HOVER: Homophilic Oversampling via Edge Removal for Class-Imbalanced Bot Detection on Graphs

Bradley Ashmore
Wright State University
Dayton, OH, USA
ashmore.3@wright.edu

Lingwei Chen
Wright State University
Dayton, OH, USA
lingwei.chen@wright.edu

ABSTRACT

As malicious bots reside in a network to disrupt network stability, graph neural networks (GNNs) have emerged as one of the most popular bot detection methods. However, in most cases these graphs are significantly class-imbalanced. To address this issue, graph oversampling has recently been proposed to synthesize nodes and edges, which still suffers from graph heterophily, leading to suboptimal performance. In this paper, we propose HOVER, which implements **H**omophilic **O**versampling **V**ia **E**dge **R**emoval for bot detection on graphs. Instead of oversampling nodes and edges within initial graph structure, HOVER designs a simple edge removal method with heuristic criteria to mitigate heterophily and learn distinguishable node embeddings, which are then used to oversample minority bots to generate a balanced class distribution without edge synthesis. Experiments on TON IoT networks demonstrate the state-of-the-art performance of HOVER on bot detection with high graph heterophily and extreme class imbalance.

CCS CONCEPTS

• **Security and privacy** → *Network security*; • **Computing methodologies** → *Neural networks.*

KEYWORDS

Bot detection, Graph convolutional networks, Imbalanced classes, Homophily and heterophily

ACM Reference Format:
Bradley Ashmore and Lingwei Chen. 2023. HOVER: Homophilic Oversampling via Edge Removal for Class-Imbalanced Bot Detection on Graphs. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3583780.3615264

1 INTRODUCTION

Malicious bots are compromised victim computers or IoT devices in a network by communicating in peer-to-peer (P2P) structures [27], which can infect a system or commit other malicious activities [3, 21]. As such, graph neural networks (GNNs) have emerged as one of the most popular bot detection methods [1, 15, 26, 30]. Bot detection is formulated as a node classification problem, where various GNNs [9, 10, 13] can be devised to perform neighborhood aggregation and label propagation through graph structure. For example, Zhao et al. [30] customized GNNs to detect botnet nodes within Internet communication graphs by identifying their topological patterns. Lo et al. [15] built a graph isomorphism network (GIN) with grouped reversible residual connections to handle performance degradation on botnet communication graphs. The use of GNNs for bot detection is appealing, as they provide a method to boost the expressiveness of node presentations from network traffic.

The nodes within such communication graphs fall into two classes: malicious bots and benign devices, where the imbalance ratio r = #malicious bots/#benign devices ($0 \leq r \leq 1$) can be used to quantify the extent of class imbalance. We calculate r over TON IoT networks [2, 19], a dataset widely used for bot detection (detailed in Section 4.1), where in most cases the imbalance ratios are below 0.03. This indicates a significant disparity where malicious bots are greatly outnumbered by benign devices. When trained on class-imbalanced graphs, GNNs may favor the majority class over the minority class and degrade their detection performances [11]. Unfortunately, existing graph-based bot detection methods have rarely explored this class-imbalanced issue. Oversampling has recently been generalized to imbalanced graphs [8, 28], which synthesizes nodes in the embedding space and then generates edges to connect them with existing nodes. However, this formulation has two shortcomings: (1) class imbalance in graphs is often accompanied by heterophily [29], where minority nodes tend to connect with majorities and their embeddings can be easily smoothed [25, 31]; and (2) edges generated using weighted inner products of node embeddings may not represent real-world relations between nodes with constraints (e.g., device communications can be only enabled by TCP), leading to noisy neighborhoods for message passing.

In this paper, we propose HOVER, which implements **H**omophilic **O**versampling **V**ia **E**dge **R**emoval for class-imbalanced bot detection on graphs. Instead of oversampling nodes and edges within initial graph, HOVER follows the GNN learning steps to first acquire higher-level node representations that capture node features and graph structure by employing heterophily-aware neighborhood aggregation, and then apply an oversampling algorithm over the learned minority node representations to generate a balanced class distribution for bot detection, without the need for edge synthesis. Unlike adaptive aggregation techniques [7, 20], HOVER leverages a simpler yet effective edge removal method to mitigate homogenization and over-smoothing on minority node embedding resulting from heterophily. With examination of edge patterns, we design a set of heuristic criteria to guide edge removal, which increases

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10…$15.00
https://doi.org/10.1145/3583780.3615264

graph homophily and the effectiveness of subsequent oversampling, and thus improves class-imbalanced bot detection performance.

