

Article

FN-GNN: A Novel Graph Embedding Approach for Enhancing Graph Neural Networks in Network Intrusion Detection Systems

Dinh-Hau Tran ¹  and Minhho Park ^{2,3,*} ¹ Department of Information and Telecommunication Engineering, Soongsil University, Seoul 06978, Republic of Korea; hautransns@soongsil.ac.kr² School of Electronic Engineering, Soongsil University, Seoul 06978, Republic of Korea³ Department of AI Convergence Security, Soongsil University, Seoul 06978, Republic of Korea

* Correspondence: mhp@ssu.ac.kr

Abstract: With the proliferation of the Internet, network complexities for both commercial and state organizations have significantly increased, leading to more sophisticated and harder-to-detect network attacks. This evolution poses substantial challenges for intrusion detection systems, threatening the cybersecurity of organizations and national infrastructure alike. Although numerous deep learning techniques such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and graph neural networks (GNNs) have been applied to detect various network attacks, they face limitations due to the lack of standardized input data, affecting model accuracy and performance. This paper proposes a novel preprocessing method for flow data from network intrusion detection systems (NIDSs), enhancing the efficacy of a graph neural network model in malicious flow detection. Our approach initializes graph nodes with data derived from flow features and constructs graph edges through the analysis of IP relationships within the system. Additionally, we propose a new graph model based on the combination of the graph neural network (GCN) model and SAGEConv, a variant of the GraphSAGE model. The proposed model leverages the strengths while addressing the limitations encountered by the previous models. Evaluations on two IDS datasets, CICIDS-2017 and UNSW-NB15, demonstrate that our model outperforms existing methods, offering a significant advancement in the detection of network threats. This work not only addresses a critical gap in the standardization of input data for deep learning models in cybersecurity but also proposes a scalable solution for improving the intrusion detection accuracy.

Keywords: intrusion detection system (IDS); graph neural network (GNN); deep learning; flow-based characteristic; feature engineering



Citation: Tran, D.-H.; Park, M. FN-GNN: A Novel Graph Embedding Approach for Enhancing Graph Neural Networks in Network Intrusion Detection Systems. *Appl. Sci.* **2024**, *14*, 6932. <https://doi.org/10.3390/app14166932>

Academic Editor: Juan-Carlos Cano

Received: 18 June 2024

Revised: 3 August 2024

Accepted: 5 August 2024

Published: 8 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In today's era, with the widespread use of the Internet, network systems are becoming increasingly vast and significantly more complex. Despite their benefits, these network systems also pose numerous risks that can cause harm to businesses and organizations. Indeed, cyberattacks are increasing both in quantity and sophistication, presenting a substantial challenge for protective systems such as intrusion detection systems (IDSs) and intrusion protection systems (IPSs). Network-based intrusion detection systems (NIDSs) [1] consistently play a pivotal and essential role in any business or organization's network system. NIDS relies on various characteristics to determine whether a network flow is normal or malicious. Different characteristics are considered depending on whether the detection mechanism is signature-based or anomaly-based. For signature-based detection [2], the system typically monitors characteristics such as known attack patterns, lists of malicious IP addresses and domains (blacklists), and the use of suspicious ports to identify unauthorized intrusions. For anomaly-based detection [3], characteristics such as traffic volume and frequency, unusual user or device behaviors, and flow attributes like duration,

packet size, and inter-arrival times are considered. As a sensitive shield, NIDS detects external threats or potential risks within the network system. In the face of these challenges, NIDS systems are experiencing significant limitations in effectively detecting unknown attacks or zero-day attacks [4]. Indeed, the primary detection mechanisms of NIDS such as signature-based or anomaly-based detection are easily circumvented by modern attacks or generate false alarms. Therefore, integrating various new techniques into NIDS to enhance the detection performance is always considered an urgent requirement for modern network systems.

Recently, machine learning (ML) and deep learning (DL) have been employed in various fields, such as image processing [5,6], storage systems [7–9], wireless communication [10], and cybersecurity [11]. Many deep learning approaches, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and traditional multi-layer perceptrons (MLPs), have also been applied to NIDS to enhance the network monitoring efficiency. However, these techniques exhibit limited effectiveness when applied to datasets comprising network flows due to a mismatch between the models and the type of data being monitored by the IDS. Conventional DL models are often trained on flat data structures, such as vectors or grid data, rendering them incapable of exploiting the complex structures of network flows. The information embedded in these complex structures is crucial for detecting advanced persistent threats (APTs) or zero-day attacks. Furthermore, the employed ML techniques focus on analyzing individual network flows, neglecting their inter-dependencies, as seen in [12,13].

Among the various research techniques in deep learning, graph neural network (GNN) models are particularly well suited for analyzing traffic data. GNN is a subclass of deep learning techniques designed to operate on graph-structured data, consisting of ‘nodes’ and the connections between them, called ‘edges’. This type of structure is well suited for representing relationships in various domains, such as social networks, transportation networks, and molecular structures. Similarly, network traffic, which consists of multiple flows, can naturally be represented as graph data. Moreover, the mechanism of GNN models to aggregate the information from neighboring nodes allows them to exploit the complex structures present in network data. Instead of relying on predefined features, GNNs can learn relevant features directly from the data. This reduces the dependency on manual feature engineering and enables the model to discover intricate patterns that might indicate malicious activity. The information contained in these patterns is crucial for detecting APT and zero-day attacks. Thus, GNN can significantly improve the performance of NIDS by utilizing the inherent graph structure of network traffic.

However, GNN-based IDS still has not achieved the desired reliability and stability level, as the model’s input data has not been optimized before training. Previous research has mainly focused on creating graph data based on network topology or only using a component of the graph data, such as nodes or edges. Meanwhile, both nodes and edges are crucial elements of the graph that need to be simultaneously exploited for the model to learn contextually relevant information. Therefore, in this study, we propose a model called the flow-node graph neural network (FN-GNN) to design graph data from network flows. In this proposed model, the graph data are formed using a completely new approach. Specifically, the set of the most important features of flows is represented as nodes. Simultaneously, the edges of the graph are formed by utilizing the correlation between flows that share the same source IP address. This approach helps generate graph data that already contain information about the relationships within them. Additionally, this method preserves the maximum amount of network information since it considers the entire dataset as a whole rather than treating each data point independently. Thus, this approach is the most appropriate and effective for network flow data.

Furthermore, our research also proposed a new graph model architecture based on a combination of existing models, GCN and SAGEConv. This combination helps the new model overcome the limitations of previous models and significantly enhances the performance of network attack detection. We implemented the model on two benchmark

datasets, CIC-IDS2017 and UNSW-NB15. The experimental results show that the accuracy of the proposed model reached 99.76% for the CIC-IDS2017 dataset and 98.65% for the UNSW-NB15 dataset.

We summarize our contributions as follows:

- We proposed the FN-GNN model, a novel approach to represent the network flow data into nodes and edges in graph data.
- We proposed a new graph model architecture combined with the GCN and SAGEConv models, which significantly improve the performance of the intrusion detection system.
- The proposed model was applied to two standard datasets and we supplied the simulation results to prove the effectiveness of the proposed model.

The remainder of this paper is organized as follows. Section 2 reviews the relevant literature and related work. Section 3 provides background information necessary for understanding the research, including an overview of the NIDS system and GNN models. Our proposed FN-GNN model is introduced in Section 4. Section 5 describes the experimental setup, detailing the datasets and evaluation methodology used to assess the model's performance. Section 6 presents and analyzes the experimental results, evaluating the effectiveness of the FN-GNN model compared to the existing method. Finally, Section 7 concludes the paper while also discussing limitations and outlining potential future work.

2. Related Works

In this section, we focus on presenting some recent studies, which are NIDS models based on graph neural networks. These are the most common approaches to using graph models to represent data collected from network traffic. Additionally, we also highlight the differences between the proposed methods and those previously existing.

A common approach to applying GNN models to NIDS is to represent network flow data as graphs, with nodes identified by hosts or devices in the network, while the remaining data are placed into edges. For instance, ref. [14] represented flow data in a graph format, with network traffic flows mapped to the graph edges and the endpoints as nodes, whereas paper [15] proposed a method to represent network flows as a graph, where each node includes flow features in sort of -tuple (IP src, IP dst, port, protocol, request, response). Another graph representation approach was also presented in paper [16], where all data are represented as nodes of the graph. Specifically, the authors introduced a heterogeneous graph constructed from network flows, with three nodes created for each flow: the source host node, the flow node, and the destination host node. Paper [17] even proposed a model with graph data generated without including any flow or node features but only considering the network's topology. They ignored the edge features and initialized the node features with vectors of all ones. We recognized the commonality among these studies that the graphs are constructed based on the inherent network structure, thus resulting in graph data resembling the topology of computer networks. This approach did not allow the model to fully exploit the relationships between the generated flows in the network. This can be explained from the perspective of the GNN model's concept. According to the theory of GNN, nodes that are similar to each other are often connected through edges, while the network system's topology structure cannot generalize that relationship. This is one of the main differences between our method and previous methods. In this study, the graph data are generated by considering the relationships between flows, with each node in the graph representing the characteristics of a flow. Thus, the nodes in this graph represent their inherent similarities when connected by edges. This approach aligns with the theory of graph data representation we discussed earlier.

While approaches like the above primarily classify flows based on edge features, some studies [18–21] consider both node and edge features. However, leveraging edge features is negligible as they mainly focus on node features and only use edge features to improve the passing of messages between nodes. Moreover, in papers [22,23], threats to the system were not extensively considered, as the proposed models only accounted for packets and flows transported between specific endpoints within the network. Likewise, the studies [23,24]

concentrate solely on individual flows, resulting in these models exhibiting limitations when attempting to classify network flows that were not presented in the training data. Meanwhile, our model effectively exploits the correlation between flows appearing between any two endpoints in the network system. For real-world applications, the proposed model can detect malicious flows in general, rather than solely focusing on a few types of attacks as indicated in previous studies.

