand network dynamics at a macro level. The GraphSAGE-Transformer (ETG) model is proposed to optimize the edge features through the self-attention mechanism to exploit the potential of network streaming data and improve the accuracy of intrusion detection. The experimental results show that the model outperforms the existing techniques in key performance metrics Tests on several standard datasets (BoT-IoT, NF-BoT-IoT, NF-ToN-IoT) validate the broad applicability and robustness of the ETG model, especially in complex network environments.**

Keywords—GNN, GraphSAGE, Transformer, NIDS

I. INTRODUCTION

With the widespread adoption of the Internet of Things (IoT) and cloud computing, the structure of network systems is becoming more complex, and the types and numbers of devices are increasing dramatically. This environment provides more vulnerabilities and points of entry for cyber attackers, making traditional cyber defense systems face serious challenges[1]. Modern network attacks are varied, including distributed denial-of-service (DDoS) attacks, malware spread, and data breaches but also more subtle and adaptable, frequently targeting multiple layers of the network and various nodes. In addition, with the rapid development of attack techniques, new and unknown zero-day vulnerability attacks frequently appear, and these attacks are able to bypass the signature-based intrusion detection system easily[2]. Therefore, there is a need to develop new detection techniques that not only recognize known attack patterns but also can predict and adapt to unknown threats.

To overcome these limitations, recent research has increasingly focused on leveraging machine learning and deep learning techniques. Among these, Transformer architectures have gained attention for their self-attention mechanism, which effectively captures long-range dependencies in sequential data. Originally developed for natural language processing, Transformers have been successfully adapted for cybersecurity applications, offering the ability to analyze complex interdependencies within network traffic.

Graph neural networks (GNNs), known for their ability to handle graph-structured data, offer significant potential in cybersecurity applications. By capturing the complex relationships between nodes (e.g., IP addresses or devices) and edges (i.e., data transmissions or sessions) in a network, GNNs are able to efficiently map the overall pattern of network behavior. This capability makes GNN particularly suitable for identifying and analyzing complex network intrusions that are difficult to detect through conventional detection means[3]. GNNs can analyze network traffic graphs by representing hosts or servers as nodes and their communications as edges. By learning the normal and abnormal characteristics of these communication patterns, the GNN is able to identify anomalous behavior in the network, such as unauthorized data access or abnormal data traffic. In addition, a key advantage of GNNs is their ability to integrate data from multiple sources and extract deep network characteristics, which is particularly important for detecting advanced persistent threats (APTs) and multi-stage attacks.

GNN not only enhances the detection of known threats, but more importantly, it provides a mechanism to understand and predict new or variant attack behaviors that are difficult to identify with traditional methods. Therefore, the introduction of GNN into network security systems, especially network intrusion detection systems, will greatly enhance the system's ability to defend against complex network threats[4].

This research aims to develop an enhanced Network Intrusion Detection System (NIDS) by integrating Graph Neural Networks (GNNs) with Transformer architectures. The goal is to improve the efficiency and accuracy of detecting complex and previously unknown attack patterns by leveraging the

Transformer's ability to capture long-range dependencies in network traffic. This integration seeks to enhance the model's capability to analyze network flows on both local and global scales, improving overall performance in detecting sophisticated cyber threats.

The proposed study will use a hybrid approach, combining GNNs and Transformers to analyze network traffic. GNNs will be employed to construct graph representations of network entities and interactions, while the Transformer's self-attention mechanism will capture long-range dependencies and global patterns[5]. This integrated model aims to enhance understanding of network dynamics and improve detection and prediction of both known and emerging threats. The model's effectiveness will be evaluated through experiments on benchmark datasets, comparing its performance with existing intrusion detection systems.

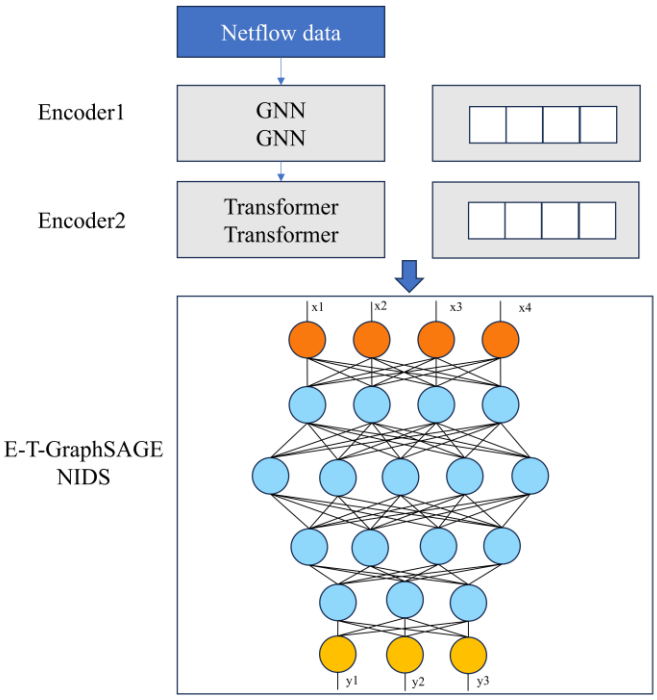

Fig. 1. Network flow data graph structuring.

As shown in Fig. 1, we utilize both GNN and Transformer to encode the raw stream data successively to obtain the desired graph data structure, which is input to the model for training.

(1) The core contribution of this research is the development of a NIDS model that combines GNN and Transformer. The edge features optimized by the self-attention mechanism fully exploit the potential of network streaming data and significantly improve the detection accuracy of network intrusion.

