mathematics

MDPI

Article

Research on Virus Propagation Network Intrusion Detection Based on Graph Neural Network

Xianer Ying [1], Mengshuang Pan [1], Xiner Chen [1], Yiyi Zhou [2], Jianhua Liu [1,*], Dazhi Li [3,*], Binghao Guo [1] and Zihao Zhu [1]

[1] Department of Computer Science and Engineering, Shaoxing University, Shaoxing 312000, China; c1571153285@163.com (X.C.); 18636961259@163.com (B.G.); zzh66192021@163.com (Z.Z.)
[2] College of Letters & Science, University of California, Berkeley, Berkeley, CA 94720, USA
[3] College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, Shanghai 200234, China
* Correspondence: ljh_541@163.com (J.L.); lijunzhi@shnu.edu.cn (D.L.)

Abstract: The field of network security is highly concerned with intrusion detection, which safeguards the security of computer networks. The invention and application of intrusion detection technology play indispensable roles in network security, and it is crucial to investigate and comprehend this topic. Recently, with the continuous occurrence of intrusion incidents in virus propagation networks, traditional network detection algorithms for virus propagation have encountered limitations and have struggled to detect these incidents effectively and accurately. Therefore, updating the intrusion detection algorithm of the virus-spreading network is imperative. This paper introduces a novel system for virus propagation, whose core is a graph-based neural network. By organically combining two modules—a standardization module and a computation module—this system forms a powerful GNN model. The standardization module uses two methods, while the calculation module uses three methods. Through permutation and combination, we obtain six GNN models with different characteristics. To verify their performance, we conducted experiments on the selected datasets. The experimental results show that the proposed algorithm has excellent capabilities, high accuracy, reasonable complexity, and excellent stability in the intrusion detection of virus-spreading networks, making the network more secure and reliable.

Keywords: virus propagation; intrusion detection; deep learning; graph neural networks

MSC: 68M25

Citation: Ying, X.; Pan, M.; Chen, X.; Zhou, Y.; Liu, J.; Li, D.; Guo, B.; Zhu, Z. Research on Virus Propagation Network Intrusion Detection Based on Graph Neural Network. *Mathematics* **2024**, *12*, 1534. https://doi.org/10.3390/math12101534

Academic Editors: Zhaoquan Gu and Jianxin Li

Received: 4 April 2024
Revised: 3 May 2024
Accepted: 10 May 2024
Published: 14 May 2024

1. Introduction

Along with the consistent development of the Internet, people's livelihoods have undergone significant changes. However, in this era in which everything is interconnected, various network viruses have emerged, and network security issues have gradually surfaced. Multiple kinds of intrusions and attacks cause serious harm to the network and equipment. Among these concerns, network intrusion through virus infection has recently gained attention. The attack methods update constantly while their scope expands, and pattern is hidden yet frequent. Traditional virus-infected network intrusion detection algorithms are no longer suitable for new emerging virus-infected network intrusions. Therefore, finding a new detection method to accurately detect these new network intrusions is urgent.

Based on the study of references [1–10], virus propagation networks often present complex structural forms, such as grid topology and ring topology, which are difficult to detect when they spread among networks.

Moreover, existing intrusion detection methods have limitations in detecting complex virus transmission networks. These methods mainly consist of traffic-based detection

(ref. [11–15]) and graph-based detection (ref. [16–20]). The traffic-based detection method uses statistical analysis to detect link-type data, though this method does not apply to topology. Meanwhile, the traditional graph-based detection method is unsuitable for complex topologies. Moreover, the conventional methods have the following drawback: they usually use a static model to represent the network structure, but the network is dynamic. This drawback means that these methods cannot capture the dynamic evolution and real-time changes in the network. They often only focus on the local network structure or traffic characteristics and ignore global information. Virus spreading involves the topology of the whole network and the complex relationship between nodes, so it may not be practical to detect the spreading behavior by relying solely on local information. The existing methods usually lack a deep understanding of node behavior and context, and virus propagation may be affected by many factors, such as node attributes, behavior patterns, etc. However, traditional methods often fail to take these factors into account. Therefore, an efficient, accurate, and complex detection method is necessary when dealing with a virus propagation network model with a complex topology.

Considering that GNN can capture complex topologies with high precision and that the GNN deep learning algorithm is efficient, we chose GNN as the detection model for the virus propagation network. This paper presents an intrusion detection model based on a graph neural network for a virus propagation network. The contributions of this paper are as follows:

(1) We construct a GNN model. Our GNN model consists of the standardization module and the computation module—the two modules stack on top of each other to form a powerful GNN model.
(2) The normalized module is constructed, and it consists of symmetry and random walk (ref. [21–23]).
(3) The computing module consists of three parts: multi-layer perceptron (MLP) (ref. [24–26]) and additive and incremental learning methods. MLP is a classical neural network model with a wide range of applications; the additive method is a simple and effective algorithm that can quickly adapt to new input data; the incremental learning rule is a method of updating model parameters, which can improve the training effect and generalization ability of the model. These three parts constitute an efficient and reliable computing module that performs well in data classification and text clustering tasks.

