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 users, small companies, and even large corporations. They are known for spamming, mass downloads, and launching distributed denial-of-service (DDoS) attacks that have a destructive impact on large corporations. With the rise of internet-of-things (IoT) devices, they are also used to mine cryptocurrency, intercept data in transit and send logs containing sensitive information to the master botnet. Many approaches have been developed to detect botnet activities. A few approaches employ graph neural networks (GNN) to analyze the behavior of hosts using a directed graph to represent their communications. However, while designed to capture structural graph properties, GNN may overfit, and therefore fail to capture these properties when the network is unknown. In this work we hypothesize that structural graph patterns can be used to effectively detect Botnets. We then propose a structural iterative representation learning approach for graph nodes, which is designed to perform well on unseen data, called Inferential SIR-GN. Our model creates a vector representation for each node that epitomizes its structural information. We demonstrate that this set of node representation vectors can be used with a neural network classifier to identify bot nodes within an unknown network with better performance than the current state-of-the-art GNN based method.**

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

I. Introduction

Current works on botnet detection heavily depend on operators' ability to identify their behavior, and requires extensive monitoring. Additionally, many works rely on additional data, such as further traffic patterns, packet sizes, prior blacklisted addresses and port numbers. This data can unavailable or manipulated to create unidentifiable patterns. In previous works focusing on the topology features of botnets, the size of the communications networks can make it difficult to discern botnet communication patters from background internet traffic. A few promising works focus on the communication network of the machines to extract patterns defining the bot's behavior using GNN. However, we will show that GNN do not generalize well to unseen data. This is likely because GNN, while capable of capturing structural information, also rely partially upon a notion of similarity that is based upon node proximity. In this case, machines interacting with bot machines will be considered likely to be a bot as well. This can result in poor detection performance if the GNN is trained on a network different from that for which inference is needed. In this work, we present the graph representation learning technique Inferential SIR-GN, combined with a neural network classification model, to automatically identify topology features that belong to botnets within large graphs. Using Inferential SIR-GN, we aim to better preserve the structural information of a graph even in the inference phase, on networks entirely unseen, and leverage that to detect bot machines. We show that Inferential SIR-GN plus a neural network classifier performs nearly identically to the state-of-the-art GNN model on tasks where the training data and test data topology are the same. Moreover, Inferential SIR-GN outperforms the state-of-the-art GNN when training topologies differ from the test data. Given the ever-changing nature of botnets, this feature suggests that our model is a superior automatic botnet detection method. This paper is organized as follows. In Section 2, previous detection approaches are presented. A description of the datasets used in this work is presented in Section 3. The Inferential SIR-GN methodology is described in Section 4, and we evaluate our approach in Section 5. Finally, conclusions are presented in Section 6.

II. Related Work

A. Botnet Detection

Due to the versatile nature of botnets, there is an extensive list of use cases. As a result, botnets and their attacks are ever evolving, and increasing in complexity. As they evolve, so too must methods to detect them. It is well known that botnets take actions to thwart detection strategies. For example, many honeypots, designed to attract botnets, can be identified and therefore avoided, allowing these botnets to remain undetected [1].

The emerging trait that has made botnets most difficult to combat lies with P2P. Past botnet detection approaches were able to isolate the C&C control node, and so end the entire botnet. P2P botnets are able to share C&C command when seized, and have only limited information about the remaining botnets [2]. Previous works such as BotMiner [3] used node clusters with similar communication and malicious traffic to perform cross-cluster correlation and isolate the central control node. P2P botnets have rendered these approaches ineffective. Additionally, botnets are able to intentionally manipulate their C&C server address frequently, using fast flux server networks for example, to evade traffic monitoring [4]. This hinders works,