(2) Tests on multiple standard datasets show that our model outperforms existing techniques in key performance metrics such as accuracy, recall, and F1 score.

The rest of the paper's organizational sequel will detail the design and experimental evaluation of the E-T-GraphSAGE (ETG) model. Part II will explore the development of NIDS, as well as research related to GNNs and Transformers. Part III details the model architecture and key technologies. The fourth section shows the experimental results on a variety of cyberattack datasets and compares them with other methods. The concluding section will summarize the research results and discuss future research directions.

II. Related Work

In recent years, various approaches have been proposed to enhance the performance of Intrusion Detection Systems (IDS) Alowaidi et al.[6] proposed a hybrid Intrusion Detection System (IDS) combining Machine Learning (ML) and Deep Learning (DL) techniques, which enhances IDS performance and prediction accuracy while lowering computational costs. However, the model's generalization relies on the diversity and representativeness of the training data. If the training data is biased, it negatively impacts the model's real-world performance. Gupta et al.[7] proposed an anomaly-based NIDS, this approach considers multiple performance metrics, along with training time and resource usage, but remains limited by dataset dependency and average generalization capabilities. Kumar et al.[8] proposed a bi-directional long short-term memory (BiLSTM) based anomaly detection system for Internet of Things (IoT) networks. The BiLSTM model effectively improves the accuracy by preprocessing and feature selection through normalization and gain ratio.

Suárez-Varela et al.[9] introduced the use of GNNs in the modeling control, and management of communication networks, demonstrated their advantages in terms of generalization capabilities and data-driven solutions, and discussed their potential in network modeling control and management. Hnamte et al.[10] proposed an approach using Deep Convolutional Neural Networks (DCNN) and validated its performance with the InSDN dataset. While DCNN achieves high accuracy, it demands significant data and computational resources for training.

Kisanga et al.[11] proposed a new Activity and Event Network (AEN) graph framework that focuses on capturing long-term stealthy threats that are difficult to detect by traditional security tools, and is very promising in detecting long-term threats in cybersecurity. L et al.[12] proposed an end-to-end anomalous edge detection method based on unified graph embedding, which enhances the model's ability to learn task-relevant patterns by combining embedding learning and anomaly detection into the same objective function, and accurately estimates the probability distributions of edges through the local structure of the graph to identify anomalous edges. Superior accuracy and scalability are demonstrated on multiple publicly available datasets.

Sun et al.[13] proposed a framework combining Graph Neural Network (GNN) and Transformer for self-supervised heterogeneous graph representation learning. The Metapath-aware Hop2Token method is designed to efficiently convert neighbors with different hop counts in heterogeneous graphs into Token sequences, reducing the computational complexity in Transformer processing. GTC enhances information fusion, improves learning efficiency, and reduces the demand for computational resources by contrasting learning tasks between graph pattern views and hop count views.

Nguyen et al.[14] proposed a Transformer-based GNN model for learning graph representation. With an unsupervised conduction learning approach, UGformer is able to solve the problem of limited category labels, but for large-scale datasets to construct graphs, UGformer may still need to be optimized to deal with extremely large graph structures, despite the sampling mechanism that UGformer is designed for.

Unlike previous studies, our method focuses on extracting data edge features from network streams and develops E-GraphSAGE models that incorporate transformer modules. Combining local and global features to achieve more accurate feature representations, making full use of the structural and topological information and inherent in network streaming data to achieve better feature representations and network intrusion detection performance. The T-E-GraphSAGE method introduced in this paper addresses the shortcomings of traditional graph embedding techniques by capturing topological details and edge features in network flow data, leading to more precise detection. while its ability to effectively classify samples with unseen node features. Three NIDS standard datasets are used to evaluate our model, which verifies the broad applicability accuracy, and robustness of our model in different types of network scenarios, which is effective in comparison with traditional ML methods, especially in complex network environments. Through these improvements, the performance of our system in network intrusion detection has been significantly improved, and it is able to effectively respond to various network attacks in complex network environments.

III. THE PROPOSED METHOD

A. GraphSAGE

Graph Neural Networks (GNN) are becoming increasingly popular in the field of machine learning. Its power stems from the effective utilization of graph-structured data. These data are widely available in application areas such as social media networks, biological research, and telecommunication systems[15]. The primary reason for using GNN in NIDS is their capability to leverage the structural data present in network streams, which can be represented graphically. Although some conventional machine learning approaches also handle graph data, they usually involve intricate processes and depend heavily on manually crafted features, leading to more cumbersome and less efficient applications.

GraphSAGE[16] is an efficient graph neural network technique that generates embedded representations of nodes by sampling and aggregating the features of their neighbors. It is particularly suitable for processing large-scale graph data. The main steps include sampling neighboring nodes, aggregating features, and updating node features, which effectively solve the computation and storage bottlenecks of traditional graph neural networks. As a result, GraphSAGE has been widely used in many fields.

GraphSAGE : learning node representation through local aggregation, and its core steps include three aspects: neighbor node sampling, feature aggregation, and node feature update, as shown in Fig. 2.

In neighbor node sampling, for each node, a fixed number of neighbor nodes are randomly sampled to reduce the computation and storage requirements. Suppose a node in the graph is v, and its set of neighbor nodes is $N(v)$, and the set of neighbor nodes obtained from sampling is $\tilde{N}(v)$. This process can be represented as:

$$\tilde{N}(v) = Sample(N(v), K) \quad (1)$$

where K denotes the number of neighbor nodes sampled. This phase seeks to manage computational complexity by limiting the number of adjacent nodes for each vertex in extensive graphs.

