Citation: Zhian Cui, Hailong Li, Xieyang Shen. HGS-ATD: A hybrid graph convolutional network-graphSAGE model for anomaly traffic detection. *Journal of Harbin Institute of Technology (New Series)*. DOI: 10.11916/j.issn.1005-9113.2025008

HGS-ATD: A hybrid Graph Convolutional Network-GraphSAGE Model for Anomaly Traffic Detection

Zhian Cui, Hailong Li and Xieyang Shen*

(*Rocket Force University of Engineering, Xi'an 710025, China*)

Abstract: With network attack technology continuing to develop, traditional anomaly traffic detection methods that rely on feature engineering are increasingly insufficient in efficiency and accuracy. Graph Neural Network (GNN), a promising deep learning (DL) approach, has proven to be highly effective in identifying intricate patterns in graph-structured data and has already found wide applications in the field of network security. In this paper, we propose a hybrid Graph Convolutional Network (GCN)-GraphSAGE model for Anomaly Traffic Detection, namely HGS-ATD, which aims to improve the accuracy of anomaly traffic detection by leveraging edge feature learning to better capture the relationships between network entities. We validate the HGS-ATD model on four publicly available datasets, including NF-UNSW-NB15-v2. The experimental results show that the enhanced hybrid model is 5.71% to 10.25% higher than the baseline model in terms of accuracy, and the F1-score is 5.53% to 11.63% higher than the baseline model, proving that the model can effectively distinguish normal traffic from attack traffic and accurately classify various types of attacks.

Keywords: anomaly traffic detection; graph neural network; deep learning; graph convolutional network

CLC number: TP393 **Document code**: A **Article ID**: 1005-9113(2025)00-0000-18

0 Introduction

As computer network attacks become more frequent and sophisticated, the effectiveness and efficiency of anomaly traffic detection have become significant challenges in ensuring network security. To address these challenges, many enterprises and governments depend on Network Intrusion Detection Systems (NIDS)[1] to safeguard vital systems, sensitive information, and overall network integrity. These systems typically analyze raw network traffic to detect potential intrusion activities. Common formats for traffic records include packet capture and flow-based records, which outline the characteristics of network traffic and categorize them accordingly. Traditional NIDS methods are generally divided into two paradigms: signature-based[2] and behavior-based detection[3]. The former approach uses predefined rules or patterns to identify known threats. In contrast, behavior-based methods employ more sophisticated techniques, often integrating Machine Learning (ML) approaches to identify evolving and novel attack patterns.

Existing NIDS methods frequently overlook the topological distinctions between normal and attack traffic, a factor critical for accurate intrusion detection. Incorporating topological information[4], especially the relationships between nodes and lateral movement paths, can significantly enhance intrusion detection performance. Through improved feature extraction, this deeper analysis helps detect attacks like data theft, DDoS[5], and reconnaissance[6-8].

Lately, Graph Neural Network (GNN)[9] have gained prominence as powerful tools for modeling, yielding impressive results in areas such as Natural Language Processing (NLP)[10], speech recognition[11], and computer vision [12-13]. GNN are particularly well-suited for handling graph-structured data with complex relationships. By propagating information through the connections between nodes and edges, GNN can effectively model traffic behavior patterns within networks. In order for a GNN to map IP and port to the nodes and network flow to edges, network

Received 2025-01-16.

Sponsored by the National Natural Science Foundation of China (Grant No.62103434), the National Science Fund for Distinguished Young Scholars (Grant No.62176263).

* Corresponding author: Zhian Cui, Master's student. Email: zhian_cui@163.com.

traffic could be symbolized as a graph with nodes (such as port number, IP address, etc.) and edges (such as data packet, protocol, byte number, etc.). This creates an ideal application environment for problems involving the detection of anomaly traffic.

In addition, GNN can effectively aggregate information from neighboring nodes through their unique message passing mechanism[14], thus capturing complex structures and relationships in network data. This enables GNN to analyze network traffic by not solely relying on predefined features, but by autonomously learning the most relevant patterns from the data. As a result, the need for traditional manual feature engineering is minimized, giving the model greater flexibility to adapt to diverse and complex network topologies and traffic behaviors.

The fundamental limitations of current GNN-based techniques are threefold. First, the majority of research (including E-GraphSAGE[15] and (Graph Convolutional Network) GCN-based techniques[16]) ignores edge attributes, which are essential for capturing traffic interactions, in favor of concentrating on node aspects. In addition, when working with large-scale network graphs, typical GNN frameworks are computationally inefficient, which restricts their use in practical situations. Lastly, current models perform worse on unbalanced datasets, particularly for a large percentage of attacks that are likely to result in high false positive rates.

In this paper, our research goal is to design and implement an anomaly traffic detection model based on hybrid GCN and GraphSAGE, which utilizes the respective advantages of GCN's global structure modeling and GraphSAGE's local efficient sampling to improve the detection accuracy. At the same time, feature coding and edge-based batching strategies were introduced to optimize the processing efficiency of large-scale network data, and the problems of large computation and high memory consumption of traditional GCN in large-scale graph data were solved. Specifically, we transform the network traffic data for a graph format, with each edge representing a traffic-related characteristic like protocol type, flow size, and more. Additionally, we label the edges of the graph to enable supervised learning during the subsequent training process. To efficiently handle large-scale network graph data[17], we have designed an edge-based batching method combined with a node expansion technique. In each batch, a certain number of edges are selected, and the corresponding neighboring nodes are expanded, ensuring that each batch contains a sufficient number of nodes and edges. During training, this tactic aids in preserving the model's robustness and diversity. Our model's architecture is built on a hybrid framework that blends the GraphSAGE and GCN models. By aggregating information from neighboring nodes, this hybrid model learns feature representations for both nodes and edges, allowing it to capture potential anomalous behaviors within the network and accurately identify various types of attack patterns. According to experimental results, our model outperforms existing approaches across several public datasets, demonstrating significant improvements in detection performance.

In conclusion, this paper has three main contributions:

· We propose a new anomaly traffic detection model, called HGS-ATD, utilizing both graph convolution and edge features to improve the accuracy of anomaly traffic detection.

· An edge-based batch method is designed to ensure sample diversity and sufficient node coverage by dynamically expanding the set of nodes in each batch. This strategy can be applied to large-scale data, and enhances computational efficiency during training.

· The efficiency of the suggested strategy in reliably recognizing different types of attacks and distinguishing between attack and benign traffic is validated by substantial experiments conducted on four datasets.

The remaining parts of the document are structured as follows: the associated studies are presented in Section 1, the pertinent background is summarized in Section 2, Section 3 details the proposed model and methodology, the experimental setup and results analysis are presented in Section 4, and the work is summarized in Section 5.

1 Related Work

Network intrusion detection system is crucial for safeguarding computer networks against a variety of attacks. With the continuous evolution of network attack methods, traditional signature-based and rule-based methods have gradually been unable to deal with complex and new attacks, not only zero-day

attacks[18] and advanced persistent threats, but also DDoS[19] attacks, large-scale Botnets attacks, and new covert attacks using encrypted communication channels[20]. As a result, in the past few years, ML-based approaches, particularly DL and GNN, have attracted significant interest as promising advancements in the area of NIDS.

Traffic data in a network is generally composed of devices, IP addresses, or other communication nodes, with their interactions represented as edges. GNN excel at modeling the complex relationships between nodes and edges, particularly by using convolutional operations to capture the intricate structural features of network communication. Busch et al.[21] extracted directed edge attribute graphs from network traffic, where each node corresponds to network endpoint, and each edge represents a communication feature vector between two endpoints. Zhao et al.[22] introduced a sextuple representation of network traffic, encompassing details such as source IP, destination IP, ports, protocols, requests, and responses. Pujol-Perich et al.[23] developed a host connection graph in which nodes signify hosts or traffic entities, and edges represent the connections between them. Zhou et al.[24] presented network traffic as a communication graph comprising a node set and an adjacency matrix. The node set includes multiple distinct nodes observed in the traffic records, representing devices or entities in the network, while the adjacency matrix captures the connections between these nodes.

Many existing studies tend to overlook edge features, often initializing node features using vectors derived from all edge features. These methods all rely on graphs that are directly built from the underlying network architecture, producing graph data that closely resembles computer network topology. Nevertheless, this method restricts the model's capacity to completely utilize the connections among traffic flows produced within the network. In GNN models, similar nodes are typically linked by edges, but traditional network topologies struggle to generalize this relationship effectively. By creating graph-structured data based on the connections between traffic flows, we overcome this constraint in this work. Since every node in the graph reflects the properties of a flow, the connections made by edges are able to capture the nodes' innate commonalities.

Hamilton et al.[25] introduced the GraphSAGE approach, which improves the computational efficiency of GNN on large-scale graphs by sampling neighboring nodes, making it suitable for real-time traffic detection. While this method allows for generating low-dimensional vector representations of graphs, it has a significant limitation: it overlooks edge features. The E-GraphSAGE method, an extension of GraphSAGE intended for intrusion detection tasks, was created by Lo et al.[15] in order to overcome this limitation. This inductive model integrates edge features into the GraphSAGE framework. The method entails constructing the flow representation by aggregating the nearby flows of the source and target nodes. However, one limitation of this approach is that traffic features themselves have insufficient influence on the embedding representation. The real-time urban traffic analysis method proposed by Chandramohan et al.[26] optimizes routing decisions by modeling vehicle interaction through graph structure, which provides a cross-domain idea for efficient utilization of edge features in dynamic networks.

