ISSN 2616-5775 Vol. 7, Issue 4: 17-23, DOI: 10.25236/AJCIS.2024.070403

Debiasing Frequency Adaptive Graph Neural Network-based Fraud Detector

Hongjia Pi[1,a,*]

[1]College of Information Engineering, Nanjing University of Finance & Economics, Nanjing, China
[a]2162624523@qq.com
[*]Corresponding author

Abstract: *The detection of anomalies in graph data is a critical task across various domains, such as fraud detection in social and commercial networks. Traditional Graph Neural Network (GNN) models often struggle with dataset biases, including label bias and keyword bias, which impair their generalization abilities. This paper introduces a novel Debiasing Frequency Adaptive GNN (DFA-GNN) that addresses these challenges by enhancing model accuracy and reducing dataset biases. Unlike previous approaches, DFA-GNN adapts to the complexity of node relationships and the frequency of interaction signals, making it particularly effective for node-based anomaly detection. By decomposing the input graph into several relation graphs and employing a frequency adjusting filter, DFA-GNN selectively processes signals of varying frequencies, catering to both homophily and heterophily conditions. Additionally, our counterfactual inference mechanism mitigates unintended dataset biases, further enhancing the model's prediction accuracy. Our extensive experiments on fraud detection datasets such as Yelp have shown that DFA-GNN has excellent performance in identifying subtle and complex anomalies, outperforming existing models in accuracy and debiasing ability.*

Keywords: *Graph Anomaly Detection, Counterfactual Inference, Heterogeneous Graph*

1. Introduction

In the digital era, fraud detection has emerged as a paramount challenge across various platforms[2], including e-commerce, social networks, and financial systems. GNN-based models excel in capturing complex interactions between entities, making them particularly suited for identifying fraudulent activities. However, current models often struggle with dataset biases[5], including label and keyword biases, which can severely impact the generalizability and fairness of fraud detection systems. These biases can lead to overfitting on specific data distributions, rendering models less effective when confronted with new or evolving fraudulent schemes. Addressing these challenges, we introduce a novel GNN-based model that significantly enhances fraud detection capabilities in the face of heterophily and dataset biases. Our model employs a debiasing frequency adaptive mechanism. This mechanism allows for dynamic adjustment to the model's focus, enabling it to capture both low-frequency and high-frequency signal patterns within the graph[6]. Furthermore, our approach integrates a counterfactual inference [9] to mitigate the effects of dataset biases, enhancing the model's robustness and ensuring fairer fraud detection outcomes. Through extensive experiments on anomaly detection datasets, our model demonstrates good performance in detecting camouflaged fraudsters, outperforming existing benchmarks in both accuracy and debiasing capabilities.

The paper's contributions are as follows: We propose a novel GNN architecture that effectively addresses these challenges through a debiasing frequency adaptive mechanism and counterfactual inference, enhancing the model's ability to detect camouflaged fraudsters. We have validated its potential in advancing the field of graph anomaly detection through extensive experiments conducted on real-world datasets.

2. Methodology

In this work, our goal is to process graphs with fraudsters to demonstrate the role of the model in the field of fraud detection. The framework of DFA-GNN consists of relation learning module, frequency adjusting module and dataset debiasing module. In Figure 1, as you can see, we can clearly see the overall framework of the model. Given an input graph $\mathcal{G} = (\nu, \varepsilon)$, comprised of a set ν representing N nodes characterized by features $\chi = (x_1, x_2, \ldots, x_N) \in R^{N\times d}$ and an edge set $\varepsilon \subseteq \upsilon \times \upsilon$, where each node $v \in \upsilon$ has feature, so let's use $x_v \in \chi$ to represent it. Each edge $e_{u,v} \in \varepsilon$.

