# Detecting Botnet Nodes via Structural Node Representation Learning

Justin Carpenter
*Computer Science Dept.*
*Boise State University*
Boise, Idaho, United States
JustinCarpenter836@u.boisestate.edu

Janet Layne
*Computer Science Dept.*
*Boise State University*
Boise, Idaho, United States
janetlayne@boisestate.edu

Edoardo Serra
*Computer Science Dept.*
*Boise State University*
Boise, Idaho, United States
edoardoserra@boisestate.edu

Alfredo Cuzzocrea
*iDEA Lab*
*University of Calabria,*
Rende, Calabria, Italy
alfredo.cuzzocrea@unical.it

***Abstract*—Botnets are an ever-growing threat to private user's machines, small companies, and even large corporations. Botnets are known for spamming, mass downloads and launching distributed denial of service (DDoS) attacks that made a destructive impact on large corporations. Although, with the rise of internet of things (IoT) devices, they are also used as another resource to mine crypto currency, intercept data in transit, send logs containing sensitive information for the master botnet. There have been many approaches to detect botnet activities with few approaches using graph neural networks (GNN) that analyze the behavior of the hosts by using their communications represented by a directed graph. Unfortunately, GNN even if designed for preserving structural graph properties due to over-fitting they are not able to preserve all the structural property especially when the network is unknown, i.e. it is different from the one used for the training. In this work we hypothesize the existence of structural graph patterns that can be used to effectively detect botnet nodes. Using a structural iterative representation learning approach for graph nodes, called inferential SIR-GN, we create for each node a vector representation which represents the nodes' structural information. Then, we use a supervised 3 fully connected layers neural network, to identify bot nodes within an unknown network.**

***Index Terms*—Botnet Detection,Structural Graph Representation Learning, Machine Learning.**

## I. Introduction

Botnets are a system of devices that are controlled by a host to usually perform harmful activities or for personal gain, such as DDoS attacks, spamming, information stealing, or mining crypto currency. The system of Botnets is controlled using either a single botmaster though command-and-control (C&C) or a handful of botmasters though peer-to-peer (P2P). Using only the topological features of communication between devices.

Current works on botnet detection heavily depends on operators being able to identify botnet behavior and requires extensive monitoring. Additionally, many works rely on additional traffic patterns, packet sizes, prior blacklisted addresses, and port numbers. This data can be unavailable or manipulated which can cause unidentifiable patterns. In previous works which focus more on the topology features of botnets, the size of the network communications makes it difficult to differentiate botnet communication patterns from background internet traffic. Other promising works focus on the communication network of the machines to extracts the patterns defining the bot's behavior. GNNs are used to detect such structural patterns and identify the bots among the other traffic. Unfortunately, as shown in our experiments GNNs do not generalize well such patterns, and they work in part for proximity, i.e., machines interacting with bot machines have high likelihood to be bot. This means, that if the GNN is trained on a network of a certain topology and used against a different one, an unknown network, the detection performance of the GNN decreases. In this work, we propose to use the graph representation learning technique Inferential SIR-GN combined with standard GNN for the classification to automatically identify topology features which belong to botnets within large graphs. By using Inferential SIR-GN, that better preserves the structural information even in the inference phase, we aim to extract general structural patterns that can be used on unknown networks to detect bot machines.

This paper is organized as follows. In Section 2, we present our datasets selection and usage. We explain the functionality of our model in Section 3. We evaluate our approach in Section 4. Section 5 provides related works of previous detection approaches. We conclude in Section 6.

## II. Related Work

### A. Botnet Detection

Due to the versatile nature of botnets, there is an extensive list of use cases. This leads to the growing evolution of botnet attacks and complexity. As botnets evolve, so must botnet detection. It is known that botnets can be aware of different botnet detection methods and remain hidden. Honeypots are designed to attract botnets, current botnets are able to identify the Honeypots and avoid them in order to remain undetected. [1]

The most difficult change in how botnets used to behave and how they operate now lies with P2P. Past botnet detection approaches were able to isolate the C&C control node and end the entire botnet. P2P Botnets are able to share the C&C command when seized, they have only limited information of the remaining botnets. [2] The P2P design renders previous works like BotMiner [3] which used node clusters with similar communication traffic and similar malicious traffic and then performs cross-cluster correlation to isolate the central control node.

