

Structural Node Representation Learning for Detecting Botnet Nodes

Justin Carpenter[1], Janet Layne[1], Edoardo Serra[1], Alfredo Cuzzocrea[2(✉)], and Carmine Gallo[2]

[1] Computer Science Department, Boise State University, Boise, ID, USA
JustinCarpenter836@u.boisestate.edu, {janetlayne, edoardoserra}@boisestate.edu
[2] iDEA LAB, University of Calabria, Rende, Italy
alfredo.cuzzocrea@unical.it

Abstract. Private consumers, small businesses, and even large enterprises are all more at risk from *botnets*. These botnets are known for spearheading *Distributed Denial-Of-Service (DDoS) attacks*, *spamming* large populations of users, and causing critical harm to major organizations. The development of *Internet-of-Things (IoT)* devices led to the use of these devices for *cryptocurrency mining*, in transit data *interception*, and sending *logs* containing private data to the *master botnet*. Different techniques have been developed to identify these botnet activities, but only a few use *Graph Neural Networks (GNNs)* to analyze host activity by representing their communications with a *directed graph*. Although GNNs are intended to extract *structural graph properties*, they risk to cause *overfitting*, which leads to failure when attempting to do so from an unidentified network. In this study, we test the notion that *structural graph patterns* might be used for efficient botnet detection. In this study, we also present SIR-GN, a *structural iterative representation learning methodology for graph nodes*. Our approach is built to work well with untested data, and our model is able to provide a *vector representation* for every node that captures its structural information. Finally, we demonstrate that, when the collection of node representation vectors is incorporated into a *neural network classifier*, our model outperforms the state-of-the-art GNN based algorithms in the detection of bot nodes within unknown networks.

Keywords: Machine Learning · Botnet Detection

1 Introduction

In this paper, we present a method for detecting *botnets*. When developing this technique, we considered the key problems with detecting botnets, such as the rise in devices that are always connected to the network. Due to the constant active connection features of these devices, the exponential growth of IoT devices has increased the chances for cybercriminals to expand and grow these botnets. Wide-range cyberattacks have also become more common, which are challenging to perform without having a lot of devices being connected to the network. *Spam*, *DDoS* attacks, the transmission of *viruses* and

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023, LNCS 13956, pp. 731–743, 2023.
https://doi.org/10.1007/978-3-031-36805-9_47

malware, password and credential *theft*, and even *cryptocurrency mining* are some of the main cyberattacks that use botnets.

Many effective botnet detection techniques rely on the use of *GNN (Graph Neural Networks)* models to extract patterns that characterize the behavior of the botnet. Although GNNs are capable of acquiring structural information, they still have limitations in relation to the nodes' proximity, which, based on concepts of *similarity*, will allow them to interact with machines that, despite being thought of as reliable, are actually botnet-affiliated machines. As a result, we may assert that botnet identification using the GNN model may be poor if the model was trained on a different network than the network that would be evaluated. Therefore, in this work, we provide the learning technique of the *inferential SIR-GN* graphical representation in order to achieve superior outcomes. Even across very vast networks, it is possible to identify the nodes belonging to the botnet by combining this technique with a *neural network* classification model. Significant benefits of inferential SIR-GN include the preservation of a *graph's structural information*, which can be utilized to identify malicious software even in networks that are completely transparent. For networks when the topology of the training data and the test data is the same, the behavior of the Inferential SIR-GN is almost exactly like that of the GNN model during the *inference* phase. Additionally, it significantly improves when these data topologies are different. Since botnets are constantly developing, we can therefore say with certainty that the Inferential SIR-GN model is a more effective detection strategy than GNN models.

The remaining Sections of this work are structured as follows: Sect. 2 presents the earlier detection techniques, and Sect. 3 describes the datasets that were employed. The Inferential SIR-GN methodology is finally demonstrated in Sect. 4 before being evaluated in Sect. 5. Finally, we give our work's conclusions in Sect. 6.

2 Analysis of State-of-the-Art-Approaches

2.1 The Detection of Botnets

Cybercriminals must first persuade people to install *malware* on their systems in order to gain control over the majority of devices, and the majority of malware used for this purpose is freely available online. We use the *Mirai malware* as an illustration, which preys on Linux-based IoT systems including *routers*, *IP cameras*, and any *home automation* equipment. This botnet was utilized against organizations like *Krebs on Security*, the French *web hosting service OVH*, and even the *DNS provider Dyn*, which is a crucial service for regular internet communications, to cause widespread disruptions and produce internet traffic of up to 1 Tbit/s. The Mirai malware is regarded as the first of its kind, and even though its developers were found and turned over to the police, numerous additional varieties have since emerged, including the *PureMasuta*, *Okiru*, *Satori*, and *Masuta* malware. Figure 1 depicts the conceptual design of a Mirai-based bot that transforms IoT devices into *proxy servers*.