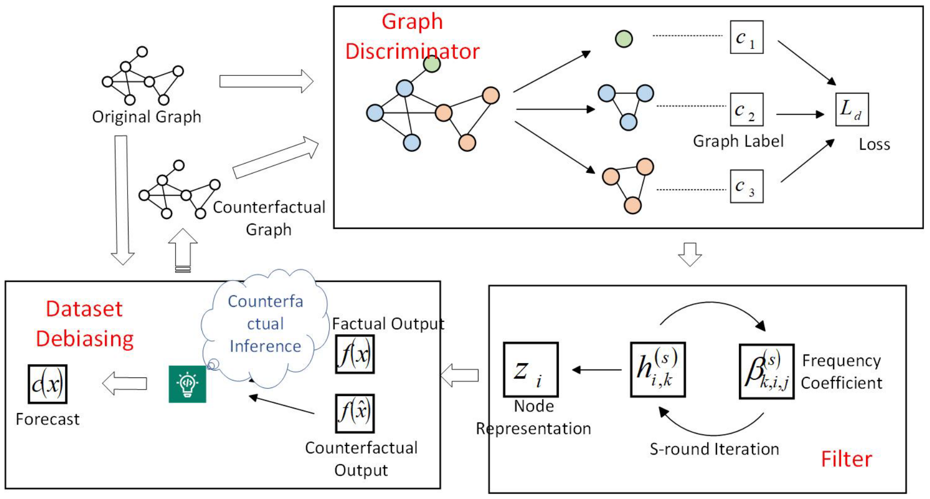

Figure 1: DFA-GNN framework.

2.1. Relation Learning Module

We set the objective of the relation learning unit to separate the input graph $\varsigma = (\upsilon, \varepsilon)$, so the graph ς becomes K relation graphs $\{G_k\}_{k=1}^{K}$. We can imagine it as approximate $\phi^{-1}(\cdot)$ with $\psi(\cdot)$. $h_i' = Wx_i$, where $W \in R^{F'\times d}$ and F' is the dimension of hidden space. In the initial step, we make the input nodes into a lower-dimensional space. In next step, the relation coefficients between nodes i and j regarding relation $k(1 \le k \le K)$ is as follows:

$$G_{k,i,j} = \sigma(\Omega_k(h_i', h_j')) \tag{1}$$

where $\sigma = tanh(\cdot)$ and $\Omega_k(\cdot)$ is structured as MLP. A particular relation graph G_k can be on behalf of a relation k, as all the coefficients are computed. We learn a graph label $c_k \in R^K$ for each relation graph G_k to guarantee that the learned $\{G_k\}_{k=1}^{K}$ is distinctive. Graph label is defined as

$$c_k = \sigma(f(Readout(GAE(G_k, H')))) \tag{2}$$

where $GAE(\cdot)$ is a two-layer GAE[12] which generates new features for each node. $H = (h_1', h_2', \ldots, h_N')$ and relation graph G_k are inputs. $f(\cdot)$ is a fully connected layer. $Readout(\cdot)$ conducts global average pooling and $\sigma(\cdot) = softmax(\cdot)$. By maximizing the differences between different $\{c_k\}_{k=1}^{K}$, the relation learning module maximize the differences between those relation graphs $\{G_k\}_{k=1}^{K}$ indirectly and we define the loss function training the graph discriminator as

$$L_d = -\frac{1}{K}\sum_{k=1}^{K} log(c_k[k]) \tag{3}$$

where c_k is predicted label distribution vector, $c_k[k]$ is its kth element in relation graph G_k.

2.2. Frequency adjusting Module

The aim of frequency adjusting unit is to pick signals of various frequencies adaptively, with the frequency adjusting filter playing an important role. For the convolution kernel f and signal x, the convolutional $*_G$ is

$$f *_G x = U((U^T f) \odot (U^T x)) = U g_\theta U^T x \tag{4}$$

where g_θ denotes a diagonal matrix and $\odot$ is the elementwise product of two vectors. Parameterizing g_θ with a polynomial expansion $g_\theta = \sum_{k=0}^{K-1} a_k \Lambda^k$ by GCN-Cheby. The convolutional kernel[13] is $g_\theta = I - \Lambda$ in GCN. In raw features, the frequency adjusting filter can choose the high- and low- frequency signals[14] adaptively, which is as follows:

