Hindawi
Security and Communication Networks
Volume 2023, Article ID 9796912, 10 pages
https://doi.org/10.1155/2023/9796912

Research Article

Deep Graph Embedding for IoT Botnet Traffic Detection

Bonan Zhang,[1] **Jingjin Li**,[1] **Lindsay Ward**,[2] **Ying Zhang**,[1] **Chao Chen**,[2] **and Jun Zhang**[1]

[1]*School of Physics and Electronic Information, Yunnan Normal University, Kunming 650000, China*
[2]*College of Science and Engineering, James Cook University, Townsville QLD 4811, Australia*

Correspondence should be addressed to Ying Zhang; yingzhang27@ynnu.edu.cn and Jun Zhang; junzhang@ynnu.edu.cn

Received 15 September 2021; Revised 6 March 2022; Accepted 23 May 2022; Published 25 October 2023

Academic Editor: Ding Wang

Copyright © 2023 Bonan Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Botnet attacks have mainly targeted computers in the past, which is a fundamental cybersecurity problem. Due to the booming of Internet of things (IoT) devices, an increasing number of botnet attacks are now targeting IoT devices. Researchers have proposed several mechanisms to avoid botnet attacks, such as identification by communication patterns or network topology and defence by DNS blacklisting. A popular direction for botnet detection currently relies on the specific topological characteristics of botnets and uses machine learning models. However, it relies on network experts' domain knowledge for feature engineering. Recently, neural networks have shown the capability of representation learning. This paper proposes a new approach to extracting graph features via graph neural networks. To capture the particular topology of the botnet, we transform the network traffic into graphs and train a graph neural network to extract features. In our evaluations, we use graph embedding features to train six machine learning models and compare them with the performance of traditional graph features in identifying botnet nodes. The experimental results show that botnet traffic detection is still challenging even with neural networks. We should consider the impact of data, features, and algorithms for an accurate and robust solution.

1. Introduction

Botnets are networks consisting of multiple compromised devices connected to the Internet [1]. These compromised devices can be used to perform malicious activities such as distributed denial-of-service attacks, data theft, and spamming [2, 3]. For botnet, it is not necessary to have a powerful host to accomplish complex tasks. To achieve the attack task, it needs to infect as many devices as possible. So many botnets are now targeting their infections on IoT devices. A number of IoT botnets have now emerged, such as BASHLITE [4], Carna [5], and Mirai [6]. One of them, Mirai, temporarily crippled Krebs with a denial-of-service attack in September 2016 [7]. Moreover, its attack volume exceeded 600 Gbps, one of the largest attacks on record. For IoT devices, their connection to the Internet makes it easier for botnets to spread. Furthermore, many IoT devices use initial passwords, making it easier for botnets to infect these devices. How to detect these botnet-infected IoT devices is the current key research problem.

From the perspective of structures, botnets can be classified as centralized command and control (C&C) structures or decentralised peer-to-peer (P2P) structures [8]. The bot header can direct the bot nodes efficiently for malicious activities through a hierarchical structure in centralized forms. The apparent hierarchical nature of this structure makes it easily detectable. To make botnets harder to detect, attackers are now building botnets using P2P structures. There are no hosts in the P2P structure for assigning tasks. The owner of the botnet can join any part of the botnet. Furthermore, any part of the botnet can control the entire botnet for attacks. This makes the detection of botnets more difficult than in the centralized model [9]. The focus of this paper is on detecting such P2P structured botnets.

In order to distinguish botnet traffic from background traffic, previous works mainly detect botnets from the following three categories: The first is botnet node detection by traffic patterns. This type of approach trains machine learning algorithms to find botnet traffic in the background

traffic by using traffic information, including communication time, port number, packet size, communication protocol, etc. as features. The second approach is to identify bot nodes by using prior knowledge, such as domain names and DNS blacklists. The attacker may evade both methods of detection by modifying the traffic packets. On the other hand, these two detection methods may also cause the leakage of user privacy. The last method is to distinguish botnet traffic from normal traffic by identifying the network topology. In previous studies, researchers have found that botnets often have particular topologies, such as mixing rates [10] and features of connection graph components [11, 12]. Centralized botnets manage individual nodes for attacks through a hierarchical structure. For botnets with a P2P structure, a high rate of mixing can often be found, as they need to pass information quickly to organise their attacks. On the other hand, to protect users from privacy breaches, more and more algorithms are now avoiding this problem by using federated learning. The details of each of these approaches will be discussed in the "Related Work."

Machine learning has shown its success in various cybersecurity applications, such as malware detection [13, 14], cyber threats/incidents detection [15–18], and software vulnerability detection [19–21]. Researchers have also applied machine learning in the detection of botnet traffic. However, most existing detection models require a lot of detailed traffic information for analysis. These content-based features have many limitations in practical use. The traffic details may be encrypted or may even have been intentionally modified by the attacker. Now some models solve this problem by identifying the traffic topology [10]. However, these models are labour-intensive in defining topological features and perform multiple prefiltering steps. In practice, the models also need to be tuned to data characteristics. This project achieves graph feature extraction by using graph neural network model to identify the network topology. The advantage of using a GNN model is that the relationships between nodes in the network can be effectively identified. Nodes at each layer in the model pass information to their neighbours to exchange information and update their state. As the number of layers in the model increases, the nodes will contain information about their neighbours at more hops, thus enabling the identification of the network topology. In this project, we first trained a GNN (graph neural network) model for generating graph embedding data and then tested the performance of six machine learning models when using this data for detection.

To summarise, the contributions we made in this paper are as follows:

(i) We provide detailed analysis of the algorithms previously used to detect botnet nodes and their suitable environments. The advantages and disadvantages of the different algorithms are summarised.

(ii) A GNN model has been tailored to transform traffic data into graph embedding data. The model effectively captures the relationships between nodes and provides a basis for identifying the topology of the network structure.