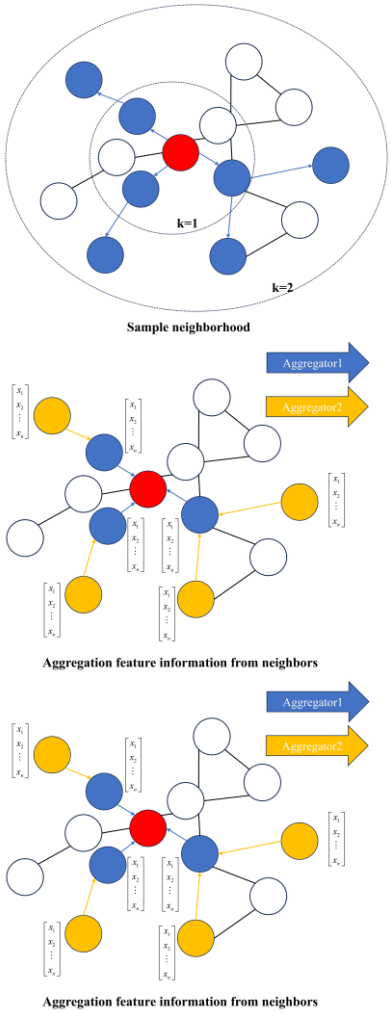

Fig. 2. GraphSAGE model diagram

In feature aggregation, a feature aggregation operation is performed on the sampled set of neighbor nodes $\tilde{N}(v)$ to generate neighbor feature representations. Common aggregation methods include mean value aggregation, pooling, and LSTM. The following are the formulas for several aggregation methods:

1) Mean aggregation: Mean aggregation computes the average of neighboring node features. Its formula is:

$$h_{\tilde{N}(v)}^{(k)} = mean(\{h_u^{(k-1)}, \forall u \in \tilde{N}(v)\}) \quad (2)$$

where $h_u^{(k-1)}$ denotes the feature representation of the neighboring node at the $k-1$ th layer of u, and $h_{\tilde{N}(v)}^{(k)}$ denotes

the representation of the node v after aggregating the features of its neighboring nodes at the k layer.

2) Maximum pooling: Maximum pooling is used to take the maximum value in the features of neighboring nodes. The formula for this is:

$$h_{\tilde{N}(v)}^{(k)} = max(\{h_u^{(k-1)}, \forall u \in \tilde{N}(v)\}) \quad (3)$$

3) LSTM aggregation: LSTM aggregation uses LSTM network for neighbor node features with the formula:

$$h_{\tilde{N}(v)}^{(k)} = LSTM(\{h_u^{(k-1)}, \forall u \in \tilde{N}(v)\}) \quad (4)$$

For node feature update, the algorithm combines the node's own features with the aggregated neighbor features and updates the node feature representation through a neural network. A common way of combining is a concatenation operation (concatenation) followed by a transformation through a fully connected layer. Its formula is:

$$h_v^{(k)} = \sigma(W^{(k)} \cdot concat(h_v^{(k-1)}, h_{\tilde{N}(v)}^{(k)})) \quad (5)$$

where σ denotes the activation function (e.g., ReLU), $W^{(k)}$ denotes the weight matrix of the k-th layer, and $h_v^{(k)}$ denotes the feature representation of node v in the k-th layer.

In the specific process, the features are first initialized and each node's feature can be its attribute vector x_v. Then multi-layer sampling and aggregation is performed, for the k-th layer, each node v randomly samples a fixed number of K neighbors from its neighborhood to form the sampling set $\tilde{N}(v)$ and aggregates the features of the neighboring nodes using the selected aggregation function (e.g., mean, maximum pooling, or LSTM) to obtain $h_{\tilde{N}(v)}^{(k)}$. Then the node v own features are connected to the aggregated neighboring features in a join operation and nonlinearly transformed through the fully connected layer to obtain a new node feature representation $h_v^{(k)}$. Finally, after multi-layer (usually 2 to 3 layers) sampling and aggregation operations, the embedding representation of each node is finally generated h_v. Through the above steps, the GraphSAGE algorithm is able to efficiently deal with large-scale graph data, and generate high-quality node embedding representations through sampling and aggregation operations.

B. E-Transformer-GraphSAGE Methods

The traditional GraphSAGE method mainly focuses on the analysis and utilization of node features for node classification, but is deficient in dealing with edge features. The primary objective of NIDS aims to detect and identify malicious traffic. In our study, we focus on the application of edge features and improve the GraphSAGE model by using the edge embedding method and introducing the Transformer layer method.