$$F_A = \alpha I + \beta \widetilde{D}^{-1/2} \tilde{A} \widetilde{D}^{-1/2} = (\alpha + \beta) I - \beta L \tag{5}$$

where β is a learnable parameter. α is a scaling hyperparameter and $\alpha \in (0, 1]$. The convolutional kernel f is replaced by F_A, so the signal x is filtered by F_A as

$$F_A *_G x = U[(\alpha+\beta)I - \beta\Lambda]U^T x. \tag{6}$$

Therefore, the convolutional kernel of F_A become $g_\theta = (\alpha+\beta)I - \beta\Lambda$, that is, $g_\theta(\lambda_i) = \alpha + \beta - \beta\lambda_i$. When $\beta < 0$, F_A is a high-pass filter[8]; When $\beta > 0$, F_A is a low-pass filter. Hence, a adaptive and shared mechanism is used to learn $\{\beta_{i,j}\}_{i,j=1}^{N}$, which are local node-specific frequency coefficients and given by

$$\beta_{i,j} = \sigma\left(a^T\left[h_i \| h_j\right]\right) \tag{7}$$

where $a \in R^{2F}$ is a shared convolutional kernel and $h_i \| h_j$ is the concatenation operation. $\sigma(\cdot) = tanh(\ \cdot)$. After calculating $\{\beta_{i,j}\}_{i,j=1}^{N}$, the features of neighbor node can be aggregated by:

$$\widetilde{h_i} = \alpha h_i + \sum_{j \in N_i} \frac{\beta_{i,j}}{\sqrt{|N_i||N_j|}} h_j \tag{8}$$

where the neighborhood nodes of nodes i and j are N_i and N_j.

2.3. Dataset Debiasing Module

Let X_{text} and Y respectively denote the input (text features) and output (category) spaces. The core idea of CORSAIR[9] is as follows: We firstly train a "poisoned" classifier without considering the bias of the dataset. According to the reasons for the biases, we post-adjust the biased predictions in inference. CORSAIR has three main sections which are biased learning, bias distillation, and bias removal.

2.3.1. Label Bias Distillation

The in-coming links of a cause variable X_{text} are eliminated by causal intervention operation. In pursuit of this objective, we designate $\hat{x}$ to represent the conceived fully blindfolded counterfactual dataset. To generate a counterfactual embedding, all words within the test dataset x are uniformly masked. The corresponding counterfactual output, denoted as $f(\hat{x})$, is obtained through a feedforward process using the trained model. Consequently, the fully blindfolded counterfactual output can be expressed as:

$$\begin{gathered} P\big(Y \big| do(X_{text})\big) = f(\hat{x}) = f(\langle w_1, w_2, \cdots, w_n\rangle) \\ \forall w_i \in \hat{x}, w_i \leftarrow [MASK] \end{gathered} \tag{9}$$

Manifests as the label bias captured by the trained model M, with the placeholder $[MASK]$ representing a special mark used for single-word masking. In practice, we utilize the average text feature across the entire training set as the embedding for the counterfactual dataset.

2.3.2. Keyword Bias Distillation

Partially blindfolded counterfactual dataset can also be used to removing bias[1]. We use an effective masking strategy to extract the main content of the text features by use discriminative text summarization methods. We mask content words which are important classification clues and subsequently expose others as potentially harmful biasing factors. Thus, the partially blindfolded counterfactual output:

$$\begin{gathered} f(\tilde{x}) = f(\langle w_1, w_2, \ldots, w_n\rangle) \\ \forall w_i \in \tilde{x}, \begin{cases} w_i \leftarrow [MASK] & \text{if } w_i \in x_{content} \\ w_i \leftarrow w_i & \text{if } w_i \in x_{context} \end{cases} \end{gathered} \tag{10}$$

Reflects as the keyword bias captured by model M, where $x_{context}$ and $x_{content}$ are the context and the main content of x. We can employ the Jieba tool to implement discriminative text summarization as its TextRank-based interface can identify words that are likely to impact the semantic meaning of a sentence, designating them as content. Conversely, words deemed potentially discriminatory or unfair, such as stop words, are retained as context.