Botnets can intentionally manipulate their C&C server address frequently using ideas like fast flux server networks to evade traffic monitoring. [4] Which hinders works like [5] which rely on statistical feature representation computed from the network traffic. Additional approaches require more knowledge of the network including the uncompromised botnet information such as domain names [6] and DNS blacklists [7]. These approaches work very well if the data is available and has not been altered by a botnet.

In our proposed work, we use a random-walk of clustering aggregated to maintain the entirety of the graph's structure in a vector representation that can be fit for a classifier. A related work [8]that used random-walk and clustering is our baseline as the approach is tailored to only look at the topology of the network. Our approach takes the entirety of the structure which is why we propose it is better fit for a real world application.

### *B. Graph Representation Learning*

There is an ever growing interest in developing effective and efficient unsupervised representation learning techniques for graphs. The changing botnet structure and interactions with known detection methods opens an area for automated detection using graph representation learning. In order for Machine Learning (ML) applications to use a graph's features, representation learning methods are needed to render the data of use.

There has been many different representation learning techniques applied in different areas of study that have been meet with great success. Some of wich are DeepWalk [9] which continued the with the well known NLP Skip-Gram model, a.k.a. World2Vec [10] [11] which generated word representations by taking advantage of word sequences (sentences) and using them to optimize a neighborhood. DeepWalk generalized the Skip-Gram model from sequences of words to graphs. DeepWalk used a procedure similar to a Depth-First traversal of the graph in combination with the neighborhood which results in a connectivity-based representation learning method where nodes sharing similar neighbors in a direct (first-order proximity) or indirect(higher-order proximity) fashion are located closer in the resulting latent space. Following DeepWalk's idea to use Depth-First traversal, LINE [12] proposed using a Breadth-First traversal where nodes sharing the same edge (first-order proximity) are located closer in the latent space. The problem arises when the graph is not fully connected which is a requirement for Breadth-First. Performance of each are similar compared to Node2vec [13] which uses a random walk procedure that interpolates between both the Depth-First and Breadth-First topology. This removes the connectivity requirement while maintaining a better performance.

All this approach tend to preserve connectivity information among nodes, however they lack in preserving structural properties which are crucial to detect bot machine. The most relevant representation learning approaches preserving structure are: 1) graph neural networks [14] such as Graph Convolutional Neural Network, Struct2Vec [15], GraphWave [16] and Iterative procedures [17], [18]. Among them, only graph neural networks and the iterative approach Inferential SIR-GN are capable to perform inference, i.e., they are capable once trained to produce prediction for graphs different from the one used in the training. Then structural representation learning techniques with inference ability are our core techniques for the bot detection task.

## III. Data Description

There have been many communication patterns of botnets observed in networks that are considered in our approach. C&C botnets are easily identified as they have a single bot which is centralized and has a star pattern. P2P botnets are decentralized without a star pattern and contains a handful of nodes which connect most the network with one or two hops. The P2P botnet clusters are harder to identify as there is no single center point. In these experiments, the P2P is used to prove that graph representation learning is able to detect anomalies that are more difficult to detect using other methods.

The datasets used are composed of real background traffic collected in 2018 from the IP backbone from CAIDA [19] (2018)'s monitors. The traffic graph is aggregated as it would be in a real case scenario for user's protection. Then at random, a subset of nodes is selected from the background traffic as botnet nodes for embedding the different P2P topologies. There are four different P2P topologies used to create controlled networks in our experiments. The synthesized networks are DE BRUIJN [20], KADEMLIA [21], CHORD [22], and LEETCHORD [23]. Also, included in the datasets used is a real P2P network which captures botnets attack from 2011 [24] which contain attack behavior with communication traffic.

The Log-Log plot shown in Figure 1 represents the degree distribution within our graphs. As the degree (number of edges) of nodes increase, the frequency (number of nodes) decreases. This shows that there are few nodes that are highly connected, and majority of nodes have less than two degrees. The average cluster in this graphs are 0.007.