Additionally, the selection of features contained within the network flow for the GNN model also significantly impacts the model's effectiveness. Indeed, each network flow comprises a set of features that characterize the flow, but not all of them are considered for the identification and classification of that flow. Only specific features are selected for training the classification model based on certain methods. However, to the best of our knowledge, there have been no specific studies utilizing feature selection methods for NIDS models based on graph neural networks. Therefore, in this paper, we apply a feature selection method based on random forest regression, which can assess the necessity of each feature according to the 'important weight' index. This feature selection method is detailed in Section 5.

Specifically, this research proposes a modified GCN model to address the limitations of the current GCN models. The GCN model is one of the most popular and effective models for node classification. However, this model is not very efficient when handling large graph data. Indeed, the GCN model requires the storage of the entire adjacency matrix with the corresponding features in memory, which makes it unusable for very large graphs [25]. In the paper [26], the authors used the GCN model provided in Figure 1 to classify malicious network flows. Although this model is designed to be simple with two GCN layers and one fully connected layer, it still requires a considerable amount of memory and computation time. Furthermore, the predictive performance of this model is not outstanding, with a detection rate of only around 92% on the CIC-IDS2017 dataset. Meanwhile, the papers [17,27] propose GCN models for botnet detection in network systems. To capture the dependencies in large botnet architectures, the authors used 12 GCN layers. Constructing a GCN model with such a large number of layers also makes the model prone to over-smoothing [28], reducing its prediction effectiveness. Our model is proposed based on a combination of the GCN model and the SAGEConv model. The sampling mechanism of the proposed model helps to improve efficiency in processing large graph data.

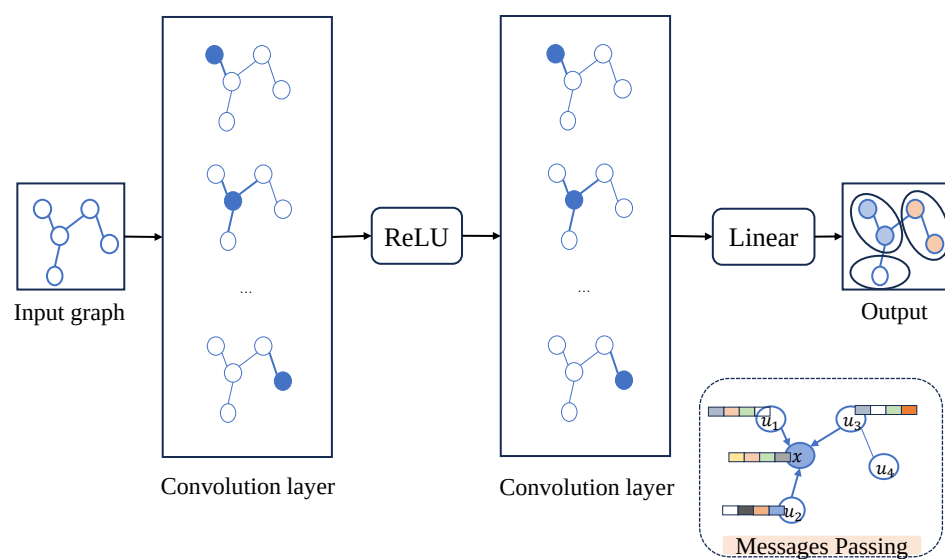


Figure 1. GCN model.

3. Background

3.1. Intrusion Detection Systems

An IDS is a widely used network security technology employed across various enterprise and organizational networks. As the name suggests, an IDS is a system or software established in a network system that has the role of monitoring the traffic and immediately sending alerts to administrators if malicious activities or policy violations are detected. The most optimal and common position for an IDS to be placed is “behind-the-firewall” (strategic points) because this placement enables the IDS to have high visibility into incoming network traffic while ensuring that it does not intercept traffic between users and the network.

The NIDS, which is shown in Figure 2 is one of the two main types of IDS and it is strategically positioned within the network to analyze the traffic originating from all devices on the network. Traditional NIDS systems use two main attack detection methods: signature-based and anomaly-based methods. The signature-based method uses pre-defined attack signatures through rule sets to effectively identify known threats and malicious patterns. This method helps the system respond accurately and promptly to known attacks, but it is largely ineffective at detecting new attacks or other variations of known attacks. Additionally, the anomaly-based method establishes the baselines of normal network behavior, allowing it to detect anomalous activities that do not match regular patterns, thereby identifying them as suspicious behaviors and sending alerts to the administrator. This method still does not guarantee reliability, as it still faces limitations such as high rates of false positives and false alarms. To overcome the limitations of traditional NIDS systems, many deep learning techniques, such as CNN [29,30], RNN [31], and GNN [32] were applied to NIDS to improve the accuracy and performance of detecting cyber-attacks.

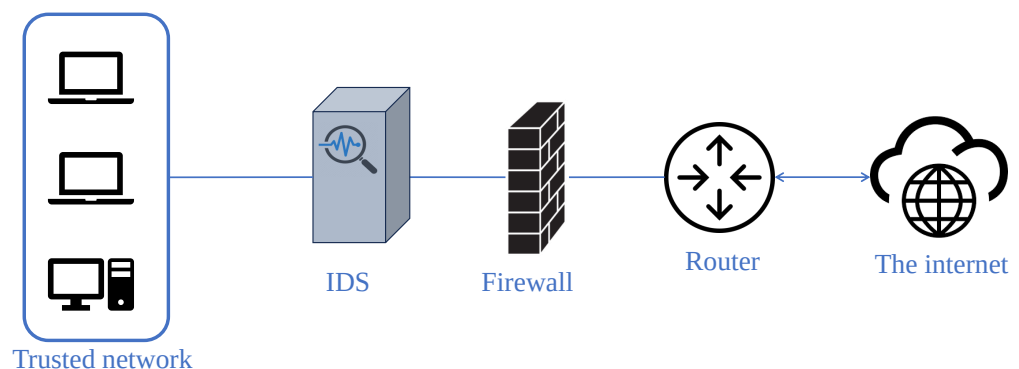


Figure 2. Diagram of the IDS model.

3.2. Graph Neural Network

In recent years, GNN has gradually become prominent in the field of deep learning because of the flexibility and high efficiency they bring. GNN is a subclass of deep learning techniques that work with graph data. Unlike image or text data, graph data allow GNN models to take advantage of the inherent graph structure of many real-world non-Euclidean data, such as relationships in telephones, social networks, and molecules. A graph is created by nodes, and the connections between them are called edges. By effectively leveraging the correlation relationship among the components of the graph, the GNN model outperforms conventional DL models like CNN and RNN when handling non-Euclidean data. The objective of GNN is to learn an embedding state that encapsulates the information of the neighborhood for each node. This state is then utilized to generate the output. There are many different ways to perform deep learning on graphs, and the best approach for a particular problem depends on the data structure and desired output. Some types of tasks on GNN can be mentioned such as node classification, link prediction, and graph

classification. The most crucial concept of a graph neural network is the message-passing mechanism that is presented in Figure 3.

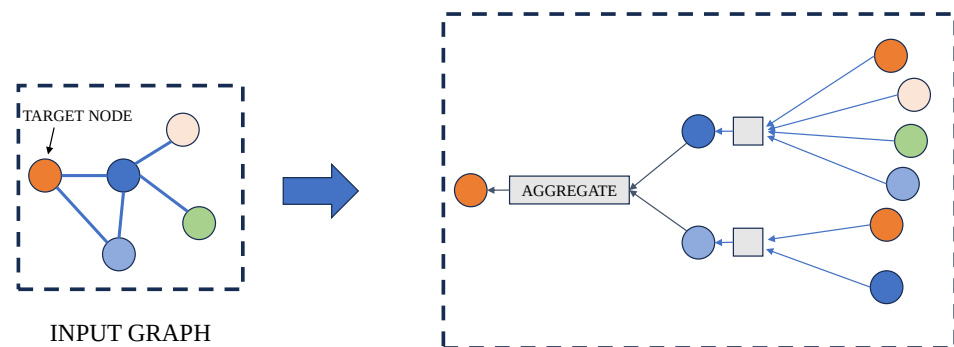


Figure 3. Message -passing mechanism in GNN.

The GNN propagates information across the graph through a series of message-passing steps. This mechanism allows the GNN layer to update the hidden state of each node from its neighborhood nodes. This process is repeated, in parallel, for all nodes in the graph and thus, the hidden state of the graph is also aggregated and updated continuously through each GNN layer. In the GNN model, the message-passing mechanism takes place through two stages: aggregation and update. Initially, information on neighboring nodes is compiled and sent to the node that needs to be updated as a ‘message’. Then, this information is updated for that node along with the information statuses it is storing in the previous layer. After passing through multiple GNN layers, the resulting output is the final embedded representation of the graph’s nodes. These embeddings are subsequently employed to address various tasks, including node classification, graph classification, and link prediction in different methods.

Observing the structure of a graph, we notice that nodes with similar features or properties are often connected. Therefore, the GNN exploits this fact to learn how and why specific nodes connect while others do not. This is why the message-passing mechanism is widely regarded as the most critical strength of GNNs.

3.3. Types of Graph Neural Network

There are several types of graph neural networks, but within the scope of this paper, we will only provide an overview of two related models: the GCN and the graph sample-aggregation (GraphSAGE) models.