(iii) We test and analyze the performance of six machine learning algorithms for detecting botnet nodes when using graph embedding data.

The remainder of the paper is structured as follows: in Section 2, we review the reason why IoT devices are vulnerable to botnet infection and state-of-the-art techniques that are used for detecting bot nodes. In Section 3, we present in detail how the data was generated and the GNN model we used to extract the graph embedding features. In Section 4, we describe the datasets and model evaluation criteria that were used for evaluating model performance, along with the evaluation results. We discuss the limitations of our work and point out some future directions in Section 5. Section 6 concludes our work.

2. Related Work

In this section we will first describe the new trend that IoT devices are compromised to become botnets. We will then review the three categories of algorithms currently in use to detect botnets. Each of these methods has its own applicable scenarios and drawbacks, and we will elaborate them in this section to provide researchers with a reference for their choice of algorithm. On the other hand in order to be able to detect anomalous nodes, machine learning algorithms will inevitably use the user's private data for prediction. To address this problem, it has recently been proposed to avoid privacy breaches by using federated learning [22]. We will discuss this point in this section as well.

2.1. Botnet Attack from IoT Devices. DDoS attacks by infecting IoT devices have now become very common [6, 23]. The reasons are twofold. First, most of the IoT devices in widespread use today do not have powerful computing capabilities, so most IoT devices need to be connected to the Internet or other IoT devices to achieve more functionalities [24]. This structure makes IoT devices vulnerable to intrusion. Second, most IoT devices use insecure default passwords or simple passwords. For most consumers, they will probably only use a simple password or even just a default password when they purchase an IoT device [25]. For botnets aiming to infect IoT devices, it is possible to infect a large number of devices by simply brute-force logging in with randomly selected usernames and passwords from a list of preconfigured credentials [6, 26]. In cases where the password is changed to a simple password, it can also be easily guessed. The model designed by Wang et al. fits popular ciphers well and obtains a coefficient of determination greater than 0.97 [27]. In another paper, Wang et al. show that an attacker can achieve a success rate of up to 73% in 100 guesses against a regular user if he has some information about the user of the device being attacked [28]. There is a trend that the widespread use of IoT devices causes an increasing number of DDoS attacks to be launched from infected IoT devices.

2.2. Detection by Traffic Pattern. Identifying botnets by analyzing traffic patterns is an effective method. Communication between botnets is often characterized by command and control channels and often uses specific protocols for communication. Botnet traffic can be effectively identified by analyzing the communication patterns between networks. BotSniffer is a typical approach to identifying bot nodes by analyzing traffic patterns [29]. The model consists of the monitoring engine and the correlation engine. The monitoring engine inspects network traffic and passes suspected traffic records to the relevant engine for analysis. The model identifies botnets in background traffic by assuming that most nodes in the botnet network should react similarly. These botnets should have a similar structure and content of messages or a similar distribution of IP addresses and port ranges for scanning activity. The model clusters traffic messages by analyzing the similarity between messages. Thus, the set of botnet nodes is aggregated into a cluster. In 2016, Bartos et al. proposed a method to detect malicious traffic based on information from network packets as well [30]. This method uses the statistical features computed from network traffic as a training set and identifies malicious behaviour in network traffic by training a classifier. The method has a high precision (reaching 90%) and successfully identifies malware and variant samples not defined in the previous training.

These methods of botnet identification by using detailed information about network traffic packets generally have a high degree of precision and accuracy. However, a 2014 paper by Rndic and Laskov suggests that attackers can evade detection by manipulating information about network traffic packets [31]. In their experiments, the attackers were able to successfully avoid detection through automated inference algorithms and approximation algorithms without any detailed information about the detection system. The accuracy of the classification detector drops dramatically (from almost 100% to 28–33%) for these malicious behaviours that avoid detection by manipulating network traffic packets. Another study has also shown that attackers can also evade traffic monitoring by deliberately manipulating their communication patterns or encrypted channels [29]. And all these methods need to collect the privacy data of nodes into a central server for processing, which may cause the risk of user privacy leakage.

2.3. Detection by DNS Blacklist. Some botnet identification methods need to be supported by prior knowledge. These methods identify botnets by analyzing traffic data access information to define whether this access is associated with a botnet by comparing the domain name that appears against a blacklist of network domains obtained through prior knowledge. These methods avoid malicious behaviour by disabling botnet nodes from communicating with hosts [32]. This type of defence is passive and can effectively prevent botnets from launching malicious attacks. However, the blacklist used by the model is generally public, and there is a possibility that the information is not updated on time.

2.4. Detection by Graph Feature. As graph neural network technology evolves, there are more and more methods to identify botnets by their particular network topology. For botnets with a P2P structure there is generally a high rate of mixing. This is due to the fact that, in order to organise malicious attacks, botnets need to be more efficient than normal networks in terms of disseminating information. In the case of centralized botnets, a hierarchical structure is evident in the topology diagram. Based on this feature, previous research has proposed that P2P botnets can be detected based on their fast mixing rate [10]. Some studies have also shown that botnet detection can also be achieved by analyzing the number and size of connection graph components [11]. However, distinguishing the topology of a botnet from that of a normal network from the vast volume of network traffic is very difficult. Using this detection method first requires manual work to label topological features and narrows the range by performing several prefiltering steps. The method also requires parameters to be adjusted according to the data. Zhou et al. published a method to identify topologies from massive network traffic using GNNs in 2020 [33]. This method achieved good prediction results without prior knowledge and manual processing. However, when the training dataset was small, the model was unable to output good prediction results.