Each network contains 960 P2P graphs each with an average of 144k nodes with over 1.5 million edges and 10k botnet nodes within each synthetic graph as shown in Figure 2. This number is based on the real botnet network which contains 144k nodes and 3k botnet nodes. The datasets used to train contain 10k botnet nodes then the testing set uses a mix of 10k/1k/100 botnet nodes within the networks. All the networks are highly unbalanced with less than a tenth of the network containing botnet nodes.

## IV. Inferential SIR-GN Proposed Methodology

Our mythology is based on the Inferential SIR-GN [18] a structural iterative representation learning procedure with inference ability. In Table I are reported the symbols we use for the methodology explanation.

Inferential SIR-GN is used for extracting node representations from directed graph, and is described in detail in Layne and Serra [18]. The model relies upon the methodology of SIR-GN, first described in [17], wherein a node's representation is iteratively updated by describing then aggregating its neighbors.

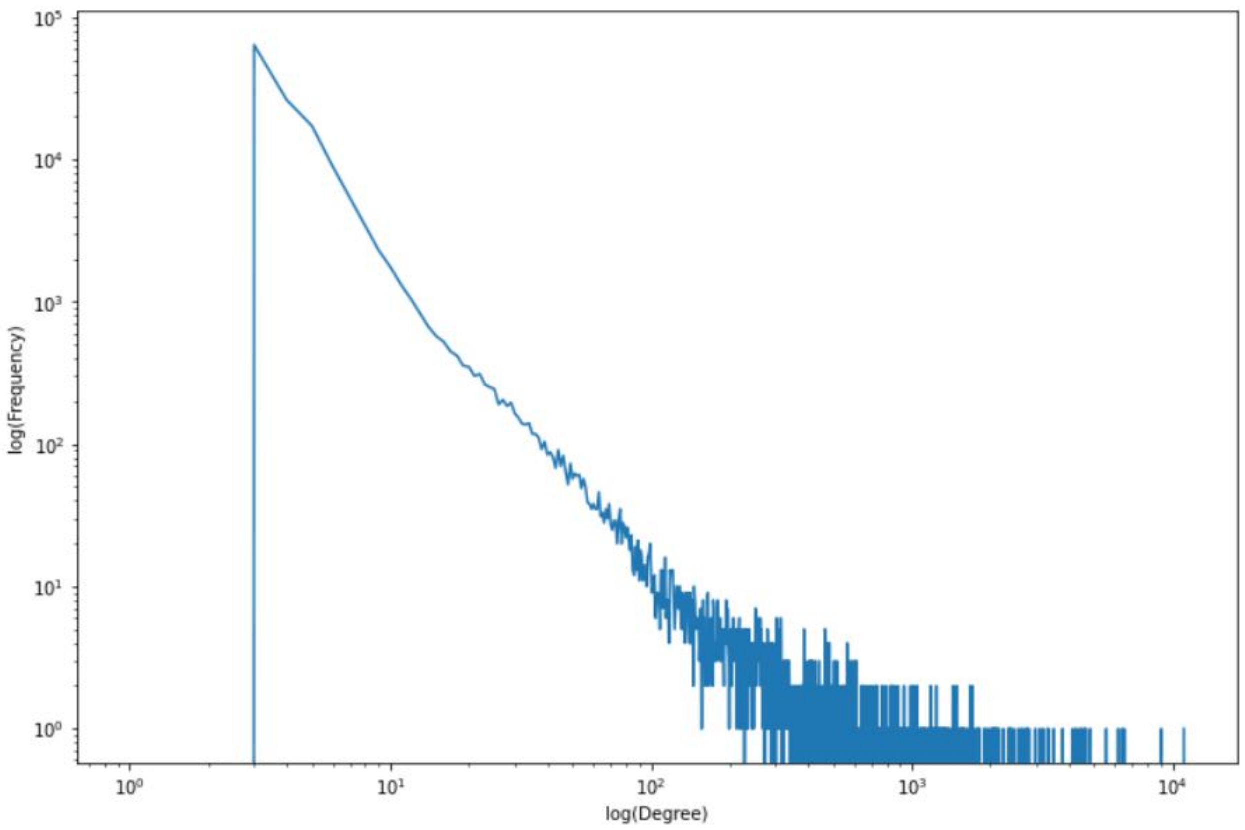
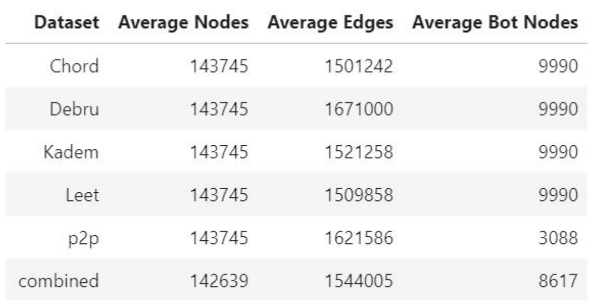