1) E-GraphSAGE: In order to handle graph structure data efficiently, we designed and implemented the GraphSAGE layer (SAGELayer). This layer updates the representation of each node by aggregating the features of the node's neighbors to capture the relationships between nodes in the graph. GraphSAGE accomplishes the updating of node representations through message passing and apply updates, and employs the ReLU activation function to improve the model's nonlinear representation[17]. The main differences from the original GraphSAGE algorithm are the algorithmic inputs, the message passing aggregation functions and the outputs. In the SAGE layer, edge embedding is incorporated into the messaging process to provide richer information. Unlike the traditional GraphSAGE module, the aggregated embedding of sampled neighboring edges is generated at the kth layer for edge features. using a mean aggregation function as shown in the following equation.

$$h_{\tilde{N}(v)}^{(k)} = mean(\{e_{uv}^{(k-1)}, \forall u \in \tilde{N}(v), uv \in \varepsilon\}) \quad (6)$$

where $e_{uv}^{(k-1)}$ is the feature of the edge uv in the k-1 layer of the sampling neighborhood $\tilde{N}(v)$ of node v, and the set $\{\forall u \in \tilde{N}(v), uv \in \varepsilon\}$ represents the sampling edges within the neighborhood $\tilde{N}(v)$ .The edge features of the uv of the kth layer are spliced by the following equation, which represents the final result of the forward propagation phase.

$$h_{uv}^{k} = CONCAT(h_u^k, h_v^k), uv \in \varepsilon \quad (7)$$

In our study, we constructed a two-layer E-GraphSAGE model with each layer consisting of an E-SAGELayer.

Neighboring node features are aggregated to generate the embedded representation of the node and a mean value aggregation method is used, where the features of the node are the mean value of the features of its neighboring nodes. The first layer E-SAGELayer in this model aggregates the input features to generate the first layer of node embedding; The second layer takes the first layer of node embeddings as input and again performs aggregation to generate the final node embeddings. Through this multi-layer aggregation, we are able to capture more complex node characteristics and neighbor relationships. A Dropout operation is used to avoid overfitting. The advantage of stacking multiple layers of GraphSAGE is the ability to capture more complex node relationships and form richer node representations to improve the performance of the model.

2) Transformer: The traditional GraphSAGE method mainly focuses on the analysis and utilization of node features for node classification, but is deficient in dealing with edge features. The primary aim of NIDS is to detect and identify malicious traffic, aligning with the edge classification problem in network flow classification. Our study emphasizes the use of edge features and enhances the GraphSAGE model by incorporating the edge embedding method and introducing the Transformer layer technique.

The Transformer Encoder Layer (TEL) is the basic component of the Transformer model, which mainly consists of the MultiheadAttention mechanism, Feed-forward Neural Network (Linear Layer), and Normalization Layer (LayerNorm), and Dropout is applied between the layers to prevent overfitting. In the Transformer Encoder Layer, the inputs are node features (generated by the SAGE layer) and this layer does not explicitly process edge features. Its main function is to capture the dependencies between node features and global information through a multi-head attention mechanism along with a feed-forward neural network.

a) Multi-head attention: The self-attention mechanism allows the model to capture global dependencies by focusing

Identify applicable funding agency here. If none, delete this text box.

on all other elements in a sequence while processing each element in the sequence. The multi-head self-attention mechanism improves the model's sensitivity to different features by performing multiple self-attention computations in parallel. The specific formula is as follows:

$$\begin{cases} Attemtion(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \\ MultiHead(Q,K,V) = Concat(head_1,\cdots head_i,\cdots,head_h)W_O \end{cases} \quad (8)$$

where $Attemtion(Q,K,V)$ is the single-head self-attention computation, Q denotes the computational query matrix, K denotes the key matrix, V denotes the value matrix, d denotes the input vector dimension, and $MultiHead(Q,K,V)$ denotes the multi-head self-attention splicing the results of the h heads together and obtaining the final output by a linear transformation, where $head_i = Attention(Q_i,K_i,V_i)$, and $W_O \in \mathbb{R}^{hd_k \times d_{model}}$ is the output weight matrix and d_{model} is the input feature dimension.

Specifically, the MultiheadAttention mechanism captures the global dependencies of the input data by processing the input data in parallel through multiple Attention Heads. Each Attention Head performs self-attention computation independently, which is able to focus on different features in the input data and enhance the sensitivity of the model to multiple features. The multi-head attention mechanism's output is linked to the feed-forward neural network via a linear transformation.

b) Feed-forward neural network: Feed-forward neural networks (FFN) are fully connected neural networks applied independently at each position in each Transformer coding layer. The specific formula is as follows:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \quad (9)$$

where $W_1 \in \mathbb{R}^{d_{model} \times dff}, W_2 \in \mathbb{R}^{d_{ff} \times d_{model}}, b_1 \in \mathbb{R}^{d_{ff}}, b_2 \in \mathbb{R}^{d_{model}}$ is the parameter of the science department and d_{ff} is the hidden layer dimension of the FNN.

The feedforward neural network used in this paper includes two fully connected layers with a ReLU activation function and Dropout applied in between. The first fully connected layer maps the input dimension from the embedded dimension (embed_dim) to a higher hidden dimension (ff_hidden_dim), the ReLU activation function introduces a nonlinear transformation, and the Dropout operation is used to prevent overfitting. The second fully connected layer maps the hidden dimension back to the embedded dimension, thus keeping the dimensionality of the inputs and outputs the same.

c) Normalization layer: The normalization layer is implemented following each sublayer, including both self-attention and the feed-forward neural network, to ensure regularization and stabilize the training process. The specific formulas are as follows:

$$LayerNorm(x) = \frac{x-\mu}{\sigma+\varepsilon} \bullet \gamma + \beta \quad (10)$$

where μ and σ are the mean and standard deviation of the inputs respectively, γ and β are the learnable scaling and offset parameters and ε is a small constant.