Both the complexity and attacks of botnets are constantly evolving. Fortunately, the methods for identifying botnets have evolved along with the evolution of the networks themselves. It is well known that botnets have the ability to function in a way that tries to *evade* detection. In this instance, a lot of *honeypots* are made to draw in botnets, especially

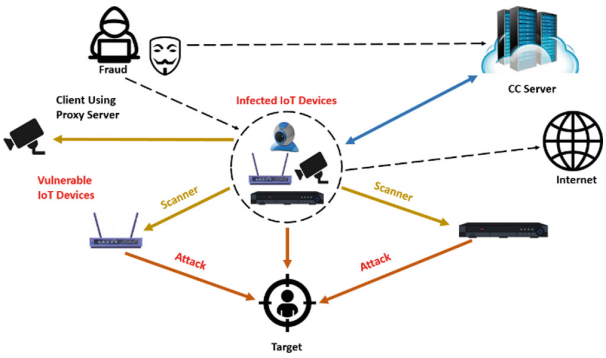


Fig. 1. A Mirai-based bot turning IoT devices into proxy servers

ones that are known to them and that they avoid, as they are able to *bypass* the controls and therefore avoid detection [1]. The *proliferation* of *P2P (Peer-To-Peer)* connections has made it challenging to detect botnets. The difference between a Client-Server mode and P2P mode connection is seen in Fig. 2.

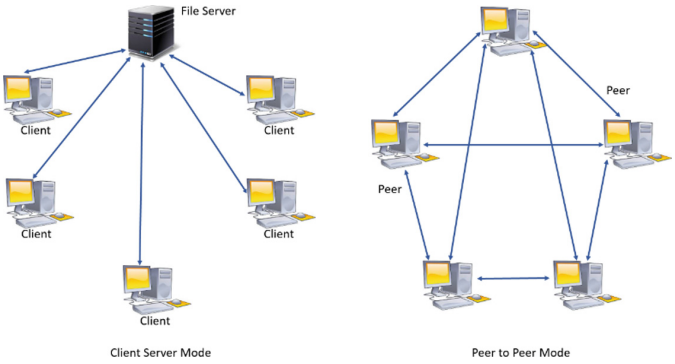


Fig. 2. Client-server and P2P mode connections

In the case of client-server botnets, if the *C&C control node* is located and isolated, the entire botnet can be found and subsequently stopped. Contrary to P2P-type botnets, which are capable of sharing *C&C commands* when discovered and have little knowledge of the other botnets [2], making detection even more difficult. In order to locate the central control node, *BotMiner* [3], for instance, uses *clusters* of nodes with similar communications and *malicious traffic*. Using *fast-flowing server networks*, P2P botnets can change the *addresses* of the C&C server node, enabling *traffic monitoring* between nodes to bypass them [4]. Because of this, the traditional approaches presented in [5] which mostly rely on static properties to characterize network traffic have been rendered worthless. These traditional methods, like *domain names* [6] and *DNS black-lists* [7], also call for a deeper understanding of the network and botnets. Therefore, this type of

strategy only works successfully provided the data is available and hasn't been altered by the botnet.

2.2 Graph Representation Learning

The use of *unsupervised learning* techniques for representing *graph data* is growing in popularity. Networks store a significant amount of information regarding relationally *structured data* in a simple *data structure*, which is a list of entities named nodes. And, *edges* which are the links that connect these nodes. Additionally, a graphical representation can be used to instantly recognize the structure of a botnet. But typical *machine learning* programs operate on a set of attributes that must be represented by graphical data structures. The similarity between nodes must be fully extracted using this type of representation. However, the structural position of the node within the network as well as that of its neighbors can serve as the foundation for the concept of similarity. For instance, the highly connected nodes will be close to one another in the feature space when a file is acquired nearby a node. Additionally, it suffices that the structures of surrounding nodes are comparable in order to classify them as similar nodes that represent a node's structural role; and a *path* between them is not necessary.

