You Can't Ignore Either: Unifying Structure and Feature Denoising for Robust Graph Learning

Tianmeng Yang*
Peking University
Beijing, China
youngtimmy@pku.edu.cn

Jiahao Meng*
Peking University
Beijing, China
mengjiahao@stu.pku.edu.cn

Min Zhou
Huawei Cloud
Shenzhen, China
zhoumin27@huawei.com

Yaming Yang
Peking University
Beijing, China
yamingyang@stu.pku.edu.cn

Yujing Wang
Xiangtai Li
Peking University
Beijing, China
{yujwang,lxtpku}@pku.edu.cn

Yunhai Tong
Peking University
Beijing, China
yhtong@pku.edu.cn

Abstract

Recent research on the robustness of Graph Neural Networks (GNNs) under noises or attacks has attracted great attention due to its importance in real-world applications. Most previous methods explore a single noise source, recovering corrupt node embedding by reliable structures bias or developing structure learning with reliable node features. However, the noises and attacks may come from both structures and features in graphs, making the graph denoising a dilemma and challenging problem. In this paper, we develop a unified graph denoising (UGD) framework to unravel the deadlock between structure and feature denoising. Specifically, a high-order neighborhood proximity evaluation method is proposed to recognize noisy edges, considering features may be perturbed simultaneously. Moreover, we propose to refine noisy features with reconstruction based on a graph auto-encoder. An iterative updating algorithm is further designed to optimize the framework and acquire a clean graph, thus enabling robust graph learning for downstream tasks. Our UGD framework is self-supervised and can be easily implemented as a plug-and-play module. We carry out extensive experiments, which proves the effectiveness and advantages of our method. Code is avalaible at https://github.com/YoungTimmy/UGD.

CCS Concepts

• **Computing methodologies** → **Neural networks**.

Keywords

Graph Neural Networks, Graph Denoising, Robust Graph Learning.

ACM Reference Format:

Tianmeng Yang, Jiahao Meng, Min Zhou, Yaming Yang, Yujing Wang, Xiangtai Li, and Yunhai Tong. 2024. You Can't Ignore Either: Unifying Structure and Feature Denoising for Robust Graph Learning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3627673.3680007

*Both authors contributed equally to this research.

CIKM '24, October 21–25, 2024, Boise, ID, USA.

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3680007

1 Introduction

By extending deep learning on graphs, graph neural networks have achieved great success and been applied to many important scenarios, such as social networks [11, 16], recommendation [18, 21] and financial transaction [3, 20]. Most GNNs [6, 10, 17] follow a message passing framework [5] to learn node representations for downstream tasks, relying on a favorable neighborhood aggregation including both structure connections and node features.

Unfortunately, real-world graphs are often noisy or incomplete due to error-prone data measurement or collection. For example, in a user-item graph, users may have imperfect profiles (feature missing) or misclick some unwanted items (error link) [14]. Malicious accounts may camouflage themselves, including adjusting their behaviors (feature camouflage) and connecting with several benign entities (structure camouflage) [3]. What is worse, recent studies show that GNNs are vulnerable to adversarial attacks [2, 7]. Disturbing graph structure or node features would greatly degrade the performance of GNNs and may lead to severe consequences for critical applications requiring high safety and privacy [24].

To ensure robust graph learning, many previous efforts are devoted to refining corrupted node features or identifying problematic edges. According to the homophily assumption, AirGNN [12] designs an adaptive message passing scheme with residual connection to boost the resilience to abnormal features. MAGNET [23] automatically detects locally corrupted feature attributes and reconstructs robust estimations with sparsity promoting regularizer. Meanwhile, structure learning methods, such as ProGNN [8] and GSR [22], aim to learn an optimal graph structure from noisy input and thus alleviate the structure noise or attacks.

However, most of these approaches assume a single noise source of edge perturbation or feature poisoning, purify the graph structures based on feature similarity, or refine node features based on reliable structures. They would face heavy performance degradation when feature noise and structure perturbation exist *simultaneously*. In addition, we find that simply combining the two processes into

a pipeline cannot achieve a satisfactory performance and sometimes even gets worse (detail at Sec 3.2). Therefore, reconciling this dilemma remains an unexplored challenge.

To unravel the deadlock, we present a **U**nified **G**raph **D**enoising framework (termed as UGD) that considers both structure and feature noises. Specifically, we propose to conduct structure denoising by developing high-order neighborhood proximity instead of computing the directed similarity of pair nodes, which may be easily disturbed by feature noise. For feature denoising, we leverage a graph auto-encoder to reconstruct node features guided by local smoothness. Furthermore, an iterative updating algorithm is adopted to infer the preserved edges and jointly optimize the node features step by step.