2 PROBLEM STATEMENT

In this paper, we explore graph-based class-imbalanced bot detection, where bots and benign devices are distributed in a network. We represent this network as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V}(n = |\mathcal{V}|)$ is the set of devices, $\mathcal{E}$ is the set of edges, and $\mathbf{X} \in \mathrm{R}^{n \times d}$ is feature matrix. Each labeled node has a ground truth $y \in \{0, 1\}$ where 0 denotes benign device and 1 denotes malicious bot. Edges E can be organized as an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{A}_{ij} = \{0, 1\}$, where if $(v_i, v_j) \in E$, then $\mathbf{A}_{ij} = 1$; otherwise, $\mathbf{A}_{ij} = 0$ [12]. We cast bot detection as a node classification problem, and aim to mitigate the impact of imbalanced classes, such that a GNN model f can be learned on $\mathcal{G}$ to correctly identify malicious bots and benign devices. Specifically, this model is to train $f_{\mathbf{W}}(\mathbf{A}, \mathbf{X})$ in a semi-supervised manner by minimizing the training loss:

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \mathcal{L}(f_{\mathbf{W}}(\mathbf{A}, \mathbf{X}), \mathbf{y}_l) \tag{1}$$

where $\mathbf{W}$ is weight matrix, and $\mathbf{y}_l$ is class vector. We use a graph convolutional network (GCN) [10] to facilitate node representation learning that performs neighborhood aggregation as:

$$\mathbf{H}^{(l)} = \sigma\left(\tilde{\mathbf{A}}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}\right) \tag{2}$$

At layer l ($l \geq 1$), $\mathbf{H}^{(l-1)}$ and $\mathbf{H}^{(l)}$ are its input and output, $\mathbf{W}^{(l)}$ is weight matrix, σ is an activation function, $\tilde{\mathbf{A}} = \hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}$, $\hat{\mathbf{A}} = \mathbf{A}+\mathbf{I}$, and $\hat{\mathbf{D}}$ is the diagonal degree matrix defined on $\hat{\mathbf{A}}$.

3 PROPOSED MODEL: HOVER

In this section, we present our designed class-imbalanced bot detection model HOVER, the overview of which is illustrated in Figure 1.

3.1 Unsupervised Edge Removal

Given a communication graph $\mathcal{G}$, edges can be divided into three types: bot edge that connects two bot nodes, benign edge that connects two non-bot nodes, and inter-class edge that connects a bot node to a benign node. Bot nodes are often connected with benign nodes, leading to heterophily issue [29, 32]. The heterophily ratio in $\mathcal{G}$ can be defined as follows:

$$h = \frac{|\{(v_i, v_j) : (v_i, v_j) \in \mathcal{E} \wedge y_{v_i} \neq y_{v_j}\}|}{|\mathcal{E}|} \tag{3}$$

We calculate h over TON IoT networks, where the heterophily ratios for some graphs are above 0.70, indicating high heterophily.

Due to neighborhood aggregation, bot node embeddings generated by GCNs in a graph with high heterophily tend to be homogenized by benign node embeddings, resulting in smoothed and indistinguishable embedding space for subsequent oversampling. To address graph heterophily, the prevalent methods attend to refine neighborhood aggregation to adaptively exploit homophilic and heterophilic information [7, 14, 16, 20], which improve the learning performance, but complicate the models with extra computational cost. In this paper, we take a different direction to formulate a simple yet effective edge removal method to mitigate the impact of heterophily. Edge removal has proved to yield advantages for graph learning, such as preventing over-smoothing [5, 22], enhancing robustness [6, 17], and increasing homophily [18]. This thus provides a great potential to address heterophily problem.