2.5. Federated Learning. Traditional machine learning algorithms require a central server for centralized data collection and processing. Each edge node needs to upload its data to the central server, which poses a risk to protecting the data privacy of edge nodes. On the other hand, due to various privacy laws and regulations, companies do not directly access users' private data. This creates difficulties in how to detect anomalous nodes in the network legally. To address this problem, Konečný et al. proposed a federated learning approach that trains machine learning models without compromising privacy [34]. This method is a distributed machine learning method, where nodes at the edge do not need to upload training data to contribute to the global model training. At the same time, the privacy leakage problem is faced by traditional distributed deep learning algorithms. A collaborative distributed deep learning paradigm was proposed by Liu et al. to address this problem [35]. The method is similar to the federated learning approach proposed by Google, where nodes at the edge also do not need to upload data to a central server to train deep learning models. For federated learning, the technical bottleneck is how to reduce its communication overhead for use in a wide range of devices. Previous algorithms generally reduce the communication overhead of federated learning by designing efficient stochastic gradient descent algorithms [36] and compression models [37].

3. Data Processing and Graph Feature Engineering

3.1. Data Processing. We used the botnet dataset collected by Zhou et al. in 2020 [33] (the dataset can be downloaded at

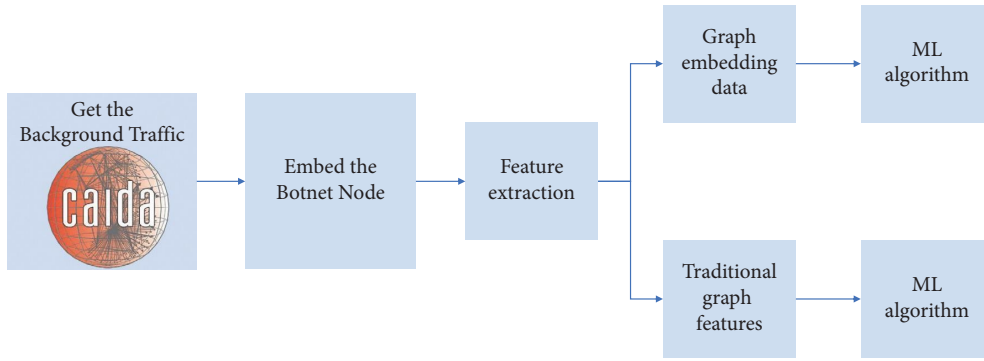

Figure 1: Flowchart for deep graph embedding feature botnet node detection experiments.

https://zenodo.org/record/). The background traffic for this dataset is all traffic information collected by CAIDA's monitors in 2018. After aggregating the traffic maps, the subnet traffic was selected for the experiment. The selection of subnet level traffic matches the network traffic topology map pattern available for the enterprise or research organisation. A subset of nodes from the subnet traffic is then randomly selected as botnet nodes. Botnet generation is achieved by embedding the botnet with the P2P structure captured by García et al. into these nodes [38]. The fact that decentralised botnets are not as well defined as centralized botnets with a hierarchical structure makes them more challenging to identify. Therefore, this project focuses on detecting decentralised P2P botnets. We use anomalous traffic detection as a binary classification problem to classify the nodes in a graph. The traffic graph uses IPs as nodes and traffic between IPs as edges. Figure 1 illustrates how we perform the data processing and feature extraction process.

The dataset we used in this experiment has 100 graphs, each containing an average of 140 000 nodes and 700 000 edges. Each of these graphs contains 3000 botnet nodes. The 100 graphs used in this experiment were randomly divided into a training set and a test set in a ratio of 9 : 1. In order to protect user privacy, specific IP information is hidden in this project to achieve data masking.

In this experiment, we process the dataset in two ways. One is to generate graph embedding data by the GNN algorithm introduced in the previous section. The other is to select traditional graph features for training. In the dataset using graph features, each node contains the following features: its own degree, the maximum, minimum, and average values of the degrees of its neighbours.

3.2. Graph Feature Engineering. In order to compare our proposed method of extracting graph features with state-of-the-arts, we have designed two sets of experiments in the evaluation. The first set is our proposed method. The method converts raw network traffic data into graph embedding data by using a trained GNN model. The second set uses traditional graph features as training data. Figure 2 shows the flowchart of the extracted graph feature project.

When training the GNN model, the last layer of the model is the linear layer, which outputs the prediction results for each node. When taking the graph embedding data, we remove the last layer and take the output of its previous layer as the graph embedding features for each node.

In this work, in order to automatically identify the topological features of a botnet under massive background Internet communication graphs, a neural graph networks (GNN) model was designed to accomplish this task [33, 39]. GNN is a neural network model that uses multiple convolutional layers to represent data using vectors. Capturing nodes in this way will contain important information about the node. GNN is ideally suited for identifying the network topology. In each layer of the GNN model, each node updates its state and passes information to its neighbouring nodes. Thus, after multiple layers of messaging, the model will automatically identify the topological relationships of the nodes in the graph.

The GNN model, which we used in this experiment, contains 12 layers. As the number of model layers increases, the nodes will contain more information about neighbouring nodes within hops, leading to better topology identification. Between layers, Zhou et al. used ReLu as the nonlinear activation function [33]. After each layer, we added the bias vector. The model uses the Adam optimiser [40], which uses cross-entropy to calculate the loss, with the learning rate set to 0.005 and the weight decay of the optimiser set to $5e-4$. The model implementations are based on Pytorch [41] and Pytorch Geometric [42]. The model structure is shown in Figure 3.

For input data, we used $G = \{V, A\}$ to define each graph. We define the set of nodes $\{v_1, ...v_n\}$ consisting of n unique nodes as V. And A is an adjacency matrix to show whether there exists a connection between v_i and v_j. When there is a connection between nodes v_i and v_j, we set $a_{ij} = 1$. The diagonal node degrees Diag$(d_1, ...d_n)$ is denoted by D, and the degrees are calculated by the formula $d_i = \sum_{j=1}^{n} a_{ij}$.

The node feature matrix of the data after layer l is $X^{(l)} \epsilon R^{n*h}$. The vector $x_i^{(l)}$ in each row is the feature vector of node v_i. h is the size of the node feature vector.