Figure 1: LogLog Plot to show the degree distribution of the Data used in the experiments

Table I: 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 |



| Dataset | Average Nodes | Average Edges | Average Bot Nodes |
|---|---|---|---|
| Chord | 143745 | 1501242 | 9990 |
| Debru | 143745 | 1671000 | 9990 |
| Kadem | 143745 | 1521258 | 9990 |
| Leet | 143745 | 1509858 | 9990 |
| p2p | 143745 | 1621586 | 3088 |
| combined | 142639 | 1544005 | 8617 |

Figure 2: Average node structure in the Datasets used in Data Description

The size of a node's representation at each iteration is equal to a user-chosen hyperparameter $nc$. Node descriptions are generated by clustering the current node description (which initializes as the node degree) into $nc$ KMeans clusters. Normalization of the representation occurs before the clustering step at each iteration, then the distance from each cluster centroid is converted into a probability of membership of the node in each cluster. Once a node's structural description has been updated, its neighbors are aggregated into its description by summing for each cluster all neighbors' probabilities of membership per cluster. The resulting node representation is equal to the expected number of neighbors that node possesses in each cluster. Each iteration corresponds to an added depth of exploration, where $k$ iterations will generate a node description incorporating the k-hop neighborhood structure of a node. Inferential SIR-GN differs from the standard model via multiple modifications, the first being that at the end of each iteration, we concatenate each node's structural description into a larger representation that captures the evolution of the structural information through deeper neighborhood exploration. After the final iteration, a Principle Component Analysis (PCA) is used to prevent degradation of the information as the representation size grows. The final representation is condensed to a size chosen as a hyperparameter. For directed graphs, a node's initial representation begins as two vectors of size $nc$, one containing the node's in-degree, the other containing its out-degree. These two are concatenated together before clustering. At each iteration, clustering of this larger node vector is performed, followed by aggregation of the neighbors. For directed data, the aggregation is performed separately for a node's in-neighbors and out-neighbors into two intermediate vectors, then once again concatenated together for the next iteration. Inferential capability of our proposed model is accomplished by pre-training the KMeans and scalers for each iteration - a new KMeans and Scaler are used for every depth of exploration - along with the PCA model that will be used to generate the final node embedding. We pre-train on random graphs and store each model for use in inference. At inference time, repeated normalization followed by clustering and aggregation is accomplished using the pre-trained models, and the PCA fit during training is used to generate the final node representations. This drastically increases inference time, and the same pre-trained model can be used on a variety of different data sources. This is demonstrated extensively in Layne and Serra, along with a detailed algorithm and description of the time complexity of the model. The SIR-GN vectors which represents our graphs structural information can be used in any classifier to learn the botnets topology for automated detection. Our SIR-GN vectors represent and aggregated vector of all the layers in each node. Once the structural representation of each node (each machine of the network) are achieved they are passed through the Neural Network Classification algorithm to classify if a node/machine is a bot or not.

## V. Experiment

### A. Methods

Using the 4 different topologies mentioned in section 2 there are 4 different collections of datasets created. For each topology, 960 graphs are created by applying the botnet topology to real world traffic. The actual P2P attack contains 960 graphs of real botnet attacks. These graphs were split into an 8:2 ratio for training and testing. The ABD-GN utilizes all graphs in a GNN in order to fit the model for that specific topology. With SIR-GN the models only need to train on 50 graphs to fully learn the topologies. The models were tested against the same topology and other topologies. This was done to test the graph representation learning model on all the different topologies simultaneously with out the need to over fit for a single P2P topology.