3.3.1. GCN

The GCN [33] is one of the most basic graph neural networks designed to operate on graphs. As the name suggests, it is inspired by the CNN model. GCN can be understood as performing a convolution in the same way that traditional CNN performs a convolution-like operation when operating on images. Fundamentally, a GCN takes as input a graph together with a set of feature vectors where each node is associated with its feature vector. The GCN is then composed of a series of graph convolutional layers that iteratively transform the feature vectors at each node. The GCN layers use the message-passing mechanism previously mentioned to aggregate information from neighboring nodes and reflect it into the current node’s representation. This same procedure is carried out at every node. The output of each GCN layer serves as the input data for the subsequent GCN layer. Consequently, the graph data are transformed into new embedding through the layers, and the final neural network layer utilizes these embeddings to address tasks such as node classification and graph classification. The GCN model is described as shown in Figure 1.

The hidden states of nodes at each layer are made up of two consecutive processes: aggregation and update. This is where the idea of ‘convolutional’ comes into play. The hidden states of each GCN layer can be updated through the following formula:

$$\mathbf{H}^{(l)} = \sigma(\mathbf{A}\mathbf{H}^{(l-1)}\mathbf{W}^{(l-1)} + \mathbf{b}^{(l)}), \quad (1)$$

where

A: the adjacency matrix of the graph;

H: the node feature matrix;

W: the GCN layer's weight matrix;

b: the bias number;

σ : the activation function.

At each node updated in the l th layer, the information is updated based on the connections between neighboring nodes with that node, which can be seen as the 'mask' of a node. This mask plays a role similar to the concept of a kernel in a CNN model. The nodes of the graph are sequentially updated by sliding these 'masks' over each vertex and performing information aggregation right there. This aggregation is typically achieved by multiplying two matrices **H** and **W**, as outlined in the formula above. However, at its core, it still embodies the essence of 'convolution' in aggregating information from each node's neighbors. This is the reason behind the name of the GCN.

3.3.2. GraphSAGE

GraphSAGE [34] stands for graph sample and aggregate. It is a GNN model for large graphs, and was introduced for the first time in 2017. Unlike other models such as GCN and GAT that aggregate information from all neighboring nodes of a node, GraphSAGE pre-specifies the number of neighboring nodes at each node that is aggregated to update its embedding information. This aggregator method of GraphSAGE helps the model overcome the limitations of traditional GNN models when processing large graph data.

Based on the above idea, the message-passing mechanism in GraphSAGE includes two processes: neighborhood sampling and aggregation. The sampling operation is denoted by:

$$N_s(u) = \text{SAMPLE}(N(u), s), \quad (2)$$

where $N_s(u)$ represents the set of neighborhood samples of node u , with s denoting the number of nodes selected from the total number of neighboring nodes $N(u)$ of node u . By choosing s neighbors for each node, GraphSAGE helps the model significantly reduce the size of the computational graph and memory requirements, thereby reducing the space and time complexity of the algorithm. After specifying the number of neighbors for each node, GraphSAGE utilizes aggregation functions to synthesize information from them:

$$\mathbf{h}_{N_s(u)}^{(l)} = \text{AGG}(\{\mathbf{h}_v^{(l-1)}, \forall v \in N_s(u)\}), \quad (3)$$

where $\mathbf{h}_{N_s(u)}^{(l)}$ represents the information aggregated from the selected neighbors and AGG is the aggregation function. In GraphSAGE, many types of aggregation functions can be applied, including sum, mean-pooling, max-pooling, LSTM, and so on.

The aggregated data from that neighborhood are utilized to calculate and update the embedding for each node, akin to other GNN models:

$$\mathbf{h}_u^{(l)} = \sigma([\mathbf{h}_u^{(l-1)}, \mathbf{h}_{N_s(u)}^{(l-1)}] \cdot \mathbf{W}^{(l)}), \quad (4)$$

where $\mathbf{h}_u^{(l)}$ is the embedding of node u at layer l^{th} , calculated from embedding $\mathbf{h}_u^{(l-1)}$ of node u in the layer $(l-1)^{\text{th}}$ and the information aggregated from its neighbors $\mathbf{h}_{N_s(u)}^{(l-1)}$. **W** is the weight matrix used at layer l^{th} of the model.

In general, GraphSAGE can solve the limitations of GCN and GAT models when it works effectively with large graphs and fast training. However, through experiments, we can see that GraphSAGE does not significantly improve accuracy compared to other models.

4. Proposed Method

In prior research, the authors typically utilized nodes or edges to represent the features extracted from network traffic flows. However, given the capacity of GNN models to leverage both nodes and edges within graph data, this paper introduces a novel method for extracting features from traffic flows, encompassing both nodes and edges. Our proposed model is presented in Figure 4.

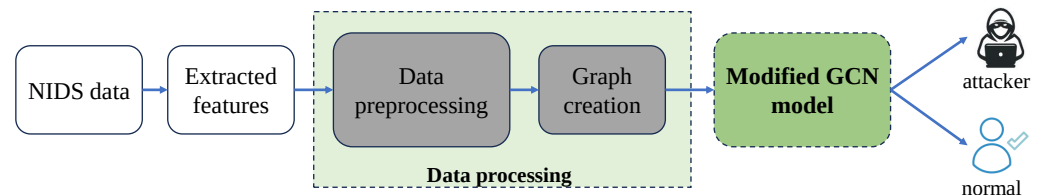


Figure 4. The overall view of the proposed model.

Within the model, flow data represent the communication exchanged between two computers or devices within the network system. These data are characterized by features such as source IP, source port, destination IP, destination port, protocol, etc. We employed the random forest regression algorithm to evaluate the impact and necessity of each feature based on the ‘important weight’ index. This allowed us to select a subset of relevant features from the data to be used in our proposed model. Subsequently, these selected features were used to construct the nodes and edges in the graph network representation, as illustrated in Figure 5.

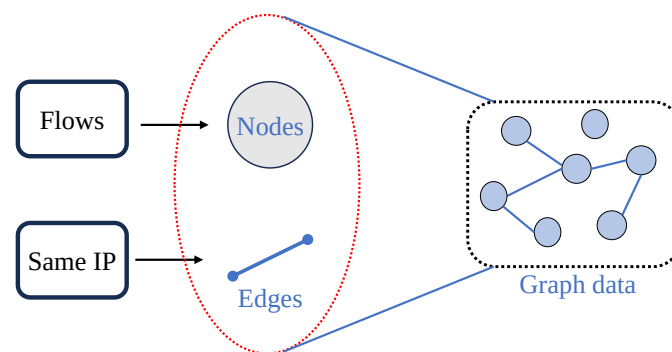


Figure 5. The data pre-processing steps.

This feature selection method helps our approach focus on meaningful information in the flow data and avoid distortions. By doing so, the proposed model can more effectively exploit the characteristic patterns in the data. This allows the model to achieve higher accuracy compared to other models that use only a few features from the flow data.

Next, we initialize a graph using the extracted features as nodes. For edge creation, we leverage the IP addresses presented in each flow of data. Specifically, edges were established between flows sharing the same source IP. The output of the pre-processing data is the graph data, which is used to feed the next GNN model.

After the preprocessing step, we obtained graph data. These data are used as input for the node classification model to find suspicious network flows in the network. In this study, we proposed a modified version of the GCN model to perform this classification task. It is presented in Figure 6 with two SAGEConv layers and one fully connected layer as the output layer. Batch normalization layers and ReLU activation functions are also used immediately after each SAGEConv layer. The softmax function at the end of the model helps generate the most efficient predictions based on the class with the highest probability output.

Specifically, the feature vector at each node is aggregated through each SAGEConv layer. This information is then normalized and non-linearized using BatchNorm and the ReLU function immediately after each SAGEConv layer, as shown in Figure 6. In the fully connected layer, the current feature vector of the nodes is transformed into vectors with dimensions equal to the number of classes to be classified. This transformation is achieved using flattening techniques and the weight matrix multiplication of this layer. Finally, the softmax activation function is applied to the vector at each node to generate classification probabilities. The result at each node is a probability vector where the sum of the distributions is equal to 1. The model classifies nodes based on the highest probability in this vector, corresponding to the one-hot encoding matrix presented in the above figure.

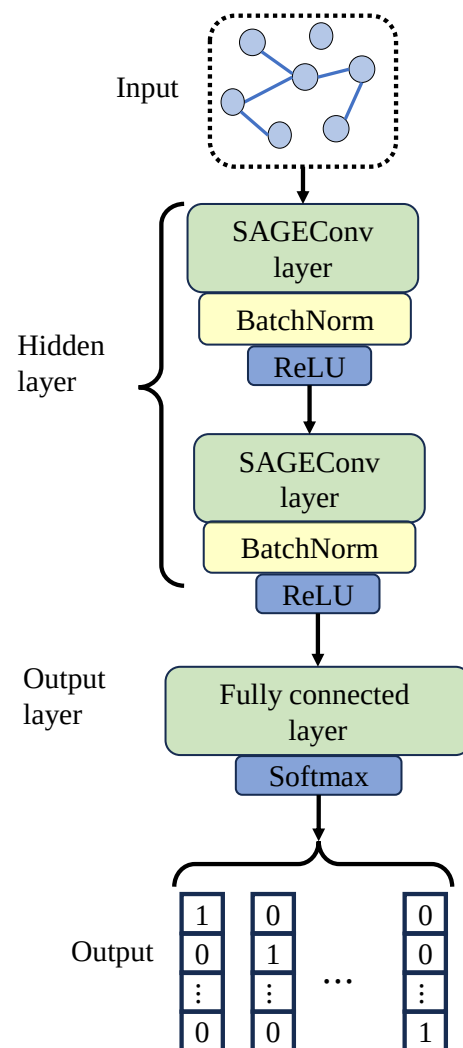


Figure 6. Diagram of the modified GCN model.