Numerous representative learning techniques have been used with considerable success in a variety of academic subjects. For instance, the *DeepWalk* method [8], which employs *NLP (Neuro Linguistic Programming)* and is based on the *Word2Vec* algorithm [9, 10] for the *Skip-Gram* model, seeks to anticipate the words in the context starting from a core word in order to optimize all the nearby nodes. In addition, DeepWalk extends the Skip-Gram model by switching from word sequences to graphs. It uses a *randomized path traversal* mechanism to establish this transformation and to provide *insights* about localized network topologies. Here's how the DeepWalk procedure operates:

1. For each node, perform *N "random passes"* starting at that node.
2. Treat each walk as a sequence of node-id strings.
3. Train a Word2Vec model using the Skip-Gram algorithm on the string sequences obtained previously.

This manifests as a node representation learning approach with similar neighbor nodes that is based on *node sharing* and *connectivity*. Due to the necessity of traversing the graph, a major issue arises when the graph is partially connected because there are no links between the nodes. The *Node2vec* algorithm [12] uses random walks around a network beginning at a *target node*, and this methodology is comparable to that used by that algorithm. It removes the connectivity requirement and results in performance upgrades. Each of these techniques is effective at capturing connectivity data between nodes, but they fall short when it comes to maintaining crucial network architecture characteristics for bot machine detection. The most effective techniques for learning representations while maintaining structural information are Graph Neural Networks [13] such as *Graph Convolutional Neural Network*, *Struct2Vec* [14], *GraphWave* [15] and *Iterative Procedures*. Only the inferential SIR-GN iterative method and neural networks are able to perform inferences that offer predictions for graphs that are entirely different from those used for training. Our botnet detection method's primary goal is to learn

the structural representation of inference-capable algorithms. However, other relevant efforts in the areas of *AI* and machine learning are mentioned in [18–30].

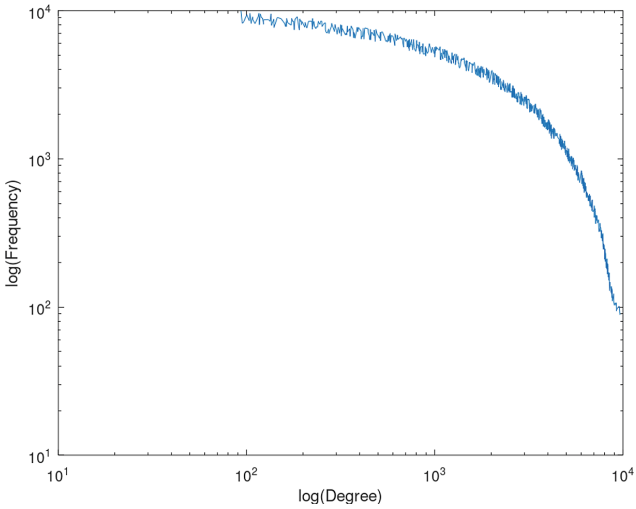


Fig. 3. Log-log distribution of the experimental data

3 Description of Datasets

We have so far discussed two different botnet models, the first of which can be quickly recognized via C&C due to its *centralized* botnet and *star pattern.* The second form, known as P2P, is *decentralized* and has nodes that connect the majority of network nodes in just one or two *hops.* P2P botnet clusters, as we previously noted, are more challenging to identify because there isn't a single core hub. The P2P model is used in the following experiments to show how representation learning applied to the graph can identify *anomalies* that are more difficult to identify when using other techniques. For our purposes, real background traffic data obtained in 2018 related to IPs on a *backbone* monitored by *CAIDA* [31] was employed. Since the monitored traffic is *aggregated*, it is impossible to identify the users who generated it. We chose this background traffic at random on a selection of nodes thought to be part of the botnet in order to take into account the various P2P botnet topologies. In these investigations, we built controlled networks using the P2P topologies De Bruijn [32], Kademlia [33], Chord [34], and Leetchord [35]. Additionally, in order to detect botnet attacks in communication flow, we also use a genuine P2P network [36]. The Log-Log graph, which depicts the degree of distribution within the graphs, is shown in Fig. 3.

As seen in Fig. 3, the *frequency* (number of nodes) decreases as the degree (number of edges) of the nodes increases. This pattern suggests that most nodes have degrees under two and that there aren't many nodes that are well related. Each network has 960 P2P graphs with an average of 144,000 nodes, over 1.5 million edges, and 10,000 botnet

nodes within each synthetic graph. The average cluster size in the graphs is 0.007, and each network contains 960 P2P graphs. Based on the existing botnet network, which consists of 144k nodes and 3k botnet nodes, this figure was calculated. 10,000 botnet nodes were utilized in the datasets we used for training, while 10,000, 1,000, and 100 botnet nodes were used in the test dataset. As less than 10% of the network's nodes are part of botnets, all of these networks are severely imbalanced. Figure 4 displays the values for the mean node structure across the various data sets.