When starting this experiment, a split of each different networks in an 8:2 ratio is used in the training and testing. With only the communication information (edge relations) between the nodes, the SIR-GN is used to create a vector representation for each nodes' structure in each graph. Then the nodes and vectors are passed into a classifier (3 layered Neural Network) to then test on.

The model used in [8] (ABD-GN) is another GNN that has been tailored for botnet detection. This baseline was chosen to compare our graph representation learning, SIR-GN inside a classifier, which can be applied for any GNN problem. With having the ability to quickly generate vector representations without losing any structural information, we are able to get similar scores as a tailored GNN approach.

The evaluations of the different trained models include the average false positive rate, false negative rate, accuracy, and F1 scores for an equal comparison. This test our features generated automatically without any additional tailoring against another GNN that has been tailored for the best results. This experiment of testing the automation of a vector representation within a classifier further proves botnet detection using GNN as an effective approach compared with non-learning methods.

### B. Evaluations

In Table II the classifier using SIR-GN vector representation (isirgn1), F1 scores on all the synthetic datasets come within 5 percent of the GNN model (ABD-GN) which was tailored for the datasets. The F1 score is based on the false positive, false negative and true positive scores which accurately represents our goal in detecting botnets automatically.

Table II: Botnet detection results on synthetic and real botnet topologies. FP represents the false positive rate, FN represents the false negative rate, ACC represents the accuracy, F1 represents the F1 score. All the scores are rounded to the nearest two decimals, and are an average over all the graphs in the test set.

Chord

| Model | FP | FN | ACC | F1 |
|---|---|---|---|---|
| ABD-GN | 0.02 | 1.47 | 99.88 | 99.12 |
| isirgn1 | 0.01 | 0.60 | 99.87 | 99.39 |

DE Bruijn

| Model | FP | FN | ACC | F1 |
|---|---|---|---|---|
| ABD-GN | 0.00 | 0.09 | 99.99 | 99.93 |
| isirgn1 | 0.00 | 0.32 | 99.98 | 99.50 |

KADEM

| Model | FP | FN | ACC | F1 |
|---|---|---|---|---|
| ABD-GN | 0.03 | 2.05 | 99.83 | 98.77 |
| isirgn1 | 0.02 | 2.69 | 98.31 | 99.10 |

LEET

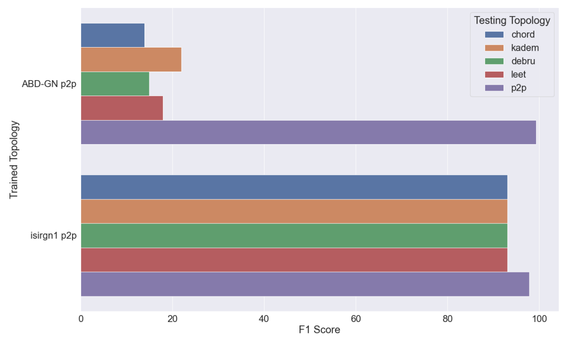
| Model | FP | FN | ACC | F1 |
|---|---|---|---|---|
| ABD-GN | 0.02 | 1.18 | 99.90 | 99.30 |
| isirgn1 | 0.00 | 0.36 | 99.92 | 99.78 |

P2P

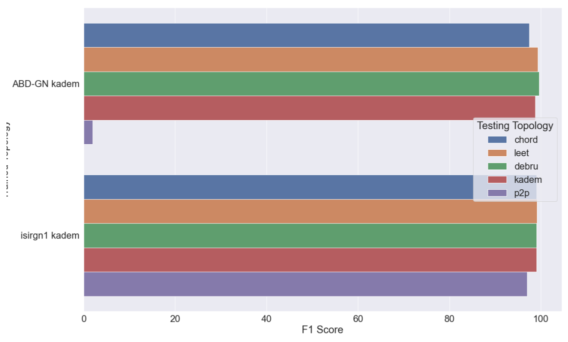
| Model | FP | FN | ACC | F1 |
|---|---|---|---|---|
| ABD-GN | 0.01 | 0.96 | 99.97 | 99.29 |
| isirgn1 | 0.02 | 2.15 | 99.00 | 97.85 |