After a detailed and in-depth exploration, we conducted many experiments on the network intrusion detection dataset, CTU-13, to verify the performance of the new detection model. The experimental results revealed that the proposed model has significant advantages over the traditional model in terms of detection efficiency and accuracy.

2. Related Work

Intrusion detection systems have numerous shortcomings in traditional virus transmission network detection methods. Mithlesh Kumar et al. [25] proposed using modified and optimized machine learning intrusion detection systems that incorporated MLP and RNN architectures to address these limitations. While this approach demonstrated a promising performance when encountering small amounts of intrusion data, its effectiveness was limited when the virus transmission network containing vast amounts of data was detected.

Flow-based detection (ref. [11,15]) has numerous advantages. It focuses explicitly on analyzing IP stream records, which provide aggregated information about packet headers and summarize network traffic as IP streams. Doing so reduces the amount of data intrusion detection systems need to process. As a result, flow-based intrusion detection is particularly suitable for detecting complete network traffic backbone links that may pose computational challenges. Although it excels at efficiently and accurately detecting link-type data, this model could be more effective at detecting the complex topology involved in virus transmission networks.

Based on the study of references [16–20], graph-based intrusion detection involves processing each stream into a graph by utilizing the interactive information between packets. The graph is then classified using various methods to learn its vector representation. This approach effectively transforms the problem of traffic detection into a graph classification problem. However, when it comes to detecting the virus transmission network with its complex topology, this method still has limitations associated with ensuring high efficiency and precision of detection.

Traditional machine learning algorithms (ref. [27,28]), including random forest, support vector machine (SVM) (ref. [29,30]), and adaptive boosting, have been widely applied to network intrusion detection with varying degrees of success. However, two significant limitations exist in applying virus-spread network intrusion detection. Firstly, these algorithms rely heavily on carefully selected features, and inappropriate selection can severely compromise their detection effectiveness. Secondly, traditional virus spread network detection methods require better adaptability and are primarily suitable for smaller-scale datasets. These methods often fail to produce satisfactory experimental results for large-scale network security datasets.

General deep learning detection methods (ref. [31–34]), such as Recurrent Neural Networks (RNN) (ref. [35,36]) and Convolutional Neural Networks (CNN) (ref. [37]), have been developed to enhance traditional detection algorithms. While they offer some improvements, selecting the best model for virus-spread network intrusion detection can be challenging, as it involves comparing multiple model groups using data from a single experiment. This singleness and blind adherence may adversely affect the experimental results.

This paper proposes a novel intrusion detection model based on a graph neural network. It considers the traditional detection methods of virus propagation networks and universal deep learning detection methods. The detection accuracy and detection ability are further improved to ensure stability and adaptability. Contrast experiments select the most suitable intrusion detection model for virus spreading networks.

3. Virus Transmission Detection Model Based on Graph Neural Network

3.1. Model Overview

We systematically explore a graph neural network (GNN) model based on symmetry, random walk, multi-layer perceptron (MLP), addition (additive), and the incremental learning method (IL). This model comprises carefully designed modules, including six GNN models with unique characteristics, aiming to comprehensively address the complexity and diversity of graph structural data. First, we introduce a symmetric normalization module, which aims to improve the model's sensitivity to the symmetry between the nodes in the graph. Considering the importance of symmetry in graph neural networks, we process the node adjacency information through normalization to ensure the accuracy and stability of the model when processing graph data. Second, applying random walk normalization modules provides an effective graph traversal strategy. By randomly selecting the starting nodes and walking between the adjacent nodes of the graph, this module encourages the model to further understand the relationship between the nodes in the graph. Then, the generalization ability of the model is improved.

Regarding computing modules, we choose multi-layer perceptron (MLP) as a feed-forward neural network structure. MLP has a high degree of nonlinear transformation ability and can extract high-order features of the graph data. This design allows the model to better resolve the complex patterns in the graph and enhance its performance in dealing with complex graph structures. Furthermore, we use an addition to integrate the output of different modules to produce the final graph representation. This design helps to synthesize the information of different modules and further improve the expression ability of the model. The addition methods perform excellently in multiple tasks, such as node classification, graph classification, and link prediction. Finally, introducing the incremental learning method (IL) allows the model to update new data without retraining the entire

model. This feature is critical when processing dynamic graph data or gradually growing datasets, enabling the model to adapt to new information and make timely adjustments. The incremental learning method improves the model's generalization ability and enhances its utility in real-world applications. To validate the performance of the proposed GNN model, we perform a comprehensive experimental validation of the CTU-13 invasion detection dataset. The experimental results clearly show that the six proposed GNN models perform well in node classification, graph classification, and link prediction tasks. These results fully confirm the validity and reliability of our proposed method and provide new perspectives and tools for the in-depth analysis of graph structure data.

We propose a GNN model based on symmetry, random walk, MLP, additive, and incremental learning methods. Six GNN models with unique features are constructed by carefully combining the modules. The experimental results fully demonstrate these models' excellent performance and good generalization ability in the graph structure data analysis task. We will continue to study more efficient graph normalization modules and computational modules to improve the GNN model's overall performance.