Chang and Branco[27] proposed E-ResSAGE and E-ResGAT algorithms, which are based on GraphSAGE and Graph Attention Network (GAT) algorithms respectively. In terms of model construction, E-ResSAGE algorithm retains the original traffic feature information by adding residual connections in the aggregation operation, and enhances the detection ability of minority categories. On the basis of E-ResSAGE, the E-ResGAT algorithm further introduces the attention mechanism to perform weighted aggregation of the neighborhood information of the traffic data, so that the model can better pay attention to the influence of different neighbor nodes on the current node.

Bao et al.[28] proposed a hybrid model based on E-GraphSAGE and LSTM (Long Short-Term Memory) to improve detection performance by combining spatial structure and time series features. E-GraphSAGE is used to perform hierarchical edge sampling and aggregation of the network traffic graph to retain the topological characteristics of communication between nodes. The residual design is introduced to concatenate the original edge features and the aggregated features to alleviate the feature dilution problem.

Ji and Meng[29] applied GNN-based representation learning methods to the problem of network traffic

classification. Instead of directly extracting features from the traffic, they transformed raw traffic data into an image format and used a GCN for image classification, ultimately achieving traffic classification. Initial experimental results indicated strong performance on smaller networks, which showed a 97.35% classification accuracy. Ying et al.[30] introduced a network intrusion detection system leveraging GCN. This approach represented traffic data as a graph structure, enabling GCN to capture intricate dependencies between network nodes and edges. Experiments demonstrated that this method could better identify anomalous behavior patterns in the network compared with conventional ML-based approaches.

The use of GCN for heterogeneous information network (HIN) was examined by Yang et al.[16]. However, several significant limitations constrained the effectiveness of GCN on HIN, affecting both their precision and effectiveness. They developed an comprehensible and effective heterogeneous graph convolutional network (ie-HGCN) to overcome these difficulties. Experiment results showed that their approach successfully addressed the drawbacks of current latest methods, allowing for a deeper comprehension and learning of HIN.

As shown in Fig. 1, Tran and Park[31] classified malicious network traffic using a GCN model. The model required a significant amount of memory and computing time, even though its architecture was simple—it just had two GCN layers and one fully connected layer. Additionally, it only achieved around 90% identification rate on the CIC-IDS2017 dataset, indicating a moderate detection performance.

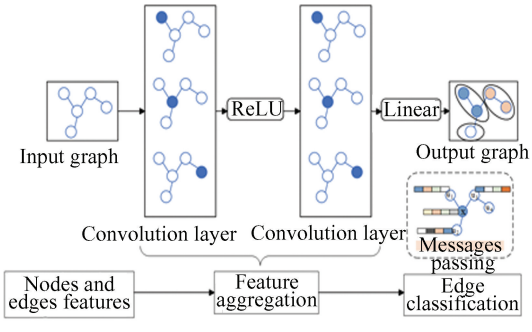


Fig. 1 GCN model

Deng and Huang[32] proposed a novel intrusion detection model, EMA-IDS (Edge-featured Multi-hop Attention Graph Neural Network for Intrusion Detection System). The core idea is to use GNN combined with multi-hop attention mechanism and edge features to improve the effect of network intrusion detection. It effectively alleviates the over-smoothing problem often encountered by traditional GNN in deep networks, and also improves the computational efficiency through the multi-hop attention mechanism.

Du et al.[33] designed an intrusion detection model combining GCN and LSTM networks, named UGL (UAV-GCNLSTM). A comprehensive extended dataset is constructed by adding three dimensions to the original dataset: Message Cycle, Hamming Distance, and Entropy. At the same time, the efficiency of GCN in capturing network topology and the ability of LSTM in processing time series data are used to efficiently and accurately intercept network attacks in the real UAV working environment.

In the GNN architecture, GAT[34] dynamically learns the association weights between nodes through the attention mechanism, which can capture complex interaction patterns more flexibly. GAT has been applied to various anomaly detection tasks. For example, Jahin et al.[35] proposed to combine contrastive Learning Graph Network (CAGN) and GAT, which fuses attention mechanism and contrastive loss through a two-stream architecture to improve feature discrimination. However, GAT's high computational complexity and lack of scalability to large-scale graph data limit its application in real-time traffic detection. Notably, the introduction of GATv2 improves the attention mechanism, making it a potential candidate for further research in this area.

This work offers an enhanced GCN-based model to solve the drawbacks of existing GCN approaches. Although the GCN method is widely acknowledged as one of the best methods for classifying nodes, it performs poorly on huge graph datasets. The model's efficiency is greatly impacted when working with huge graph data since it must store the complete adjacency matrix and its accompanying characteristics in memory. The suggested batch processing technique greatly enhances the model's capacity to effectively handle massive amounts of graph data. In addition to being able to differentiate between regular and abnormal traffic, our proposed model can also accurately detect different sorts of attacks by utilizing edge features and topological information.

2 Background

2.1 Graph Neural Network

GNN[36] is a DL framework tailored for processing graph-structured data. GNN transforms graph data into standardized feature representations by applying certain algorithms to a graph's nodes and edges, which can subsequently be used in various neural network models for training. In applications including node classification, propagating edge information, and graph-based aggregation, GNNs have shown remarkable performance.

GNN was first developed in 2005, when Recurrent Neural Network (RNN)[37] was first used to handle graph data. Subsequently, researchers proposed GCN, which incorporated properties such as translation invariance, local perception, and weight sharing from Convolutional Neural Network (CNN)[38-39] into graph structures. This laid the groundwork for the construction and improvement of subsequent GNN frameworks.

The key feature of GNN is its ability to efficiently learn graph representations through a message passing mechanism, which plays a key role in anomaly traffic detection. It allows the model to effectively aggregate information between nodes and capture complex relationships and patterns in network traffic, as shown in Fig. 2. Nodes in the graph are connected to each other by edges, and messages are passed from one node to neighboring nodes along edges. In this process, each node updates its hidden state based on the messages it receives from its neighbors. For example, the hidden state of node *A* is affected by the information of its neighboring nodes *B*, *C*, and *D*. By continuously iterating this message passing and state updating, the model is able to learn the feature representation of the entire graph, which makes GNN particularly effective in dealing with graphs with complex structures or attributes, so as to detect abnormal traffic. In addition to natural language processing, image analysis, trajectory prediction, physical chemistry, and drug discovery, they are versatile in a number of other areas as well.

In the domain of anomalous traffic detection, GNN are used to analyze abnormal patterns in network traffic. By converting network traffic into a graph structure, GNN can capture relationships between data packets and distinguish between attack traffic and normal traffic by aggregating node features. The application of GNN in anomaly detection is not limited to node-level classification; it also encompasses analysis at the level of graph connections and entire network structures, such as the detection of anomalous nodes, edges, subgraphs, and entire graphs.

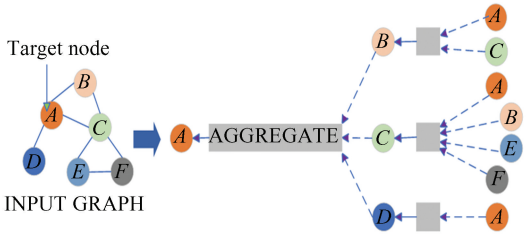


Fig. 2 Message passing mechanism

2.2 Graph Convolutional Network

GCN[40] can handle non-Euclidean structures of data[41], such those present in social media networks, molecular networks, and user behavior data in recommendation systems, in contrast to typical CNN, which are mostly employed for grid data (like photos). Nodes and edges make up graph data, the nodes stand for discrete entities, while the edges specify the connections or exchanges between them. In many applications, nodes not only contain attribute information (such as text or image features), but edges often carry additional features as well (such as relationship strength or timestamps).

The fundamental concepts behind GCN involve propagating and combining information across the graph's adjacency structure, allowing each node to incorporate information from its neighborhood during the learning process. Unlike traditional neural networks with fixed structures, GCN leverages the graph's topology to aggregate features from neighboring nodes, this allows GCN to capture local as well as global dependencies inside the structure of the graph.

The hidden state of each node undergoes two sequential operations: aggregation and update. This is where the concept of "convolution" becomes relevant. The following formula can be used to update the concealed state of nodes in each GCN layer:

$$\boldsymbol{H}^{(l+1)} = \sigma(\boldsymbol{A}\boldsymbol{H}^{(1)}\ \boldsymbol{W}^{(l)}) \tag{1}$$

where σ is the activation function (such as ReLU), $\boldsymbol{W}$ is the weight matrix, $\boldsymbol{A}$ is the graph's normalized adjacency matrix, and $\boldsymbol{H}$ is the node feature matrix,

and $\boldsymbol{H}^{(l+1)}$ is the new node features obtained through the graph convolution operation.

As a potent tool for processing graph data, GCN may enhance performance through end-to-end learning and efficiently capture the dependencies within the graph structure using graph convolution operations. By using the topological information of the graph and the properties of its nodes and edges to perform complex pattern recognition, GCN offers strong assistance for efficient detection and classification in the context of network traffic anomaly detection.

2.3 GraphSAGE

GraphSAGE (Graph Sample and Aggregation), introduced by Hamilton et al.[25] in 2017, is a variant of GNN. GraphSAGE proposes an innovative approach that significantly improves the scalability and efficiency of GNN by extracting partial data from each node's neighbors and combining these data to generate node features.