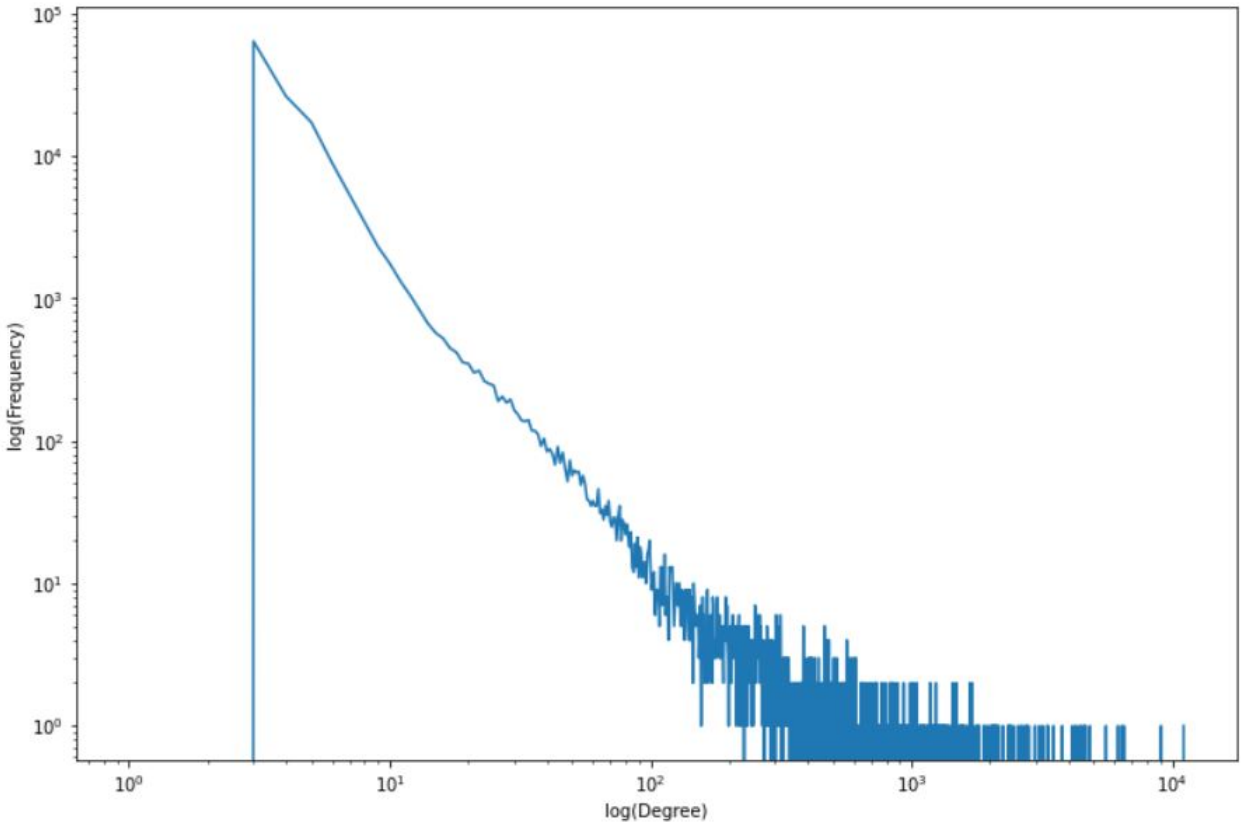


Figure 1: LogLog Plot to show the degree distribution of the Data used in the experiments

such as those presented in [5], which rely on statistical feature representations computed from the network traffic. Additional approaches require more extensive knowledge of the network, including uncompromised botnet information, such as domain names [6] and DNS blacklists [7]. These approaches work well unless the data is unavailable, or has been altered by the botnet, as is often the case.

B. Graph Representation Learning

The need for effective and efficient unsupervised representation learning techniques for graph data are becoming increasingly recognized. Networks capture a wealth of information about any dataset with relational structure, in a simple data structure as a list of entities, called nodes, and their connections, referred to as edges. The botnet structure lends itself well to automated detection using graph representation learning. However, standard machine learning applications operate upon a list of features, requiring graph data structures to be transformed into representations. It is imperative that these representations fully capture the similarity between nodes in the feature space. The notion of similarity, however, can be based upon the nodes structural role in the graph, or it's proximity to neighbors. As an example, when capturing a node's proximity, highly connected nodes will be found close to one another in the feature space. However, representations that capture a node's structural role should not require that a path even exists between two nodes to consider them similar, only that the structure of their neighborhood is alike.

There have been many different representation learning techniques applied in multiple areas of study that have met with great success. DeepWalk [8] utilizes the well known NLP Skip-Gram model, a.k.a. Word2Vec [9] [10] which generates word representations by taking advantage of word sequences (sentences) to optimize a neighborhood. DeepWalk generalizes the Skip-Gram model from sequences of words to graphs using a procedure similar to a Depth-First traversal of the graph in combination with the neighborhood. This 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 [11] proposed using a Breadth-First traversal where nodes sharing the same edge (first-order proximity) are located closer in the latent space. A notable problem arises when the graph is not fully connected, a requirement for Breadth-First graph traversal. The above-described approaches are similar to Node2vec [12], which uses a random walk procedure that interpolates between both the Depth-First and Breadth-First topology. This removes the connectivity requirement and demonstrates improved performance. Each of these approaches tends to capture connectivity information among nodes, but is weaker in preserving structural properties

that are crucial for detecting bot machines in a network.