3.2. GNN Model of Virus Transmission Network Detection

Firstly, we design the space of the GNN layer and use the GNN model with different layer structures to detect the virus-spreading network. We define the virus transmission network topology as g ∈ {V, E, M}, where V is the node{v1, . . ., vn}and E are edge sets, and M is the adjacency matrix. We also design the matrix M: a_{ii}=$\sum_{j=1}^{n} m_{ij}$. The GNN layer is calculated as follows:

$$O^{(l)} = L_\beta(\overline{O}^{(l-1)}, \overline{S}), \overline{O}^{(l-1)} = C\left(O^{(l-1)}\right) \quad (1)$$

where $O^{(l)}$ is the output of the l layer and $O^{(l-1)}$ the characteristic output of the previous layer. $l(\cdot)$ represents the execution function and is its learning parameter set. In (1), $C(\cdot)$ corresponds to the selection of the calculation module with multi-layer perceptron, the additive method, and the incremental learning method. Considering the selection of the normalization module, two methods are available: symmetric and random walk. These two modules constitute the design space for the GNN layer, as depicted in Figure 1. The various designs of GNN models correspond to different choices of these two modules. Now, let us delve into a closer examination of these two modules and their expressions.

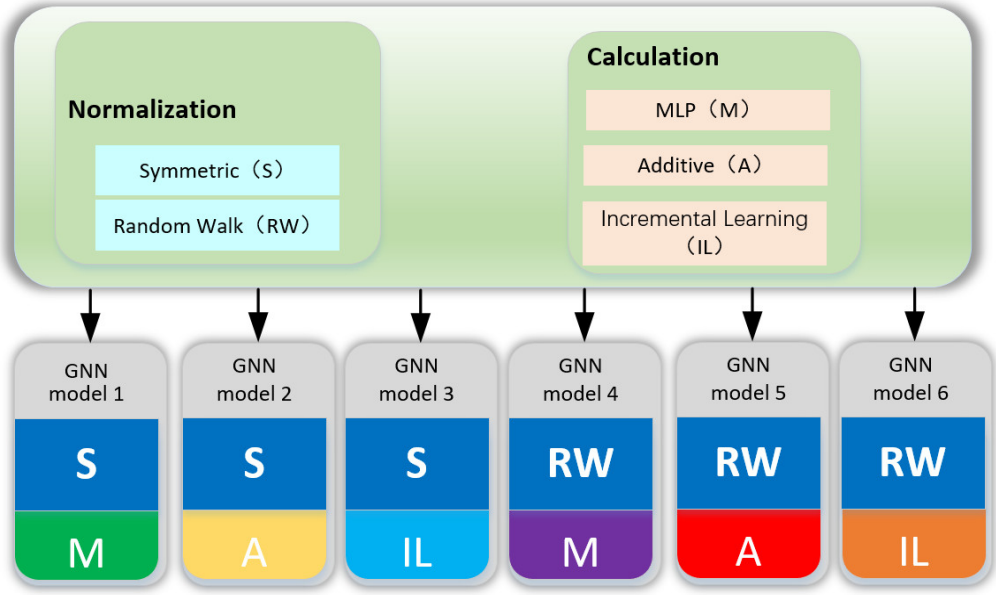

Figure 1. Schematic diagram of GNN model design.

Normalization module: The two methods are expressed as:

$$\text{Symmetric}: \ O^{(l)} = \sigma((\hat{A}_a X \theta^{(0)})\theta^{(l)}) \tag{2}$$

$$\text{Random walk}: \ O^{(l)} = \sigma((\hat{A}_a X W^{(0)})W^{(l)}) \tag{3}$$

$\sigma(\cdot)$ is the activation function, and Softmax is used as the activation function. Equation (2) is the calculation formula of the symmetric method, and Equation (3) is the calculation formula of the random walk method, where $\hat{A}_a$ is the symmetric normalized adjacency matrix; X is the input node characteristic matrix, which is used to standardize the features before aggregation; and θ and W represent the weight matrix. Both normalization methods are defined below as $\overline{F}$.

The advantage of symmetric as a normalization module is that it can maintain the symmetry of the graph data and will not break the symmetry property in the graph structure. It is also beneficial for some graph tasks, as preserving symmetry better captures the relationships between nodes and can produce stable feature representations that are relatively stable to noise and variation in the graph data. The advantage of random walk as a normalization module is its ability to capture local and global information between nodes, thereby generating rich feature representations. This feature representation includes the structural properties of nodes in the graph, which helps to improve the performance of the model. It has a wide range of applications, and its design does not depend on the specific graph structure, so it has strong generality and applicability.