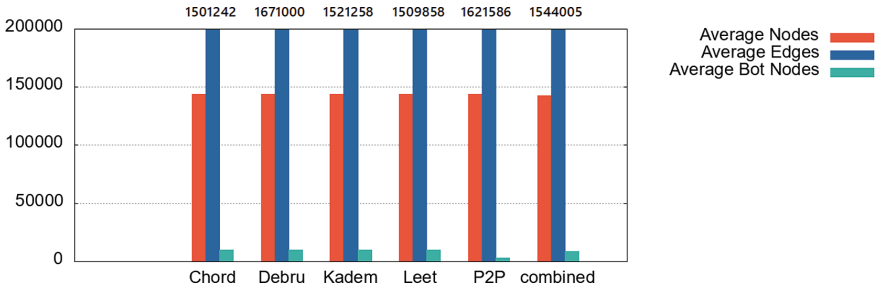


Fig. 4. Average node structure of datasets

4 The Proposed Methodology: Inferential SIR-GN

Our proposed approach is based on the inferential SIR-GN, which is an iterative structural representation learning process with inference capabilities. The symbols we utilize in our methodological explanation are shown in Table 1.

Table 1. Notations used in model description

Notation	Description
nc	The number of clusters chosen for node representation
ngc	The number of clusters chosen for graph representation
k	The depth of exploration, equal to a node's k-hop neighborhood

Layne and Serra provided a detailed description of the Inferential SIR-GN, which is used to extract node representations from directed graphs, in their work [17]. The model is built using the SIR-GN methodology, which was first presented in M. Joaristi and E. Serra [16]. Using this methodology, a node's representation is iteratively updated by first characterizing, then aggregating, its neighbors. Each iteration size of a node representation is determined by a user-selected *nc* hyperparameter. The actual node description, which first starts as the node degree, is clustered into a *nc K-Means* cluster to provide node descriptions. At each iteration, the representation is *normalized* before

clustering, thus the distance from each cluster *centroid* is changed into a node probability of being a member of that cluster. The neighbors of a node are then aggregated in its structural description once it has been updated by adding the probability of all the nearby nodes that are part of each cluster for that cluster.

The number of neighbors held by each node in each cluster is represented by the same number of nodes as predicted. Each iteration represents a deeper level of study, and after k iterations, a node description will include information about its k-hop neighborhood structure. The inferential SIR-GN differs from the conventional model in some ways. The structural descriptions of each node are really concatenated into a broader representation, which captures the evolution of the structure and information through a more in-depth examination of the neighborhood. First, it is tied to each interaction. The final representation is compressed into a dimension selected as a hyperparameter in a *Principal Component Analysis (PCA)* to prevent information degradation as a result of the size growth.

For *directed graphs*, a node's initial representation consists of two vectors of dimension *nc*, one of which represents the node's internal degree and the other its exterior degree. Prior to clustering, these two vectors are concatenated. This vector is then clustered after each iteration, and the nearby nodes are then gathered together. For the next iteration, two intermediate vectors one for neighbors and one for non-neighbors are concatenated together to execute the aggregate in the case of direct data. By pre-training the K-Means algorithm and scaling at each iteration, our suggested model is able to make inferences. This procedure is performed along with the PCA model that will be used to create the final node embedding for each exploration depth where the training is established on saved and used to draw inference random graphs.

In order to make inferences, we repeatedly do normalization using pre-trained models, then clustering, then aggregation, utilizing PCA to create file representations of the end nodes during training. This approach enables the use of the same pre-trained model across several data sources while also decreasing the inference time which Layne and Serra [17] present in great depth and include a thorough algorithm that explains the *time complexity* of the model. Any *classifier* can use the structural representation vectors of SIR-GN nodes to understand the topology of botnets for automatic botnet discovery. A 3-layer neural network is also used in this work to make a final prediction about a node (machine) state as a bot.