2.4. DFA-GNN Architecture

Firstly, the input nodes are transformed to a low-dimensional space, $h_{i,k}^{(0)} = \sigma(\overline{W} x_i)$, where $1 \le k \le K$, $\sigma(\cdot) = ReLu(\cdot)$ and the shared weight matrices are $\overline{W} \in R^{F \times d}$. Subsequently, we execute the relation-based multihop frequency adjusting message-passing through iteration S, as follows:

$$h_{i,k}^{(s)} = \alpha h_{i,k}^{(0)} + (1-\alpha)\sum_{j\in N_{i,k}} \frac{\beta_{k,i,j}^{(s-1)}}{\sqrt{|N_{i,k}||N_{j,k}|}} h_{j,k}^{(s-1)} \tag{11}$$

for relation k in the sth iteration, $h_{i,k}^{(s)} (1 \le s \le S)$ is the representation of node i. Between nodes i and j for relation k, $\beta_{k,i,j}^{(s-1)}$ is the frequency factor of the s-order message. $N_{i,k}$ is the neighborhood of node i in relation graph G_k. $\beta_{k,i,j}^{(s-1)}$ is defined as

$$\begin{gathered} \beta_{k,i,j}^{(0)} = G_{k,i,j} \quad for\, s = 1 \\ \beta_{k,i,j}^{(s-1)} = tanh\left(a_{s,k}^{T}\left[h_{i,k}^{(s-1)} \| h_{j,k}^{(s-1)}\right]\right) \quad for\, 2 \le s \le S \end{gathered} \tag{12}$$

for relation k in the sth iteration, $a_{\mathrm{s,k}} \in R^{1\times 2F}$ is the attention coefficient. Finally, output features are acquired through

$$z_i = \|_{k=1}^{K} \left(\widetilde{W}_k h_{i,k}^{(S)}\right) \tag{13}$$

where $\widetilde{W}_k \in R^{F\times F}$ is the weight matrix regarding relation k.

In the testing phase, we use the Dataset Debiasing Module for bias removal. We assume that $f(x)$ and $c(x)$ respectively represent traditional factual prediction and our counterfactual prediction. Mitigating the label bias and keyword bias inherent in the training data is our final objective, by leveraging the direct effect from X to Y for debiased prediction. Process can be formalized as

$$c(x) = f(x) - \hat{\lambda} f(\hat{x}) - \tilde{\lambda} f(\tilde{x}) \tag{14}$$

where $f(\hat{x})$ and $f(\tilde{x})$ respectively represent the label bias and the keyword bias distilled from the trained base model; $\hat{\lambda}$ and $\tilde{\lambda}$ is factors balancing two biases. To search two adaptive parameters on the validation set, we introduce the elastic scaling mechanism, which use scaling factors $\hat{\lambda}^*$ and $\tilde{\lambda}^*$. Elastic scaling can be executed through grid beam search[3] within a scoped two-dimensional space:

$$\hat{\lambda}^*, \tilde{\lambda}^* = arg \max_{\hat{\lambda},\tilde{\lambda}} \psi\left(D_{dev}, c\left(x; \hat{\lambda}, \tilde{\lambda}\right)\right) \quad \hat{\lambda}, \tilde{\lambda} \in [a, b] \tag{15}$$

where ψ is a metric function (such as recall) on the validation set $D_{dev} = (X_{dev}, Y_{dev})$.

2.5. Loss Function

The total loss function is denoted as $L = L_t + \gamma * L_d$. L_d is the loss of graph discriminator, as discussed in (3). The weight balancing these two losses is γ. The loss L_t is defined as:

$$L_t = \frac{1}{|V_L|} \sum_{i\in V_L} \sum_{c=1}^{C} y_{i,c} log \hat{y}_{i,c} \tag{16}$$

where $y_{i,c}$ equals 1 if the label is c, and 0 otherwise, for node i. The number of classes is C and the set of nodes having labels is V_L. $\hat{y}_i = soft\,max\left(fc(z_i)\right) \in R^C$ where $fc(\cdot)$ is a MLP and the number of layers is one.