Calculation module: According to the different calculation methods of the GNN layer for the feature topology structure, the following two methods are proposed:

$$MLP: \ O^{(l)} = \sigma(MLP(\overline{F}O^{(l-1)}W^{(l)})) \tag{4}$$

$$\text{Additive}: \ O^{(l)} = \sigma(\overline{F}\widetilde{O}^{(l-1)}W^{(l)}), \ \widetilde{O}^{(l-1)} = f(O^{(l-1)}) \tag{5}$$

$$\text{IL}: \ O^{(l)} = \sigma(\eta_0 e^{-\frac{1}{l_p}}) \tag{6}$$

Equation (4) is the calculation formula of the multi-layer perceptron, where MLP$(\cdot)$ is the multi-layer perceptron function. Equation (5) is the calculation formula for the additive method and $f(\cdot)$ is the original characteristic function. Equation (6) is the calculation formula of the incremental learning method, where η_0 is the initial learning rate and l_p is the learning rate parameter.

The MLP contains two hidden layers, each containing 128 neurons, and uses ReLU as the activation function. In order to improve the efficiency of the model, by increasing the depth and width of the hidden layer, the complex features in the network data can be better captured to improve the accuracy and generalization ability of the model. In addition, we experimentally select a learning rate of 0.001, a regularization parameter of 0.01, and a batch size of 64. Through hyper parameter optimization, these hyper parameters are adjusted to improve the model's performance further and make it more suitable for this model. The appropriate learning rate and batch size can speed up the convergence speed of the model to save training time. The appropriate regularization can improve the model's generalization ability, reduce the error on unseen data, and control the complexity of the model. Therefore, the efficiency and practicability of the model improve. Moreover, the choice of activation function can reduce the model's computational complexity and accelerate the model's reasoning speed so that the network intrusion detection system can detect and respond to the network traffic faster.

For the additive calculation module, we experimentally select a learning rate of 0.001, a regularization parameter of 0.01, and a batch size of 64. For the IL calculation module, we experimentally select a learning rate of 0.01, a regularization parameter of 0.001, and a batch size of 64.

The most notable characteristic of MLP as a calculation module is that it is a feedforward neural network and can deal with nonlinear relationships. The multi-layer structure

of MLP allows it to capture complex nonlinear relationships in graph data, and it can also adapt to different data complexity by adjusting the size and number of hidden layers, making it suitable for tasks that need to capture complex relationships in graph data. The characteristic of additive as a calculation module is that it can combine features in an additive way and capture the interaction between features. The additive method is generally fast to compute and easy to implement and debug, making it suitable for graph data tasks that require simple and efficient feature representations and strong model interpretability. The characteristic of IL as a calculation module is that it allows the model to learn online after receiving new data without retraining the entire model. IL can process new data in real time so that the model can be continuously optimized over time. It also avoids the overhead of repeatedly training the whole model, saves computing resources, and is suitable for tasks that need to process dynamic graph data or continuously receive new data.

By combining the different methods of the normalization and calculation modules, we obtain six GNN models. We then train these models and collect the corresponding experimental data. The pseudo-code of the algorithm is in Algorithm 1:

Algorithm 1 GNN model training process of virus transmission network

Input: original datasets D;
Output: Accuracy and interpretability scores r of six GNN models;
1. Raw data preprocessing;
2. **while** original datasets D **do**;
3. Edges with scores greater than or equal to the critical value are taken out to form a star topology model;
4. **end while;**
5. Design and training of GNN model for virus transmission network detection:
6. GNN layer design: Determine the normalization method and calculation method;
7. Train the GNN model;
8. **while** star topology A, node feature X **do**;
9. Through $O = GNN_{\Phi}(A, X)$, get input A get;
10. **end while;**
11. GNN model analysis;
12. Calculate the detection accuracy ACC_d of the GNN model according to (7);
13. Calculate the interpretability score r of the GNN model according to (9);
14. return ACC_d, r;

4. Results and Analysis

4.1. Experimental Datasets

We conducted experiments on the CTU-13 network intrusion detection dataset, which was created by the Czech University of Technology (CTU) in the Czech Republic in 2011. The purpose of this dataset was to capture real network traffic mixed with regular and background traffic on a large scale. It comprises 13 scenarios representing virus transmission networks, including IRC, SPAM, CF, PS, DDoS, FF, P2P, US, and HTTP. Each scenario depicts a specific virus propagation network that utilizes different protocols and performs various operations. The characteristics of the virus spread network scenarios are summarized in Table 1. From this dataset, we selected specific scenarios and generated 1320 graphs. These graphs were randomly divided into training, validation, and test sets, with proportions of 70%, 20%, and 10%, respectively. Each graph has a varying number of nodes and edges, approximately containing 28,200 nodes and 719,000 edges per graph. In addition, all graphs include a predetermined number of abnormal nodes, with 3000 abnormal nodes present in each graph.

Table 1. CTU-13 datasets analysis and statistics.