We use the learnable matrix $W^{(l)}$ to update the node vector of each layer:

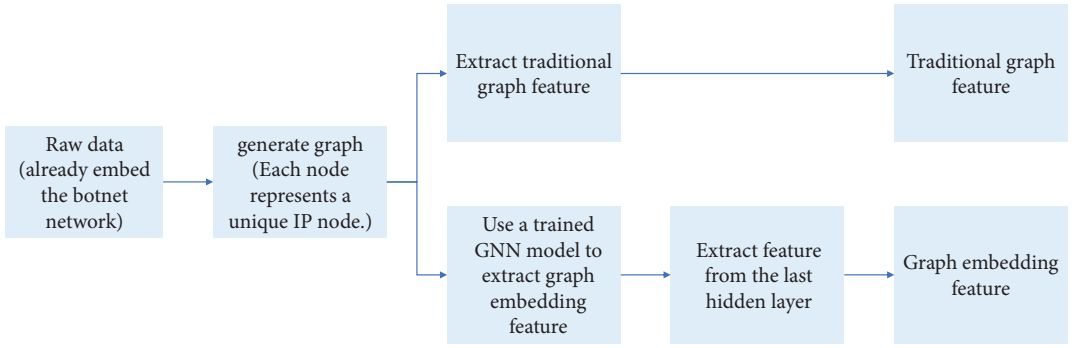

Figure 2: Flowchart for extracting graph features.

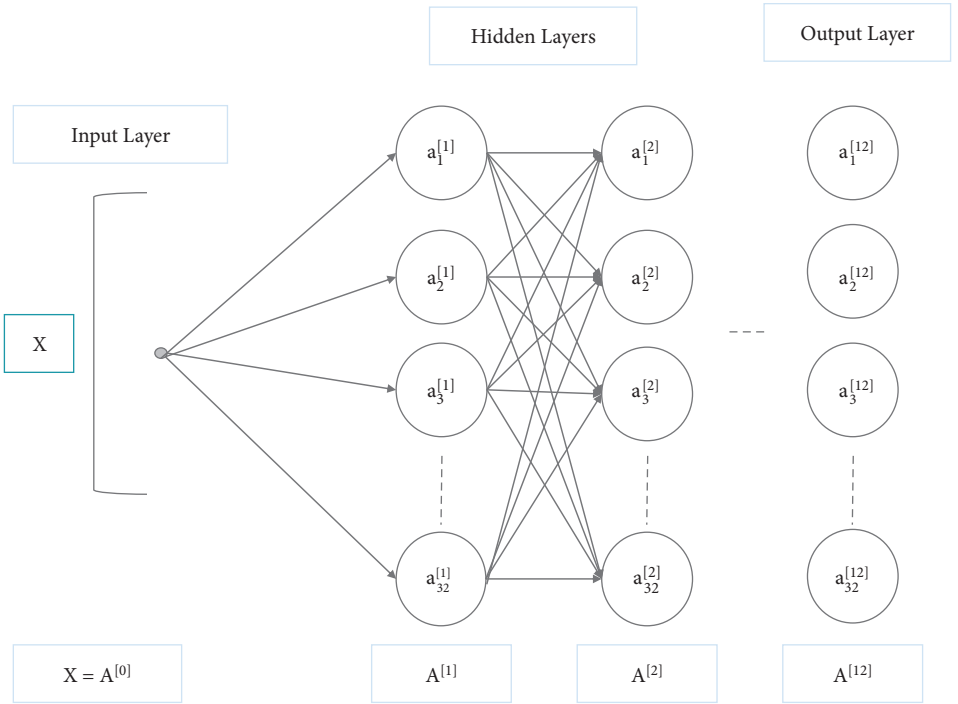

Figure 3: A 12-layer GNN model with all nodes of each graph as input layers. After 12 layers of GNN training, each node will be represented by 32 vectors.

$$x_i^{(l)} = x_i^{(l-1)} W^{(l)}. \tag{1}$$

In order to obtain information about neighbouring nodes, the representation of each node is the average of its direct neighbours.

$$x_i^{(l)} = \sum_{j=1}^{n} \frac{a_{ij}}{\sqrt{d_i d_j}} x_j^{(l)}. \tag{2}$$

In this experiment, we used a GNN model with many layers and normalisation at each layer to avoid numerical instability. It has been proved that GNN's model performance can be significantly improved by using normalisation [39]. The data processing for each layer can be expressed with the following expressions:

$$\begin{aligned} X^{(l)} &= \sigma\left(\overline{A} X^{(l-1)} W^{(l)}\right), \\ \overline{A} &= D^{-1/2} A D^{-1/2}. \end{aligned} \tag{3}$$

σ represents the nonlinear activation function ReLU. In our previous description, each node's data is updated according to the data of its neighbouring nodes. So after the L-level transformation, each node's data contains information about its adjacent nodes within its L-hop. This helps us to identify its topology. This is why the multilayer GNN model was chosen for the data transformation. When training the GNN model, we added a linear layer to output the predictions for each node.

$$X^{(l)} = \left(X^{(l-1)} U^{(l)} \sigma\left(\overline{A} X^{(l)} W^{(l)}\right)\right). \tag{4}$$

The learnable transformation matrix at layer l is denoted by $U^{(l)}$. The node data after the L layer will be input into the linear layer, and the softmax algorithm is used for the final classification. By adjusting $\overline{A}$, it is possible to control how neighbouring node features are normalised prior to aggregation, resulting in different variants of the GNN model. In $\overline{A} = D^{-1/2}AD^{-1/2}$, $\overline{A}$ is calculated from the degree of the source node and the degree of the target node. The graph attention network is also a GNN model variant that calculates $\overline{A}$ using a learnable nonlinear function based on node features, with each edge being independently normalised.