The ABD-GN results from [8] performs well when targeting a single topology as shown in Table II. When these same trained models are tested against a different topology, however, the results very from good to very poor. As shown in Figure 3 Using the SIR-GN on a combined topology, the SIR-GN is able to identify patterns within all the different topology for a more rounded classification.

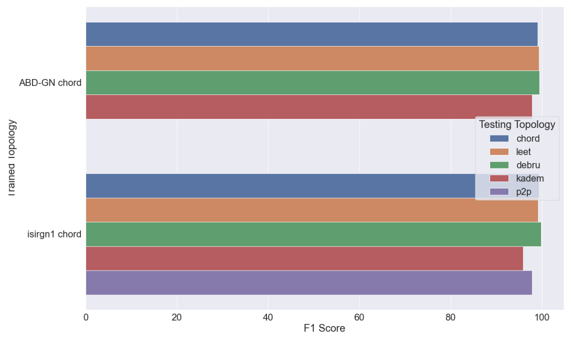
As shown in Figure 3, using only a GNN approach to learn a single topology fails at detecting many different topologies. The real world botnet attack can not be detected by ABD-GN without prior knowledge of the botnet used in the attack's structure. Even with using the methodology mentioned in



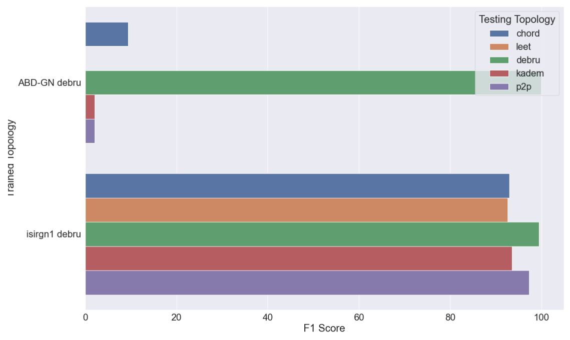
(a) Trained on P2P



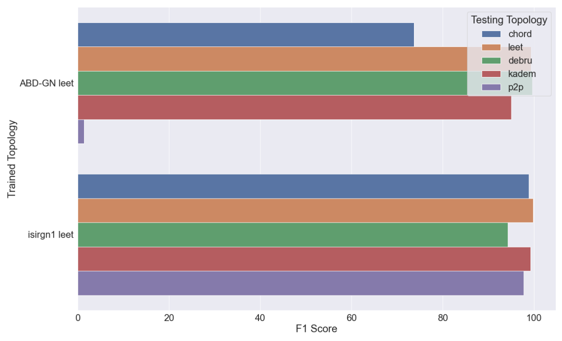
(b) Trained on Kadem



(c) Trained on Chord



(d) Trained on Debru



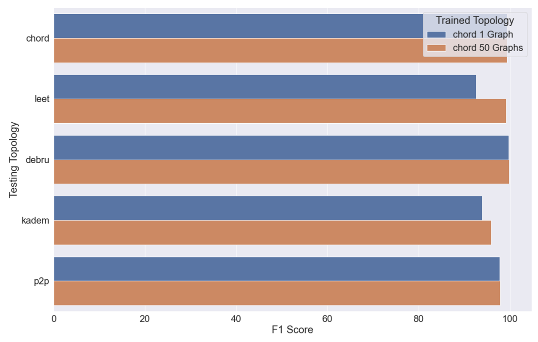
(e) Trained on Leet

Figure 3: F1 Scores of the models trained on one topology and tested on another compared between the methodology mentioned in Inferential SIR-GN (isirgn1) and the base GNN from [8] (ABD-GN). F1 represents the F1 score, each image represents the different topology trained on, each bar represents the topology tested against. All the scores are rounded to the nearest two decimals, and are an average over all the graphs in the test set.)