5 Experimental Assessment and Analysis

5.1 Setup

We discussed the four basic botnet topologies used to create *synthetic datasets* in Sect. 2 of this article. Where 960 distinct graphs are produced for each topology by using that topology on actual traffic. The graphs size and number are scaled to match a real P2P dataset with 960 graphs of actual botnet attacks. For each set of graphs, representations of the structural nodes are produced using our inferential SIR-GN model. We trained Inferential SIR-GN using a collection of randomly generated graphs, as shown in the technique. Then, we calculated the nodes representation in a way that effectively allowed us to extract each node structural description. This technique produced outstanding

results utilizing a relatively small fraction of these representations that had been trained using a classifier.

A neural network (NN) classifier that outperforms all previous models that have been trained similarly can be trained using inferential SIR-GN to train the upstream of any classifier and to transfer learning. Using the ABD-GN model, which is a GNN specifically designed for identifying botnets, we will elaborate on the contrast. We will use the following comparisons to demonstrate how inferential SIR-GN is useful for generalizing invisible data:

1. To begin, we contrast the inferential classifier SIR-GN plus a neural network that was trained on 50 graphs from a dataset (botnet topology) and used to classify 96 graphs from that dataset with the ABD-GN one, which was trained on 80% (768) of the dataset graphs and used to classify a test set of 20% (on the same 96 graphs) from the same topology.
2. Next, we contrast the ABD-GN classifier, which is trained on 768 graphs from a topology and used to classify the test set, with the Inferential SIR-GN plus classifier, which is trained on 50 graphs from a single topology and used to classify the test set of 96 graphs from each of the other topology datasets and real P2P attack data. We will refer to Inferential SIR-GN plus with neural network classifier as isirgn1 in the results report to keep things simple.

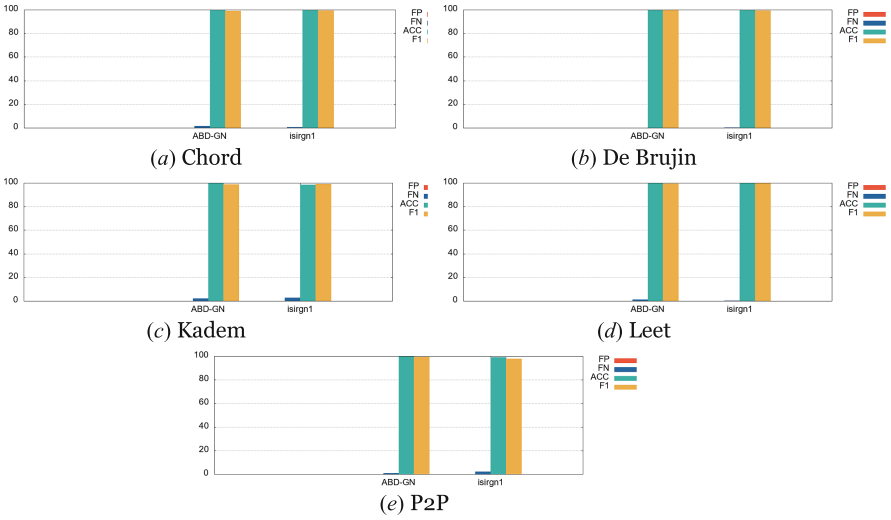
5.2 Results

Figure 5 compares the node representations created by Inferential SIR-GN and ABD-GN and displays the results of the NN classifier. We can see that, even though the model was trained to represent nodes using randomly generated graphs and specifically trained on 50 graphs from a dataset, isirgn1 performs very similarly to ABD-GN for all datasets, in contrast to the ABD-GN neural network model, which has been trained on 80% (768) of dataset graphs.

Figure 5 compares the node representations created by Inferential SIR-GN and ABD-GN and displays the results of the NN classifier. We can see that, even though the model was trained to represent nodes using randomly generated graphs and specifically trained on 50 graphs from a dataset, isirgn1 performs very similarly to ABD-GN for all datasets, in contrast to the ABD-GN neural network model, which has been trained on 80% (768) of dataset graphs.

Figure 5 illustrates the outstanding performance of the ABD-GN model for a single topology. The outcomes, however, range from excellent to very poor when these same trained models are tested against an invisible topology, as seen in Table 2. When evaluated on all other synthetic datasets, isirgn1 and ABD-GN perform equally when trained on the Kadem topology. Similar to training on Leet and Chord data, the F1-Score for ABD-GN falls below 10%, with no variance in performance on the other synthetic datasets. It's interesting to note that when Debru data is used to train each model, ABD-GN performs poorly in classification across all other topologies in addition to failing to accurately detect bots in the real-world dataset, whereas isirgn1 works well in this situation.