We proposed this model based on the idea of combining the GCN model with the SAGEConv module of the GraphSAGE model. GCN is the most popular model of GNN presented in Figure 1. In the GCN model, GraphConv layers play a key role in learning graph representations. These layers help the model effectively extract complex features and structural information through multiple convolution calculations. This architecture leads to the GCN model achieving high accuracy, but it has some limitations when working with large graph data. With large graphs, the information of each node is aggregated from all neighbors, making the data huge and causing system resource requirements and computing time to increase significantly. Furthermore, information taken from all these

neighbors may cause embedding nodes to tend to be similar. This phenomenon is called over-smoothing and reduces the accuracy of the model.

On the other hand, SAGEConv is a variation of the GraphSAGE model, as introduced in the previous section. It represents an improvement over GraphSAGE by employing a more expressive convolutional operator. Unlike GraphSAGE, SAGEConv utilizes the average of neighbor representations, normalized by the degree of each neighbor, as the aggregate representation. This enhancement enables SAGEConv to capture more fine-grained information about the graph's structure.

Our model is the result of combining the advantages of both the GCN and SAGEConv models. The main difference compared to the old model is that GraphConv layers are replaced by SAGEConv layers. This combination makes the model more suitable for training on large graph data while not sacrificing its accuracy and performance. To apply this model most effectively, choosing model parameters such as hidden units and learning rate appropriately helps the model achieve the best accuracy and stability. We conducted experiments on benchmark datasets using the proposed model and achieved superior results compared to previous methods. The detailed results and evaluation are presented in the next section.

5. Experiment

In this section, the paper outlines the datasets chosen for training and testing, details the evaluation criteria used in the experiments, and describes the experimental setup as well as the selection of parameters for our model.

5.1. Datasets

A dataset for the network intrusion detection system includes many network traffic flows combined with information about the network system, network devices, servers, and user behavior. Raw data are firstly collected by capturing the network traffic generated in the system through network devices such as routers and switches. They are then processed using specific techniques to create dataset flows. These datasets are especially important and necessary to evaluate malicious patterns and attacker behavior during cyber-attacks. Datasets play an important role in training deep learning models. It has a direct impact on the model's prediction performance. Therefore, it is necessary to choose quality datasets that are suitable for the model and the purpose to be achieved. In our experiment, CIC-IDS2017 and UNSW-NB15 datasets were used to train and evaluate the performance of the proposed model.

5.1.1. CIC-IDS2017 Dataset

CIC-IDS2017 [35] is an intrusion detection evaluation dataset created in 2017 by the University of Brunswick (UNB) and the Canadian Institute of Cybersecurity (CIC). Generating realistic background traffic was a top priority when this dataset was built. CICIDS2017 dataset contains 2,830,743 flows, including benign and most up-to-date common attack flows. To generate benign traffic, they used the previously proposed B-Profile system [36], which describes the abstract behavior of human interactions and generates natural background traffic. The B-Profile system is described in Figure 7 below. In this dataset, they abstracted the behavior of 25 users based on HTTP, HTTPS, FTP, SSH, and email protocols.

On the other hand, many different tools are used to create attack flows based on simulating common attacks such as Brute Force attacks, PortScan attacks, Denials of Service, Distributed Denials of Service, and so on. At the end of the above process, the obtained data include benign traffic and 7 different types of attacks, with a total of 13 labeled attacks. However, these raw data are saved as .pcap files and it needs to be converted into flow data to reduce its huge size, as its pure raw form is not very useful for deep learning models. The CIC-FlowMeter processor developed by them is used for the above conversion process. It ensures that the data features are extracted consistently from the same features found in the .pcap file. The transformed data are distributed as a .csv file to use deep learning models.

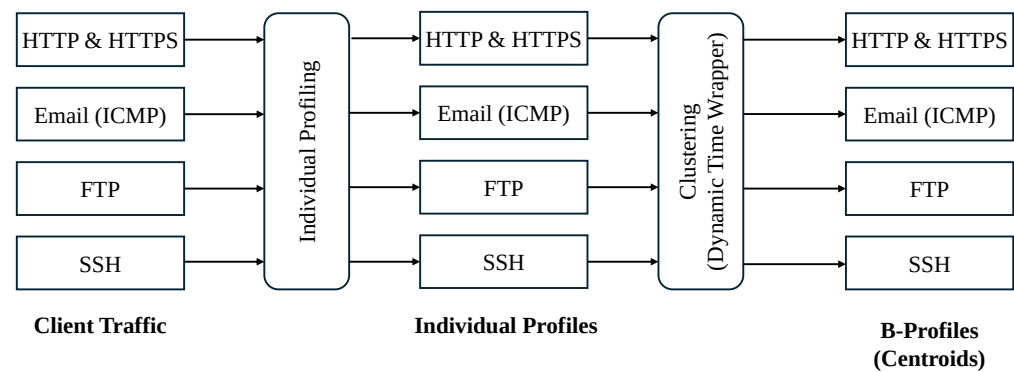


Figure 7. The B-profile system.

The resulting dataset was labeled based on the timestamp, source and destination IPs, source and destination ports, protocols, and types of attacks. We experimented using 424,155 flows randomly selected from the dataset, comprising 340,598 (80.3%) normal flows and 83,557 (19.7%) malicious flows. Each flow in this dataset includes over 80 network flow features extracted from captured network traffic.

5.1.2. UNSW-NB15 Dataset

The UNSW-NB15 dataset [37], released by the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS), combines real-world normal network activities with contemporary synthesized attack behaviors. Captured in a private environment, the dataset includes 2,218,761 benign flows (87.35%) and 321,283 attack flows (12.65%), categorized into ten classes: benign, analysis, backdoor, DoS, exploits, fuzzers, generic, reconnaissance, shellcode, and worms. The significant imbalance between normal and malicious flows necessitates preprocessing before use. This preprocessing involves removing excess normal flows from the dataset before experimenting. From the processed dataset, we used 50,000 flows, including 30,614 (61.22%) normal flows and 19,386 (38.78%) malicious flows, for the experiment. Each flow in the dataset is characterized by 49 features, including the class label. In the experiments of this paper, we examined these features using feature selection techniques before incorporating the data into the training model.

5.2. Evaluation Criteria

The results of this study were evaluated according to four criteria, namely accuracy, precision, f-measure, and recall. All these criteria take a value between 0 and 1. When it approaches 1, the performance increases. If it approaches 0, it decreases. The way to calculate these metrics is described in Table 1.

Table 1. Model performance metrics.

Metric	Definition
Recall	$TP/(TP + FN)$
Precision	$TP/(TP + FP)$
F1-score	$(2 \times \text{Recall} \times \text{Precision})/(\text{Recall} + \text{Precision})$
Accuracy	$(TP + TN)/(TP + FP + TN + FN)$

5.3. Implementation

The steps of our experimental process are described in Figure 8. Initially, we used an appropriate feature selection technique to select a set of valuable features from each of the CIC-IDS2017 and UNSW-NB15 datasets. Then, the obtained data were divided into training and testing sets to create graph data for the proposed model. After completing the training process, a node classification model with optimally selected parameters was used for classifying malicious flows. We simulated the experiment on the hardware with:

- CPU: Intel(R) Core i7-8700
- GPU: NVIDIA GeForce GTX 1060 3GB
- Memory: 32GB

5.3.1. Feature Selection

Data preprocessing is a crucial step that significantly enhances the proposed model's performance. Initially, the normalization of the dataset was conducted. For the CIC-IDS2017 dataset, flows containing attributes with malformed values, or featuring “infinity” or “NaN” values in the columns “Flow Bytes/s” and “Flow Packets/s” were eliminated. Additionally, the “Fwd Header Length” attribute is duplicated in columns 41 and 62; hence, one instance is removed. Some columns of the dataset, such as Flow ID, Source IP, Destination IP, Timestamp, and External IP, encompass string and categorical attributes. To facilitate numerical processing, these attributes were converted into numerical format utilizing the `LabelEncoder()` [38] function from the Sklearn library. As for column labels, they were divided into two types of labels to serve the classification of the model. Specifically, label 0 is assigned to benign flows, while label 1 is designated for the remaining attack flows.

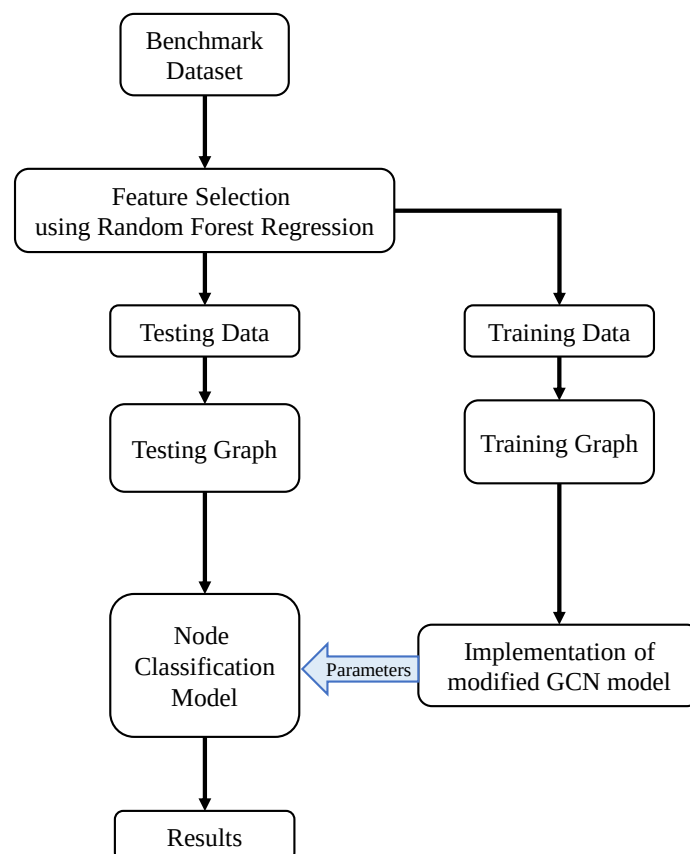


Figure 8. The implementation process.