ID	IRC	SPAM	CF	PS	DDOS	FF	P2P	US	HTTP	Abnormal Edge	Normal Edge	Background Flow
1	✓	✓	✓							39,933	30,387	2,753,290
2	✓	✓	✓							18,839	9120	1,778,061
3	✓			✓				✓		26,759	116,887	4,566,929
4	✓				✓			✓		1719	25,268	1,094,040
5		✓		✓					✓	695	4679	124,252
6				✓						4431	7494	546,795
7									✓	37	1677	112,337
8				✓						5052	72,822	2,875,282
9	✓	✓	✓	✓						179,880	43,340	2,525,565
10	✓				✓			✓		106,315	15,847	1,187,592
11	✓				✓			✓		8161	2718	96,369
12							✓			2143	7628	315,675
13		P		✓					✓	38,791	31,939	1,853,217

4.2. Scoring Criteria

After several pre-tests, we determined a critical value based on several groups of data obtained in advance. We removed the edges with scores greater than or equal to the essential value to form a model of the star topology structure. Our experiment must improve the accuracy of abnormal edge detection and reduce the percentage of selected edges. Therefore, we set two factors to reflect the detection precision: ACC_d and extraction density ρ, respectively. After many experiments, we found that the general rule is that the abnormal edges of the star topology are usually distributed outward from the central node. Therefore, when we count the number of abnormal edges, we can start to search from the peripheral father node, find the first abnormal node, stop searching, and then count the number of normal edges on this link. When we have traversed all the links, we can identify the number of abnormal edges by subtracting the number of normal edges from the total number of edges. Since the number of normal edges is much smaller than the number of abnormal edges in most individuals of the star topology, counting the number of abnormal edges in this way can significantly optimize the algorithm time. The calculation formula for detection precision ACC_d and extraction density ρ is given below:

$$ACC_d = \frac{S_{all} - S_{nor}}{G_{abn}} \tag{7}$$

$$\rho = \frac{S_{all}}{G_{all}} \tag{8}$$

where S_{all} is the total number of edges in the star topology, S_{nor} is the statistical number of normal edges in the star topology, G_{abn} is the number of abnormal edges in the initial graph, and G_{all} is the total number of edges in the initial graph. Then, through integrated analysis of these two factors, we obtained a model scoring formula r:

$$r = \frac{ACC_d - \ln^{\rho}}{2 - 2 \times \ln^{\rho_{\min}}} \tag{9}$$

4.3. Experimental Results and Analysis

(1) We must determine the optimal depth of each GNN model based on the performance of different GNN models at different depths to ensure that different GNN models are in the best state during the comparison. We conducted experiments on the virus transmission network using different models. To demonstrate objectivity and simplicity, we used the line chart in Figure 2 to show the changes in the models with varying performances under different depths. Depth 5–9 was selected for the chart description. It can be seen from Figure 2 that different design models all performed well. The F1, Acc, and AUC values of varying design models at different depths are

more significant than 0.96, 0.99, and 0.97, respectively. Since the value of F1 has the most significant variation amplitude under the same model, we choose the value of F1 as the index to select the optimal depth under the same model.

(2) We can evaluate the model numerically using the extraction density, detection accuracy, and model score value, making it more standardized, specific, and reliable.

(3) As can be seen from Table 2, when the extraction densities of the six models are close, the detection accuracy of GNN model 4, which is composed of the random walk method and multi-layer perceptron, is 91.7%, which is much higher than that of the other six models. Its model score is 0.683, which is the highest among the six groups. Moreover, we find that when the calculation modules are the same, and the normalization module is the random walk method, the variance in the detection accuracy is generally smaller than that when the normalization module is symmetric, indicating better detection stability.

(4) It can be seen from Figure 3 that the variation curves of each model at different depths and the comparison between each model show that the GNN model formed by stacking random walk and MLP has higher detection accuracy than other models at different depths. According to Table 3, the detection accuracy ACC_d of the GNN model formed by random walk and the MLP stack is higher than that of the machine learning model and deep learning model, and the model score value r of this GNN model is much higher than that of other models.