Each node's representation in GraphSAGE is modified based on the feature information of the nodes that are adjacent to it. To avoid operating on the entire graph, a fixed number of neighboring nodes, denoted as $N_s(v)$, are sampled from the neighbor set $N(v)$ of node v. If the mean aggregation method is used, the sampled neighbor set is $N_s(v) = \{v_1, v_2, \cdots, v_k\}$.

The aggregation step is the core of GraphSAGE, which creates a new feature vector that represents the node and the aggregated information of its neighbors by combining the features of the sampled nearby nodes. Let the feature vectors of the neighbors of node v be $\boldsymbol{h}_{v1}, \boldsymbol{h}_{v2}, \cdots, \boldsymbol{h}_{vk}$, where $\boldsymbol{h}_v$ represents the feature of node v. The aggregation function, AGGREGATE, aggregates these neighboring feature vectors into a new vector $\boldsymbol{h}_v^{(k)}$:

$$\boldsymbol{h}_v^{(k)} = \mathrm{AGGREGATE}(\{\boldsymbol{h}_{v1}, \boldsymbol{h}_{v2}, \cdots, \boldsymbol{h}_{vk}\}) \quad (2)$$

Common aggregation functions include mean aggregation, pooling aggregation, and LSTM aggregation.

A neural network is then used to update the node through integrating the aggregated neighbor features with its own features. Let $\boldsymbol{h}_v^{(k-1)}$ represent the node v feature vector in layer $(k-1)$, then the feature of node v in the k-th layer is:

$$\boldsymbol{h}_v^{(k)} = \sigma(\boldsymbol{W}_k \cdot [\boldsymbol{h}_v^{(k-1)}, \mathrm{Agg}(N_s(v))]) \quad (3)$$

Here, $\boldsymbol{W}_k$ is the learned weight matrix, $\mathrm{Agg}(N_s(v))$ represents the aggregation operation (such as mean) on the neighboring nodes, and σ is the activation function. In the final layer of the graph, the representations of nodes are employed for downstream tasks such as regression, classification, etc.

2.4 Comparative Analysis of Models

The traditional GCN model effectively captures the complex dependencies in the graph structure by graph convolution operation, which is suitable for processing data with non-Euclidean structure. However, it will lead to memory explosion when processing datasets with 1000 edges (e.g. NF-CSE-CIC-IDS2018-v2). GraphSAGE significantly improves the scalability and efficiency of GNN on large-scale graphs by sampling neighbor nodes and aggregating their features. GraphSAGE supports a variety of aggregation functions, such as mean aggregation, pooling aggregation and LSTM aggregation, and can dynamically weight neighbor nodes according to their importance. However, GraphSAGE ignores edge features, which limits its performance in some scenarios. GAT introduces an attention mechanism to improve feature discrimination, but multi-head attention leads to 3-5 times increase in computational cost. Although the E-GraphSAGE method interates edge attributes, the fixed window sampling strategy is difficult to adapt to dynamic traffic changes. The specific comparative analysis is shown in Table 1.

Table 1 comparative analysis of state-of-art models

Model	Strengths	Limitations
GCN[29-30]	Captures global dependencies via adjacency matrix propagation.	High memory consumption for large graphs; ignores edge features.
GraphSAGE[25]	Efficient neighbor sampling; inductive learning for dynamic graphs.	Edge features are not integrated into aggregation.
E-GraphSAGE[15,28]	Extends GraphSAGE with edge feature integration.	Full-batch training limits scalability; lacks dynamic subgraph sampling.
GAT[35]	Dynamically weights node interactions via attention mechanisms.	High computational complexity; struggles with billion-edge graphs.
EMA-IDS[32]	Multi-hop attention enhances edge feature utilization; reduces over-smoothing.	Requires pre-defined graph structure; limited support for dynamic edge updates.
UGL[33]	Combines GCN with LSTM for spatiotemporal modeling.	Fixed batch size; no adaptive node expansion for imbalanced data.

3 Methodology

Traditional GNN, such as GCN, GAT, and GraphSAGE, have been successfully utilized across various applications. These methods mainly emphasize node features for node classification and do not presently account for edge features in the context of edge classification. On the other hand, the hybrid algorithm combining GCN and GraphSAGE that we propose allows us to simultaneously incorporate both node and edge feature information during the embedding process. This approach lays the groundwork for calculating edge embeddings and conducting edge classification, thus facilitating the effective differentiation of normal and attack traffic and ensuring accurate classification of different attack types.

3.1 Data Preprocessing

Data preprocessing plays a crucial role in transforming raw NetFlow data into graph data for training and testing. A detailed description of our data preprocessing procedure is provided below. Firstly, the key information of each flow is extracted from the large amount of NetFlow data in the dataset, which includes IP address, port number, packet and byte count, and other useful packet statistics. NetFlow data supports the conversion between flow records and those in graphical format because nodes can be represented by IP addresses, and ports, and flow information can represent edges.

For this study, the original network traffic data is first converted into the relationship between graph nodes and edges through feature engineering, the source IP and port, the destination IP and port are merged into a unique node identifier, redundant port fields are deleted, and the edge list based on communication relationship is constructed. After dealing with infinite values and missing values, the numerical features are standardized, and the features are scaled to zero mean and unit variance by StandardScaler. This process improves the convergence efficiency of the model. Finally, the stratified sampling strategy was adopted in the data segmentation stage, and the data set was divided into a 60% training set and 40% test set, and the proportion of samples of each category in the training set and the test set was the same.

3.2 Graph Construction

Information including source and destination IP addresses, source and destination ports, and traffic characteristics like duration, transaction bytes, and transmission packet size are commonly included in network traffic data. During the graph construction process, the raw data needs to be cleaned and preprocessed. We handle missing values by imputation, correct outliers, and standarding as well as encoding categorical features such as protocol type and TCP flags.

When building the network traffic graph, each traffic record (i. e., a network connection) is represented as an edge. The source IP address and port, along with the target IP address and port, serve as the graph's nodes, while other traffic details are stored as features of the edges. This approach effectively frames the anomaly traffic detection task as an edge classification problem.

3.3 The Proposed HGS-ATD Model

With a focus on unusual traffic pattern identification, we provide an enhanced GCN model in conjunction with the GraphSAGE model for node classification tasks in network data. To improve generalization, the model applies ReLU activation and dropout regularization after each of the two SAGEConv layers and a fully connected output layer. Finally, the Multi-Layer Perceptron (MLP) predictor is used to forecast the assault classifications. Fig. 3 displays the model's architecture. The model first converts network traffic data into graph structure through data preprocessing and graph building module, where nodes and edges represent different network elements and traffic characteristics respectively. Then, the data passes through node coding module and edge coding module successively. In the node coding module, the GCN layer is responsible for the propagation of node information in the global scope and provides the global semantic background for the GraphSAGE layer. On this basis, the GraphSAGE layer combines the features of local neighbor nodes for dynamic aggregation and optimization of node feature representation. The edge coding module calculates the weight of edge features through the attention mechanism to capture the important information of the edge more effectively. Finally, through the edge classification module, the edge embedding is mapped to specific classification results using MLP to detect whether the traffic is abnormal. This architecture design combines the advantages of GCN and GraphSAGE to better handle large-scale graph data and accurately identify abnormal traffic.

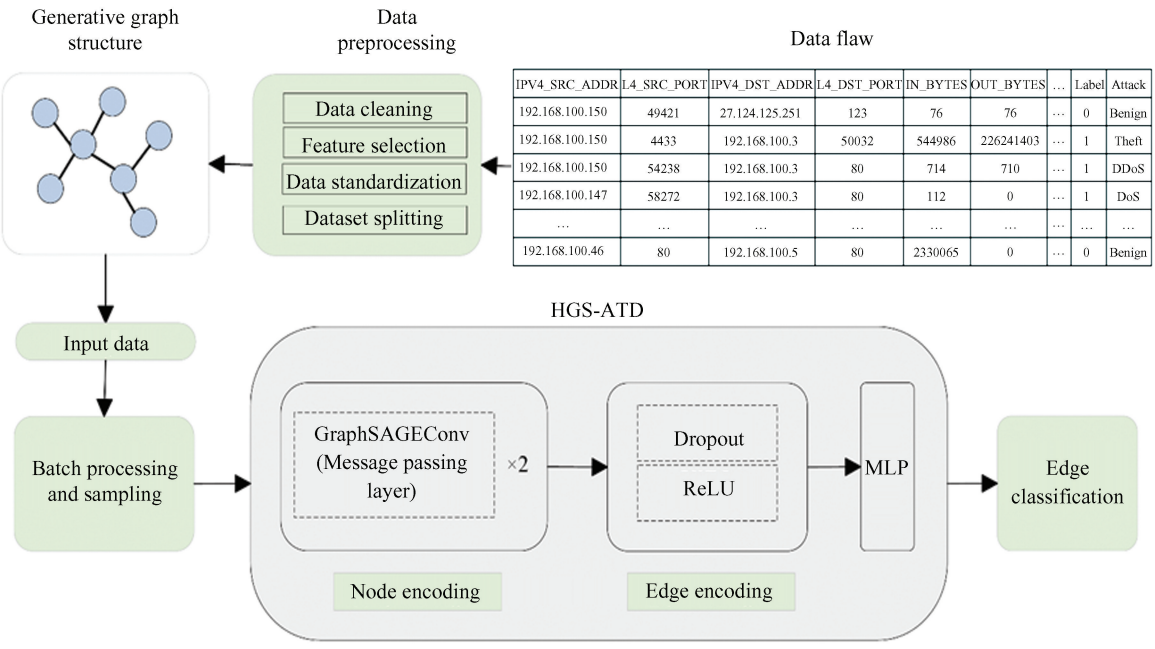


Fig. 3 The overall architecture of HGS-ATD model

The core principle of our approach lies in synergizing the strengths of GraphSAGE and GCN. A popular model in GNN, GCN relies on the graph convolutional layer to learn graph structure representations. Through multiple convolution operations, GCN can effectively capture intricate features and structural information from the graph, thereby improving the model's accuracy. However, GCN faces certain limitations when handling large-scale graph data, especially when the graph is large. In such cases, the feature representation for each node requires aggregation of information from all its neighboring nodes, which significantly increases both computational demand and memory consumption.