In this project, we modified the calculation of $\overline{A}$ according to the method proposed by Zhou et al. [33] to identify the topology more efficiently. Botnets with a P2P structure have a high rate of rapid mixing. In order to better identify the fast mixing rate [10] of the botnet, a random walk was used for normalisation. In our model, $\overline{A}$ is calculated as $\overline{A} = D^{-1}$. Unlike the previous equation, the calculation of $\overline{A}$ is only related to the degree of the source node and does not take into account the degree of the target node, thus achieving a random walk.

This experiment evaluates the botnet detection performance on six machine learning algorithms: logic regression, random forest, decision tree, Naive Bayes, k-nearest neighbour, and random forest. These six machine learning algorithms are chosen because they converge faster during training and are more suitable for anomaly detection for a large number of nodes in order to compare the difference between our graph embedding based features and the traditional graph features. Six machine learning algorithms with the same parameters are used in the experiments to training on each of the two datasets.

4. Experimental Evaluation

4.1. Evaluation Metrics. The dataset we used in this project was very unbalanced, with only 2% of the nodes being botnet nodes. It is not reasonable to evaluate model performance by accuracy in this case. This project focuses on whether the detection model can effectively find botnet nodes. In this experiment, we import some information retrieval metrics to evaluate the performance. 1) Positive and negative: Suppose there is a node N. The classifier's output is whether N belongs to a botnet or not. Researchers commonly use the true positives (TP), false positives (FP), and false negatives (FN) to evaluate the classifier. These metrics are defined below:

(a) True positive: The botnet node is correctly classified as botnet class.

(b) False positive: The normal node is incorrectly classified as botnet class.

(c) True negative: The normal node is correctly classified as normal class.

(d) False negative: the botnet node is incorrectly classified as normal class.

The relationship between these criteria and the classification results is shown in Table 1.

Table 1: Evaluation metrics.

	Predicted negative	Predicted positive
True negative	TN	FP
True positive	FN	TP

In order to test the recognition performance, we introduced true positive rate (TPR) and false positive rate (FPR) as evaluation metrics in this experiment.

TRP represents the ratio of the number of nodes correctly classified as botnet nodes to the number of all botnet nodes and is calculated as follows:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{5}$$

FPR is defined as the ratio of the number of nodes incorrectly classified as botnet nodes to the number of all normal nodes.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}. \tag{6}$$

In order to provide a more comprehensive analysis of the performance of each algorithm, we have introduced the metrics, precision, recall, and F-measure.

(a) Precision represents the ratio of the number of nodes correctly classified as botnet nodes to the number of all nodes predicted to be botnet nodes.

$$\text{Precision} = \frac{\text{TP}}{\text{FP} + \text{TP}}. \tag{7}$$

(b) Recall is also known as detection ratio. It is obtained by calculating the ratio of the total number of nodes correctly classified as botnet nodes to the total number of botnet nodes.

$$\text{Recall} = \frac{\text{TP}}{\text{FN} + \text{TP}}. \tag{8}$$

(c) The F-measure is a comprehensive evaluation of the algorithm's precision and recall criteria. The values of both detection rate and precision affect the calculation of the F-measure, making it easy for us to make comparisons. It is calculated by the following formula:

$$F - \text{measure} = \frac{2^{*}\text{Precision}^{*}\text{Recall}}{\text{Precision} + \text{Recall}}. \tag{9}$$

4.2. Comparisons among Different ML Algorithms. In this section, we evaluate the performance of each algorithm when using graph embedding data for training. In this set of experiments, each algorithm uses 90 graphs from the dataset for training and 10 graphs for validation of the detection. The dataset used in these experiments was extremely unbalanced. In order to obtain more reasonable detection results, we adjust the weights of the bot nodes and normal nodes by setting the parameter "class_weight" to be balanced, and we sampled the botnet node data.

Table 2 shows the results when the graph embedding data is used for training. From Table 2, we can see that the three algorithms logical regression, Naive Bayes, and KNN are not very effective in detecting botnets. The botnet detection rate of most algorithms is less than 50%. Tree-based algorithms such as XGBoost and decision tree are susceptible to bot nodes and achieve a bot node detection rate of over 98%. In terms of comprehensive performance evaluation, random forest has a relatively low false positive rate, a high precision rate, and the second highest *F*-measure value while retaining a high bot node detection rate. This suggests that tree-structure-based machine learning algorithms are superior to other algorithms in identifying topologies. On the other hand, tree-structure-based algorithms can find the vast majority of bot nodes, but their precision and *F*-measure values are generally low.

We use the graph embedding data to train the algorithm in this part of the experiment. We used 32 features to represent each node. However, these features are not independent of each other. Thirty-two features work together to generate a vector of nodes. For logistic regression, support vector machines, and naive Bayesian algorithms, these algorithms do not capture the relationship between the features. This makes these algorithms perform poorly when trained with graph-embedded data. For algorithms based on tree structures such as XGBoost, the prediction process requires multiple features to work together to make a prediction. This makes tree-structure-based machine learning algorithms more advantageous when using graph-embedded data for prediction. Also, we can find that the KNN algorithm has better performance in botnet node detection. This is since the KNN algorithm also works with multiple features to predict the results.

Table 2: Evaluation result.

Algorithm	Metric			
	Recall	FPR	Precision	*F*-measure
XGBoost	98.5	37.6	5.4	10.2
Decision tree	99.1	38.5	5.3	10.1
Random forest	73.8	0.6	72.0	72.9
Logistic regression	44.5	27.3	3.4	6.3
KNN	64.0	0.1	96.1	76.8
Naive Bayes	0.6	0.7	2.2	0.9

Table 3: Tradition feature result.