Each coding layer undergoes Layer Normalization and Residual Connection between and after the multi-head self-attention mechanism and the feed-forward neural network. Layer Normalization helps to stabilize and speed up the training process, while Residual Connection helps to solve the problem of vanishing gradients in deep networks.

d) Dropout: Dropout randomly discards a certain percentage of neurons during training to prevent overfitting. By stacking multiple such coding layers, the Transformer model is able to effectively capture the global dependencies of the input data and enhance the model's sensitivity to different features. The multi-head self-attention mechanism in each layer enables the model to focus on different features in the input data, and the feed-forward neural network further processes these features. Through the layer-by-layer processing of the multi-layer structure, the model is able to capture more complex and deeper feature relationships in the input data, which improves its performance in various tasks.

C. NIDS

Fig. 3 shows how the network stream data is constructed as graph data and the propagation process from the source node to the destination node. Fig. 4 shows an overview of our E-Transformer-GraphSAGE NIDS. Initially, a graph is created using the network flow data. Next, the generated network graph is fed into the E-Transformer-GraphSAGE model for supervised training. Edge embeddings are designed to classify network streams into benign or malicious categories. The following subsections explain these three steps in detail.

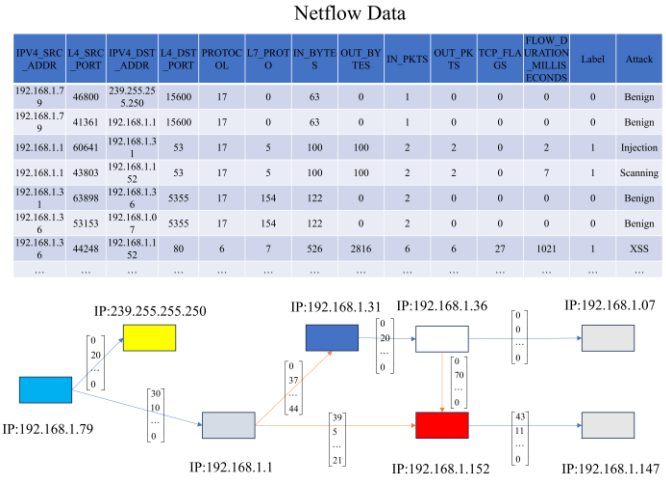

Fig. 3. Network flow data conversion diagram data

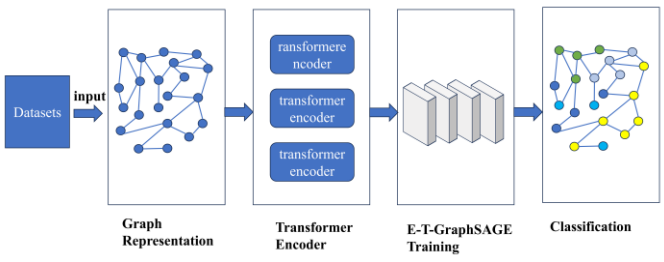

Fig. 4. E-Transformer-graphsage-based Network Intrusion Detection System

1) Graph data structure: Net-Flow is a commonly used format for logging network communications in production environments and is the predominant format in Network Intrusion Detection System (NIDS) environments. A flow record typically includes fields that identify the communication's source and destination, along with additional information like packet and byte counts, and flow duration. Graph structures naturally model this type of data. In this study, we use the source IP address, source port, destination IP address, and destination port. The first two fields form a binary group identifying the source node, and the last two form the destination node. The remaining data are used as features for that edge, making the graph nodes featureless. We assign a vector of all 1's to all nodes in the algorithm.

2) E-Transformer-GraphSAGE: Our proposed model combines the sensitivity of GNN to local structures and the ability of Transformer to capture global dependencies by first processing the graph data through E-GraphSAGE to obtain node representations. Then, Transformer is utilized to further capture global dependencies. During the training process, we utilize a weighted cross-entropy loss function (CrossEntropyLoss) to address category imbalance. We use Adam optimizer (Adam optimizer) for parameter updating. The algorithm's output is compared with the labels from the NIDS dataset and the model's trainable parameters are adjusted in the backpropagation phase. After tuning the model parameters during training, the performance of the model can be evaluated by classifying unseen test samples. The process involves converting the test stream records into graph data structures. Edge embeddings are then generated using a trained E-Transformer-GraphSAGE layer. These edge embeddings are subsequently transformed into class probabilities via the Softmax layer. The predicted class probabilities are compared with the actual class labels to evaluate the classification performance metrics.

IV. EXPERIMENT

In this section, We performed binary classification and multiclassification task comparisons to validate the effectiveness of our algorithm.

A. Experiment Setting

We modeled using Python, Pytorch, and DGL, and the server environment was performed on an Intel(R) Xeon(R) Gold 6242 CPU @ 2.80GHz total of 32 cores, a single A100 graphics card, and 192G RAM.

B. Datasets

To evaluate our proposed GNN-based NIDS, we use three publicly available datasets that include various labeled attack flows and benign network flows. The first dataset is BoT-IoT, which is widely used for evaluating ML based network intrusion detection systems in the Internet of Things, with a proprietary format and feature set. The second and third datasets are NF-ToN-IoT and NF-BoT-IoT presented in Netflow format.

1) BoT-IoT datasets: The BoT-IoT dataset[18] was generated by the Cyber Range Lab at the Australian Center for Cyber Security (ACCS) to evaluate the performance of cyber security tools. It simulates real network environments containing normal traffic and multiple types of malicious traffic such as DDoS, DoS, reconnaissance, and data theft for Intrusion Detection System (IDS) training and testing. Avoid combining SI units, like current in amperes, with CGS units, such as the magnetic field measured in oersteds, as this can cause dimensional imbalance and confusion. If using mixed units, clearly specify the units for each quantity in the equation.