To handle massive amounts of network traffic data, batch processing and node expansion strategies are also introduced. Specifically, an edge-based batching approach is employed to restrict the number of nodes and edges within each batch to a defined range, optimizing both computational efficiency and model performance. Instead of using the full neighborhood as in Ref.[15], for each batch, we first select a set of edges and then generate a subgraph by extracting the relevant node and edge features from these edges. This subgraph serves as the training input, effectively mitigating computational bottlenecks associated with processing the entire graph in memory. Specifically, the minimum number of nodes M in each batch is first determined to make sure that the number of nodes in each batch is enough to avoid insufficient information caused by too few nodes. Choose a batch of edges $E_{\text{batch}} \subseteq \varepsilon$ from the graph, which is of size B. For each edge $e \in E_{\text{batch}}$, the source and destination nodes are extracted to obtain a preliminary set of nodes V_{batch}, it is shown as follows:

$$V_{\text{batch}} = \{v_i \mid e_i = (v_i, v_j) \in \varepsilon_{\text{batch}}\} \tag{4}$$

When generating batches, if a batch contains a small number of nodes (less than a preset minimum number of nodes), we will expand the batch by selecting more adjacent nodes to ensure the diversity and representative of the batch, as shown in Eq.(5):

$$V_{\text{expanded}} = V_{\text{batch}} \cup \{\text{neighbors}(v) \mid v \in V_{\text{batch}}\} \tag{5}$$

where neighbors (v) represents the set of neighbor nodes of node v, that is, all nodes directly connected to node v in the original network topology graph.

This expansion operation adds neighboring nodes such that the number of nodes in the batch is at least M. Extract a subgraph G_{subgraph} from the original graph G based on the expanded node set V_{expanded}:

$$G_{\text{subgraph}} = (V_{\text{expanded}}, E_{\text{subgraph}}) \tag{6}$$

where E_{subgraph} is the edge containing all nodes in V_{expanded}. The subgraph is then used as the input data in the current batch for GNN training.

Similarly, SAGEConv, as a variant of the GraphSAGE model, improves on GraphSAGE by introducing more expressive convolution operations. Different from the traditional GraphSAGE method, SAGEConv captures the information in the graph

structure more finely by normalizing the representation of each neighbor node. This improvement enables SAGEConv to extract more detailed neighbor node information, enhancing the model's capacity to depict graph data. Following batch processing and the node expansion strategy, the data flows through the node encoding module, edge encoding module, and edge classification, ultimately producing the final output result.

Node encoding: Firstly, the GCN layer realizes the global scope of node information dissemination through the normalization operation of the adjacency matrix, as shown in Eq.(7):

$$\boldsymbol{H}^{(l)} = \sigma(\boldsymbol{A}\boldsymbol{H}^{(l-1)}\ \boldsymbol{W}^{(l)}) \tag{7}$$

where $\boldsymbol{H}^{(l)}$ represents the node feature matrix of the l-th layer, A is the normalized adjacency matrix of the graph, $\boldsymbol{W}^{(l)}$ represents the trainable weight matrix of the l-th layer, and σ is the activation function (such as ReLU).

By combining the characteristics of nearby nodes over the entire graph and weighting them according to the corresponding links, GCN can capture global structural traits. After receiving the output of GCN layer, GraphSAGE takes it as the initial node feature, and then combines the features of local neighbor nodes for dynamic aggregation. Aggregation functions employed by the GraphSAGE layer, such as averaging, pooling, or LSTM aggregation, enable it to assign weights based on the importance of its neighbors, it is shown as follows:

$$h_u^{(l)} = \sigma(W^{\mathrm{SAGE}} \cdot \mathrm{CONACT}(h_u^{\mathrm{GCN}}, \mathrm{AGGREGATE}(\{h_u^{\mathrm{GCN}} \mid v \in N(u)\}))) \tag{8}$$

where h_u^{GCN} is the feature passed from the GCN layer, and AGGREGATE is the aggregation function of GraphSAGE, such as pooling, mean or LSTM.

GCN provides global semantic background for GraphSAGE, which makes GraphSAGE more accurate in local feature aggregation. GraphSAGE further optimized the features generated by GCN through a dynamic weighting mechanism, and enhanced the adaptability of the model to heterogeneous graphs and abnormal relationships.

Edge encoding: Input the edge coding module's embedded and edge features that were processed by the preceding module. Better modification of the impact of various edges on the nodes is made possible by the attention-based calculations. The calculation of edge feature attention is as follows:

$$a_{uv} = \mathrm{softmax}(\mathrm{ReLu}(W_e^L f_{uv}^{(0)})) \tag{9}$$

Through the attention mechanism, a weight a_{uv} is assigned to the edge (u, v), which indicates the importance of the edge to the final result in the aggregation process. The initial feature $f_{uv}^{(0)}$ of the edge undergoes a linear conversion W_e^L, and then the ReLU function is used to enhance the expressiveness. Softmax is used to normalize the weights of all edges, so that the weight distribution of edges conforms to the probability meaning.

The attention coefficients obtained are subsequently integrated with the edge features to generate a weighted edge feature using the attention mechanism. This method makes it possible to gather important information from the edges more successfully. The following is the formula:

$$h_{uv}^{(l)} = \mathrm{AGG}(h_u^{(L)} \parallel \alpha_{uv} \parallel h_v^{(l)}) \tag{10}$$

where $h_{uv}^{(l)}$ denotes the embedding representation of the edge (u, v) at the lth layer, $h_u^{(L)}$ and $h_v^{(l)}$ are the embedding representations of the nodes at both ends of the edge, α_{uv} is the attention weight of the edge, $\parallel$ represents the vector splicing operation.

Edge classification: The edge embedding z_{uv} generated by the READOUT operation captures the interrelations between the two end nodes of the edge and its properties and is used as the input for edge classification, as shown in Eq. (11):

$$\boldsymbol{z}_{uv} = \mathrm{READOUT}(\boldsymbol{z}_u, \boldsymbol{z}_v), \forall (u,v) \in \varepsilon \tag{11}$$

where, $\boldsymbol{z}_u$ and $\boldsymbol{z}_v$ stand for the final embedding vectors of the nodes u and v, and z_{uv} denotes the embedding representation of the edge (u, v). Finally, MLP was used to map the edge embedding to the specific classification result, which was used to detect whether it was abnormal traffic.

$$y_{uv} = \mathrm{MLP}(z_{uv}) \tag{12}$$

where y_{uv} is the prediction result of edge (u,v), and MLP is responsible for mapping the edge embedding to the classification result using a fully connected layer.

Algorithm 1 outlines the entire process and training flow of our model.

Algorithm 1: The HGS-ATD algorithm

Input:

- A graph $G(V,\varepsilon)$
- Input node features $X = \{x_i \mid i \in N\}$
- Input edge features $E = \{e_{uv} \mid \forall uv \in \varepsilon\}$
- Parameters: Number of layers, aggregation type (GCN/SAGE).

Output: Edge classification scores $Y = \{y_{uv} \mid \forall uv \in \varepsilon\}$

1 For each iteration do

/* Batch processing and sampling */
batch_size = B, min_nodes = M
$V_{\text{batch}} = \{v_i \mid e_i = (v_i, v_j) \in \varepsilon_{\text{batch}}\}$
if $|V_{\text{batch}}| < M$:
$V_{\text{expanded}} = V_{\text{batch}} \cup \{\text{neighbors}(v) \mid v \in V_{\text{batch}}\}$
$G_{\text{subgraph}} = \text{subgraph}(V_{\text{expanded}}, G)$
/* node encoding */
For $l = 1, 2, \ldots, L$ do
$H^{(l)} = \sigma(AH^{(l-1)}W^{(l)})$;
$h_u^{(l)} = \sigma(W^{\text{SAGE}} \cdot \text{CONACT}(h_u^{\text{GCN}}, \text{AGGREGATE}(\{h_u^{\text{GCN}} \mid v \in N(u)\})))$;
end
/* edge encoding */
$a_{uv} = \text{softmax}(\text{ReLu}(W_e^L f_{uv}^{(0)}))$
$h_{uv}^{(l)} = \text{AGG}(h_u^{(L)} \parallel \alpha_{uv} \parallel h_v^{(l)})$
/* edge classification */
$z_{uv} = \text{READOUT}(z_u, z_v), \forall (u, v) \in \varepsilon$
$y_{uv} = \text{MLP}(z_{uv})$
end
Return $Y = \{y_{uv}\}$ // Return all edge classification scores

Overall, the proposed model combines the strengths of GCN and SAGEConv, replacing the traditional GraphConv layer. This design not only enables better handling of large-scale graph data but also maintains high accuracy and excellent performance. Choosing the right hyperparameters, including the total amount of hidden units as well as learning rate, is essential to maximizing the model's accuracy and stability. In order to compare the proposed model with current methods, we conducted experiments on a number of benchmark datasets. The outcomes reveal that our model exhibits significant gains in efficiency and accuracy. Detailed experimental results and performance assessments will be discussed in the following sections.