After normalization, the features within each flow are extracted to constitute the input data for the model. The feature selection is based on the role of each type of feature in defining a different type of cyberattack. Therefore, to effectively predict each type of attack, the deep learning model requires diverse feature types. Those features are evaluated based on the “important weight” index initially introduced in the random forest regression algorithm [38]. The random forest method provides the advantage of assessing the importance of each feature in class prediction based on its individual score. Evaluating feature scores in a high-dimensional dataset can be challenging. To address this issue, the random forest method utilizes these importance scores to select a minimal set of highly

discriminatory features automatically. Leveraging this index, the author of the paper [39] identified the sets of four features that exert the most influence on each type of attack. Twelve different types of attack were considered, each corresponding to a set of 4 features. With each set of these 4 features, the model can learn how to classify and predict each type of attack with the highest accuracy. In this study, we classified network flows into normal flows and attack flows. The way to select features was also introduced in the paper [39], which is to synthesize features from the sets of 4 features mentioned above. In this study, all attack types were classified under a common label of ‘attack’. Thus, the 4 features obtained for each of the 12 attack types resulted in a pool of 48 features. After eliminating duplicates, the number of features was reduced to 18. We selected these 18 features for the process of creating graph data. In addition, the features “Source IP” and “Destination IP” were also selected because they are necessary for the graph creation process. Consequently, 20 features were selected from more than 80 features in each flow to construct a graph for training the model. The list of these 20 features is presented in Table 2 below.

Table 2. The list of selected features for the CIC-IDS2017 dataset.

Src IP	Flow IAT Max	Fwd Packet Length Max
Dst IP	Flow IAT Mean	Fwd Packet Length Mean
Bwd Packet Length Max	Flow IAT Min	Fwd Packet Length Min
Bwd Packet Length Mean	Flow IAT Std	Fwd Packet Length Std
Bwd Packet Length Std	Fwd IAT Total	Total Length of Bwd Packet
Total Backward Packets	Flow Bytes/s	Total Length of Fwd Packets
Total Backward Packets	Flow Duration	

Similarly, for the UNSW-NB15 dataset, columns containing string data or categorical attributes were converted into numerical format. The ‘attack-cat’ column, which included a list of attack types, was removed. We utilized the ‘label’ column, which was labeled with label 0 for benign flows and label 1 for attack flows. The Random Forest Regression algorithm was used to assess the influence of the features based on the ‘important weight’ index mentioned above. This index indicates the influence of each feature in class prediction and always sums up to 1. After calculating the ‘important weight’, various threshold values are used to determine the optimal number of features to select. The optimal number of features is identified at the threshold where the model achieves the highest classification accuracy. Based on the evaluation results, we selected a set of 32 features out of the total 49 features in the data to use for experiments on the proposed model. The list of 32 features is provided in Table 3 below.

Table 3. The list of selected features for the UNSW-NB15 dataset.

dur	service	spkts	dpkts
sbytes	dbytes	rate	sttl
dttl	sload	dload	sloss
dloss	sinpkt	dinpkt	sjit
djit	stcpb	dtcpd	tcprrt
synack	ackdat	smean	dmean
ct_srv_src	ct_state_ttl	ct_dst_ltm	ct_src_dport_ltm
ct_dst_sport_ltm	ct_dst_src_ltm	ct_src_ltm	ct_srv_dst

5.3.2. Creation of Training and Testing Data

After completing data preprocessing, the CIC-IDS2017 and UNSW-NB15 datasets were split into two subsets, with 80% allocated for training data and 20% for testing data. Subsequently, 20% of the training data were extracted as validation data during the model training process. Masks were also created to learn and evaluate each subset independently. Next, we create a graph from the extracted data, following the algorithm presented in Figure 5. Specifically, each flow in the dataset was represented as a node of the graph. Additionally, flows with the same source IP feature create an edge between them. For this graph generation process, we utilize the DGL library of Python.

5.3.3. Implementation of Modified GCN Model

The above graph was fed into the modified GCN model to create a node classification model. The modified model was described in Figure 6. In each SAGEConv layer, the number of hidden units is an important factor that affects the model's representational capacity and is also the dimension of the node embedding. This parameter determines the model's ability to leverage richer features. Another critical parameter is the learning rate, which dictates the size of the gradient descent step and influences whether the model can converge to the global optimum. To select appropriate values for these parameters, numerous experiments were implemented while keeping other parameters constant and adjusting a specific parameter. The results indicated that the model achieves optimal performance when the number of hidden units is set to 500 and the learning rate is set to 0.001. The model training process consists of 500 epochs, using the cross-entropy loss function and the Adam optimization function. Details regarding the selected parameters are provided in Table 4.

Table 4. Hyperparameter values used in FN-GNN.

Parameter	Value	Parameter	Value
Hidden units	500	Loss function	Cross-entropy
Learning rate	0.001	Optimizer	Adam
Epoch number	3000		

6. Results and Discussion

6.1. Experimental Results

The results of the experiments with the CIC-IDS2017 and UNSW-NB15 datasets were primarily evaluated based on the metrics presented in Section 5. First, we focused on the stability of the model, as demonstrated by its convergence. This can be assessed through the training process depicted in Figures 9 and 10. With the selected parameters mentioned above, the proposed model nearly achieves convergence around the 800th epoch and remains stable thereafter.

Next, the accuracy achieved by the model during the testing phase demonstrates the effectiveness of this model in accurately detecting malicious flows within network systems. Based on effective detection, the proposed model becomes sufficiently robust for deployment in the network systems of organizations and enterprises. Twenty percent of each dataset was used for testing, corresponding to 84,831 flows from the CIC-IDS2017 dataset and 9997 flows from the UNSW-NB15 dataset. The model's prediction results were depicted in the confusion matrix in Figures 11 and 12. The prediction results indicated that all incorrect predictions (both false positives and false negatives) make up less than 1%. This means that the model achieves a high and consistent detection rate for both malicious and normal flow cases.

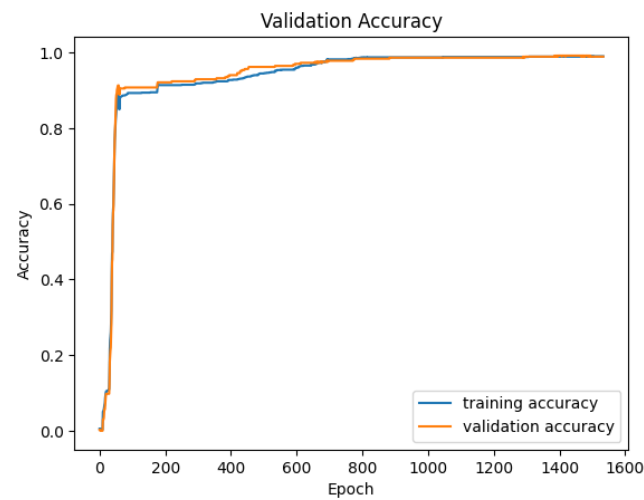


Figure 9. The accuracy of the training history on the CIC-IDS2017 dataset.

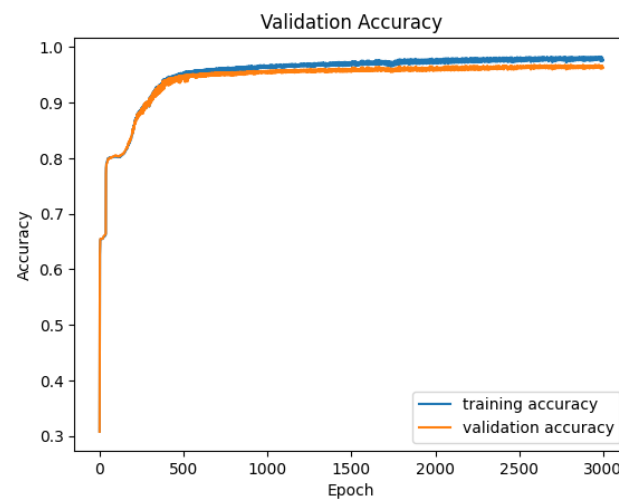


Figure 10. The accuracy of the training history on the UNSW-NB15 dataset.

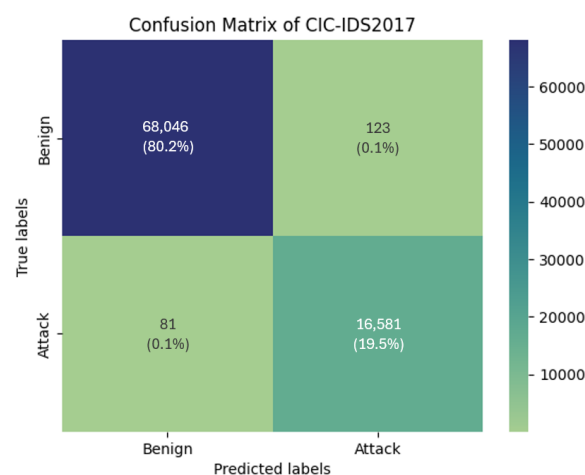


Figure 11. Prediction results on the test set of CIC-IDS2017.

The metrics used to evaluate the effectiveness of the model were calculated based on the formula presented in Table 1. The final evaluation results on two benchmark datasets are described in Table 5.