Algorithm	Metric			
	Recall	FPR	Precision	*F*-measure
XGBoost	99.1	98.9	2.2	4.3
Decision tree	94.6	94.7	2.2	4.3
Random forest	1.9	1.9	2.1	1.9
Logical regression	91.4	91.3	2.2	4.3
KNN	0	0	4.5	0
Naive Bayes	0	0	0	0

4.3. Comparisons with Traditional Graph Feature Set. This section uses the same dataset for training, but the extracted features are traditional graph features. The features include the degree of the node and the maximum, minimum, and mean of its neighbourhood degrees. We have selected the same raw data for training as in the previous section to facilitate our comparison of the performance changes of the algorithm. Table 3 shows the performance of the different algorithms when using traditional features.

From Table 3, we can see that all six machine learning algorithms show a sharp drop in performance when trained with traditional features compared to using graph-embedded features. Both the random forest algorithm and the naive Bayesian algorithm have a detection rate of less than 5%. The XGBoost algorithm, which has a high botnet node detection rate, decision tree, and logistic regression algorithms all have a false positive rate of over 90%. Such performance is unacceptable for botnet node detection. Figure 4 shows a comparison of the results of the different algorithms after training with different features.

These results show that it is not enough to use only two hops of the neighbour node for identifying botnet nodes. An algorithm using only this information cannot identify the topological features of the network. To recognise topological features, we need to obtain the hidden representations of the nodes for more sophisticated learning. The advantage of using GNNs is that the features of each node are generated by the joint influence of the features of its surrounding nodes. And with the multilayer GNN model, the features of each node will be related to the features of the nodes within its multihop. The GNN can effectively help us obtain the topology's implicit features, which allows training with graph-embedded features to outperform the performance by using only traditional features.

4.4. Impact of Imbalanced Data. This section evaluates the impact of the bot to nonbot node ratio on the machine learning algorithms described above. In previous experiments, the datasets we used for training and testing were extremely unbalanced. In this section, the training dataset retains all the botnet nodes and the same number of normal nodes when training the classifier. The training set contains 270 000 botnet nodes and 270 000 normal nodes. The same test dataset used in Section 4.2 was used when testing the classifier's performance.

From Table 4, we can find that, except for the logical regression algorithm, all the algorithms achieved a recall rate of over 90%. Compared with Table 2, the detection rates of random forest, KNN, and naive Bayesian algorithms are considerably higher. In particular, the naive Bayesian algorithm was enhanced from the original 0.6% to 99%. However, the high detection rate of the naive Bayesian algorithm comes at the cost of the false positive rate. It has a false positive rate of 99%. At the same time, we can observe that the random forest and KNN algorithms, which have a significant increase in detection rate, have a greater reduction in precision and an increased false positive rate. The false positive rate for both has increased from less than 1% originally to around 12%. The precision of the random forest

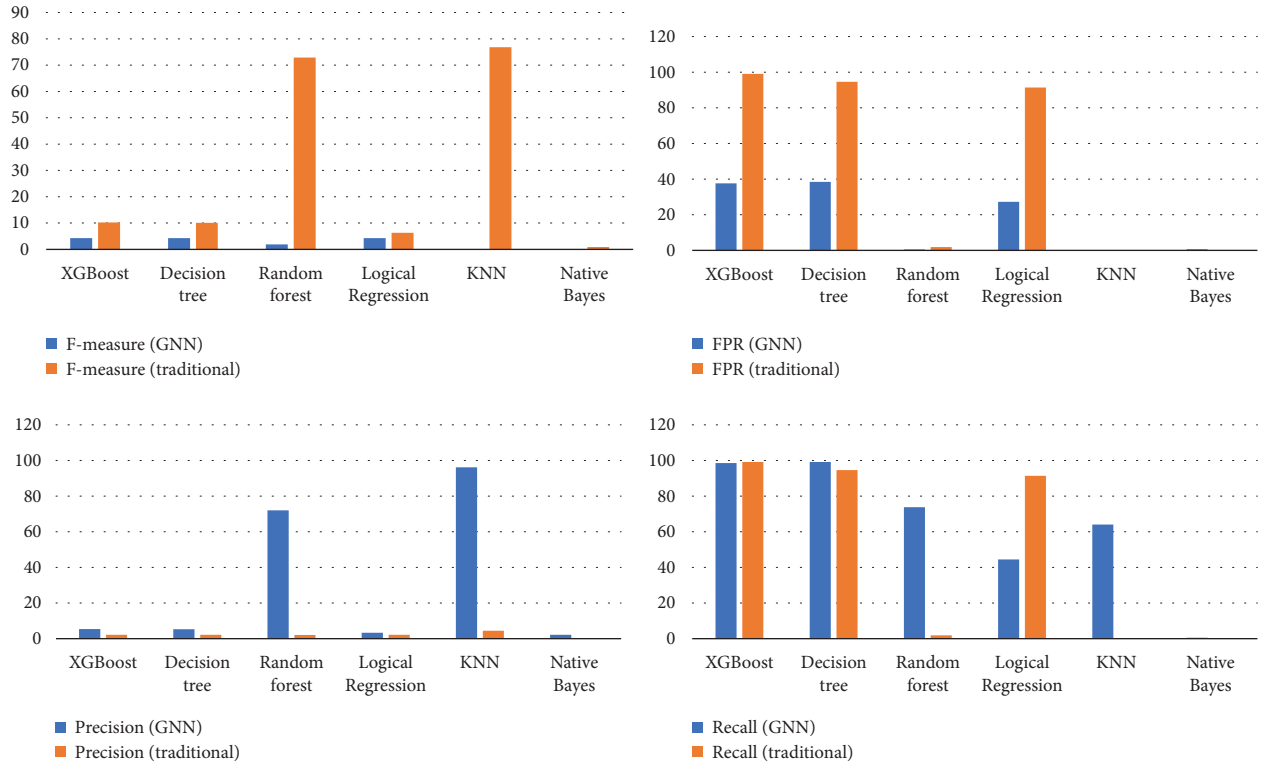

Figure 4: Comparison of the results of training with different features.

Table 4: Traditional features (balanced dataset) results.