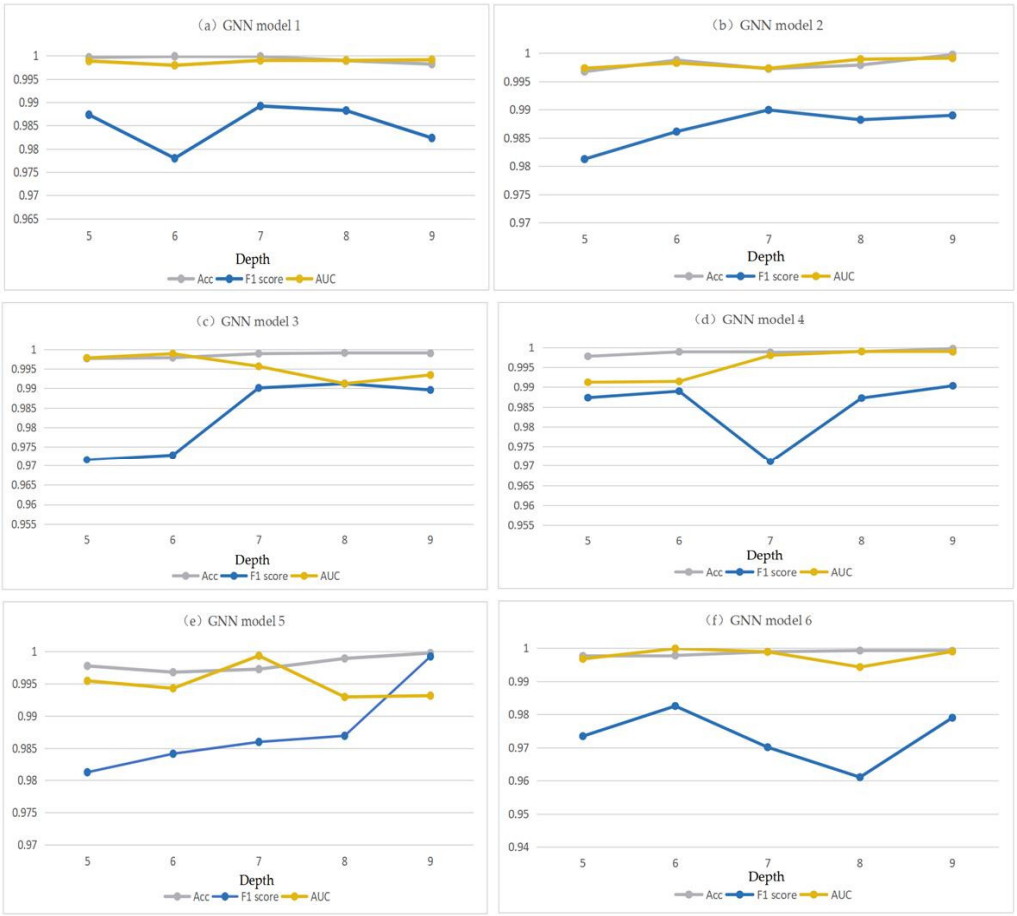

Figure 2. Performance comparison of the GNN model (**a**) symmetric and additive (**b**) symmetric and MLP (**c**) symmetric and incremental learning (**d**) random walk and additive (**e**) random walk and MLP (**f**) random walk and incremental learning at different depths.

Table 2. Six GNN models in the best state of the indicators.

GNN Model	Normalization	Calculation	Extraction Density ρ	Detection Accuracy ACC_d	Score r	Variance of Detection Accuracy
1	Symmetric	MLP	24.3%	65.1%	0.428	0.0058
2	Symmetric	Additive	24.2%	63.2%	0.459	0.0061
3	Symmetric	IL	25.1%	67.3%	0.512	0.0052
4	Random walk	MLP	25.4%	91.7%	0.683	0.0023
5	Random walk	Additive	25.2%	68.9%	0.487	0.0032
6	Random walk	IL	25.4%	63.5%	0.493	0.0029

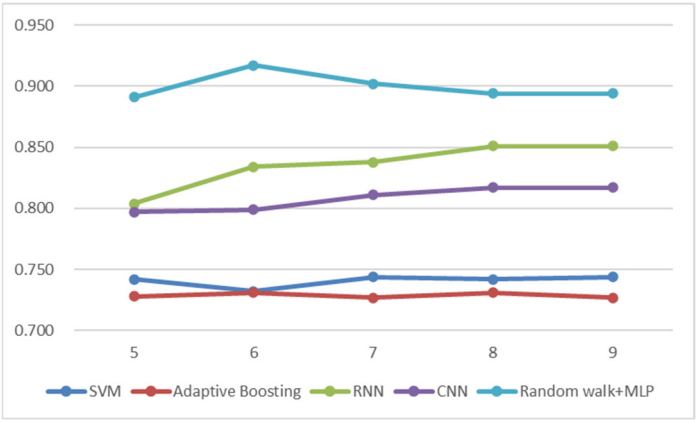

Figure 3. The detection model of each model at different depths.

Table 3. Performance comparison of each model.

Category	Algorithm	Detection Accuracy ACC_d	Score r
Machine learning	SVM	74.4%	0.446
	Adaptive Boosting	73.1%	0.439
Deep learning	RNN	85.1%	0.468
	CNN	81.7%	0.454
GNN	Random walk + MLP	91.7%	0.683

5. Conclusions

In this paper, we proposed an intrusion detection model based on a graph-based neural network for a virus propagation network. Then, we explained each part of the model in detail, including the algorithm process, the model scoring formula, etc., and compared the model with other classical neural network structure models. We used experiments and formulas to obtain the test accuracy and model evaluation values. Through theoretical derivation and comparison, we concluded that our GNN Model 4, which is composed of the random walk method and multi-layer perceptron, can detect virus propagation networks with higher precision.

This model is of great significance to the intrusion detection of virus propagation networks. However, this paper still has some shortcomings and some aspects of the findings should be studied further. For example, although the amount of data used in this paper is already large, it is still necessary to further supplement the data of different virus propagation network sample scenarios in the later stage to ensure the authenticity and accuracy of the experiment. The experiment in this paper was only implemented on the existing fixed datasets and has not been used for detection in the natural network environment. In the later stage, it needs to perform the detection in real time by crawling a large number of network data to evaluate its performance. Training and deploying a GNN model usually requires a large amount of computing resources, memory requirements,

and computational complexity, which are also potential limitations of the proposed model. Moreover, there are many aspects that can be optimized and enhanced to further improve the detection accuracy and precision of the model. For example, a new formula can be designed to find a computing module that is more suitable for a virus propagation network and to form a new GNN model with the random walk method. Researchers can also consider optimizing the parameters of the original formula. These provide the direction for further improving the precision of the model and give the research important theoretical and practical significance and scientific research value.