4 Experiment and Result

4.1 Datasets

In this study, we utilize four distinct NetFlow datasets[42]: NF-UNSW-NB15-v2, NF-ToN-IoT, NF-BoT-IoT, and NF-CSE-CIC-IDS2018-v2. The NF-UNSW-NB15-v2 dataset consists of network traffic data derived from the UNSW-NB15[43] dataset, including both normal traffic and nine distinct types of attack traffic. On the other hand, the NF-ToN-IoT dataset is specifically focused on IoT network traffic, incorporating features such as device diversity and intricate traffic patterns. The NF-BoT-IoT dataset is designed to provide insights into botnet attacks targeting IoT devices. Concurrently, the original file set of the CSE-CIC-IDS2018 dataset is used to create the NF-CSE-CIC-IDS2018-v2 dataset[44]. In order to examine network traffic and identify irregularities in IoT-based settings, these datasets are created specifically for that purpose, with standardization to the NetFlow format[45] carried out by Sarhan et al.[46]. Each dataset is specifically curated to simulate a variety of network conditions, aiding in the development and testing of NIDS. Table 2 summarizes the salient features of these datasets.

4.2 Baselines

In this experiment, we evaluate the effectiveness of HGS-ATD model using DL and ML approaches, comparing it with three baseline models widely recognized as leading techniques in network intrusion detection.

- XGBoost[47] is a tree-based ML algorithm that belongs to the gradient boosting framework. It improves prediction accuracy by progressively constructing decision trees, where each new tree attempts to address the faults (residuals) produced by the prior ones. This iterative process allows XGBoost to learn from various data features, such as network traffic patterns or packet durations, to make more accurate and informed predictions.
- E-GraphSAGE[15] serves as a foundational model for analyzing graph data. It addresses a range of tasks in graph-structured datasets, including node classification, edge prediction, and graph clustering, by effectively learning representations of both nodes and edges within the graph. However, because all of the data must be processed simultaneously on the GPU, sometimes going beyond memory restrictions, its computational efficiency may be constrained when working with huge datasets. To overcome this problem, we retain the original E-GraphSAGE framework but incorporate a mini-batch training strategy to improve scalability.
- Anomal-E[48] can efficiently identify anomalous patterns in graph data by fusing self-supervised learning techniques with GNN, particularly in cases when the node-edge relationship is aberrant. By using a graph-based self-supervised task for training, the learned node representation is able to learn more about the graph structure without the need

for a lot of label data. Large-scale graph data can still be processed effectively thanks to the graph convolution technique, which avoids the enormous expense of directly computing the entire graph.

Table 2 Salient features of the four datasets

Dataset	Total number	Class name	Number	Percentage(%)
NF-UNSW-NB15-v2	2390275	Benign	2295222	96.02
		Analysis	2299	0.10
		Backdoor	2169	0.09
		DoS	5794	0.24
		Exploits	31551	1.32
		Shellcode	1427	0.06
		Generic	16560	0.69
		Reconnaissance	12779	0.53
		Fuzzers	22310	0.93
		Worms	164	0.01
NF-ToN-IoT	1379274	Benign	270279	19.60
		Injection	468539	33.97
		DDos	326345	23.66
		Password	156299	11.33
		Dos	17717	1.28
		Backdoor	17247	1.25
		MITM	1295	0.09
		Ransomware	142	0.01
		Scanning	21467	1.56
		XSS	99944	7.25
NF-BoT-IoT	600100	Benign	13859	2.31
		Reconnaissance	470655	78.43
		DDoS	56844	9.47
		Dos	56833	9.47
		Theft	1909	0.32
NF-CSE-CIC-IDS2018-v2	18893708	Benign	16635567	88.05
		Bot	143097	0.76
		Brute force	120912	0.64
		DDOS	1390270	7.36
		DoS	483999	2.56
		Infilteration	116361	0.62
		Web Attacks	3502	0.02

4.3 Experiment Settings

The performance of the HGS-ATD model is thoroughly assessed using the four standard evaluation criteria listed below: Accuracy (ACC), Precision, Recall and F1-score. These indicators have been widely used in several studies[49-50] and can evaluate the classification performance of models from several perspectives, especially when there are imbalanced categories.

For the parameter setting of the model, we set up 2 SAGEConv layers in order to capture more information about the network structure. By comparing the performance of models with different number of layers, the 2-layer model provides the best balance between accuracy and computational efficiency. The number of hidden units is 128. Experiments show that this value can provide better performance on all data sets. A small number of hidden layer units cannot fully capture the complex relationship in the data, resulting in low accuracy. We set the number of epochs to 5 because, after several trials, we found that 5 epochs would allow the model to learn enough about the data and avoid overfitting due to too many epochs. The learning rate range is set to $[10^{-3}, 10^{-2}]$, and the grid search method is used to tune the model. The experimental results show that when the learning rate is 0.001, the model can converge well in the training process, and avoid excessive fluctuations in training. We chose 2048 as the batch size in order to balance memory usage and training speed. Dropout regularization of 0.2 was applied after each SAGEConv layer, this setting performed well on the

dataset and helped the model avoid overfitting issues during training. The optimizer is Adam[51], which combines the advantages of Adagrad and RMSProp, and can adaptively adjust the learning rate of each parameter, which helps to accelerate the convergence speed of the model and improve the training effect. Table 3 shows the detailed settings of experimental parameters.

Table 3 Setting hyperparameters

No. Layers	No. Hidden	Learning rate	Epoch
2	128	$[10^{-3}, 10^{-2}]$	5
Loss function	**Dropout**	**Batch size**	**Optimizer**
Cross-entropy loss	0.2	2048	Adam

The dataset is divided into two parts, with the training set accounting for 60% and the test set accounting for 40%. We use Python[52], PyTorch[53], PyTorch-geometric[54] and DGL[55] to implement our proposed model and training procedure. Because of the large-scale data in this experiment, we perform the evaluations using an NVIDIA Tesla V100 32GB GPU.

4.4 Results of Binary Classification

We evaluate the HGS-ATD model on a range of datasets against XGB, E-GraphSAGE, and Anomal-E for the binary classification test. According to the experimental findings shown in Table 4, HGS-ATD outperforms the other models on the majority of datasets, especially when it comes to important metrics such as ACC and F1-score.

Table 4 Binary classification results

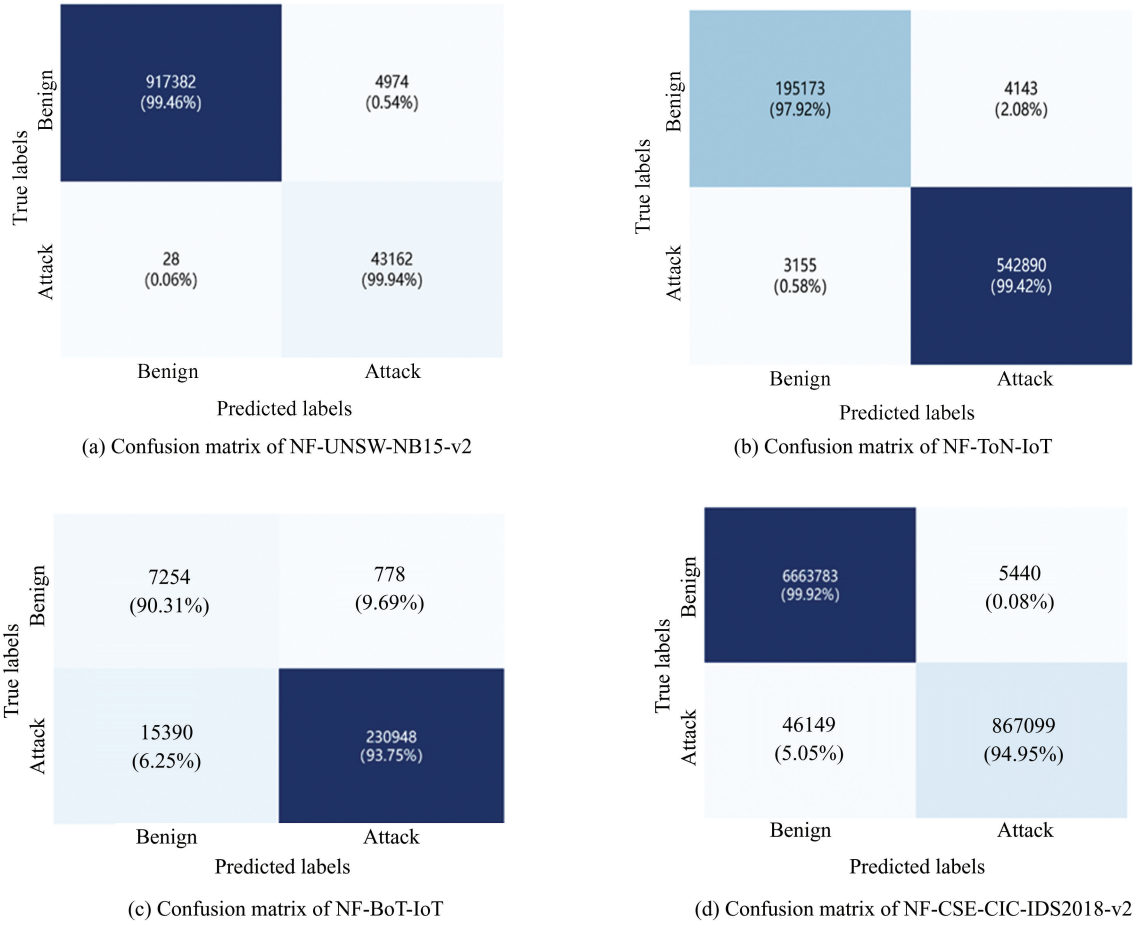
Method	Dataset	Metrics			
		Accuracy	Precision	Recall	F1-score
HGS-ATD	NF-UNSW-NB15-v2	**0.9948**	0.8967	**0.9994**	0.9452
XGB		0.9921	0.9358	0.9921	**0.9637**
E-GraphSAGE		0.9853	**0.9474**	0.9935	0.9606
Anomal-E		0.9202	0.6086	0.9351	0.6563
HGS-ATD	NF-ToN-IoT	**0.9902**	**0.9924**	**0.9942**	**0.9933**
XGB		0.9857	0.9804	0.9861	0.9833
E-GraphSAGE		0.9645	0.9457	0.9140	0.9503
Anomal-E		0.8877	0.8326	0.9184	0.8522
HGS-ATD	NF-BoT-IoT	**0.9364**	**0.9966**	**0.9375**	**0.9576**
XGB		0.9356	0.9636	0.7815	0.8413
E-GraphSAGE		0.9057	0.5967	0.8897	0.9025
Anomal-E		0.9202	0.6086	0.9351	0.6563
HGS-ATD	NF-CSE-CIC-IDS2018-v2	**0.9932**	**0.9938**	0.9495	**0.9711**
XGB		0.9925	0.9729	**0.9726**	0.9702
E-GraphSAGE		0.9913	0.9514	0.9641	0.9613
Anomal-E		0.9361	0.9409	0.9062	0.9158