We conduct extensive experiments on various datasets to evaluate the proposed UGD framework. The results validate its superior performance with noise and poisoning attacks over state-of-the-art methods. In particular, numerical comparison on downstream node classification tasks shows that UGD improves accuracy by 4.0% on the Citeseer dataset and 6.7% on the AComp dataset over the best previous methods, respectively. Moreover, detailed analyses under different ratios of perturbations and visualizations are also provided to show the strength and advantages of our UGD.

2 The Proposed UGD Framework

2.1 Problem Formulation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an graph with node set $\mathcal{V}$ and edge set $\mathcal{E}$. The feature matrix of the graph is $X \in \mathbb{R}^{n \times d}$, where $n = |\mathcal{V}|$ denotes the number of nodes and d denotes the number of input features. $A \in \{0,1\}^{n \times n}$ is the adjacency matrix, where $A_{uv} = 1$ means the node u and node v are connected. $N(u)$ is the direct neighbors of node u. A graph data can also be denoted as $\mathcal{G} = \{\mathcal{E}, X\}$.

Given a noisy or poisoned graph $\mathcal{G} = \{\mathcal{E}, X\}$, the unified denoising problem in this paper is to produce an optimized graph $\mathcal{G}^* = \{\mathcal{E}^*, X^*\}$ considering both structure and feature, and $\mathcal{G}^*$ will be used for the following downstream graph learning tasks. The overall framework of UGD is illustrated in Figure 1.

2.2 High-order Neighborhood Proximity

Most existing works in graph structure learning focus on direct relations between interconnected nodes, with metrics like similarity measures or attention mechanisms to remove anomalous edges. However, in the presence of nodes with high feature noise, these pairwise metrics may become less reliable, potentially leading to the erroneous removal of normal edges. An intuitive way is to make a node observe more neighborhood information and keep up with more friends. From the perspective of a central node, its identity can be better represented by the neighbors and thus be resistant to feature disturbance. Thus, we define the neighborhood prototype of node u as an aggregating vector P_u of its neighbors with a permutation-invariant readout function. For simplicity, we apply the mean pooling as $P_u = \sum_{v \in N(u)} \frac{x_v}{|N(u)|}$.

Subsequently, for each edge (u, v) in the noisy graph $\mathcal{G}$, we introduce a function $sim(\cdot, \cdot)$ to measure high-order neighborhood proximity $D_{(u,v)}$ between node v and prototype P_u, and distinguish whether the interaction of edge (u, v) is positive or negative. In this

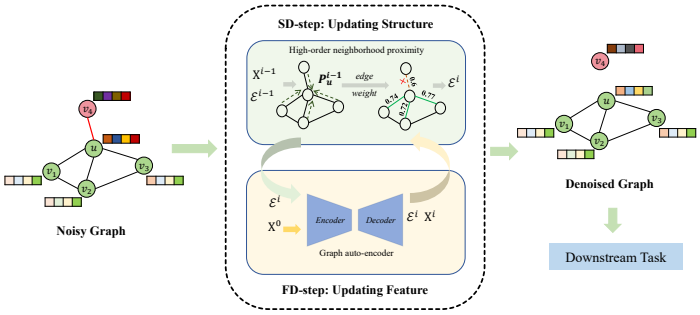


Figure 1: The overall framework of UGD. Node u is injected with feature noise and the edge (u, v_4) is an adversarial link.

work, the cosine similarity score $D_{(u,v)} = \frac{P_u \cdot x_v}{\|P_u\|_2 \|x_v\|_2}$ is adopted for its better performance. Considering the asymmetry of $D_{(u,v)}$ and $D_{(v,u)}$ in undirected graphs, the smaller of the two values is retained as weight to filter more noise and formulated as:

$$weight_{(u,v)} = weight_{(v,u)} = min\{D_{(u,v)}, D_{(v,u)}\}, \tag{1}$$

where $weight_{(u,v)}$ goes beyond simple consideration of relations between adjacent nodes, incorporating the features of second-order neighbors for a more robust judgment of noisy edges. A threshold θ is employed to selectively remove edges with lower weights, thereby preserving a credible graph structure.

2.3 Feature Reconstruction

To alleviate the impact of abnormal features and achieve local smoothness, prior approaches [12, 13] based on graph signal denoising often formulate it as a complex convex optimization problem. Nevertheless, graph anomaly detection methods, such as CONAD [19], have observed that feature compression and reconstruction can effectively identify anomalous nodes. Motivated by this, we develop a graph auto-encoder to ensure that the reconstructed features deviate from the original noise while maintaining similarity with its neighbors. In addition, we introduce a residual connection, combining the reconstructed results with the original features in a weighted manner to form the denoised features. With a fundamental reconstruction loss function $\mathcal{L}_{recon}$, the forward propagation process of node feature X can be defined as follows:

$$\hat{X} = \beta \cdot X + (1 - \beta) \cdot Dec(Enc(X|\mathcal{E})|\mathcal{E}), \tag{2}$$

$$\mathcal{L}_{recon} = ||\hat{X} - X||_2 = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} ||\hat{x}_v - x_v||_2, \tag{3}$$

where the encoder Enc and decoder Dec are two-layer GCNs [10], $\hat{X}$ is the denoised feature and β is a residual weight parameter.