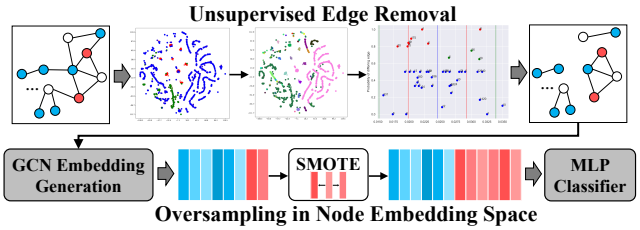


Figure 1: Overview of HOVER: unsupervised edge removal is performed to mitigate heterophily and learn distinguishable node embeddings, which are then used to oversample bots to generate a balanced class distribution for bot detection.

Heuristics. Edge removal involves selectively removing certain edges from the graph based on specific criteria to influence information flow. From a typical perspective, edges connecting nodes with dissimilar features contribute to the heterophily issue. Similarity between node features are thus measured as criteria to select such edges to drop [6, 24]. But heterophilic edges in a graph do not necessarily degrade the GNN performance [18]. This motivates us to reconsider edge removal criteria. With a close examination of edge patterns, we propose two heuristics: (1) edges with similar semantics, carrying homophily or heterophily information, exert a similar influence on information propagation with the graph; (2) edges captured by distributions of different properties should be discrepant, contributing differently to graph connections.

Edge clustering. To generalize the heuristics into edge removal criteria, the most direct way is utilizing edge clusters. We first produce a collection of edge vectors to characterize edges, each of which is defined as difference between initial feature vectors $\mathbf{X}_{v_i}$ and $\mathbf{X}_{v_j}$ of nodes v_i and v_j connected by the given edge:

$$\mathbf{E}_{v_i,v_j} = |\mathbf{X}_{v_i} - \mathbf{X}_{v_j}|, \quad (v_i, v_j) \in \mathcal{E} \tag{4}$$

$\mathbf{E}_{v_i,v_j}$ provides a quantitative measure of every edge in $\mathcal{G}$, which is easily feasible for edge clustering. Specifically, edge vectors are clustered using DB-SCAN [23], which allows for non-circular and abnormal cluster shapes to be discovered. The upper half of Figure 1 shows a visualization of edge types and the resulting clusters using DB-SCAN. To further differentiate edge distributions across clusters, we examine the properties of distributions by calculating the mean and standard deviation for a collection of cluster metrics $\mathcal{M}$, including number of edges in a cluster, width, centroid magnitude, and average intra-cluster distance. In this way, for each cluster, we can determine the number of standard deviations from the mean for each metric. Then all the clusters are binned with one standard deviation size. This produces a set of deviation bins denoted by $\mathrm{DB}_{m,b}$, where m is the metric and b is the bin index for metric m.

Criteria. According to the heuristics, for each cluster, we can utilize available training labels to estimate the cluster's probability distribution. Our goal is to understand the probability that any edge in a cluster c is a bot edge $P(\mathrm{bot}|c)$, benign edge $P(\mathrm{ben}|c)$, or inter-class edge $P(\mathrm{int}|c)$. Each probability is calculated using

the training edges in a cluster, where the number of edges of each type in training set is divided by the total number of training edges. Note that edges in test set or edges that straddle the test and training nodes are omitted from this calculation. Given the clusters in each deviation bin $\mathrm{DB}_{m,b}$, we aggregate this distribution with these probabilities to give us three conditional probabilities regarding cluster metric m and deviation bin b:

$$P(\mathrm{int}|m,b) = \frac{1}{|\mathrm{DB}_{m,b}|} \sum_{c \in \mathrm{DB}_{m,b}} P(\mathrm{int}|c) \tag{5}$$

$$P(\mathrm{bot}|m,b) = \frac{1}{|\mathrm{DB}_{m,b}|} \sum_{c \in \mathrm{DB}_{m,b}} P(\mathrm{bot}|c) \tag{6}$$

$$P(\mathrm{ben}|m,b) = \frac{1}{|\mathrm{DB}_{m,b}|} \sum_{c \in \mathrm{DB}_{m,b}} P(\mathrm{ben}|c) \tag{7}$$