2) NF-BoT-IoT datasets: The NF-BoT-IoT dataset[19] is a NetFlow characterization dataset extracted from the BoT-IoT dataset to provide a more concise representation of network traffic by summarizing IP traffic flows. The dataset includes information such as source and destination IP addresses, ports, packet counts, byte counts, and timestamps, which helps in large-scale data analysis and real-time intrusion detection.

3) NF-ToN-IoT datasets: The NF-ToN-IoT dataset is a NetFlow characterization dataset generated based on the ToN-IoT dataset and contains telemetry and operational network data from Internet of Things (IoT) devices. The dataset provides detailed traffic records that help detect network intrusions and understand traffic patterns in IoT environments and is suitable for IoT security research.

C. Results Of The Experiment

To assess the effectiveness of the proposed neural network model, we employed the standard metrics outlined in Table I. Here, TP stands for true positives, TN for true negatives, FP for false positives, and FN for false negatives.

TABLE I. EVALUATION INDICATORS

Accuracy	$\frac{TP+TN}{TP+FP+TN+FN} \times 100\%$
Precision	$\frac{TP}{TP+FP} \times 100\%$
FAR	$\frac{\text{FP}}{FP+TN} \times 100\%$
Recall	$\frac{TP}{TP+FN} \times 100\%$
F1-Score	$2 \times \frac{Precision \times Recall}{Precision \times Recall} \times 100\%$

1) Binary classification results: The datasets employed in our experiments contain dual-layer labels for each data instance. The first layer indicates whether the network flow is benign or non-benign, while the second layer specifies the attack type. For the binary classification task, we use the first layer of labels, and for the multi-class classification task, we use the second layer of labels[20,21].

TABLE II. BINARY CLASSIFFCATION RESULTS

Dataset	*Accuracy*	*Precision*	*F1-Score*	*Recall*	*FAR*
BoT-IoT	99.99%	1.00	1.00	99.99%	0.00%
NF-BoT-IoT	94.52%	1.00	0.99	97.32%	0.24%
NF-ToN-IoT	99.93%	1.00	1.00	99.84%	0.03%

Table II summarizes our model's performance metrics—accuracy, precision, F1-Score, and False Alarm Rate (FAR)—

across three datasets: BoT-IoT, NF-BoT-IoT, and NF-ToN-IoT. The findings demonstrate that our method performs exceptionally well in binary classification, a key factor for successful network intrusion detection.

In cybersecurity, datasets frequently exhibit an imbalance, with fewer attack samples compared to normal traffic. The F1-Score is particularly important in such scenarios as it balances precision and recall, providing a more accurate assessment of the model's ability to differentiate between benign and malicious traffic than accuracy alone.

Given the importance of precise intrusion detection, particularly in practical applications where the cost of missed detections is high, we prioritize the F1-Score as a more reliable indicator of our model's performance. In the following sections, we will compare our F1-Score with those from other studies to demonstrate how effectively our model handles the challenges of imbalanced datasets, ensuring dependable intrusion detection.

TABLE III. COMPARISON OF BINARY-CLASSIFICATION ALGORITHMS F1

Method	***Dataset***	***F1***
Ours	BoT-IoT	1.00
CatBoost		0.99
Ours	NF-BoT-IoT	0.99
Extra Tree Classifier		0.97
TS-IDS		0.95
Ours	NF-ToN-IoT	1.00
Extra Tree Classifier		1.00

Table III shows the F1 of our method compared with other algorithms[21,22]. The results show that our method achieves F1-Scores that are either similar to or better than those of existing approaches. This indicates that our method performs effectively in both traffic classification and binary network intrusion detection.

The comparable or superior F1-Scores demonstrate that our model is not only accurate in identifying malicious network traffic but also maintains a balanced performance across different datasets. This balance is crucial in practical applications, where high precision and recall are necessary to minimize false positives and ensure reliable intrusion detection.

In summary, the data in Table III confirms that our method is competitive with, and in some cases superior to, other leading algorithms, highlighting its effectiveness in traffic classification and network intrusion detection tasks.

2) Multiclass classiffcation results: Table IV presents the multi-classification results of our method across three standard datasets, where the classifier is tasked with distinguishing between various attack types. The multi-classification problem is more complex than binary classification, as it requires the model to accurately identify not just whether an attack is present, but also to specify the type of attack. The results in Table IV indicate that our model demonstrates strong performance, particularly on the BoT-IoT dataset. This superior performance is indicative of the model's capability to effectively differentiate between the distinct attack types within this dataset.

Table V provides further insight into the model's performance by showing the recall and F1-Score values for different attacks in the multi-classification task, specifically focusing on the ToN-IoT dataset. These metrics are crucial for understanding the model's ability to correctly identify each attack type. High recall values suggest that the model is effective in identifying the majority of true positive instances for most attack types, minimizing the risk of undetected threats. Similarly, strong F1-Score values indicate a good balance between precision and recall, reinforcing the model's robustness in handling diverse attack scenarios.