3. Experiments

3.1. Experimental Setups

3.1.1. Datasets

To understand the effectiveness of our proposed DFA-GNN, we evaluate it on 4 datasets. We use the Yelp review dataset, Amazon review dataset[6], T-Finance and T-Social[7] to study GNN-based fraud detection problem. The statistical information of the dataset is shown in Table 1. On all datasets, we use AUC and F1-macro to evaluate the overall performance of all classifiers.

Because Yelp and Amazon datasets have three ready-made relation graphs respectively, DFA-GNN does not use relation learning module. Instead, it can directly use frequency adjusting module and dataset debiasing module for the three relation graphs. Our model and baselines are run on multi-relation graphs. In this process, they deal with information from different relations in their way.

Table 1: Dataset statistics.

	Statistics				
Dataset	# Nodes	# Edges	# Features	# Anomaly	# Train
Amazon	11,944	4,398,392	25	6.87%	5%
Yelp	45,954	3,846,979	32	14.53%	5%
T-Finance	39,357	21,222,543	10	4.58%	5%
T-Social	5,781,065	73,105,508	10	3.01%	5%

3.1.2. Baselines

We compare DFA-GNN with Player2Vec[10], GraphConsis[4], CARE-GNN[6], FRAUDRE[15] and BWGNN[7], which are five state-of-the-art GNN-based fraud detectors, to verify the ability of DFA-GNN in identifying camouflaged fraudsters. In particular, our comparison also include three classical methods, GCN[16], GAT[17] and GraphSAGE[11].

3.1.3. Experimental Setting

The following hyperparameters are applied across all datasets: weight decay $decay = 5\text{e} - 4$ and Adam optimizer with learning rate $lr = 0.01$. In addition, since the percentage of fraudsters is small in datasets, for training DFA-GNN and other baselines, we adopt under-sampling and mini-batch training. Through an open-source toolbox for fraud detection, we implemented Player2Vec, GraphConsis, CARE-GNN, FRAUDRE and BWGNN. For our approach and other baselines, we conduct experiments based on the DGL library and PyTorch 1.6.0.

3.2. Evaluation and Ablation Study

We evaluated the ability of DFA-GNN to detect outliers using four real-world datasets. In Table 2, we can see the comprehensive performance indicators of the model. The bold values in bold indicate that the evaluation metric of the model is better than other benchmark models.

Table 2: Experimental results of all compared models with 5% training ratios.

Dataset	Yelp		Amazon		T-Finance		T-Social	
Metric	AUC	F1-macro	AUC	F1-macro	AUC	F1-macro	AUC	F1-macro
GCN	55.37	53.28	75.13	63.02	59.23	58.33	63.87	52.74
GAT	56.25	53.54	75.08	62.87	60.03	53.97	63.03	52.51
GraphSAGE	56.16	53.49	73.72	62.58	66.39	55.64	65.24	55.24
Player2Vec	53.87	52.47	80.54	69.26	81.27	67.08	66.04	55.28
GraphConsis	65.72	56.61	83.62	72.48	89.64	71.82	68.71	55.43
CARE-GNN	75.65	62.53	89.67	78.68	90.28	73.55	72.36	58.35
FRAUDRE	76.42	62.08	90.84	80.87	90.55	75.86	71.05	56.22
BWGNN	80.47	**65.36**	91.35	82.70	90.81	77.36	**75.63**	**61.28**
DFA-GNN	**80.55**	63.50	**91.52**	**83.08**	**91.03**	**77.61**	73.85	60.57
w/o debiasing	79.83	62.73	90.67	80.73	90.57	75.71	72.80	59.73

Our model has achieved excellent results on most datasets. Among them, DFA-GNN performs excellently on Amazon and T-Finance datasets, with higher AUC and F1-macro values than all other models. It is evident that the inclusion of relation learning and counterfactual inference mechanisms significantly enhances model performance, which verifies our previous theoretical analysis.