Note: Data in bold is the one with the best performance.

In the binary classification experiments of four data sets, HGS-ATD model shows significant advantages. For NF-UNSW-NB15-v2, HGS-ATD leads other models with an accuracy of 0.9948, and its recall rate is as high as 0.9994, which almost completely captures attack traffic, but its precision is slightly lower than that of the baseline model, indicating that a small amount of normal traffic is misclassified as an attack. In NF-ToN-IoT with a small number of samples and NF-BoT-IoT with a high imbalance (97.69% of attacks), the indicators of HGS-ATD are higher than the three baseline models, showing its applicability and robustness in IoT scenarios, as well as its reliability in data skew scenarios. For the large-scale dataset NF-CSE-CIC-IDS2018-v2, HGS-ATD performs best with 0.9932 accuracy and 0.9938 precision, but the recall rate (0.9495) slightly decreases due to the limited dataset size, indicating that the model still maintains high reliability in large traffic scenarios. However, it needs to combine dynamic feature extraction to improve the adaptability to unknown attacks.

Confusion matrix in Fig. 4 shows the specific detection results of the model on four data sets. In NF-UNSW-NB15-v2 and NF-CSE-CIC-IDS2018-v2, the model highlights its high recall characteristics with an accuracy of more than 99% and a very low miss rate, especially suitable for the scenario of zero tolerance to attacks. In NF-ToN-IoT, the model balances the precision (99.24%) and recall (99.42%), which

verifies its robustness in different network environments. However, in the face of the extremely imbalanced NF-BoT-IoT dataset, although the model maintains high accuracy (99.66%), the missed detection rate rises to 6.25%, which exposes the generalization bottleneck under class imbalance. In the future, data augmentation (such as oversampling) and dynamic loss function optimization should be used to further improve the sensitivity to rare samples, and fine-grained feature engineering should be combined to capture low-frequency attack patterns, so as to comprehensively strengthen the practicability of the model in complex scenarios.



(a) Confusion matrix of NF-UNSW-NB15-v2

(b) Confusion matrix of NF-ToN-IoT

(c) Confusion matrix of NF-BoT-IoT

(d) Confusion matrix of NF-CSE-CIC-IDS2018-v2

Fig. 4 Binary confusion matrix

In conclusion, the proposed hybrid model, which combines GCN and GraphSAGE, demonstrates exceptional overall performance across four datasets in the binary classification task, outperforming baseline models such as XGBoost, E-GraphSAGE, and Anomal-E on four evaluation criteria. This thoroughly demonstrates the method's high reliability and practicability as well as its ability to differentiate between attack and normal traffic in anomaly traffic detection.

4.5 Results of Multi-class Classification

In the multi-classification task, we also conducted experiments on the above four data sets to make a comparison of the classification accuracy of different methods. As presented in Table 5, proposed approach achieved the highest accuracy on all datasets.

For the NF-UNSW-NB15-v2 dataset, the HGS-ATD obtains an accuracy of 0.9947, this significantly outperforms the three baseline models. As shown in Fig. 5, for the classification of attack types such as exploits and fuzzers, the accuracy is close to or at 100%, indicating that the model effectively captures the unique features of these attack types, distinguishing them from other traffic. This further proves the powerful recognition ability of the HGS-ATD in dealing with various attack patterns.

Table 5 Results of multi-class classification experiments

Model	NF-UNSW-NB15-v2 accuracy	NF-ToN-IoT accuracy	NF-BoT-IoT accuracy	NF-CSE-CIC-IDS2018-v2 accuracy
XGBoost	0.9876	0.7218	0.8362	0.9898
E-graphSAGE	0.9833	0.4597	0.8144	0.9833
Anomal-E	0.9813	0.6345	0.8239	0.9820
HGS-ATD	**0.9947**	**0.9919**	**0.9283**	**0.9929**

Note: Data in bold is the one with the best performance.

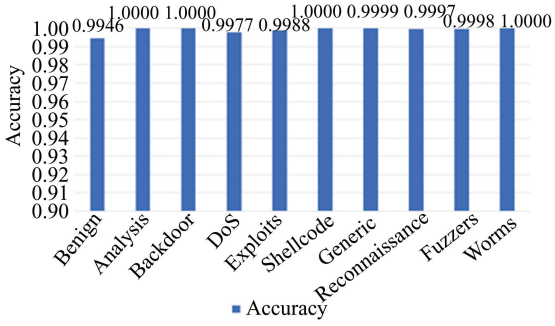


Fig. 5 Multi-class classification results of NF-UNSW-NB15-v2

The HGS-ATD performs exceptionally well for the NF-ToN-IoT dataset, with an accuracy of 0.9919, significantly outperforming other models. The accuracy was 0.7218 for XGBoost, 0.4597 for E-graphSAGE, and 0.6345 for Anomal-E. These results demonstrate how well the model can categorize different kinds of attack traffic in this multi-classification challenge. As shown in Fig. 6, the model achieves exceptionally high classification accuracy for attack types such as DDoS and injection, with minimal misclassifications. This can be attributed to the model's ability to recognize complicated traffic patterns in the dataset, as well as their correlations with attack signatures, which is the reason for its better performance.

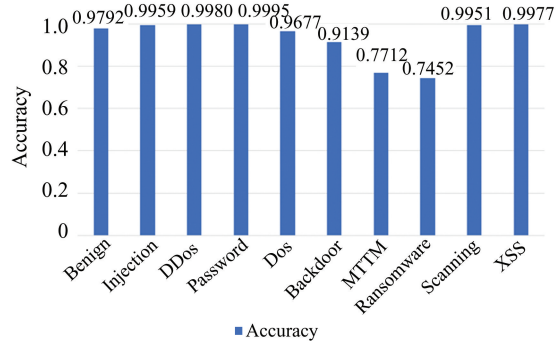


Fig. 6 Multi-class classification results of NF-ToN-IoT

The HGS-ATD obtains an accuracy of 0.9283 for the NF-BoT-IoT dataset, outperforming XGBoost (0.8362), E-graphSAGE (0.8144), and Anomal-E (0.8239). Even though the dataset has notable class imbalances that may impair the model's capacity to classify data, the HGS-ATD still manages to maintain high accuracy. This highlights its strength in handling datasets with unbalanced classes. The model effectively learns the patterns associated with different attack types, especially in the case of reconnaissance and DDoS attacks, and has excellent classification effect. As shown in Fig. 7, although the model is less accurate when dealing with categories with a small sample size (such as Theft), this may be due to the fact that there is less data for these categories, resulting in the model not learning their features adequately.

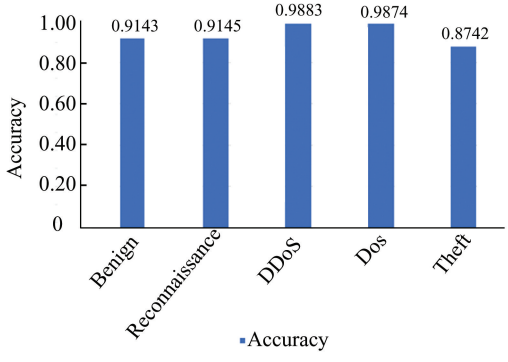


Fig. 7 Multi-class classification results of NF-BoT-IoT

Using the NF-CSE-CIC-IDS2018-v2 dataset, the suggested method once more shows a high accuracy of 0.9929, slightly higher than XGBoost (0.9898), E-graphSAGE (0.9833), and Anomal-E (0.9820). As demonstrated in Fig. 8, the model performs particularly well in identifying various attacks, including DDOS attack-HOIC and DoS attack-Hulk, maintaining a high level of accuracy. This highlights the model's strong generalization capability, enabling it to adapt effectively to diverse attack patterns.