Algorithm	Metric			
	Recall	FPR	Precision	*F*-measure
XGBoost	98.3	35.0	5.8	11.0
Decision tree	99.3	38.9	5.3	10.1
Random forest	92.5	11.5	15.1	26.0
Logical regression	25.9	13.8	4	6.9
KNN	93.2	12.2	14.4	25.0
Naive Bayes	99.0	99.0	2.2	4.2

algorithm was reduced from 72% to 15.1%. Furthermore, the precision of the KNN algorithm was reduced from 96.1% to 14.4%. There was no significant change in performance for both the XGBoost algorithm and the decision tree algorithm, with the four criteria performing in line with Table 2. The combined performance of the six machine learning algorithms does not improve significantly when trained with a balanced dataset.

5. Discussion

In this work, we focus on how to detect malicious behaviour, but both prevention and detection are equally crucial for stopping malicious behaviour. Much work is currently being done to enable nodes to perform specified actions to prove to other nodes that they are legitimate. For the current growing number of IoT devices, the limited resources on them make them vulnerable to some specific cyberattacks. Authentication and Key Agreement (AKA) needs to be established to protect communication between IoT devices and remote servers. The main issue being researched is how to balance security and usability in designing AKA protocols. The AKA protocol designed by Qiu et al. is based on chaotic map and uses the "Fuzzy-Versifiers" and "Honey" techniques [43]. The approach proposes a secure three-factor AKA protocol for lightweight mobile devices based on an extended chaotic graph. However, whether the approach can be deployed on IoT devices with more resources than mobile devices has not been validated. On the other hand, Wang et al. argue that cloud centres are necessary under the current conditions of limited node computing resources [44]. With the cloud centre, it can help make intelligent decisions and ease the pressure on nodes for computing and storage. This approach effectively reduces the computational pressure on the sensor nodes, making their computational cost only equivalent to that of the symmetric encryption algorithm. For our future research, we also plan to combine both detection and prevention of malicious behaviour in order to prevent it more efficiently.

6. Conclusion

In this paper, we propose a way for extracting botnet graph embedding features and comprehensively analyze the performance of different machine learning algorithms. To perform this evaluation, we embedded network traffic generated by 300,000 bot nodes captured from the actual network into background traffic containing 14,000,000 nodes to create a network graph. In this paper, the GNN model is applied to capture the topology of a botnet with a

P2P structure. In order to investigate the botnet detection capabilities of different classifiers, experiments were conducted using different sampling methods and feature extraction methods. A few insights were generated from our evaluations. First, the tree-structure-based machine learning algorithms outperform other algorithms. Second, machine learning algorithms cannot correctly distinguish a botnet from a normal network if the node data contains fewer hops of information about its neighbours. Third, the detection rate of botnet nodes can be improved by adjusting the data balance, but it will increase the false positive rate of the algorithm. We hope these insights can help researchers or cybersecurity professionals better detect botnet traffic.

Data Availability

The data supporting this paper were taken from previously reported studies and datasets, which have been cited. The processed data are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no potential conflicts of interest.

Acknowledgments

The research and publication of this article were funded by the National Natural Science Foundation of China (Grant no. 62062069).

References

[1] M. Feily, A. Shahrestani, and S. Ramadass, "A survey of botnet and botnet detection," in *Proceedings of the 2009 Third International Conference on Emerging Security Information, Systems and Technologies*, pp. 268–273, Athens, Greece, June 2009.

[2] N. Ianelli and A. Hackworth, "Botnets as a vehicle for online crime," *CERT Coordination Center*, vol. 1, no. 1, 2005.

[3] B. Paul, T. Holz, M. Kotter, and G. Wicherski, "Know your enemy: tracking botnets," *The Honeynet Project & Research Alliance*, vol. 2005, 2005.

[4] C. Cimpanu, "There's a 120,000-Strong IoT DDoS botnet lurking around," *Softpedia*, 2016, https://news.softpedia.com/news/there-s-a-120-000-strong-iot-ddos-botnet-lurking-around-507773.shtml.

[5] S. Christian and H. Judith, "Mapping the Internet: a hacker's secret Internet Census," *Spiegel*, 2013, https://www.spiegel.de/international/world/hacker-measures-the-internet-illegally-with-carna-botnet-a-890413.html.

[6] M. Antonakakis, T. April, M. Bailey et al., "Understanding the mirai botnet," in *Proceedings of the 26th {USENIX} security symposium*, pp. 1093–1110, Canada, Auguest 2017.

[7] B. Krebs, *"Krebsonsecurity Hit with Record Ddos"*, Website, 2016, https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/.

[8] S. Heron, "Botnet command and control techniques," *Network Security*, vol. 2007, no. 4, pp. 13–16, 2007, [Online]. Available:.

[9] B. Julian, V. Sharma, C. Nunnery, K. Brent ByungHoon, and D. Dagon, "Peer-to-peer botnets: overview and case study"," in *Proceedings of the First Workshop on Hot Topics in Understanding Botnets, HotBots'07*, N. Provos, Ed., Cambridge, MA, USA, April 2007.

[10] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov, "Botgrep: finding p2p bots with structured graph analysis," in *Proceedings of the 19th USENIX Conference on Security*, USENIX Security, Washington, DC, February 2021.

[11] M. P. Collins and M. K. Reiter, "Hit-list worm detection and bot identification in large networks using protocol graphs," in *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*, pp. 276–295, Springer, Gold Goast, Australia, September 2007.

[12] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, G. Varghese, and H. C. Kim, *Graption: Automated Detection of P2p Applications Using Traffic Dispersion Graphs (Tdgs)"*, University of California, Riverside Report, UCR-CS-2008096080, California, USA, 2008.

[13] J. Qiu, J. Zhang, W. Luo, and Y. PanNepalXiang, "A survey of android malware detection with deep neural models," *ACM Computing Surveys*, vol. 53, no. 6, pp. 1–36, 2020.