Afterwards, we use a statistics-based approach to decide the specific edge removal criteria for relative simplicity, explainability, and flexibility. Specifically, we leverage the conditional probabilities listed above to develop three edge removal criteria: maximum inter-class edge probability (MxIEP), minimum benign edge probability (MnBEP), and maximum probability difference (MxPD):

$$\text{MxIEP: } c = \underset{c \in \mathrm{DB}_{m,b}}{\mathrm{argmax}}\ P(\mathrm{int}|c)\ \ s.t.\ (m,b) = \underset{m \in \mathcal{M}, b \in \mathcal{B}}{\mathrm{argmax}}\ P(\mathrm{int}|m,b) \tag{8}$$

$$\text{MnBEP: } c = \underset{c \in \mathrm{DB}_{m,b}}{\mathrm{argmin}}\ P(\mathrm{ben}|c)\ \ s.t.\ (m,b) = \underset{m \in \mathcal{M}, b \in \mathcal{B}}{\mathrm{argmin}}\ P(\mathrm{ben}|m,b) \tag{9}$$

$$\begin{aligned} \text{MxPD: } c &= \underset{c \in \mathrm{DB}_{m,b}}{\mathrm{argmax}}\ P(\mathrm{int}|c) - P(\mathrm{ben}|c) \\ s.t.\ (m,b) &= \underset{m \in \mathcal{M}, b \in \mathcal{B}}{\mathrm{argmax}}\ P(\mathrm{int}|m,b) - P(\mathrm{ben}|m,b) \end{aligned} \tag{10}$$

Each criteria returns a cluster to guide the edge removal.

These criteria allow HOVER to reason about graphs with different properties regarding high homophily, high heterophily, or unique structural complexities: MxIEP targets inter-class edges, MnBEP avoids disturbing homophilic edges, and MxPD considers multiple probabilities to precisely target clusters of mostly inter-class edges. The edge removal process is iterative, where the edges in the cluster with the highest (or lowest) value for the criteria in use are removed. If the desired number of edges has not been reached yet, the next most appropriate cluster is returned, from which edges are removed.

3.2 Oversampling in Node Embedding Space

The unsupervised edge removal step accepts $\mathcal{G}$ and produces a reduced-edge graph $\mathcal{G}'$, which is fed to a GCN to obtain improved node embeddings that are distinguishable and encode graph structure and node features. In this node embedding space, we can directly synthesize new bot samples to balance class distribution. Here we employ SMOTE as our oversampling algorithm [4], due to its popularity and performance in oversampling. Given node embeddings $\mathbf{H}'$ generated on $\mathcal{G}'$ using Eq. (2), for a bot node v and a random bot node u from v's nearest neighbors, their embeddings are $\mathbf{H}'_v$ and $\mathbf{H}'_u$, such that a new bot sample can be synthesized as:

$$\mathbf{h} = (1-\delta)\mathbf{H}'_v + \delta \mathbf{H}'_u \tag{11}$$

where δ is a random variable ranging from 0 to 1. Since both $\mathbf{H}'_v$ and $\mathbf{H}'_u$ are bot nodes, $\mathbf{h}$ equalizes the same class distribution. In this way, we can oversample bot nodes that are appended to previous nodes for classifier training.

Table 1: Statistics of bot detection datasets

File	Bots	Benigns	Im-Ratio	Edges	Heterophily Ratio
3	32	1,604	0.020	9,492	0.349
4	38	1,623	0.023	9,644	0.730
5	41	1,638	0.024	9,716	0.731
10	56	2,048	0.027	11,928	0.332
15	57	14,009	0.004	60,372	0.067

3.3 Model Optimization

After oversampling, $\tilde{\mathbf{H}}$ is augmented by concatenating representations of initial labeled nodes with those of synthetic nodes. As $\tilde{\mathbf{H}}$ is of balanced class distribution, we feed it to an MLP directly to calculate the prediction output $\mathbf{Z} = \mathrm{MLP}(\tilde{\mathbf{H}})$ and the loss as follows:

$$\mathcal{L} = -\frac{1}{|\tilde{\mathbf{H}}|} \sum_{i=1}^{|\tilde{\mathbf{H}}|} \mathbf{y}_i * \log(\mathbf{Z}_i) \tag{12}$$

As oversampling relies on node embeddings, HOVER pre-trains a GCN on $\mathcal{G}'$ as embedding generator, and minimizes the loss $\mathcal{L}$ to optimize the final classifier. We also evaluate the difference between training GCN and MLP separately and as a whole in ablation study.