On all four datasets, the HGS-ATD shows a much higher accuracy than the baseline models in the multi-classification task. This clearly indicates that the technique not only successfully distinguishes between normal and attack traffic but also excels in accurately identifying various types of attacks. Its robust

generalization capabilities allow it to perform well on different datasets, and it displays a strong aptitude for recognizing and classifying complex attack patterns. It provides solid support for anomaly traffic detection in network security, offering a more precise way to identify and address different types of network attacks.

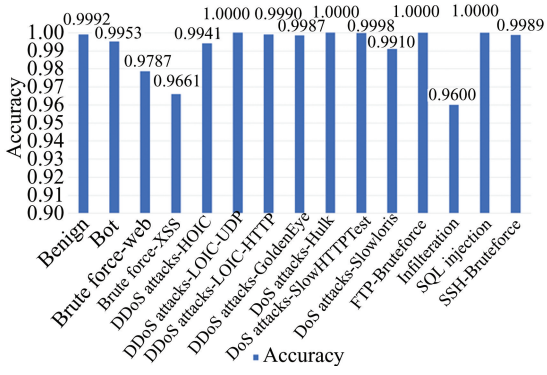


Fig. 8 Multi-class classification results of NF-CSE-CIC-IDS2018-v2

4.6 Discussion of Results

The experimental results demonstrate the practical significance of the HGS-ATD framework. In the actual network security scenario, the model has high accuracy and F1-score, and can accurately identify normal traffic and attack traffic in large-scale network traffic monitoring. In enterprise networks, for example, this helps to detect potential network attacks such as data theft and DDoS attacks in a timely manner. By accurately identifying these attacks, corresponding protective measures can be taken in time to protect the integrity of sensitive information and network systems of enterprises. In smart home systems, where numerous devices such as cameras, sensors, and smart appliances are connected to each other, the model can detect abnormal traffic patterns.

The edge-based dynamic batching strategy reduces the memory overhead and can be deployed in real-time monitoring systems for large-scale networks, such as cloud infrastructure or IoT ecosystems. In multi-class classification tasks, the proposed model's precise identification of diverse attack types offers critical support for network security management. Network security administrators can quickly identify attack types based on the output of the model, so as to formulate targeted defense strategies. Greatly improve the efficiency and effect of network security management. For example, when the model detects a DDoS attack, the administrator can adjust the network traffic policy in time to block malicious traffic and protect network services.

However, the framework exhibits certain limitations, particularly in handling imbalanced datasets, as shown in the NF-BoT-IoT dataset, where the sample size for some attack types, such as theft, is small, the ability of the model to identify these rare attack types is insufficient. Although the overall accuracy of the model is high, misclassification or missed detection of rare attack types may bring potential security risks to the network. In real-world networks, these undetected attacks can lead to data leakage or system failure.

In addition, the model currently focuses on static traffic feature analysis. In the real network environment, network traffic is highly dynamic, its traffic and patterns are constantly changing, and new attack techniques are constantly emerging. Attacks typically involve rapid changes in traffic behavior, such as sudden bursts of traffic or abnormal fluctuations in data transmission rates. However, this model does not fully consider the time series characteristics and dynamic change rules of network traffic. This can lead to missed detections or false positives when dealing with attacks that exhibit dynamic behavior. For example, for stealth attacks that exploit sudden traffic changes to evade detection, the model may not be able to accurately identify them in time.

To address these limitations, several directions for improvement and suggestions for future research can be proposed. To improve the recognition ability of the model for rare attack types, data augmentation techniques can be applied. Oversampling methods can increase the number of samples for rare attack types, and Generative Adversarial Networks (GAN) can generate synthetic samples that are similar to real-world data, enabling the model to learn more comprehensive features. In addition, adversarial training can be introduced. By adding adversarial samples in the training process, the model can learn to resist adversarial attacks and improve its robustness.

Regarding the adaptation of the model to dynamic traffic characteristics, future research could explore the combination of time series analysis methods. LSTM network and its variants can be integrated into this model to capture the time-series characteristics of network traffic. In this way, the model can better adapt to the dynamic changes of the network environment and effectively detect attacks with

changes in traffic patterns.

5 Conclusions and Future Work

This paper proposes a traffic anomaly detection method based on a hybrid model of GCN and GraphSAGE. By combining the global topology awareness of GCN and the local dynamic aggregation of GraphSAGE, as well as a novel edge-based batch processing strategy, it can effectively capture node edge relationships and complex traffic patterns. This paper implements scalable training of large-scale graphs, and compares it with traditional baseline models such as XGBoost, E-graphSAGE and Anomal-E on multiple datasets, and verifies its advantages in key indicators such as accuracy and recall. In particular, it shows strong advantages in dealing with the identification of multiple attack types, the accurate classification of attack traffic and the processing ability of unbalanced data sets, which proves its effectiveness in the field of network security.

Despite the satisfactory results, the present study has some limitations that suggest several directions for future research. Firstly, current methods mainly focus on the static feature analysis of traffic. In order to enhance the ability of the model to identify attacks in real time, future research should combine time series data and dynamic traffic characteristics, such as introducing GATv2 dynamic graph attention mechanism. In addition, although HGS-ATD has a good performance on imbalanced datasets, for some attack types (such as categories with a small number of samples), the model still has a certain miss rate. In the future, data augmentation and adversarial training can be used to further improve the recognition ability of the model for rare attack types. Finally, with the evolution of attack methods, the characteristics of attack traffic are also changing. Transfer learning techniques can be explored to further improve the adaptability of HGS-ATD model in different network environments. For example, techniques such as adversarial domain adaptation[56] can be used to align the feature distribution between the source domain and the target domain to reduce the annotation cost in new environments. The knowledge obtained in the pre-trained model can quickly adapt to the new network environment and traffic patterns, and improve its generalization ability and robustness.

In summary, the approach presented in this paper demonstrates significant potential for network anomaly traffic detection. In the future, its performance and applicability can be further improved by optimizing the model structure, enhancing the data set and combining with emerging technologies, so as to provide more effective technical support for network security protection.

References

[1] Liao H J, Lin C H R, Lin Y C, et al. Intrusion detection system: A comprehensive review. Journal of Network and Computer Applications, 2013, 36(1): 16-24. DOI: 10.1016/j.jnca.2012.09.004.

[2] Ioulianou P, Vasilakis V, Moscholios I, et al. A signature-based intrusion detection system for the internet of things. Information and Communication Technology Forum. White Rose Research Online. 2018. https://eprints.whiterose.ac.uk/133312/.

[3] Koch R, Golling M, Rodosek G D. Behavior-based intrusion detection in encrypted environments. IEEE Communications Magazine, 2014, 52(7): 124-131. DOI: 10.1109/MCOM.2014.6852093.

[4] Deng X, Zhu J, Pei X, et al. Flow topology-based graph convolutional network for intrusion detection in label-limited IoT networks. IEEE Transactions on Network and Service Management, 2022, 20(1): 684-696. DOI: 10.1109/TNSM.2022.3213807.

[5] Gupta B B, Joshi R C, Misra M. Defending against distributed denial of service attacks: issues and challenges. Information Security Journal: A Global Perspective, 2009, 18(5): 224-247. DOI: 10.1080/19393550903317070.

[6] Firdous S N, Baig Z, Valli C, et al. Modelling and evaluation of malicious attacks against the IoT MQTT protocol. Proceedings of the 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). Piscataway: IEEE, 2017: 748-755.

[7] Popoola S I, Ande R, Adebisi B, et al. Federated deep learning for zero-day botnet attack detection in IoT-edge devices. IEEE Internet of Things Journal, 2021, 9(5): 3930-3944. DOI: 10.1109/JIOT.2021.3100755.

[8] Ghafir I, Prenosil V. Advanced persistent threat attack detection: an overview. International Journal of Computer Network and Information Security, 2014, 4(4): 5054. DOI: 10.1093/chromsci/38.5.211

[9] Wu Z, Pan S, Chen F, et al. A comprehensive survey on graph neural networks. IEEE transactions on Neural Networks and Learning Systems, 2020, 32(1): 4-24. DOI: 10.1109/TNNLS.2020.2978386.

[10] Liu X, Su Y, Xu B. The application of graph neural network in natural language processing and computer

vision. 2021 3rd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI). Piscataway: IEEE, 2021: 708-714. DOI: 10.1109/MLBDBI54094.2021.00140.

[11] Sun G, Zhang C, Woodland P C. Tree-constrained pointer generator with graph neural network encodings for contextual speech recognition. arXiv preprint. 2022: arXiv: 2207.00857. DOI: 10.48550/arXiv.2207.00857.

[12] Do D P, Kim T, Na J, et al. D3T: Distinctive dual-domain teacher zigzagging across RGB-thermal gap for domain-adaptive object detection. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Computer Vision Foundation (CVF), 2024: 23313-23322.

[13] Duong M T, Lee S, Hong M C. DMT-Net: Deep multiple networks for low-light image enhancement based on Retinex model. IEEE Access, 2023, 11: 132147-132161. DOI: 10.1109/ACCESS.2023.3336411.

[14] Zhong Z, Li C T, Pang J. Hierarchical message-passing graph neural networks. Data Mining and Knowledge Discovery, 2023, 37(1): 381-408. DOI: 10.1007/s10618-022-00890-9.