The most relevant representation learning approaches preserving structural information are: 1) graph neural networks [13] such as Graph Convolutional Neural Network, Struct2Vec [14], GraphWave [15] and Iterative procedures [16], [17]. Among these methods, only graph neural networks and the iterative approach Inferential SIR-GN are able to perform inference, that is, can produce predictions for graphs entirely different than those used for training. Our core techniques for the bot detection tasks will focus on structural representation learning methods with inference capability. In [18]–[30] are reported several other related works in the field of machine learning and AI.

III. Data Description

We consider multiple communication patterns of botnets observed in networks for 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 demonstrate 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 (2018)'s monitors [31]. 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 [32], KADEMLIA [33], CHORD [34], and LEETCHORD [35]. We also consider a real P2P network which captures botnet attacks from 2011 [36] 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 the majority of nodes have degree less than two. The average cluster in the graphs are 0.007.

Each network contains 960 P2P graphs each with an average of 144k nodes, over 1.5 million edges and 10k botnet nodes within each synthetic graph (Figure 2). This number is based on the real botnet network which contains 144k nodes and 3k botnet nodes. The datasets used for training contain 10k botnet nodes, and the test 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.

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

IV. Inferential SIR-GN Proposed Methodology

Our methodology is based on the Inferential SIR-GN [17] a structural iterative representation learning procedure with inference ability. Table I reports the symbols we use for the methodology explanation.

Inferential SIR-GN is used for extracting node representations from directed graphs, and is described in detail in Layne and Serra [17]. The model relies upon the methodology of SIR-GN, first described in [16], wherein a node's representation is iteratively updated by describing then aggregating its neighbors. 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

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

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 fitted during training is used to generate the final node representations. This drastically decreases 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 node structural representation vectors can be used in any classifier to learn the botnets topology for automated detection. In this work, we use a 3-layer neural network to obtain a final prediction of a node's (machine's) status as a bot.

V. Experiments

A. Setup

Section 2 describes the four different botnet topologies that are used to create synthetic datasets. For each topology, 960 different graphs are created by applying that specific topology to real-world traffic. The number and size of the graphs is made comparable to an actual P2P dataset that contains 960 graphs of real botnet attacks. Our Inferential SIR-GN model is used to generate structural node representations for each set of graphs. As described in the methodology, Inferential SIR-GN is not trained using any of the graphs from the dataset, but rather a set of generated random graphs. Because the node representations calculated by Inferential SIR-GN so effectively capture the structural description of each node, we are able to use a drastically decreased subset of these node representations to train a downstream classifier, and still obtain excellent results. Inferential SIR-GN can be used upstream of any classifier, and we will show that transfer learning can be used to train a neural network (NN) classifier that outperforms the current state-of-art model trained in a similar manner. For comparison, we will use the model described in [37] (ABD-GN), a GNN tailored for botnet detection. We will demonstrate the effectiveness of Inferential SIR-GN to generalize to unseen data by performing the following comparisons:

1. Inferential SIR-GN plus NN classifier trained on 50 graphs from a dataset (botnet topology) and used to classify the a test set of 96 graphs from that dataset. This is for comparison to ABD-GN trained on 80% (768) of the graphs from the dataset and used to classify a test set of 20% (the same 96 graphs) from the same topology.

2. Inferential SIR-GN plus classifier 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, plus the actual P2P attack data. This will be compared to ABD-GN trained on 768 graphs from one topology and used to classify the test set.

For simplicity, we will refer to the Inferential SIR-GN plus neural network classifier as isirgn1 in the description of results.