4 EXPERIMENTAL RESULTS AND ANALYSIS

4.1 Experimental Setup

Datasets. We test HOVER on TON IoT networks [2, 19], which is an assortment of Industry 4.0/IoT and IIoT datasets. Each network is recorded in a capture file that is translated into a communication graph, where protocol used, packet size, and amount of data transmitted and received are used as node features. To represent unique graph constructs, we select graphs with high heterophily ($h > 0.6$), high homophily ($h < 0.4$), and extreme class imbalance ($r < 0.01$) respectively. Table 1 details the data statistics. We separate nodes in each graph into an 80-20 split for training and testing.

Baselines. We select baselines using rebalance and edge removal for comparison, including GCN [10], SMOTE+GCN (GCN operating on data oversampled with SMOTE), GCN+SMOTE (oversampling on graph embeddings), GraphSMOTE [28], Euclidean+SMOTE (oversampling on updated embeddings from graph with the most dissimilar edges removed), and Random+SMOTE (oversampling on updated embeddings from graph with random edges removed).

Parameters. HOVER has three hyperparameters. n_pts and ϵ are from DB-SCAN: $n\_pts = 3$ is the minimum number of edges to form a cluster; ϵ is the radius of an edge's neighborhood in a cluster. $drop\_rate$ is the ratio of removed edges to all edges. The impacts of ϵ and $drop\_rate$ are evaluated in Section 4.2. The GCN in HOVER consists of two graph convolution layers, and MLP comprises two fully connected layers. Both models employ Adam configured with 0.01 learning rate and $5e-4$ L2 regularization on the weights. F1-score is used as the primary evaluation metrics to provide insight into a model's effectiveness classifying the minority class.

4.2 Evaluation of HOVER

Effectiveness. In this section, we evaluate the performance of HOVER with $\epsilon \in [0.1, 0.5]$, $drop\_rate \in [0.1, 0.5]$, and edge removal criteria include MxIEP, MnBEP, and MxPD. The experimental results are illustrated in Figure 2. As we can see, HOVER achieves

Table 2: Comparison of HOVER with baselines

Model	File 3		File 4		File 5		File 10		File 15	
	Acc(%)	F1	Acc(%)	F1	Acc(%)	F1	Acc(%)	F1	Acc(%)	F1
GCN	98.5	0.000	97.6	0.000	97.0	0.000	95.5	0.000	99.7	0.000
SMOTE+GCN	97.6	0.526	97.0	0.546	97.9	0.666	99.1	0.900	96.8	0.165
GCN+SMOTE	98.5	0.000	97.6	0.000	97.2	0.000	95.5	0.000	99.7	0.000
GraphSMOTE	98.0	0.495	97.7	0.642	97.5	0.494	91.8	0.494	89.1	0.499
Euclidean+SMOTE	97.6	0.556	97.6	0.667	98.1	0.625	97.9	0.740	94.2	0.089
Random+SMOTE	98.8	0.667	97.6	0.334	96.7	0.417	99.0	0.556	99.7	0.000
HOVER$_{\text{MxIEP}}$	**100.0**	**1.000**	**99.1**	0.762	**99.1**	**0.824**	**99.3**	**0.914**	**99.9**	0.750
HOVER$_{\text{MnBEP}}$	99.1	0.769	98.5	**0.778**	98.5	0.667	**99.3**	**0.914**	**99.9**	**0.800**
HOVER$_{\text{MxPD}}$	**100.0**	**1.000**	**99.1**	0.769	**99.1**	**0.824**	99.1	0.889	**99.9**	0.750

*Despite the frequency of zero values (0.000) in this table, these are not errors or neglected placeholders. The zero values have been verified during review.

the state-of-the-art results of bot detection when the classes are extremely imbalanced on graphs. The best F1-scores are 1.000, 0.778, 0.824, 0.914, 0.800 for five datasets, respectively.

Impact of ϵ. As shown in Figure 2(a) (using MxIEP and $drop\_rate$ = 0.4), when ϵ is small (0.1) or large (0.5), edge removal is ineffective that leads to low F1-score. Small ϵ increases outliers and the risk of clustering only edges in test set; large ϵ renders clusters monolithic with similar probability densities. When $\epsilon = 0.3$, HOVER presents a good trade-off with best performance. Therefore, we use $\epsilon = 0.3$ throughout the following evaluations.