[15] Lo W W, Layeghy S, Sarhan M, et al. E-graphsage: A graph neural network based intrusion detection system for IoT. NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium. Piscataway: IEEE, 2022: 1-9. DOI: 10.1109/NOMS54207.2022.9789878.

[16] Yang Y, Guan Z, Li J, et al. Interpretable and efficient heterogeneous graph convolutional network. IEEE Transactions on Knowledge and Data Engineering, 2021, 35(2): 1637-1650. DOI: 10.1109/TKDE.2021.3101356.

[17] Kharal A, Bokhary S A U H, Saeed M M. Matrix manipulations and tridiagonal structures new schemes in secure data encryption. Preprints. org, Basel: MDPI, 2024: 2024102503. DOI: 10.20944/preprints202410.2503.v1.

[18] Ahmad R, Alsmadi I, Alhamdani W, et al. Zero-day attack detection: a systematic literature review. Artificial Intelligence Review, 2023, 56(10): 10733-10811. DOI: 10.1007/s10462-023-10437-z.

[19] Dahiya A, Gupta B B. A reputation score policy and Bayesian game theory based incentivized mechanism for DDoS attacks mitigation and cyber defense. Future Generation Computer Systems, 2021, 117: 193-204. DOI: 10.1016/j.future.2020.11.027.

[20] Muñoz D C, Valiente A C. A novel botnet attack detection for IoT networks based on communication graphs. Cybersecurity, 2023, 6: 33. DOI: 10.1186/s42400-023-00169-6.

[21] Busch J, Kocheturov A, Tresp V, et al. NF-GNN: network flow graph neural networks for malware detection and classification. Proceedings of the 33rd International Conference on Scientific and Statistical Database Management. New York: ACM, 2021: 121-132. DOI: 10.1145/3468791.3468814.

[22] Zhao J, Liu X, Yan Q, et al. Multi-attributed heterogeneous graph convolutional network for bot detection. Information Sciences, 2020, 537: 380-393. DOI: 10.1016/j.ins.2020.03.113.

[23] Pujol-Perich D, Suárez-Varela J, Cabellos-Aparicio A, et al. Unveiling the potential of graph neural networks for robust intrusion detection. ACM SIGMETRICS Performance Evaluation Review, 2022, 49(4): 111-117. DOI: 10.1145/3543146.3543171.

[24] Zhou J, Xu Z, Rush A M, et al. Automating botnet detection with graph neural networks. arxiv preprint. 2020: arxiv: 2003.06344. DOI: 10.48550/arXiv.2003.06344.

[25] Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs. NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems. New York: ACM, 2017: 1025-1035.

[26] Chandramohan D, Dumka A, Jayakumar L. 2M2C-R2ED: Multi-metric cooperative clustering based routing for energy efficient data dissemination in green-VANETs. Technology and Economics of Smart Grids and Sustainable Energy, 2020, 5(1): 15. DOI: 10.1007/s40866-020-00086-4.

[27] Chang L, Branco P. Embedding residuals in graph-based solutions: the E-ResSAGE and E-ResGAT algorithms. A case study in intrusion detection. Applied Intelligence, 2024, 54: 6025-6040. DOI: 10.1007/s10489-024-05404-2.

[28] Bao H, Chen M, Huo Y, et al. Network traffic intrusion detection strategy based on E-GraphSAGE and LSTM. Advanced Intelligent Computing Technology and Applications: 20th International Conference, ICIC 2024. Singapore: Springer Nature Singapore, 2024: 25-37. DOI: 10.1007/978-981-97-5606-3_3.

[29] Ji X, Meng Q. Traffic classification based on graph convolutional network. 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA). Piscataway: IEEE, 2020: 596-601. DOI: 10.1109/AEECA49918.2020.9213630.

[30] Ying R, He R, Chen K, et al. Graph convolutional neural networks for web-scale recommender systems. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mmining. New York: ACM, 2018: 974-983. DOI: 10.1145/3219819.3219890.

[31] Tran D H, Park M. Graph embedding for graph neural network in intrusion detection system. 2024 International Conference on Information Networking (ICOIN). Piscataway: IEEE, 2024: 395-397. DOI: 10.1109/ICOIN59985.2024.10572124.

[32] Deng P, Huang Y. Edge-featured multi-hop attention graph neural network for intrusion detection system. Computers & Security, 2025, 148: 104132. DOI: 10.1016/j.cose.2024.104132.

[33] Du Y, Li Y, Cheng P, et al. UGL: A comprehensive

hybrid model integrating GCN and LSTM for enhanced intrusion detection in UAV controller area networks. Computer Networks, 2025,262: 111157. DOI:10.1016/j.comnet.2025.111157.

[34] Veli kovi P, Cucurull G, Casanova A, et al. Graph attention networks. arXiv preprint. 2017: arXiv: 1710.10903.DOI:10.48550/arXiv.1710.10903.

[35] Jahin M A, Soudeep S, Mridha M F, et al. CAGN-GAT fusion: A hybrid contrastive attentive graph neural network for network intrusion detection. arXiv preprint. 2025: arXiv:2503.00961. DOI:10.48550/arXiv.2503.00961.

[36] Scarselli F, Gori M, Tsoi A C, et al. The graph neural network model. IEEE Transactions on Neural Networks, 2008, 20(1): 61-80. DOI:10.1109/TNN.2008.2005605.

[37] Yin C, Zhu Y, Fei J, et al. A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access, 2017, 5: 21954-21961. DOI:10.1109/ACCESS.2017.2762418.

[38] Vinayakumar R, Soman K P, Poornachandran P. Applying convolutional neural network for network intrusion detection. 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). Piscataway:IEEE, 2017: 1222-1228. DOI: 10.1109/ICACCI.2017.8126009.

[39] Ahmad Z, Shahid Khan A, Wai Shiang C, et al. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies, 2021, 32(1):e4150.DOI:10.1002/ett.4150.

[40] Zhang S, Tong H, Xu J, et al. Graph convolutional networks: a comprehensive review. Computational Social Networks, 2019, 6(1): 1-23. DOI:10.1186/s40649-019-0069-y.

[41] Xiao Y, Xu J, Yang J, et al. Determinate node selection for semi-supervised classification oriented graph convolutional networks. International Journal of Bio-Inspired Computation, 2025, 25(1): 1-10. DOI: 10.1504/ijbic.2025.143648.

[42] Sarhan M, Layeghy S, Moustafa N, et al. 2021. Netflow datasets for machine learning-based network intrusion detection systems. In: Deze Z, Huang H, Hou R, et al. (eds) Big data technologies and applications. BDTA WiCON 2020. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Cham:Springer, Cham, 2021,371:117-135. DOI:10.1007/978-3-030-72802-1_9.

[43] Moustafa N, Slay J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). 2015 Military Communications and Information Systems Conference (MilCIS). Piscataway: IEEE, 2015: 1-6. DOI:10.1109/MilCIS.2015.7348942.

[44] Sharafaldin I, Lashkari A H, Ghorbani A A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018). Setúbal: SCITEPRESS - Science and Technology Publications, Lda. 2018: 108-116. DOI: 10.5220/0006639801080116.

[45] Claise B. RFC 3954: Cisco systems NetFlow services export version 9. 2004: RFC 3954. DOI: 10.17487/RFC3954.

[46] Sarhan M, Layeghy S, Portmann M. Towards a standard feature set for network intrusion detection system datasets. Mobile Networks and Applications, 2022, 27(1): 357-370. DOI:10.1007/s11036-021-01843-0.

[47] Chen T, Guestrin C. XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York:ACM, 2016: 785-794. DOI:10.48550/arXiv.1603.02754.

[48] Caville E,Lo W W, Layeghy S, et al. Anomal-E: A self-supervised network intrusion detection system based on graph neural networks. Knowledge-Based Systems, 2022, 258: 110030. DOI:10.1016/j.knosys.2022.110030.

[49] Sarhan M, Layeghy S, Portmann M. Evaluating standard feature sets towards increased generalisability and explainability of ML-based network intrusion detection. Big Data Research, 2022, 30: 100359. DOI:10.1016/j.bdr.2022.100359.

[50] Powers D M W. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. arXiv preprint. 2020:arXiv:2010.16061. DOI: 10.48550/arXiv.2010.16061.

[51] Kingma D P, Ba J. Adam: A method for stochastic optimization. In: Bengio Y, LeCun Y (Eds.), 3rd International Conference on Learning Repre sentations, ICLR 2015. San Diego, 2015:arXiv:1412.6980. DOI:10.48550/arXiv.1412.6980.

[52] Van Rossum G, Jr Drake F L. Python Tutorial. Amsterdam:CWI, 1995.

[53] Paszke A, Gross S, Massa F, et al. Pytorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems 32 (NeurIPS 2019), 2019:32.

[54] Fey M, Lenssen J E. Fast graph representation learning with PyTorch Geometric. arXiv preprint. 2019: arXiv: 1903.02428. DOI:10.48550/arXiv.1903.02428.

[55] Wang M Y, Yu L F, Zheng D, et al. Deep graph library: Towards efficient and scalable deep learning on graphs. ICLR Workshop on Representation Learning on Graphs and Manifolds. 2019:arXiv:1909.01315. DOI:10.48550/arXiv.1909.01315.

[56] Ganin Y, Lempitsky V. Unsupervised domain adaptation by backpropagation. ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning. 2015,37: 1180-1189.