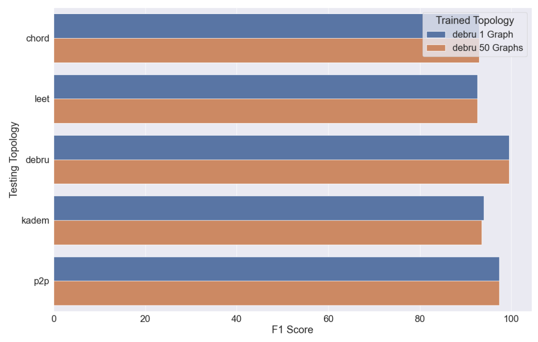
section IV the models trained on a single topology is not able to identify unknown topologies.

A large improvement seen in Figure 3 is when ABD-GN is trained on any of the 4 synthetic datasets (Figure 4a, Figure 4b, Figure 4d, Figure 4c) the GNN fails to detect the real world p2p networks with scores less than 3%.

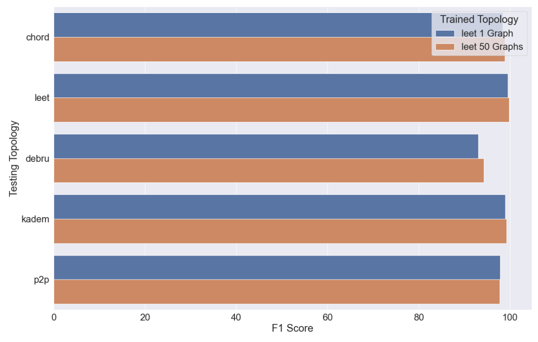
When applying SIR-GN to a any single topology, the model can be see in Figure 3 that using many different botnet attacks structures the results are very similar with the SIR-GN model approach is able to reach an F1 score within 5% of the same topology of which it was trained on. Witch is better than less over 90% lower score on a different topology as seen in Figure 3a and Figure 4b which signals over fitting for a single topology.



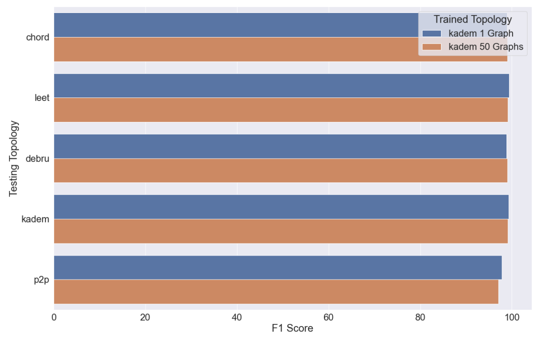
(a) Trained on Chord



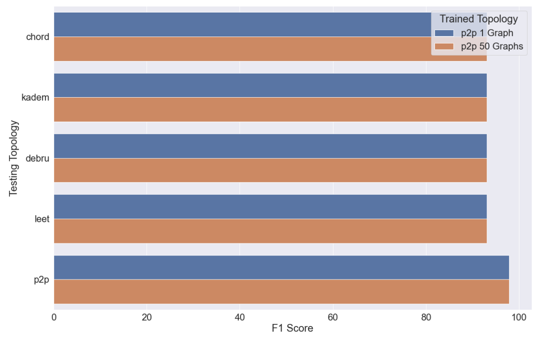
(b) Trained on Debru



(c) Trained on Leet



(d) Trained on Kadem



(e) Trained on p2p

Figure 4: Training on one graph compared to training on 50 graphs tested on the same 96 graphs.

When applying our approach to an equal split of each different topology, while also testing on an equal split, the SIR-GN is still able to achieve a higher F1 score for some topologies while maintaining a low false positive (FP) score as shown in Table II.

While developing the training set, the increase in networks trained on did not influence the testing scores. Trained on only a single graph generated less than a 5% change in the scores as shown in Figure 4. We chose 50 as a good training set size to get balance of under fitting. The scores between 50 networks trained on and 100 networks trained on was less than 0.001%. This showed a that we do not have the limitation of over fitting and there is no benefit in using more networks in the training set.