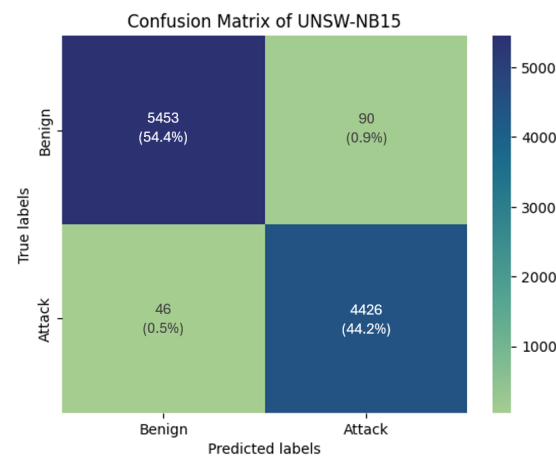


Figure 12. Prediction results on the test set of UNSW-NB15.

Table 5. Performance metrics.

Metrics/Dataset	CIC-IDS2017		UNSW-NB15	
	Benign	Attack	Benign	Attack
Precision	0.9988	0.9926	0.9916	0.9801
Recall (Detection rate)	0.9882	0.9951	0.9838	0.9897
F1-score	0.9985	0.9938	0.9877	0.9849
Weighted F1	0.9976		0.9865	
Accuracy	0.9976		0.9864	

The results indicated that the detection rates (recall) range from 98.38% to 99.51% in classifying both malicious and normal flows for both datasets. This demonstrated a high level of confidence. Figures 13 and 14 illustrate the model's accuracy in detecting malicious flows on two test datasets using ROC curves. Furthermore, due to the imbalance in the number of flow types within the datasets, the weighted F1 score was used for evaluation instead of solely relying on accuracy. Weighted F1 metrics were computed based on the F1-score values of various flows and their allocation counts in the test dataset. Accordingly, our model achieves weighted F1 scores of 99.76% and 98.65% on the CIC-IDS2017 and UNSW-NB15 datasets, respectively. We used these metrics to compare the performance of the proposed method with previous methods.

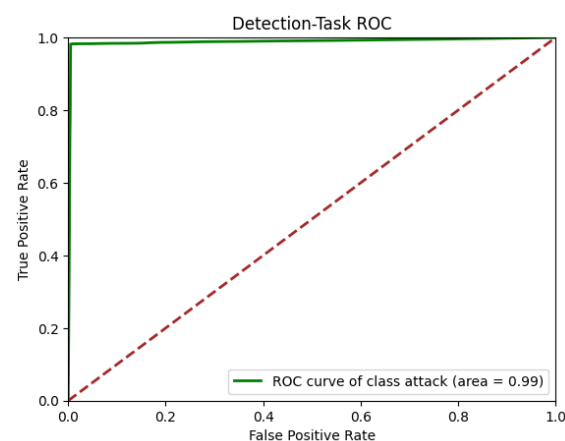


Figure 13. ROC on the test set of CIC-IDS2017.

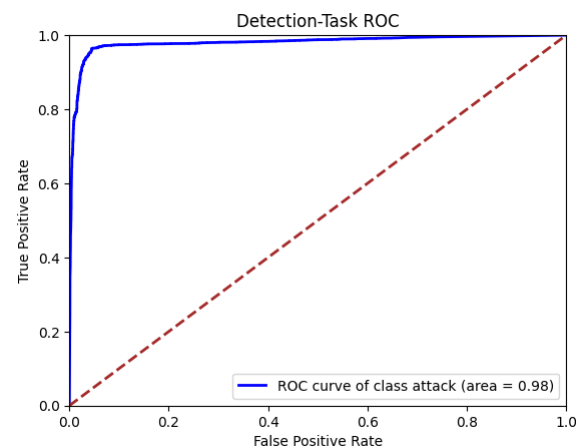


Figure 14. ROC on the test set of UNSW-NB15.

6.2. Comparison of Model Performance

We evaluated the effectiveness of the proposed model by comparing the weighted F1 scores obtained with previous models. Firstly, the proposed model was compared to GNN-based models currently considered most effective, such as E-GraphSAGE [40] and conventional GCN models. We used the E-graphSAGE and GCN models introduced in those papers to conduct the experiments classifying malicious flows in two datasets: CIC-IDS2017 and UNSW-NB15. In the case of the E-GraphSAGE model, edge features are extracted from flow data before their application in edge classification. Meanwhile, the GCN model utilizes these data as node features. The results obtained for each model are compared to the proposed model in Figure 15. We can see that the proposed model achieves a superior performance compared to the rest. This comparison result proves that the proposed model has reasonably inherited the strengths of SAGEConv layers and the GCN model.

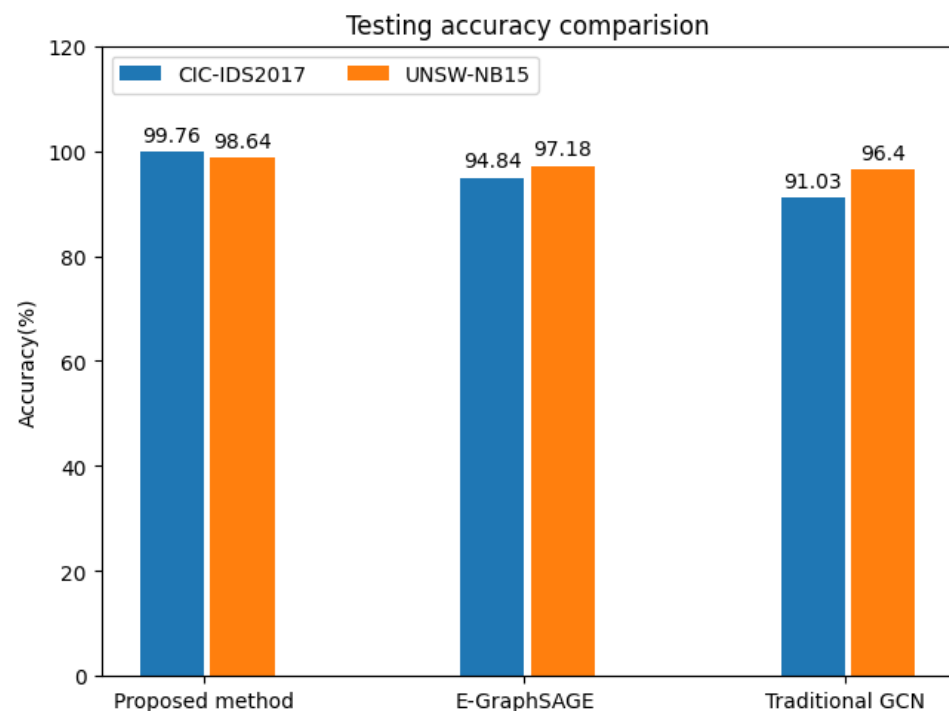


Figure 15. Comparison of the testing accuracy with previous models.

Table 6 presents comparative data on the performance and execution time of the proposed model with the E-GraphSAGE and conventional GCN models. These experiments

were performed with 424,155 flows taken from the CIC-IDS2017 dataset, in which the number of flows for the train set is 339,324 flows and the test set is 84,831 flows. Comparative data show that the proposed model always performs better than other models in precision, recall, and F1-score metrics. This means that the proposed model achieves a higher and more accurate detection rate of attack flows. However, the training and prediction times of the proposed model are not excessive compared to previous models. This can be explained based on how graph data are created from network flows as well as the architecture of each model. For the E-GraphSAGE model, graph data are created based on the network topology, in which each IP address corresponds to each node, and the number of edges represents the number of flows generated between those nodes. Meanwhile, the FN-GNN model proposes a new data preprocessing method that allows one to exploit the relationships between flows when creating graph data. In this way, each node represents each flow, and the number of edges created depends on each connection between flows in the network. We realized that, with the same amount of network flow, the graph data of the proposed model have a larger number of nodes and edges and are more complex than the graph data of E-graphSAGE. Therefore, the training and prediction time of the proposed model will also be a bit longer than that of E-GraphSAGE.

Table 6. Evaluation metrics.

Metrics	Proposal	E-GraphSAGE	Conventional GCN
Training time (s/epoch)	0.4605	0.3915	0.5735
Predicting time (s)	0.1625	0.133	0.1945
Precision	0.9926	0.95	0.92
Recall	0.9951	0.9454	0.9138
F1-score	0.9976	0.9477	0.9169

On the other hand, the proposed model has a more optimal execution time than the conventional GCN model. The GCN layer included in the GCN model uses the entire adjacency matrix to synthesize information from neighboring nodes, while the SAGEConv module applied in FN-GNN helps represent nodes by synthesizing information from several neighboring nodes. Therefore, for large graph data, the GCN model must process information from a vast number of connections, significantly increasing the system resource requirements and computation time. In contrast, the neighbor sampling mechanism enables the proposed model to efficiently utilize resources and optimize the computation time when applied in large-scale network environments.

To objectively assess the model's performance, we compare its weighted F1 scores with other existing models based on published results on the same datasets. The selected models are those that achieved the most outstanding results on the datasets used in this study.

On the CIC-IDS2017 dataset, we compared with models using machine learning techniques such as OC-SVM/RF [41], SVM, and ANN [42], as well as deep learning models like CNN-GRU [43], CNN-BiLSTM [44], and a two-phase intrusion detection system with naïve Bayes [45]. Similarly, on the UNSW-NB15 dataset, models such as AdaBoost, SVM, and DNN [46], as well as deep learning models like XGBoost-LSTM [47], AT-LSTM [48], and CNN-GRU [43], are also compared with our model. The comparison results are presented in Figures 16 and 17. Our model demonstrates a superior performance as well as high stability across multiple datasets, for both malicious and normal flow classification.