Impact of $drop\_rate$. We use MxIEP with $\epsilon = 0.3$ to run the experiments across different drop rates, and report the results in Figure 2(b). As $drop\_rate$ increases, the F1-score increases as well because of lower heterophily and higher oversampling effectiveness. When $drop\_rate = 0.3$, the performance rises to a stable high level, demonstrating the effectiveness of edge removal that removes relatively small number of edges to preserve graph structure.

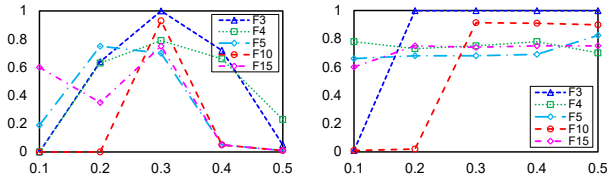
Impact of edge removal criteria. We report the results using three edge removal criteria in Table 2, where MxIEP and MxPD outperform MnBEP by 20% F1-score difference in some cases. We also see that MxIEP is generally the best choice, which is logical as MxIEP targets heterophilic edges. When the graph is extremely homophily (e.g., file 15), MnBEP has a better potential than MxIEP and MxPD for class-imbalanced node classification.

4.3 Comparison with Baselines

The comparative results are presented in Table 2. We can observe that GCN suffers from heterophily and imbalanced classes that lead to 0.000 F1-score across all datasets. As the learned node embeddings are homogenized, oversampling in such embedding space is ineffective, resulting in 0.000 F1-socre for GCN+SMOTE and low performance for GraphSMOTE. Surprisingly, SMOTE+GCN performs well, which may be due to the expressiveness of raw node features. Euclidean+SMOTE removes edges with top dissimilarity ($drop\_rate = 0.5$), delivering better performance than other baselines. However, HOVER outperforms Euclidean+SMOTE by a large margin, especially for file 15. This confirms the reasonability of our heuristics and effectiveness of our edge removal criteria.

4.4 Ablation Study

An ablation study is conducted using MxIEP on files 4 and 5. We individually eliminate one component from the model, and evaluate



(a) F1 on radius ϵ (b) F1 on drop rate

Figure 2: Evaluation on ϵ and $drop\_rate$ using MxIEP.

Table 3: Ablation study: Edge Removal (ER), Oversampling (OS), and Separate Training (ST)

ER	OS	ST	F1 on File 4	F1 on File 5
	✓	✓	0.000	0.000
✓		✓	0.667	0.667
✓	✓		0.047	0.571
✓	✓	✓	0.762	0.824

their performance. The results are reported in Table 3. A lack of edge removal prevents the GCN from generating useful embeddings for oversampling, where bot nodes are obscured by benign nodes leading to 0.000 F1-score. A lack of oversampling improves results, demonstrating the significance to address heterophily and effectiveness of edge removal in this respect. Also, model performance suffers from drastic drop when GCN and MLP are trained as a whole. This may be because that the GCN fails to receive supervision from synthesized data, which are still affected by imbalance classes. These observations reaffirm the effectiveness of our design.

5 CONCLUSION

In this paper, we introduce HOVER, an approach to detect bots on graphs with high heterophily and imbalanced classes. HOVER proceeds by designing an edge removal method with heuristic criteria to mitigate heterophily and learn high-level and distinguishable node embeddings, and then synthesizing minority bot nodes in such node embdding space to generate a balanced class distribution. The state-of-the-art bot detection results validate HOVER's effectiveness in overcoming graph heteropholy and class imbalances, and its simplicity and feasibility to detect bots in practice.

ACKNOWLEDGMENTS

This work is partially supported by the NSF under grant CNS-2245968. The authors would also like to thank the reviewers for their valuable feedback.

REFERENCES

[1] Seyed Ali Alhosseini, Raad Bin Tareaf, Pejman Najafi, and Christoph Meinel. 2019. Detect me if you can: Spam bot detection using inductive representation learning. In *Companion Proceedings of The 2019 World Wide Web Conference*. 148–153.

[2] Abdullah Alsaedi, Nour Moustafa, Zahir Tari, Abdun Mahmood, and Adnan Anwar. 2020. TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. *Ieee Access* 8 (2020), 165130–165150.