On test data produced from the same dataset, isirgn1 and ABD-GN behave similarly when trained on real-world datasets. But in the four stealth topologies, isirgn1 also performs well in classification tests whereas ABD-GN significantly underperforms. These



(*a*) Chord (*b*) De Brujin

(*c*) Kadem (*d*) Leet

(*e*) P2P

Fig. 5. Botnet detection results

Table 2. F1-Scores of the models trained on one topology and tested on another compared between Inferential SIR-GN (isirgn1) and the base GNN from [37] (ABD-GN)

Trained on Chord						Trained on Debru				
	Chord	*Kadem*	*Debru*	*Leet*	*P2P*	*Chord*	*Kadem*	*Debru*	*Leet*	*P2P*
ABD-GN	99	97.5	99.6	99.4	0	10	2.5	100	0	2.5
isirgn1	99.4	93	100	99	97	93	94	99.5	92.5	97
Trained on Kadem						**Trained on Leet**				
	Chord	***Kadem***	***Debru***	***Leet***	***P2P***	***Chord***	***Kadem***	***Debru***	***Leet***	***P2P***
ABD-GN	97	98	99.5	99	2.5	73	95	100	100	2
isirgn1	99	99	99	99	97.5	99	99.2	94.5	100	98
Trained on P2P										
	Chord		***Kadem***	***Debru***		***Leet***		***P2P***		
ABD-GN	15		22.5	16		17.5		99.5		
isirgn1	93		93	93		93		97.5		

results demonstrate that despite being trained on a much smaller data set, node representations through Inferential SIR-GN, combined with a neural network classifier, outperform even the particular botnet detection approach on stealth data. The results also show that the ABD-GN model can only recognize bots in real-world data when it is given prior information of the structure of a particular botnet.

Then, node representations for a coupled topology were created using inferential SIR-GN. Furthermore, it is clear that utilizing these representations to train a *Random Forest*

algorithm results in noticeably superior performance. In Table 2, we demonstrate how, in the situation of combined botnet attacks, the SIR-GN model offers remarkably accurate classifications on any data set, even those on real-world P2P data. Then, using the node representations produced from the inferential SIR-GN model, we examine the training requirements of a classifier for a neural network. We now show that a modest quantity of data is necessary, even for transfer learning for isirgn1, given that it outperformed ABD-GN on real-world botnet classification after transfer learning with 50 graphs.

In Table 3, we demonstrate that even on real-world test data, it is significantly effective to train a neural network classifier for node structure representation using only one graph from a synthetic dataset. Comparing Tables 2 and 3, we can see that transfer learning with a training set of 80% with ABD-GN is more efficient than transfer learning on a neural network trained with transfer learning on a single graph structural node. Additionally, the Inferential SIR-GN model, which is used to create the node representations, has been taught to transfer learning from training data to artificial random graphs.

Table 3. Training on one graph compared to training on 50 graphs tested on the same 96 graphs

Trained on Chord			Trained on Debru	
	1 Graph	*50 Graphs*	*1 Graph*	*50 Graphs*
Chord	99.5	99.5	92.5	92.5
Debru	99.9	100	99.8	99.8
Kadem	94	96	94	93
Leet	92.5	99	92.25	92.25
P2P	98	98	97	97
Trained on Kadem			**Trained on Leet**	
	1 Graph	***50 Graphs***	***1 Graph***	***50 Graphs***
Chord	99	99	99	99
Debru	98.5	99	93	95
Kadem	99.25	99	99	99.25
Leet	99.25	99	99.5	100
P2P	98	97	98	98

Trained on P2P		
	1 Graph	***50 Graphs***
Chord	93	93
Debru	93	93
Kadem	93	93
Leet	93	93
P2P	98	98

6 Conclusions and Future Work

We can draw the conclusion that Inferential SIR-GN can produce vectors to represent the structural data of every node in a network. Additionally, a machine learning technique for botnet identification can be used with the graphical depiction of the nodes. The *inclusion* of a neural network classifier with Inferential SIR-GN enables the identification of botnets, regardless of the topology used for training, which is a noteworthy advantage over models based on prior knowledge of the graph topology. We have also seen how quickly a botnet structure can alter, mostly as a result of the significant rise in P2P connections. The variety of botnet topologies has increased as a result of this circumstance. This means that, in contrast to methods that leverage GNNs, the usage of Inferential SIR-GN in tandem with a neural network classifier will result in a very high botnet detection ability. Due to the difficulty of adaptability, GNN techniques may not be able to identify previously unknown botnet topologies. Another line of research to be followed consists in coupling our framework with emerging *big data trends* (e.g., [38–44]).