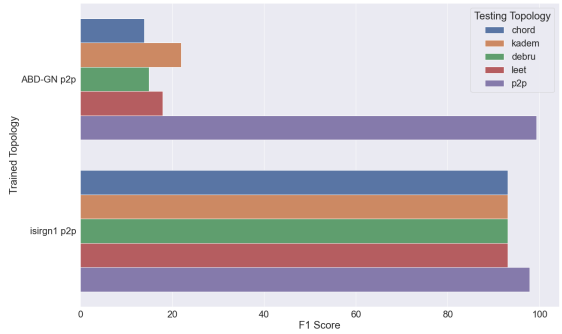
B. Results

Table II shows the results of the NN classifier on the node representations generated from Inferential SIR-GN, compared to those produced by ABD-GN. For all datasets, isirgn1 obtains very similar performance, despite that the model used to generate the node representations was trained purely on random graph data, and the neural network classifier was trained on only 50 graphs from the dataset. This is compared to the ABD-GN neural network in which 80% of the data was used for training.

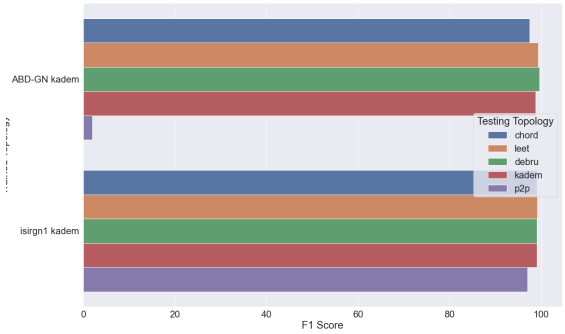
Table II: Botnet detection results on synthetic and real botnet topologies. FP represents the false positive rate, FN represents the false negative rate, ACC represents the accuracy, F1 represents the F1 score. All the scores are rounded to the nearest two decimals, and are an average over all the graphs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

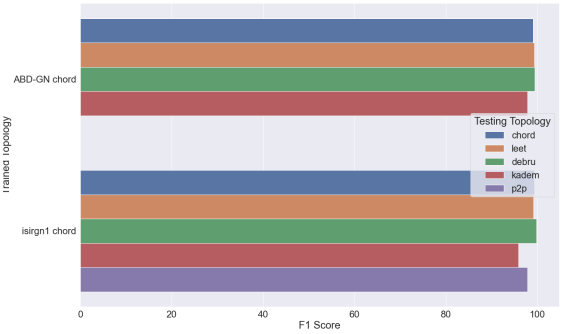
The ABD-GN model [37] produces strong results when targeting a single topology as shown in Table II. However, when these same trained models are tested against an unseen topology, the results vary from good to very poor (Figure 3). When trained on the Kadem topology (Figure 3(b)), the isirgn model and ABD-GN perform similarly with respect to testing on all of the other synthetic datasets. However, when testing classification on the real-world P2P dataset, isirgn shows excellent capability,