Furthermore, the effectiveness of the proposed model was also evaluated through comparison with several models employing the same feature selection approach. Specifically, feature selection techniques based on the random forest regression algorithm were also applied by the authors in papers [39,49] on the CIC-IDS2017 and UNSW-NB15 datasets, respectively. With the same selected features from the dataset, our model achieves significantly higher effectiveness. The evaluation results are shown in Table 7. This improvement can be explained by exploiting the relationship between flow data of the GNN model, through the synthesis of information from the neighbors of each node.

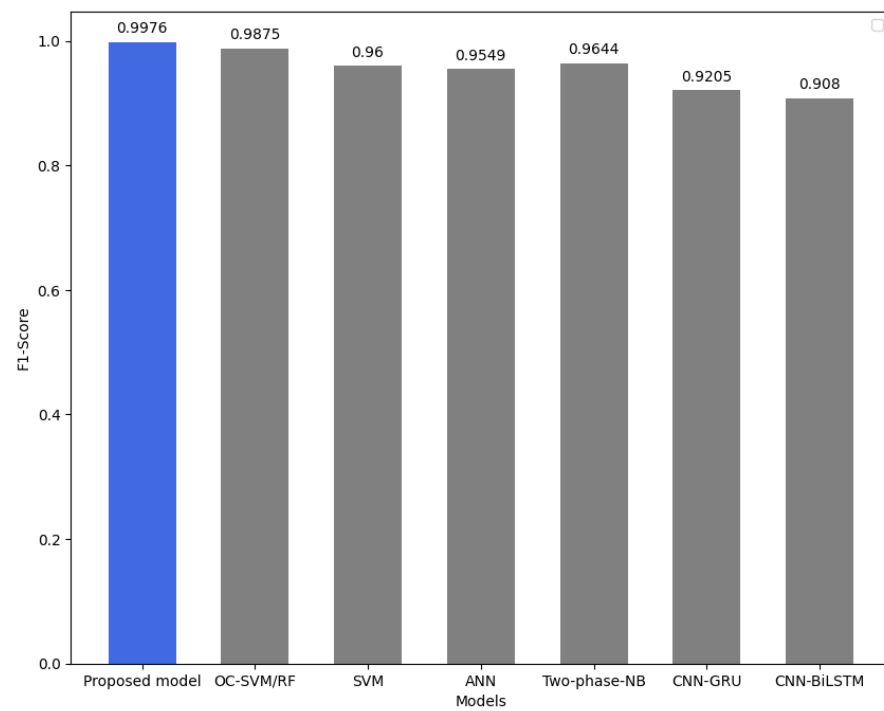


Figure 16. The comparison of the F1-score between the proposed model and the state-of-the-art models on the CIC-IDS2017 dataset.

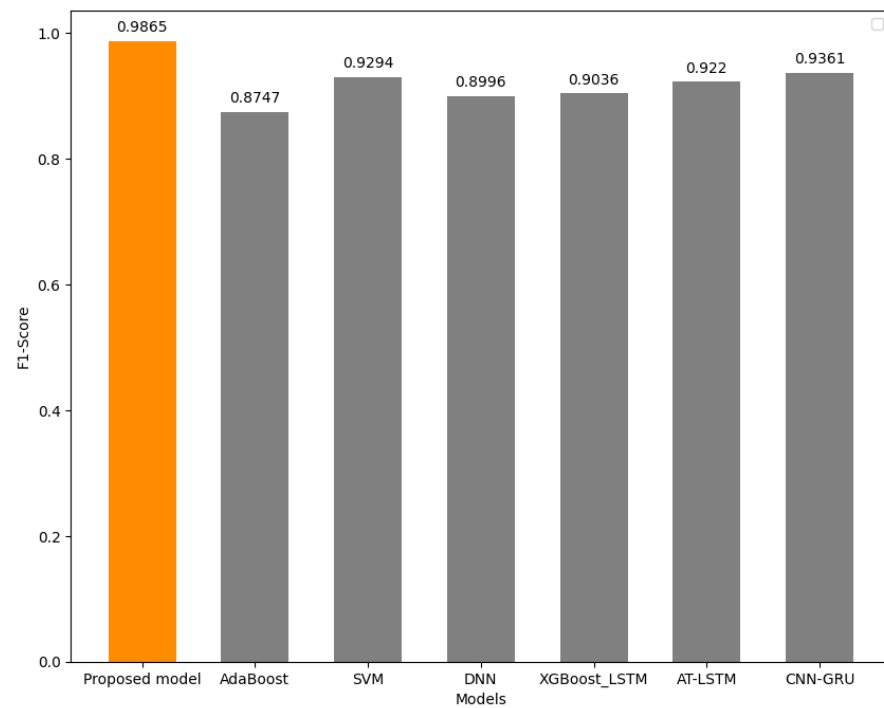


Figure 17. The comparison of the F1-Score between the proposed model and the state-of-the-art models on the UNSW-NB15 dataset.

Table 7. Comparison results of the proposed model with existing models using the same feature selection method.

Methods	Dataset	Accuracy	Model	No. Features
Proposal	CIC-IDS2017	99.76%	FN-GNN	20
[39]	CIC-IDS2017	94%	Random forest	20
		95%	ID3	20
		95%	AdaBoost	20
		81%	MLP	20
		96%	K nearest neighbors	20
Proposal	UNSW-NB15	98.65%	FN-GNN	32
[49]	UNSW-NB15	94.61%	Random forest	32

Experimental results on the two datasets CIC-IDS2017 and UNSW-NB15 show that the proposed model achieves superior performance and is more stable than existing models. To evaluate the effectiveness of the FN-GNN model when deployed in practice, the model's computational performance and ability to adapt to dynamic network conditions are aspects that need to be carefully considered. We have presented the computational efficiency of the model in Table 6. In network systems, NIDS continuously captures network traffic in the network. This traffic always has real-time characteristics and changes over time. New attack scenarios and techniques lead to changes in the nature of flows and the emergence of new traffic patterns in the data captured by the IDS. Furthermore, changes in network topology also lead to significant changes in network traffic. Those changes require NIDS systems to be constantly updated and able to adapt to new traffic patterns. The FN-GNN model provides a method for generating graph data based on evaluating the relationships of flows without depending on changes in network topology. Consequently, the graph data are generated flexibly according to the variations in network traffic. This approach allows the FN-GNN model to adapt to changes in network architecture within dynamic network environments.

Additionally, in our experiments, we used two datasets that are among the most relevant and representative of real-world environments. Specifically, these datasets include scenarios and attack types that closely resemble actual attack patterns encountered in practical settings. By using these datasets, we ensure that our experiments reflect realistic network conditions and potential threats, thereby enhancing the applicability and effectiveness of our proposed model in real-world scenarios. Moreover, the testing data used were ones that the FN-GNN model had not encountered before. Thus, the model was evaluated on data as if these were real-world data. We believe that network environments will not undergo significant changes in the near future. Therefore, the deep learning properties and the ability to model complex patterns of the proposed model help it maintain a strong performance in practical environments.

7. Conclusions

In this paper, we proposed the FN-GNN model including the data preprocessing for graph creation and the modified GCN model. In the data preprocessing, we introduced a novel approach to represent network flow data as graph data. In this model, the nodes of the graph represent a set of important features of the flow extracted by using the Random Forest Regression algorithm. The edges are created based on the relationship between flows through the source IP feature. The modified GCN model is a combination of conventional GCN and SAGEConv models. This helps overcome the limitations of previous GNN models. The proposed model achieved high stability and an accuracy of 99.76% and 98.65% for the CIC-IDS2017 and UNSW-NB15 datasets, respectively. The evaluation results demonstrated that our model performs consistently in classifying both normal and malicious flows and outperforming recent state-of-the-art models. However, we recognized that NIDS systems always face potential challenges in the future. Thus, we plan to continue researching and

updating the proposed model with newer datasets to enable the system to promptly detect complex attack scenarios when deployed in real-world environments.