Acknowledgement. This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

References

1. Zou, C.C., Cunningham, R.: Honeypot-aware advanced botnet construction and maintenance. In: International Conference on Dependable Systems and Networks, pp. 199–208. IEEE (2006)
2. Yan, G., Ha, D.T., Eidenbenz, S.: AntBot: Anti-pollution peer-to-peer botnets. Comput. Netw. **55**(8), 1941–1956 (2011)
3. Gu, G., Perdisci, R., Zhang, J., Lee, W.: Botminer: Clustering Analysis of Network Traffic for Protocol-And Structure-Independent Botnet Detection (2008)
4. Holz, T., Gorecki, C., Freiling, F., Rieck, K.: Detection and mitigation of fast-flux service networks. In: 15th Annual Network and Distributed System Security Symposium (2008)
5. Bartos, K., Sofka, M., Franc, V.: Optimized Invariant Representation of Network Traffic for Detecting Unseen Malware Variants. In: 25th {USENIX} Security Symposium, pp. 807–822 (2016)
6. Perdisci, R., Lee, W.: Method and System for Detecting Malicious and/or Botnet-Related Domain Names. Patent 10,027,688 (2018)
7. Andriesse, D., Rossow, C., Bos, H.: Reliable recon in adversarial peer-to-peer botnets. In: 2015 Internet Measurement Conference, pp. 129–140 (2015)
8. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)
9. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)

11. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: 24th International Conference on World Wide Web, pp. 1067–1077 (2015)
12. Grover, A., Leskovec, J.: Node2vec: Scalable Feature Learning for Networks. In: 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864 (2016)
13. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Netw. **20**(1), 61–80 (2008)
14. Ribeiro, L.F., Saverese, P.H., Figueiredo, D.R.: Struc2vec: Learning node representations from structural identity. In: 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 385–394 (2017)
15. Donnat, C., Zitnik, M., Hallac, D., Leskovec, J.: Learning structural node embeddings via diffusion wavelets. In: 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1320–1329 (2018)
16. Joaristi, M., Serra, E.: SIR-GN: A fast structural iterative representation learning approach for graph nodes. ACM Trans. Knowl. Discov. Data **15**(6), 1–39 (2021)
17. Layne, J., Serra, E.: INFSIR-GN: Inferential Labeled Node and Graph Representation Learning. arXiv preprint arXiv:1918.10503 (2021)
18. Ceci, M., Cuzzocrea, A., Malerba, D.: Supporting roll-up and drill-down operations over OLAP data cubes with continuous dimensions via density-based hierarchical clustering. In: SEBD. Citeseer, pp. 57–65 (2011)
19. Serra, E., Joaristi, M., Cuzzocrea, A.:, Large-scale sparse structural node representation. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 5247–5253. IEEE (2020)
20. Braun, P., Cuzzocrea, A., Keding, T.D., Leung, C.K., Padzor, A.G., Sayson, D.: Game data mining: clustering and visualization of online game data in cyber-physical worlds. Procedia Comput. Sci. **112**, 2259–2268 (2017)
21. Guzzo, A., Sacca, D., Serra, E.: An effective approach to inverse frequent set mining. In: 2009 9th IEEE International Conference on Data Mining, pp. 806–811. IEEE (2009)
22. Morris, K.J., Egan, S.D., Linsangan, J.L., Leung, C.K., Cuzzocrea, A., Hoi, C.S.: Token-based adaptive time-series prediction by ensembling linear and non-linear estimators: A machine learning approach for predictive analytics on big stock data". In: 2018 17th IEEE International Conference on Machine Learning and Applications, pp. 1486–1491. IEEE (2018)
23. Serra, E., Subrahmanian, V.: A survey of quantitative models of terror group behavior and an analysis of strategic disclosure of behavioral models. IEEE Trans. Comput. Soc. Syst. **1**(1), 66–88 (2014)
24. Bellatreche, L., Cuzzocrea, A., Benkrid, S.: F&A: A methodology for effectively and efficiently designing parallel relational data warehouses on heterogenous database clusters. In: Bach Pedersen, T., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2010. LNCS, vol. 6263, pp. 89–104. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15105-7_8
25. Korzh, O., Joaristi, M., Serra, E.: Convolutional neural network ensemble fine-tuning for extended transfer learning. In: Chin, F.Y.L., Chen, C.L.P., Khan, L., Lee, K., Zhang, L.-J. (eds.) BIGDATA 2018. LNCS, vol. 10968, pp. 110–123. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94301-5_9
26. Ahn, S., et al.: A fuzzy logic based machine learning tool for supporting big data business analytics in complex artificial intelligence environments. In: 2019 IEEE International Conference on Fuzzy Systems, pp. 1–6. IEEE (2019)
27. Serra, E., Sharma, A., Joaristi, M., Korzh, O.: Unknown landscape identification with CNN transfer learning. In: 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 813–820. IEEE (2018)