TABLE IV. COMPARISON OF BOT-IOT AND NF-BOT-IOT MULTI-CLASSIFICATION ALGORITHMS F1

	BoT-IoT		***NF-BoT-IoT***	
Class Name	Recall	F1-Score	Class Name	Recall
Benign	100.00%	0.99	Benign	100.00%
DDos	99.99%	1.00	DDos	99.99%
Dos	99.99%	1.00	Dos	99.99%
Reconnaissance	99.99%	1.00	Reconnaissance	99.99%
Theft	94.52%	0.98	Theft	94.52%
Weighted Average	99.99	1.00	Weighted Average	99.99

TABLE V. COMPARISON OF NF-TON-IOT MULTI-CLASSIFICATION ALGORITHMS

	NF-ToN-IoT	
Class Name	Recall	F1-Score
Benign	98.33%	0.99
Backdoor	98.46%	0.99
DDos	57.47%	0.73
Dos	99.72	0.46
Injection	30.59	0.46
MIMT	55.02	0.25
Ransomware	80.28	0.42
Password	100.00	0.99
Scanning	25.92	0.15
XSS	40.70%	0.28
Weighted Average	68.65%	0.67

However, the experimental plots of confusion matrices shown in Figures 5 and 6 for the NF-BoT-IoT and NF-ToN-IoT datasets reveal some nuances in the model's performance. While the recognition rate is extremely high for several attack types, the model struggles with accurately classifying DDoS attacks. This issue likely stems from the fact that during model training, DDoS and DoS attacks shared similar features, leading to a significant overlap in their learned representations. As a result, the model occasionally misclassifies DDoS attacks as DoS attacks, which suggests that the feature extraction process may need refinement to better distinguish between these two attack types.

The observed difficulty in separating DDoS from DoS attacks highlights a potential area for improvement. One possible solution could involve enhancing the feature engineering process to capture more distinctive characteristics

of these attack types. Additionally, adjusting the training process to emphasize the differences between DDoS and DoS attacks, perhaps through the use of more advanced techniques like adversarial training or ensemble learning, could further improve classification accuracy.

In summary, while our model excels in the multi-classification of several attack types, especially within the BoT-IoT dataset, there remains room for improvement in the classification of closely related attacks such as DDoS and DoS. Addressing these challenges will be crucial for further enhancing the model's overall reliability and effectiveness in real-world network security applications.

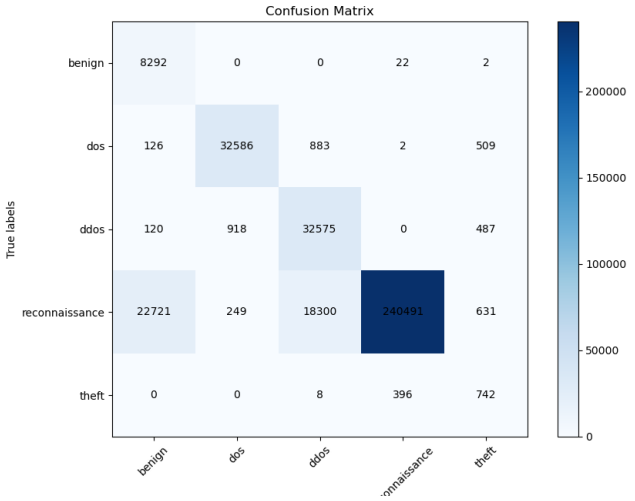

Fig. 5. NF-BoT-IoT multiclassification results

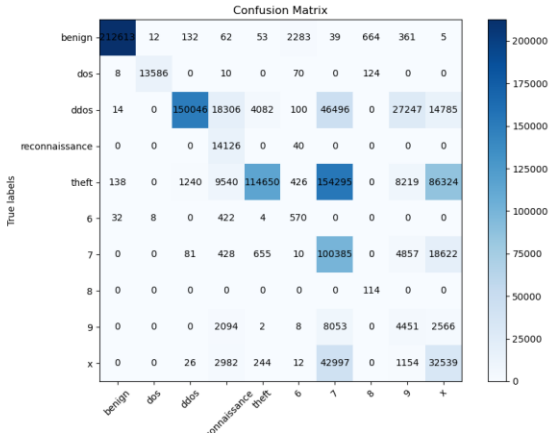

Fig. 6. NF-ToN-IoT multiclassification results

As with binary classification, we compared the performance of our model's Network Intrusion Detection System (NIDS) with other classifiers, as shown in studies [23,24] . Table VI presents the results of this comparison, focusing on the multi-classification task.

The findings reveal that our algorithm consistently achieves higher average F1-Score values compared to all existing methods. This is particularly important in multi-classification, where the ability to accurately distinguish between multiple types of network attacks is crucial. The superior F1-Score suggests that our model not only identifies attacks effectively but also excels in correctly classifying the different types of attacks, a challenge where other classifiers often fall short.

These results underscore the effectiveness of our approach in handling the complexities of multi-class network intrusion detection, proving that our model outperforms current leading methods in this critical area.

TABLE VI. COMPARISON OF MULTI-CLASSIFICATION ALGORITHMS F1

Method	***Dataset***	***W-F1***
Ours CatBoost	BoT-IoT	1.00 0.99
Ours Extra Tree Classifier TS-IDS	NF-BoT-IoT	0.88 0.77 0.83
Ours Extra Tree Classifier	NF-ToN-IoT	0.67 0.60

Overall, our method demonstrates superior performance compared to other Network Intrusion Detection System (NIDS) approaches across both binary and multi-classification tasks, as evidenced by the results from the three datasets utilized in our study. Our model not only achieves higher accuracy and F1-Scores but also shows remarkable robustness and generalizability. This indicates that it is well-equipped to handle various types of network traffic and detect both known and emerging threats effectively.

The model's ability to consistently outperform other methods highlights its advanced capabilities in accurately identifying and classifying different types of network attacks, whether it's simply distinguishing between benign and malicious traffic or correctly categorizing specific attack types. This robust performance across diverse datasets suggests that our method is adaptable to different network environments and can maintain its effectiveness even when faced with the complexities and variabilities of real-world data.