[3] Moitrayee Chatterjee, Akbar Siami Namin, and Prerit Datta. 2018. Evidence fusion for malicious bot detection in IoT. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 4545–4548.

[4] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[5] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 3438–3445.

[6] Lingwei Chen, Xiaoting Li, and Dinghao Wu. 2021. Enhancing robustness of graph convolutional networks via dropping graph connections. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part III*. Springer, 412–428.

[7] Lun Du, Xiaozhou Shi, Qiang Fu, Xiaojun Ma, Hengyu Liu, Shi Han, and Dongmei Zhang. 2022. GBK-GNN: Gated Bi-Kernel Graph Neural Networks for Modeling Both Homophily and Heterophily. In *Proceedings of the ACM Web Conference 2022*. 1550–1558.

[8] Yijun Duan, Xin Liu, Adam Jatowt, Hai-tao Yu, Steven Lynden, Kyoung-Sook Kim, and Akiyoshi Matono. 2022. Anonymity can Help Minority: A Novel Synthetic Data Over-Sampling Strategy on Multi-label Graphs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 20–36.

[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[10] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[11] Quan Li, Lingwei Chen, Yong Cai, and Dinghao Wu. 2023. Hierarchical Graph Neural Network for Patient Treatment Preference Prediction with External Knowledge. In *Advances in Knowledge Discovery and Data Mining: 27th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2023, Osaka, Japan, May 25–28, 2023, Proceedings, Part III*. Springer, 204–215.

[12] Quan Li, Xiaoting Li, Lingwei Chen, and Dinghao Wu. 2022. Distilling Knowledge on Text Graph for Social Media Attribute Inference. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2024–2028.

[13] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. 2021. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems* (2021).

[14] Yixin Liu, Yizhen Zheng, Daokun Zhang, Vincent Lee, and Shirui Pan. 2022. Beyond Smoothing: Unsupervised Graph Representation Learning with Edge Heterophily Discriminating. *arXiv preprint arXiv:2211.14065* (2022).

[15] Wai Weng Lo, Gayan Kulatilleke, Mohanad Sarhan, Siamak Layeghy, and Marius Portmann. 2023. XG-BoT: An explainable deep graph neural network for botnet detection and forensics. *Internet of Things* 22 (2023), 100747.

[16] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. 2022. Revisiting heterophily for graph neural networks. *arXiv preprint arXiv:2210.07606* (2022).

[17] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. 2021. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM international conference on web search and data mining*. 779–787.

[18] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. 2021. Is homophily a necessity for graph neural networks? *arXiv preprint arXiv:2106.06134* (2021).

[19] Nour Moustafa. 2021. A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustainable Cities and Society* 72 (2021), 102994.

[20] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287* (2020).

[21] Bruno Martins Rahal, Aldri Santos, and Michele Nogueira. 2020. A distributed architecture for DDoS prediction and bot detection. *IEEE Access* 8 (2020), 159756–159772.

[22] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903* (2019).

[23] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS)* 42, 3 (2017), 1–21.

[24] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial examples on graph data: Deep insights into attack and defense. *arXiv preprint arXiv:1903.01610* (2019).

[25] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. 2022. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1287–1292.

[26] Bonan Zhang, Jingjin Li, Chao Chen, Kyungmi Lee, and Ickjai Lee. 2022. A Practical Botnet Traffic Detection System Using GNN. In *Cyberspace Safety and Security: 13th International Symposium, CSS 2021, Virtual Event, November 9–11, 2021, Proceedings 13*. Springer, 66–78.

[27] Junjie Zhang, Roberto Perdisci, Wenke Lee, Xiapu Luo, and Unum Sarfraz. 2013. Building a scalable system for stealthy P2P-botnet detection. *IEEE transactions on information forensics and security* 9, 1 (2013), 27–38.

[28] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. 2021. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM international conference on web search and data mining*. 833–841.

[29] Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S Yu. 2022. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082* (2022).

[30] Jiawei Zhou, Zhiying Xu, Alexander M Rush, and Minlan Yu. 2020. Automating botnet detection with graph neural networks. *arXiv preprint arXiv:2003.06344* (2020).

[31] Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. 2021. Graph neural networks with heterophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11168–11176.

[32] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems* 33 (2020), 7793–7804.