Author Contributions: Conceptualization, X.Y.; writing—review and editing, M.P.; original draft, X.C.; editing and revision, Y.Z.; formal analysis, B.G.; software, Z.Z.; writing—supervision, J.L. and D.L.; project administration, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Humanities and Social Sciences Planning Foundation of the Ministry of Education of China under Grant no. 22YJAZH063, in part by the University Student Science and Technology Innovation Activity Plan (Xinmiao Talent Plan) of Zhejiang Province under Grant no. 2023R465015 and in part by the College Students Innovation and Entrepreneurship Training Program of China under Grant no. 202310349030.

Data Availability Statement: Data and material are available at https://github.com/Xian-20145131/Research_on_Virus_Propagation_Network_Intrusion_Detection_based_on_Graph_Neural_Network.git (accessed on 4 April 2024).

Acknowledgments: We are grateful to reviewer for his/her effort reviewing our paper.

Conflicts of Interest: The authors declare that they have no conflicts of interest.

References

1. Dung, N.Q.; Viet, L.H. Directed-System-Call-Graph Feature for IoT Botnet Detection. *J. Intell. Fuzzy Syst.* **2022**, *43*, 5453–5470. [CrossRef]
2. Rezaei, A. Using Ensemble Learning Technique for Detecting Botnet on IoT. *SN Comput. Sci.* **2021**, *2*, 148. [CrossRef]
3. Popoola, S.I.; Adebisi, B.; Hammoudeh, M.; Gui, G.; Gacanin, H. Hybrid Deep Learning for Botnet Attack Detection in the Internet-of-Things Networks. *IEEE Internet Things J.* **2021**, *8*, 4944–4956. [CrossRef]
4. Wan Nur Hidayah, I.; Anuar, S.; Selamat, A.; Krejcar, O.; Crespo, R.G.; Herrera-Viedma, E.; Fujita, H. Multilayer Framework for Botnet Detection Using Machine Learning Algorithms. *IEEE Access* **2021**, *9*, 48753–48768.
5. Alothman, Z.; Alkasassbeh, M.; Al Haj Baddar, S. An efficient approach to detect IoT botnet attacks using machine learning. *J. High Speed Netw.* **2020**, *26*, 241–254. [CrossRef]
6. Hiebeler, D.E.; Audibert, A.; Strubell, E.; Michaud, I.J. An epidemiological model of internet worms with hierarchical dispersal and spatial clustering of hosts. *J. Theor. Biol.* **2017**, *418*, 8–15. [CrossRef]
7. Maniriho, P.; Mahmood, A.N.; Chowdhury, M.M. A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends and challenges. *J. Theor. Biol.* **2017**, *418*, 8–15. [CrossRef]
8. Ashik, M.; Jyothish, A.; Anandaram, S.; Vinod, P.; Mercaldo, F.; Martinelli, F.; Santone, A. Detection of Malicious Software by Analyzing Distinct Artifacts Using Machine Learning and Deep Learning Algorithms. *Electronics* **2021**, *10*, 1694. [CrossRef]
9. Dounavi, H.M.; Mpanti, A.; Nikolopoulos, S.D.; Polenakis, I. A graph-based framework for malicious software detection and classification utilizing temporal-graphs. *Electronics* **2021**, *10*, 1694. [CrossRef]
10. Ali, M.; Shiaeles, S.; Clarke, N.; Kontogeorgis, D. A proactive malicious software identification approach for digital forensic examiners. *J. Inf. Secur. Appl.* **2019**, *47*, 139–155. [CrossRef]
11. Awad, M.; Fraihat, S.; Salameh, K.; Al Redhaei, A. Examining the Suitability of NetFlow Features in Detecting IoT Network Intrusions. *Sensors* **2022**, *22*, 6164. [CrossRef] [PubMed]
12. Tao, Y.; Ruiqi, Y. Detecting Abnormal Interactions among Intranet Groups Based on Netflow Data. *IOP Conf. Ser. Earth Environ. Sci.* **2020**, *428*, 012039.
13. Tania, K.P.; Putra, N.S.; Made, D.W. NetFlow dalam Monitoring Penggunaan Internet. *Maj. Ilm. Teknol. Elektro* **2017**, *16*, 86.
14. Liu, L.; Wang, P.; Lin, J.; Liu, L. Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning and Deep Learning. *IEEE Access* **2021**, *9*, 7550–7563. [CrossRef]
15. Li, Z.; Hou, J.; Wang, H.; Wang, C.; Kang, C.; Fu, P. Ethereum Behavior Analysis with NetFlow Data. *IEICE Proceeding Ser.* **2019**, *59*, TS2–TS4.
16. Sun, A.Y.; Jiang, P.; Yang, Z.L.; Xie, Y.; Chen, X. A graph neural network (GNN) approach to basin-scale river network learning: The role of physics-based connectivity and data fusion. *Hydrol. Earth Syst. Sci.* **2022**, *26*, 5163–5184. [CrossRef]