3.3. Hyper-parameter Sensitivity

We conducted a comparative analysis between the performance of DFA-GNN and GCN across various iteration steps S. As shown in Figure 2, GCN exhibits optimal performance at a depth of two layers, with its effectiveness diminishing sharply with additional layers. This trend underscores the susceptibility of GCN to the over-smoothing dilemma. In contrast, DFA-GNN demonstrates consistent and superior performance across graphs of varying heterophily ratios, evidencing its robustness and superior ability to counteract over-smoothing.

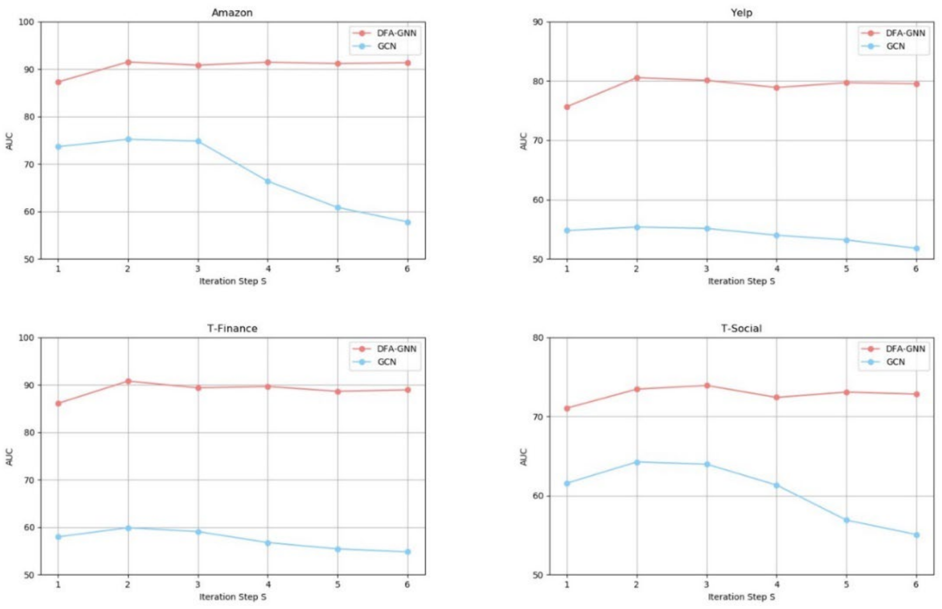

Figure 2: Sensitivity of DFA-GNN and GCN to iteration step S (evaluation protocol is AUC).

As shown in Figure 3, we conducted hyperparameter sensitivity analysis experiments using the T-Finance and T-Social datasets to estimate the influence of K on model expression. The results suggest that as K increases, the performance of DFA-GNN also enhances; However, excessively high K values can lead to meaningless redundant calculations in downstream tasks, which in turn can weaken performance.

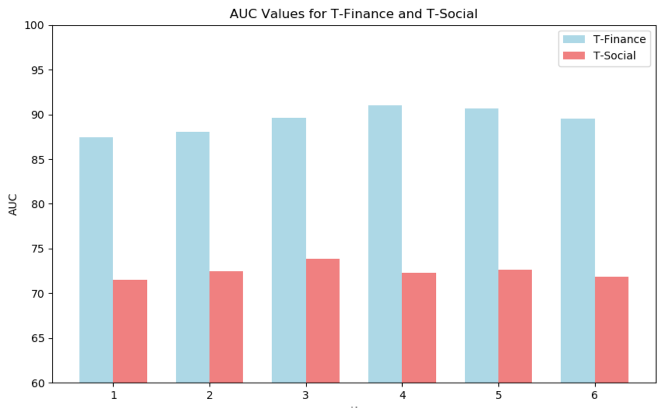

Figure 3: Sensitivity of DFA-GNN to relation numbers K (evaluation protocol is AUC).