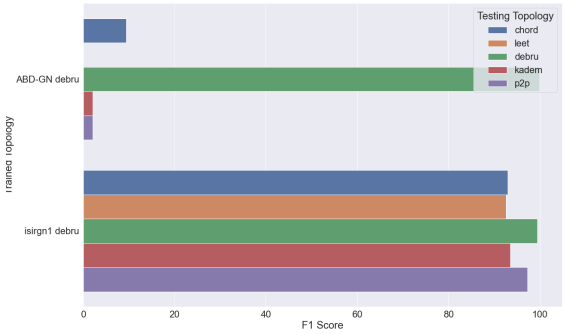
(a) Trained on P2P



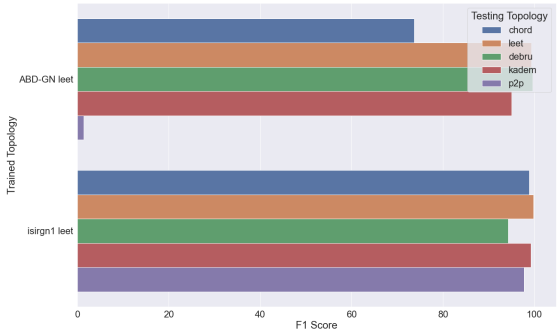
(b) Trained on Kadem



(c) Trained on Chord



(d) Trained on Debru

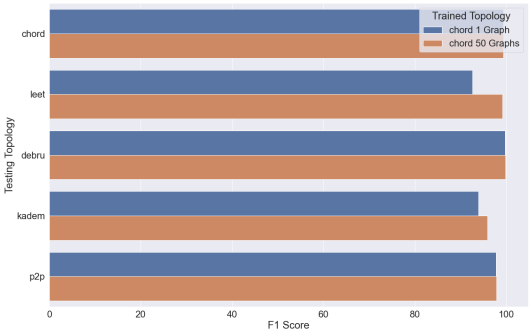


(e) Trained on Leet

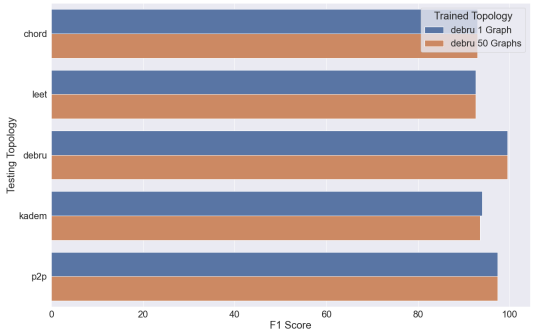
Figure 3: F1 Scores of the models trained on one topology and tested on another compared between Inferential SIR-GN (isirgn1) and the base GNN from [37] (ABD-GN). Individual bar graphs represent training topology used, while individual bars represent performance (F1 score) of classification on test data from each indicated topology. All scores are an average over all the graphs in the test set.)

while the F1 score for ABD-GN drops below 10%. The same is true for training on Leet and Chord data, with small variations in performance on the other synthetic datasets. Interestingly, when the Debru data is used to train each model, ABD-GN not only fails to correctly identify bots in the real-world dataset, but also performs poorly at classification in all other topologies as well, while isirgn maintains this capability. When trained on the real-world dataset (Figure 3(b)), isirgn and ABD-GN perform similarly on a testset generated from the same dataset. However, isirgn also performs well at classification tasks in the four unseen topologies, while ABD-GN drastically loses performance.

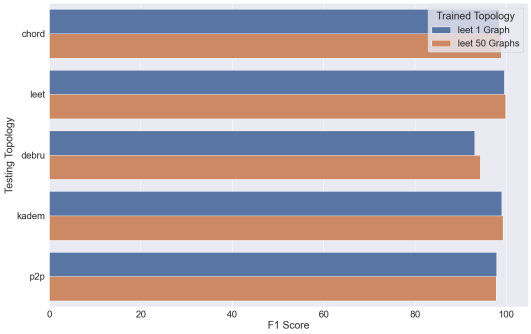
This data suggests that Inferential SIR-GN node representations, coupled with a neural network classifier, far outperform



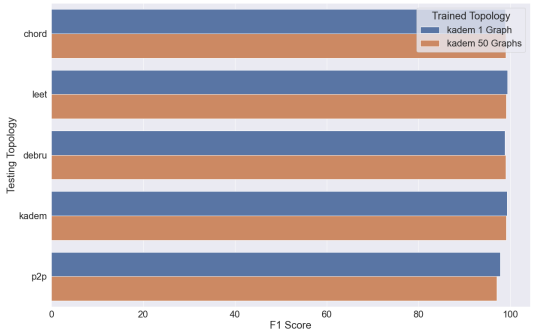
(a) Trained on Chord



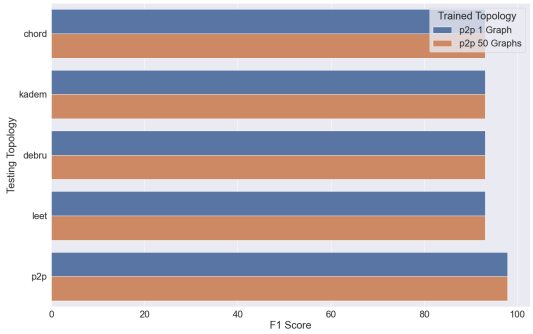
(b) Trained on Debru



(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.

even the state-of-the-art botnet detection method on unseen data, despite being trained on a significantly smaller dataset. Additionally, the results demonstrate that the ABD-GN model is only able to detect bots in real-world data if given prior knowledge about the structure of that botnet.

Inferential SIR-GN was then used to generate node representations for a combined topology. When these representations are used to train the Random Forest, a highly improved performance is observed. Figure **??** shows that using many different botnet attacks structures combined with the SIR-GN model approach obtains highly performant classifications on each dataset, including vastly improved performance on the real-world P2P dataset.