17. Govindaraju, S.; Vinisha WV, R.; Shajin, F.H.; Sivasakthi, D.A. Intrusion detection framework using auto-metric graph neural network optimized with hybrid woodpecker mating and capuchin search optimization algorithm in IoT network. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e7197. [CrossRef]
18. Sun, L.; Liu, T.; Wang, D.; Huang, C.; Xie, Y. Deep learning method based on graph neural network for performance prediction of supercritical CO_2 power systems. *Appl. Energy* **2022**, *324*, 119739. [CrossRef]
19. Chen, Y.; Tang, X.; Qi, X.; Li, C.G.; Xiao, R. Learning graph normalization for graph neural networks. *Neurocomputing* **2022**, *493*, 613–625. [CrossRef]
20. Peng, L.; Hu, R.; Kong, F.; Gan, J.; Mo, Y.; Shi, X.; Zhu, X. Reverse Graph Learning for Graph Neural Network. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *35*, 4530–4541. [CrossRef]
21. Zhou, J.; Li, L.; Zeng, A.; Fan, Y.; Di, Z. Random walk on signed networks. *Phys. A Stat. Mech. Its Appl.* **2018**, *508*, 558–566. [CrossRef]
22. Xu, X.K.; Zhu, J.J. Flexible sampling large-scale social networks by self-adjustable random walk. *Phys. A Stat. Mech. Its Appl.* **2016**, *463*, 356–365. [CrossRef]
23. Hilário, M.; Den Hollander, F.; Sidoravicius, V.; Soares dos Santos, R.; Teixeira, A. Random walk on random walks. *Electron. J. Probab.* **2015**, *20*, 1–35. [CrossRef]
24. Naskath, J.; Sivakamasundari, G.; Begum, A.A.S. A Study on Different Deep Learning Algorithms Used in Deep Neural Nets: MLP SOM and DBN. *Wirel. Pers. Commun.* **2022**, 21–24. [CrossRef] [PubMed]
25. Kumar, M.; Verma, G. Machine Learning Intrusion Detection System Based on MLP and RNN Stochastic Optimization Technology. *J. Res. Sci. Eng.* **2022**, *4*. [CrossRef] [PubMed]
26. Ding, Q.; Yin, S.; Liu, L.; Wang, C. Hardware Trojan detection research based on MLP. *J. Phys. Conf. Ser.* **2020**, *1684*, 012065. [CrossRef]
27. Buczak, A.L.; Guven, E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun. Surv. Tutor.* **2015**, *18*, 1153–1176. [CrossRef]
28. Ahmad, I.; Basheri, M.; Iqbal, M.J.; Rahim, A. Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. *IEEE Access* **2018**, *6*, 33789–33795. [CrossRef]
29. Hu, C.J.; Wang, J. The SVM and Layered Intrusion Detection System Based on Network Hierarchical. In *Internet of Things, Proceedings of the International Workshop, IOT 2012, Changsha, China, 17–19 August 2012*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 312, p. 312.
30. Manghnani, T.; Thirumaran, T. Computational CBGSA–SVM Model for Network Based Intrusion Detection System. In *Applications and Techniques in Information Security, Proceedings of the 10th International Conference, ATIS 2019, Thanjavur, India, 22–24 November 2019*; Springer: Singapore, 2019; Volume 1116, p. 1116.
31. Louk, M.H.L.; Tama, B.A. Dual-IDS: A bagging-based gradient boosting decision tree model for network anomaly intrusion detection system. *Expert Syst. Appl.* **2023**, *213*, 119030. [CrossRef]
32. Srikanth, Y.M.; Kalpana, R. Recurrent nonsymmetric deep auto encoder approach for network intrusion detection system. *Meas. Sens.* **2022**, *24*, 100527.
33. Shen, Y. An Intrusion Detection Algorithm for DDoS Attacks Based on DBN and Three-way Decisions. *J. Phys. Conf. Ser.* **2022**, *2356*, 012044. [CrossRef]
34. Wang, J.; Liu, H.; Liu, F. Research on Deep Learning Method Based on Blockchain and Intrusion Detection Model. *J. Phys. Conf. Ser.* **2022**, *2356*, 012057. [CrossRef]
35. Deore, B.; Bhosale, S. Intrusion Detection System Based on RNN Classifier for Feature Reduction. *SN Comput. Sci.* **2022**, *3*, 114. [CrossRef]
36. Sheikhan, M.; Jadidi, Z.; Farrokhi, A. Intrusion detection using reduced-size RNN based on feature grouping. *Neural Comput. Appl.* **2012**, *21*, 1185–1190. [CrossRef]
37. Gan, B.; Chen, Y.; Dong, Q.; Guo, J.; Wang, R. A convolutional neural network intrusion detection method based on data imbalance. *J. Supercomput.* **2022**, *78*, 19401–19434. [CrossRef]