28. Serra, E., Shrestha, A., Spezzano, F., Squicciarini, A.: Deeptrust: An automatic framework to detect trustworthy users in opinion-based systems. In: 10th ACM Conference on Data and Application Security and Privacy, pp. 29–38 (2020)
29. Joaristi, M., Serra, E., Spezzano, F.: Inferring bad entities through the panama papers network. In: 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 767–773. IEEE (2018)
30. Joaristi, M., Serra, E., Spezzano, F.: Detecting suspicious entities in offshore leaks networks. Soc. Netw. Anal. Min. **9**(1), 1–15 (2019)
31. CAIDA. The CAIDA UCSD Anonymized Internet Traces-2018. (2018). Accessed 16 Sept. 2017. https://www.caida.org/data/passive/passivedataset.xml
32. Kaashoek, M.F., Karger, D.R.: Koorde: A simple degree-optimal distributed hash table. In: Kaashoek, M.F., Stoica, I. (eds.) IPTPS 2003. LNCS, vol. 2735, pp. 98–107. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45172-3_9
33. Maymounkov, P., Mazières, D.: Kademlia: A peer-to-peer information system based on the XOR metric. In: Druschel, P., Kaashoek, F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 53–65. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45748-8_5
34. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H., Chord, A.: A scalable peer-to-peer lookup service for internet applications. Lab. Comput. Sci., Massachusetts Inst. Technol., Tech. Rep. TR-819 (2001)
35. Jelasity, M., Bilicki, V., et al.: Towards automated detection of peer-to-peer botnets: On the limits of local approaches. LEET **9**, 3 (2009)
36. Garcia, S., Grill, M., Stiborek, J., Zunino, A.: An empirical comparison of botnet detection methods. Comput. Secur. **45**, 100–123 (2014)
37. Zhou, J., Xu, Z., Rush, A.M., Yu, M.: Automating botnet detection with graph neural networks. arXiv preprint arXiv:2003.06344 (2020)
38. Coronato, A., Cuzzocrea, A.: An innovative risk assessment methodology for medical information systems. IEEE Trans. Knowl. Data Eng. **34**(7), 3095–3110 (2020)
39. Leung, C.K., Cuzzocrea, A., Mai, J.J., Deng, D., Jiang, F.: Personalized deepinf: Enhanced social influence prediction with deep learning and transfer learning. In: 2019 IEEE International Conference on Big Data, pp. 2871–2880. IEEE (2019)
40. Leung, C.K., Braun, P., Hoi, C.S.H., Souza, J., Cuzzocrea, A.: Urban analytics of big transportation data for supporting smart cities. In: Ordonez, C., Song, I.-Y., Anderst-Kotsis, G., Tjoa, A.M., Khalil, I. (eds.) DaWaK 2019. LNCS, vol. 11708, pp. 24–33. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-27520-4_3
41. Leung, C.K., Chen, Y., Hoi, C.S., Shang, S., Wen, Y., Cuzzocrea, A.: Big data visualization and visual analytics of COVID-19 data. In: 24th International Conference Information Visualisation, pp. 415–420. IEEE (2020)
42. Leung, C.K., Chen, Y., Hoi, C.S., Shang, S., Cuzzocrea, A.: Machine learning and OLAP on big COVID-19 data. In: 2020 IEEE International Conference on Big Data, pp. 5118–5127. IEEE (2020)
43. Barkwell, K.E., et al.: Big data visualisation and visual analytics for music data mining. In: 22nd International Conference on Information Visualisation, pp. 235–240. IEEE (2018)
44. Camara, R.C., et al.: Fuzzy logic-based data analytics on predicting the effect of hurricanes on the stock market. In: International Conference on Fuzzy Systems, pp. 1–8. IEEE (2018)