The difference in training sizes show how the Neural Network Classification receives enough information from the SIR-GN to determine the classifications. Unlike the ABD-GN, which relies on a very large amount of data to train on to fit the Neural Network classification.

## VI. Conclusion

We proposed that using SIR-GN we can generate vector representations that retain all the necessary structural information to create a graph representation learning solution which can be implemented in automated botnet detection. The SIR-GN can be implemented in other GNN applicable problems for a efficient and effective unsupervised representation learning technique.

Unlike previous works [8] which relied on the graphical topology structure, the SIR-GN is able to retain the entirety of the graph's structure and identify botnets with any topology.

Since botnet's topology is changing very rapidly and as botnet's applicability increases, the use of different new and old typologies is expanding. If a GNN is to be expected for an automated detection application for all botnets, the SIR-GN can be adapted and applied to new and unseen botnets with low false positive rates.

The large loss in detection while applying same concept to the real world dataset as the synthetic datasets is seen as a problem. There is room for further experiments on real world applicability with different gathered datasets of networks. It has been shown that single application of a GNN for a particular set of topologies is more effective than many previous attempts. While with limited data, a GNN is fast and effective enough for an automated detection application with more room to adapt to new botnet designs.

## Acknowledgment

This research was made possible by the National Science Foundation award #1820685 and Idaho Global Entrepreneurial Mission/Higher Education Research Council #IGEM22-001.

## References

[1] C. C. Zou and R. Cunningham, "Honeypot-aware advanced botnet construction and maintenance," in *International Conference on Dependable Systems and Networks (DSN'06)*. IEEE, 2006, pp. 199–208.

[2] G. Yan, D. T. Ha, and S. Eidenbenz, "Antbot: Anti-pollution peer-to-peer botnets," *Computer networks*, vol. 55, no. 8, pp. 1941–1956, 2011.

[3] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection," 2008.

[4] T. Holz, C. Gorecki, F. Freiling, and K. Rieck, "Detection and mitigation of fast-flux service networks," in *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS)*, 2008.

[5] K. Bartos, M. Sofka, and V. Franc, "Optimized invariant representation of network traffic for detecting unseen malware variants," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 807–822.

[6] R. Perdisci and W. Lee, "Method and system for detecting malicious and/or botnet-related domain names," Jul. 17 2018, uS Patent 10,027,688.

[7] D. Andriesse, C. Rossow, and H. Bos, "Reliable recon in adversarial peer-to-peer botnets," in *Proceedings of the 2015 Internet Measurement Conference*, 2015, pp. 129–140.

[8] J. Zhou, Z. Xu, A. M. Rush, and M. Yu, "Automating botnet detection with graph neural networks," *arXiv preprint arXiv:2003.06344*, 2020.

[9] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[12] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 1067–1077.

[13] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[14] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[15] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 385–394.

[16] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1320–1329.

[17] M. Joaristi and E. Serra, "Sir-gn: A fast structural iterative representation learning approach for graph nodes," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 15, no. 6, pp. 1–39, 2021.

[18] J. Layne and E. Serra, "Infsir-gn: Inferential labeled node and graph representation learning," *arXiv preprint arXiv:1918.10503*, 2021.

[19] CAIDA, "The caida ucsd anonymized internet traces- ¡2018¿," 2018, last accessed 16 September 2017. [Online]. Available: https://www.caida.org/data/passive/passivedataset.xml.

[20] M. F. Kaashoek and D. R. Karger, "Koorde: A simple degree-optimal distributed hash table," in *International Workshop on Peer-to-Peer Systems*. Springer, 2003, pp. 98–107.

[21] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 53–65.

[22] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, and A. Chord, "A scalable peer-to-peer lookup service for internet applications," *Lab. Comput. Sci., Massachusetts Inst. Technol., Tech. Rep. TR-819*, 2001.

[23] M. Jelasity, V. Bilicki *et al.*, "Towards automated detection of peer-to-peer botnets: On the limits of local approaches." *LEET*, vol. 9, p. 3, 2009.

[24] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *computers & security*, vol. 45, pp. 100–123, 2014.