4. Conclusions

In this article, our proposed DFA-GNN can handle the problem of graph heterogeneity in an end-to-end manner in fraud detection datasets. Our proposed DFA-GNN applied the idea of counterfactual reasoning to the testing phase, eliminating two types of bias in the dataset and improving the model's ability to identify disguised fraudsters. Although some progress has been made, in future work, there are still some limitations and challenging issues that need to be addressed: If the dataset provides little information, what methods can we use to improve the model's ability to estimate the number of relation K? Thereby further improving the detection ability of the model.

Acknowledgement

This work is supported by Postgraduate Research & Practice Innovation Program of Jiangsu Province under Grant KYCX22_1720.

References

[1] K. Tang, Y. Niu, J. Huang, J. Shi, and H. Zhang. Unbiased Scene Graph Generation from Biased Training. In the Conference on Computer Vision and Pattern Recognition (CVPR), 2020.pages 3716–3725.
[2] M. Jiang, P. Cui, and C. Faloutsos. Suspicious behavior detection: Current trends and future directions. IEEE Intelligent Systems (2016).
[3] C. Hokamp and Q. Liu. Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search. In the Annual Meeting of the Association for Computational Linguistics (ACL), 2017. pages 1535–1546.
[4] Z. Liu, Y. Dou, P. S. Yu, Y. Deng, and H. Peng. Alleviating the Inconsistency Problem of Applying Graph Neural Network to Fraud Detection. SIGIR. 2020.
[5] J. Zhao, T. Wang, M. Yatskar, V. Ordonez, and K. Chang. Men Also Like Shopping: Reducing Gender Bias Amplification using Corpus-level Constraints. In the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2017. pages 2979–2989.
[6] Dou, Y., Liu, Z., Sun, L., Deng, Y., Peng, H., and Yu, P. S. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In CIKM, pp. 315–324, 2020.
[7] J. Tang, J. Li, Z. Gao, J. Li. Rethinking Graph Neural Networks for Anomaly Detection. Proceedings of the 39th International Conference on Machine Learning, PMLR 162:21076-21089, 2022.
[8] L. Wu, H. Lin, B. Hu, C. Tan, Z. Gao, Z. Liu, and S. Z. Li, "Beyond homophily and homogeneity assumption: Relation-based frequency adaptive graph neural networks," IEEE Transactions on Neural Networks and Learning Systems, 2023, doi: 10.1109/TNNLS.2022.3230417.
[9] Qian, C., Feng, F., Wen, L., Ma, C., & Xie, P. Counterfactual inference for text classification debiasing. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 5434–5445, 2021.
[10] Y. Zhang, Y. Fan, Y. Ye, L. Zhao, and C. Shi. Key Player Identification in Underground Forums over Attributed Heterogeneous Information Network Embedding Framework. In CIKM. 2019.
[11] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In NeurIPS. 2017.
[12] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in Proc. NIPS Workshop Bayesian Deep Learn., 2016, pp. 1–3.
[13] J. Ma, P. Cui, K. Kuang, X. Wang, and W. Zhu, "Disentangled graph convolutional networks," in Proc. Int. Conf. Mach. Learn., vol. 97, 2019, pp. 4212–4221.
[14] F. R. Chung and F. C. Graham, Spectral Graph Theory. Providence, RI, USA: Amer. Math. Soc., 1997.
[15] G. Zhang, J. Wu, J. Yang, A. Beheshti, S. Xue, C. Zhou, and Q. Z. Sheng, "Fraudre: Fraud detection dual-resistant to graph inconsistency and imbalance," in Proc. IEEE Int. Conf. Data Mining, 2021, pp. 66–77.
[16] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proc. Int. Conf. Learn. Represent. (ICLR), 2017, pp. 1–14.
[17] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in Proc. Int. Conf. Learn. Represent. (ICLR), 2018, pp. 1–12.