V. CONCLUSION AND FUTURE WORK

In this paper, we have introduced a novel GNN-based network intrusion detection method called E-T-GraphSAGE, which has enhanced attack flow detection by capturing edge features and topology patterns within network flow graphs. Our focus has been on applying E-T-GraphSAGE to detect malicious network flows in the context of network intrusion detection. Experimental evaluations have shown that our model performs very well on the three NIDS benchmark datasets and generally outperforms currently available network intrusion detection methods. In the future, we plan to build unsupervised graph neural network intrusion detection models, as well as lighten the E-T-GraphSAGE model and apply it to edge network servers, especially small and medium-sized network devices, for better timely network intrusion detection at the edge.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant 62101299.

REFERENCES

[1] Chaabouni N, Mosbah M, Zemmari A, et al. Network intrusion detection for IoT security based on learning techniques[J]. IEEE Communications Surveys & Tutorials, 2019, 21(3): 2671-2701.

[2] Naeem H. Analysis of Network Security in IoT-based Cloud Computing Using Machine Learning[J]. International Journal for Electronic Crime Investigation, 2023, 7(2).

[3] Deng X, Zhu J, Pei X, et al. Flow topology-based graph convolutional network for intrusion detection in label-limited IoT networks[J]. IEEE Transactions on Network and Service Management, 2022, 20(1): 684-696.

[4] Zhong X, Wan G. Six-GraphSecurity: Industrial Internet Intrusion Detection Based On Graph Neural Network[C]//2023 IEEE 7th Information Technology and Mechatronics Engineering Conference (ITOEC). IEEE, 2023, 7: 1340-1344.

[5] Sukhbaatar S, Grave E, Bojanowski P, et al. Adaptive attention span in transformers[J]. arXiv preprint arXiv:1905.07799, 2019.

[6] Alowaidi M. Modified Intrusion Detection Tree with Hybrid Deep Learning Framework based Cyber Security Intrusion Detection Model[J]. International Journal of Advanced Computer Science and Applications, 2022, 13(10).

[7] Gupta N, Jindal V, Bedi P. LIO-IDS: Handling class imbalance using LSTM and improved one-vs-one technique in intrusion detection system[J]. Computer Networks, 2021, 192: 108076.

[8] Kumar P J, Nedunchelivan S, Adnan M M, et al. Anomaly-Based Intrusion Detection System Using Bidirectional Long Short-Term Memory for Internet of Things[C]//2024 Third International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE). IEEE, 2024: 01-04..

[9] Suárez-Varela J, Almasan P, Ferriol-Galmés M, et al. Graph neural networks for communication networks: Context, use cases and opportunities[J]. IEEE network, 2022, 37(3): 146-153.

[10] Hnamte and J. Hussain, "Network Intrusion Detection using Deep Convolution Neural Network," 2023 4th International Conference for Emerging Technology (INCET), Belgaum, India, 2023, pp. 1-6, doi: 10.1109/INCET57972.2023.10170202.

[11] Kisanga P, Woungang I, Traore I, et al. Network anomaly detection using a graph neural network[C]//2023 International Conference on Computing, Networking and Communications (ICNC). IEEE, 2023: 61-65..

[12] Ouyang L, Zhang Y, Wang Y. Unified graph embedding-based anomalous edge detection[C]//2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 2020: 1-8.

[13] Sun Y, Zhu D, Wang Y, et al. GTC: GNN-Transformer Co-contrastive Learning for Self-supervised Heterogeneous Graph Representation[J]. arXiv preprint arXiv:2403.15520, 2024.

[14] Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. 2022. Universal Graph Transformer Self-Attention Networks. In Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), April 25–29, 2022, Virtual Event, Lyon, France. ACM, New York, NY, USA,.

[15] Zhou J, Cui G, Hu S, et al. Graph neural networks: A review of methods and applications[J]. AI open, 2020, 1: 57-81.

[16] Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs[J]. Advances in neural information processing systems, 2017, 30.

[17] Lo W W, Layeghy S, Sarhan M, et al. E-graphsage: A graph neural network based intrusion detection system for iot[C]//NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2022: 1-9.

[18] Koroniotis N, Moustafa N, Sitnikova E, et al. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset[J]. Future Generation Computer Systems, 2019, 100: 779-796.

[19] Sarhan M, Layeghy S, Moustafa N, et al. Netflow datasets for machine learning-based network intrusion detection systems[C]//Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings 10. Springer International Publishing, 2021: 117-135.

[20] Sarhan M, Layeghy S, Portmann M. Evaluating standard feature sets towards increased generalisability and explainability of ML-based network intrusion detection[J]. Big Data Research, 2022, 30: 100359.

[21] Tanha J, Abdi Y, Samadi N, et al. Boosting methods for multi-class imbalanced data classification: an experimental review[J]. Journal of Big data, 2020, 7: 1-47.

[22] Lawal M A, Shaikh R A, Hassan S R. An anomaly mitigation framework for iot using fog computing[J]. Electronics, 2020, 9(10): 1565.

[23] Churcher A, Ullah R, Ahmad J, et al. An experimental analysis of attack classification using machine learning in IoT networks[J]. Sensors, 2021, 21(2): 446.

[24] Nguyen H, Kashef R. TS-IDS: Traffic-aware self-supervised learning for IoT Network Intrusion Detection[J]. Knowledge-Based Systems, 2023, 279: 110966.