Author Contributions: Conceptualization, D.-H.T. and M.P.; methodology, D.-H.T. and M.P.; software, D.-H.T.; validation, D.-H.T. and M.P.; formal analysis, D.-H.T.; investigation, D.-H.T. and M.P.; writing—original draft preparation, D.-H.T.; writing—review and editing, D.-H.T. and M.P.; supervision, M.P.; project administration, M.P.; funding acquisition, M.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was jointly supported by the National Research Foundation of Korea (NRF) via a grant provided by the Korea government (MSIT) (grant no. NRF-2023R1A2C1005461), and by the MSIT (Ministry of Science and ICT), Korea, under the Convergence Security Core Talent Training Business Support Program (IITP-2024-RS-2024-00426853) supervised by the IITP (Institute of Information & Communications Technology Planning & Evaluation).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Liao, H.J.; Richard Lin, C.H.; Lin, Y.C.; Tung, K.Y. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* **2013**, *36*, 16–24. [\[CrossRef\]](#)
2. Hubballi, N.; Suryanarayanan, V. False alarm minimization techniques in signature-based intrusion detection systems: A survey. *Comput. Commun.* **2014**, *49*, 1–17. [\[CrossRef\]](#)
3. Bhuyan, M.H.; Bhattacharyya, D.K.; Kalita, J.K. Network Anomaly Detection: Methods, Systems and Tools. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 303–336. [\[CrossRef\]](#)
4. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20. [\[CrossRef\]](#)
5. Do, D.P.; Kim, T.; Na, J.; Kim, J.; Lee, K.; Cho, K.; Hwang, W. D3T: Distinctive Dual-Domain Teacher Zigzagging Across RGB-Thermal Gap for Domain-Adaptive Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 17–21 June 2024; pp. 23313–23322.
6. 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. [\[CrossRef\]](#)
7. An Nguyen, T.; Lee, J. Design of Non-Isolated Modulation Code with Minimum Hamming Distance of 3 for Bit-Patterned Media-Recording Systems. *IEEE Trans. Magn.* **2023**, *59*, 1–5. [\[CrossRef\]](#)
8. Nguyen, T.; Lee, J. Interference Estimation Using a Recurrent Neural Network Equalizer for Holographic Data Storage Systems. *Appl. Sci.* **2023**, *13*, 11125. [\[CrossRef\]](#)
9. Nguyen, T.A.; Lee, J. A Nonlinear Convolutional Neural Network-Based Equalizer for Holographic Data Storage Systems. *Appl. Sci.* **2023**, *13*, 13029. [\[CrossRef\]](#)
10. Dang, X.T.; Nguyen, H.V.; Shin, O.S. Optimization of IRS-NOMA-Assisted Cell-Free Massive MIMO Systems Using Deep Reinforcement Learning. *IEEE Access* **2023**, *11*, 94402–94414. [\[CrossRef\]](#)
11. Nguyen, T.A.; Park, M. DoH Tunneling Detection System for Enterprise Network Using Deep Learning Technique. *Appl. Sci.* **2022**, *12*, 2416. [\[CrossRef\]](#)
12. Sarhan, M.; Layeghy, S.; Moustafa, N.; Portmann, M. NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. In Proceedings of the Big Data Technologies and Applications, Virtual Event, 11 December 2020; Deze, Z., Huang, H., Hou, R., Rho, S., Chilamkurti, N., Eds.; Springer: Cham, Switzerland, 2021; pp. 117–135.
13. Tomar, K.; Bisht, K.; Joshi, K.; Katarya, R. Cyber Attack Detection in IoT using Deep Learning Techniques. In Proceedings of the 2023 6th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 3–4 March 2023; pp. 1–6.
14. Busch, J.; Kocheturov, A.; Tresp, V.; Seidl, T. NF-GNN: Network Flow Graph Neural Networks for Malware Detection and Classification. In Proceedings of the 33rd International Conference on Scientific and Statistical Database Management, Tampa, FL, USA, 6–7 July 2021.
15. Zhao, J.; Liu, X.; Yan, Q.; Li, B.H.; Shao, M.; Peng, H. Multi-attributed heterogeneous graph convolutional network for bot detection. *Inf. Sci.* **2020**, *537*, 380–393. [\[CrossRef\]](#)
16. Pujol-Perich, D.; Suarez-Varela, J.; Cabellos-Aparicio, A.; Barlet-Ros, P. Unveiling the potential of Graph Neural Networks for robust Intrusion Detection. *SIGMETRICS Perform. Eval. Rev.* **2022**, *49*, 111–117. [\[CrossRef\]](#)

17. Zhou, J.; Xu, Z.; Rush, A.M.; Yu, M. Automating Botnet Detection with Graph Neural Networks. *arXiv* **2020**, arXiv:2003.06344.
18. Gong, L.; Cheng, Q. Exploiting edge features for graph neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 9211–9219.
19. Jiang, X.; Zhu, R.; Ji, P.; Li, S. Co-Embedding of Nodes and Edges With Graph Neural Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 7075–7086. [[CrossRef](#)]
20. Casas, P.; Vanerio, J.; Ullrich, J.; Findrik, M.; Barlet-Ros, P. GRAPHSEC—Advancing the Application of AI/ML to Network Security Through Graph Neural Networks. In Proceedings of the International Conference on Machine Learning for Networking, Paris, France, 28–30 November 2022; Springer: Cham, Switzerland, 2022; pp. 56–71.
21. Schlichtkrull, M.; Kipf, T.; Bloem, P.; van den Berg, R.; Titov, I.; Welling, M. Modeling Relational Data with Graph Convolutional Networks. In Proceedings of the Extended Semantic Web Conference, Portoroz, Slovenia, 28 May–1 June 2017.
22. Pang, B.; Fu, Y.; Ren, S.; Wang, Y.; Liao, Q.; Jia, Y. CGNN: Traffic Classification with Graph Neural Network. *arXiv* **2021**, arXiv:2110.09726.
23. Bekerman, D.; Shapira, B.; Rokach, L.; Bar, A. Unknown malware detection using network traffic classification. In Proceedings of the 2015 IEEE Conference on Communications and Network Security (CNS), Florence, Italy, 28–30 September 2015; pp. 134–142.
24. Xiao, Q.; Liu, J.; Wang, Q.; Jiang, Z.; Wang, X.; Yao, Y. Towards Network Anomaly Detection Using Graph Embedding. In Proceedings of the Computational Science—ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, 3–5 June 2020.
25. Bilot, T.; Madhoun, N.E.; Agha, K.A.; Zouaoui, A. Graph Neural Networks for Intrusion Detection: A Survey. *IEEE Access* **2023**, *11*, 49114–49139. [[CrossRef](#)]
26. Tran, D.H.; Park, M. Graph Embedding for Graph Neural Network in Intrusion Detection System. In Proceedings of the 2024 International Conference on Information Networking (ICOIN), Ho Chi Minh City, Vietnam, 17–19 January 2024; pp. 395–397.
27. Zhang, B.; Li, J.; Chen, C.; Lee, K.; Lee, I. *A Practical Botnet Traffic Detection System Using GNN*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 66–78.
28. Rusch, T.; Bronstein, M.; Mishra, S. A Survey on Oversmoothing in Graph Neural Networks. *arXiv* **2023**, arXiv:2303.10993.
29. Vinayakumar, R.; Soman, K.P.; Poornachandran, P. Applying convolutional neural network for network intrusion detection. In Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 13–16 September 2017; pp. 1222–1228.
30. Ahmad, Z.; Khan, A.S.; Cheah, W.S.; bin Abdullah, J.; Ahmad, F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Trans. Emerg. Telecommun. Technol.* **2020**, *32*, e4150. [[CrossRef](#)]
31. Yin, C.; Zhu, Y.; Fei, J.; He, X. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access* **2017**, *5*, 21954–21961. [[CrossRef](#)]
32. Caville, E.; Lo, W.W.; Layeghy, S.; Portmann, M. Anomal-E: A self-supervised network intrusion detection system based on graph neural networks. *Knowl.-Based Syst.* **2022**, *258*, 110030. [[CrossRef](#)]
33. Zhang, S.; Tong, H.; Xu, J.; Maciejewski, R. Graph convolutional networks: A comprehensive review. *Comput. Soc. Netw.* **2019**, *6*, 11. [[CrossRef](#)] [[PubMed](#)]
34. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1024–1034.
35. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Proceedings of the International Conference on Information Systems Security and Privacy, Funchal—Madeira, Portugal, 22–24 January 2018.
36. Sharafaldin, I.; Gharib, A.; Habibi Lashkari, A.; Ghorbani, A. Towards a Reliable Intrusion Detection Benchmark Dataset. *Softw. Netw.* **2017**, *2017*, 177–200. [[CrossRef](#)]
37. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, NSW, Australia, 10–12 November 2015; pp. 1–6.
38. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
39. Kostas, K. Anomaly Detection in Networks Using Machine Learning. Ph.D. Thesis, University of Essex, Essex, UK, 2018.
40. Lo, W.W.; Layeghy, S.; Sarhan, M.; Gallagher, M.R.; Portmann, M. E-GraphSAGE: A Graph Neural Network based Intrusion Detection System for IoT. In Proceedings of the NOMS 2022—2022 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 25–29 April 2022; pp. 1–9.
41. Verkerken, M.; D’hooge, L.; Sudyana, D.; Lin, Y.D.; Wauters, T.; Volckaert, B.; De Turck, F. A Novel Multi-Stage Approach for Hierarchical Intrusion Detection. *IEEE Trans. Netw. Serv. Manag.* **2023**, *20*, 3915–3929. [[CrossRef](#)]
42. Chua, T.H.; Salam, I. Evaluation of Machine Learning Algorithms in Network-Based Intrusion Detection Using Progressive Dataset. *Symmetry* **2023**, *15*, 1251. [[CrossRef](#)]
43. Bakhshi, T.; Ghita, B. Anomaly Detection in Encrypted Internet Traffic Using Hybrid Deep Learning. *Secur. Commun. Netw.* **2021**, *2021*, 5363750. [[CrossRef](#)]
44. Ghani, H.; Virdee, B.; Salekzamankhani, S. A Deep Learning Approach for Network Intrusion Detection Using a Small Features Vector. *J. Cybersecur. Priv.* **2023**, *3*, 451–463. [[CrossRef](#)]

45. Vishwakarma, M.; Kesswani, N. A new two-phase intrusion detection system with Naïve Bayes machine learning for data classification and elliptic envelop method for anomaly detection. *Decis. Anal. J.* **2023**, *7*, 100233. [[CrossRef](#)]
46. Wang, Z.; Liu, Y.; He, D.; Chan, S. Intrusion detection methods based on integrated deep learning model. *Comput. Secur.* **2021**, *103*, 102177. [[CrossRef](#)]
47. Kasongo, S.M. A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework. *Comput. Commun.* **2023**, *199*, 113–125. [[CrossRef](#)]
48. Alsharaiah, M.; Abualhaj, M.; Baniata, L.; Al-saaidah, A.; Kharma, Q.; Al-Zyoud, M. An innovative network intrusion detection system (NIDS): Hierarchical deep learning model based on Unsw-Nb15 dataset. *Int. J. Data Netw. Sci.* **2024**, *8*, 709–722. [[CrossRef](#)]
49. Kharwar, A.; Thakor, D. A Random Forest Algorithm under the Ensemble Approach for Feature Selection and Classification. *Int. J. Commun. Netw. Distrib. Syst.* **2023**, *29*, 426–447.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.