We next test the training requirement for a neural network classifier using node representations obtained from the Inferential SIR-GN model. Keeping in mind that isirgn outperformed ABD-GN on classification of real world botnets after transfer learning with 50 graphs, we now demonstrate that a vanishingly small amount of data is required, even for transfer learning, for isirgn. Figure 4 shows that using only one graph from

a synthetic dataset to train a neural network classifier on structural node representations (output from Inferential SIR-GN) is remarkably successful, even on real-world test data. Indeed, comparing Figures 3 and 4 shows that a neural network trained with transfer learning on a single graph's structural node representations from Inferential SIR-GN is more successful than transfer learning with an 80% training set with ABD-GN. This is also considering that the Inferential SIR-GN model used to generate the node representations is itself trained using transfer learning on random synthetic graphs.

VI. Conclusion

We propose that Inferential SIR-GN can generate vector representations that retain all necessary structural information of nodes in a network. We further propose that this graph representation learning solution can be implemented in automated botnet detection. Additionally, Inferential SIR-GN can be implemented in other GNN-applicable problems for an efficient and effective unsupervised representation learning technique.

Unlike works [37] which rely on the knowledge of the target graph's exact topological structure, Inferential SIR-GN, coupled with a neural network classifier, is able to identify botnets with any topology, regardless of the training topology.

As previously stated in section 3.1, the appearance of a botnet's network structure can change very rapidly. As botnet's applicability increases, the variety of observed topologies is expanding. We propose that employing Inferential SIR-GN coupled with a neural network for classification, will show superior ability to detect a variety of botnets, including those with never-before-seen topologies, where other GNN methodologies may fail 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] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] M. Ceci, A. Cuzzocrea, and D. Malerba, "Supporting roll-up and drill-down operations over olap data cubes with continuous dimensions via density-based hierarchical clustering." in *SEBD*. Citeseer, 2011, pp. 57–65.

[19] E. Serra, M. Joaristi, and A. Cuzzocrea, "Large-scale sparse structural node representation," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 5247–5253.

[20] P. Braun, A. Cuzzocrea, T. D. Keding, C. K. Leung, A. G. Padzor, and D. Sayson, "Game data mining: clustering and visualization of online game data in cyber-physical worlds," *Procedia Computer Science*, vol. 112, pp. 2259–2268, 2017.

[21] A. Guzzo, D. Sacca, and E. Serra, "An effective approach to inverse frequent set mining," in *2009 Ninth IEEE International Conference on Data Mining*. IEEE, 2009, pp. 806–811.

[22] K. J. Morris, S. D. Egan, J. L. Linsangan, C. K. Leung, A. Cuzzocrea, and C. S. Hoi, "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 (ICMLA)*. IEEE, 2018, pp. 1486–1491.

[23] E. Serra and V. Subrahmanian, "A survey of quantitative models of terror group behavior and an analysis of strategic disclosure of behavioral models," *IEEE Transactions on Computational Social Systems*, vol. 1, no. 1, pp. 66–88, 2014.

[24] L. Bellatreche, A. Cuzzocrea, and S. Benkrid, "$F\&A$: A methodology for effectively and efficiently designing parallel relational data warehouses on heterogenous database clusters," in *International Conference on Data Warehousing and Knowledge Discovery*. Springer, 2010, pp. 89–104.

[25] O. Korzh, M. Joaristi, and E. Serra, "Convolutional neural network ensemble fine-tuning for extended transfer learning," in *International Conference on Big Data*. Springer, 2018, pp. 110–123.

[26] S. Ahn, S. V. Couture, A. Cuzzocrea, K. Dam, G. M. Grasso, C. K. Leung, K. L. McCormick, and B. H. Wodi, "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 (FUZZ-IEEE)*. IEEE, 2019, pp. 1–6.

[27] E. Serra, A. Sharma, M. Joaristi, and O. Korzh, "Unknown landscape identification with cnn transfer learning," in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2018, pp. 813–820.

[28] E. Serra, A. Shrestha, F. Spezzano, and A. Squicciarini, "Deeptrust: An automatic framework to detect trustworthy users in opinion-based systems," in *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, 2020, pp. 29–38.

[29] M. Joaristi, E. Serra, and F. Spezzano, "Inferring bad entities through the panama papers network," in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2018, pp. 767–773.

[30] ——, "Detecting suspicious entities in offshore leaks networks," *Social Network Analysis and Mining*, vol. 9, no. 1, pp. 1–15, 2019.

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

[32] 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.

[33] 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.

[34] 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.

[35] 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.

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

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