[14] X. Chen, C. Li, D. Wen, Y. ZhangNepalXiangRen, and K. Ren, "Android hiv: a study of repackaging malware for evading machine-learning detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 987–1001, 2020.

[15] L. Liu, O. De Vel, Q.-L. Han, J. Xiang, and Y. Xiang, "Detecting and preventing cyber insider threats: a survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1397–1417, 2018.

[16] Y. Miao, C. Chen, L. Han, J. ZhangXiang, and Y. Xiang, "Machine learning-based cyber attacks targeting on controlled information," *ACM Computing Surveys*, vol. 54, no. 7, pp. 1–36, 2022.

[17] J. Zhang, L. Pan, Q.-L. Han, C. Chen, S. Wen, and Y. Xiang, "Deep learning based attack detection for cps security: a survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, 2021.

[18] N. Sun, J. Zhang, P. Rimba Rimba, and Y. GaoZhangXiang, "Data-driven cybersecurity incident prediction: a survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1744–1772, 2019.

[19] G. Lin, S. Wen, Q.-L. Han, J. Xiang, and Y. Xiang, "Software vulnerability detection using deep neural networks: a survey," *Proceedings of the IEEE*, vol. 108, no. 10, pp. 1825–1848, 2020.

[20] S. Liu, M. Dibaei, Y. Chen, J. ZhangXiang, and Y. Xiang, "Cyber vulnerability intelligence for internet of things binary," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2154–2163, 2020.

[21] M. Wang, T. Zhu, T. Zhang, S. YuZhou, and W. Zhou, "Security and privacy in 6g networks: new areas and new challenges," *Digital Communications and Networks*, vol. 6, no. 3, pp. 281–291, 2020.

[22] E. Jonsson and E. Jonsson, "Anomaly-based intrusion detection: privacy concerns and other problems," *Computer Networks*, vol. 34, no. 4, pp. 623–640, 2000.

[23] An Wang, W. Chang, S. Mohaisen, and A. Mohaisen, "Delving into internet ddos attacks by botnets: characterization and analysis," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2843–2855, 2018.

[24] B. L. Risteska Stojkoska, "A review of internet of things for smart home: challenges and solutions," *Journal of Cleaner Production*, vol. 140, pp. 1454–1464, 2017.

[25] C. Macdonald, *Passwords Used in the Biggest Ever Cyberattack Revealed - and '12345' and 'password' Were Top"*, Dailymail, 2016, https://www.dailymail.co.uk/sciencetech/article-3825740/Passwords-used-biggest-DDoS-attack-revealed-12345-password-top.html.

[26] T. Easton, *"Chalubo Botnet Wants to Ddos from Your Server or Iot Device"*, Sophos, 2018, https://news.sophos.com/en-us/2018/10/22/chalubo-botnet-wants-to-ddos-from-your-server-or-iot-device/.

[27] D. Wang, H. Cheng, P. Wang, X. Jian, and G. Jian, "Zipf's law in passwords," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2776–2791, 2017.

[28] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, "Targeted online password guessing: an underestimated threat," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 1242–1254, Vienna, Austria, July 2016.

[29] G. Gu, J. Zhang, and W. Lee, *Botsniffer: Detecting botnet command and control channels in network traffic*, in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, California, USA, January 2008.

[30] K. Bartos, M. Sofka, and V. Franc, "Optimized invariant representation of network traffic for detecting unseen malware variants," in *Proceedings of the 25th USENIX Security Symposium (USENIX Security 16)*, Austin, TX, August 2016, https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/bartos.

[31] N. Rndic and P. Laskov, "Practical evasion of a learning-based classifier: a case study," in *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, Berkeley, CA, USA, May 2014.

[32] R. Perdisci and W. Lee, "Method and system for detecting malicious and/or botnet-related domain names," *US Patent*, vol. 10, p. 688, 2018.

[33] J. Zhou, Z. Xu, A. M. Rush, and M. Yu, "Automating botnet detection with graph neural networks," *AutoML for Networking and Systems Workshop of MLSys 2020 Conference*, vol. 1, 2020.

[34] J. Konečnỳ, H. B. McMahan, and X. Felix, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: strategies for improving communication efficiency," 2016, https://arxiv.org/abs/1610.05492.

[35] Y. Liu, J. J. Q. Yu, J. Kang, and S. Zhang, "Privacy-preserving traffic flow prediction: a federated learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7751–7763, 2020.

[36] N. Agarwal, A. T. Suresh, F. Yu, S. Kumar, and H. Brendan Mcmahan, "cpsgd: communication-efficient and differentially-private distributed sgd," 2018, https://arxiv.org/abs/1805.10559.

[37] Yi Liu, S. Garg, J. Zhang, J. XiongKangHossain, and M. S. Hossain, "Deep anomaly detection for time-series data in industrial iot: a communication-efficient on-device federated learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6348–6358, 2021.

[38] S. García, M. Grill, J. Zunino, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, vol. 45, pp. 100–123, 2014, [Online]. Available:.

[39] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, https://arxiv.org/abs/1609.02907.

[40] D. P. Kingma and Ba Jimmy, "Adam: a method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.

[41] P. Adam, S. Gross, F. Massa, and A. Lerer, J. Bradbury, G. Chanan, T. Killeen et al., Pytorch: an imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, pp. 8026–8037, 2019.

[42] M. Fey and J. Eric Lenssen, "Fast graph representation learning with pytorch geometric," 2019, https://arxiv.org/abs/1903.02428.

[43] S. Qiu, D. Wang, G. Kumari, and S. Kumari, "Practical and provably secure three-factor authentication protocol based on extended chaotic-maps for mobile lightweight devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, 2020.

[44] C. Wang, D. Wang, and D. He, "Efficient privacy-preserving user authentication scheme with forward secrecy for industry 4.0," *Science China Information Sciences*, vol. 65, no. 1, Article ID 112301, 2022.