To better achieve local smoothness in the reconstructed features, we also introduce a neighborhood smoothness loss ($\mathcal{L}_{smooth}$) to enhance the representational capacity of denoised features. Formally, the objective is to force a neighborhood proximity:

$$\mathcal{L}_{smooth} = \frac{1}{2} \sum_{v \in \mathcal{V}} \sum_{u \in N(v)} || \frac{\hat{x_u}}{\sqrt{d_u}} - \frac{\hat{x_v}}{\sqrt{d_v}} ||_2^2 = tr(\hat{X}^T L \hat{X}), \tag{4}$$

where L is the normalized Laplacian matrix derived on the structure of graph $\mathcal{G}$, d_v is the degree of node v. The reconstructed feature $\hat{X}$

can be optimized by minimizing the two losses:

$$\mathcal{L} = \mathcal{L}_{recon}(\hat{X}|X) + \gamma \cdot \mathcal{L}_{smooth}(\hat{X}|\mathcal{E}, X), \quad (5)$$

with γ being a balance weight of the reconstruction and smoothness.

2.4 Optimization with Iterative Updating

Since the graph structure and node features are interrelated, the overall objective function $\mathcal{L}$ is hard to optimize. To address this, we build the optimization of structure denosing (SD) and feature denoising (FD) with an iterative updating (IU) algorithm.

In each iteration i, the SD-step aim to find a credible edge subset while minimizing the ratio of noisy edges. At this point, node features are fixed, and the edges are updated by our defined high-order neighborhood proximity in section 2.2:

$$update\ \ \mathcal{E}^i = \{(u,v)|\ (u,v) \in \mathcal{E}^{i-1} and\ weight^{i-1}_{(u,v)} \geq \theta\}, \quad (6)$$

where $\mathcal{E}^i$ is the updated structure.

Then the target is redirected to update node feature in FD-Step. Model parameters of the graph auto-encoder are optimized with the objective function in section 2.3:

$$minimize\ \mathcal{L} = \mathcal{L}_{recon}(X^i|X^0) + \gamma \cdot \mathcal{L}_{smooth}(X^i|\mathcal{E}^i, X^0). \quad (7)$$

These two steps would alternate until the estimation $\mathcal{E}^i$ achieves convergence that the difference between two iterations is not more than a stop number ϵ.

With the designs above, we build our proposed UGD framework and purify the graph data to support subsequent learning of downstream tasks. UGD is self-supervised and can be easily implemented as a plug-and-play module for various models and tasks.

3 Experiments

3.1 Experimental Setup

Datasets. We evaluate the semi-supervised node classification performance of UGD on five widely used benchmark datasets [10, 15]. The dataset statistics are summarized in Table 1.

Baselines. We compare the proposed UGD with representative GNNs and defense models, including: **(1) Vanilla GCN** [10] that ignores noises. **(2) Feature denoising methods**: MAGNET [23] uses a sparsity promoting regularizer to recover robust estimations of features; AirGNN [12] and ElasticGNN [13] focus on local smoothness by combining l_1 and l_2 loss. **(3) Structure learning methods**: Pro-GNN [8] directly takes the adjacency matrix as parameters for learning; IDGL [1] and GSR [22] utilize metric learning to refine the structure. **(4) Combination** of feature denoising and structure learning methods, including different models and orders.

Settings. For a fair comparison, we closely follow the experiment setting in previous works [8, 23]. We employ *attribute injection* and *metattack* [25] to inject the feature noise and modify the graph structure, respectively. For UGD, the optimal threshold θ for structure denoising varies across different datasets. We employ a smaller threshold during the early iterations to prevent removing excessive edges due to noisy features. Other hyper-parameters are tuned from: (1) β: {0, 0.5}; (2) γ: {1e-5, 5e-4, 3e-4, 1e-3}; (3) learning rate η: {5e-4, 1e-3}; (4) stop number ϵ: {0, 2, 10, 30}; The backbone model for downstream node classification is a two-layer GCN [10] for all methods, with a learning rate of 0.01, weight decay of 1e-3, and trained for 100 epochs. Adam optimizer [9] is used for all experiments. Our methods are implemented using PyTorch and PyTorch Geometric [4]. All experiments are conducted on a machine with an NVIDIA 3090 GPU (24GB memory).

Table 1: Statistics of the benchmark graphs.

Dataset	#Nodes	#Edges	#Features	#Classes	#Train/Val/Test
Cora	2708	5278	1433	7	140/500/1000
Citeseer	3327	4552	3703	6	120/500/1000
Pubmed	19717	44324	500	3	60/500/1000
AComp	13752	245861	767	10	200/300/Rest
Coauthor	18333	81894	6805	15	300/450/Rest

Table 2: Mean accuracy ± stdev over benchmark graphs.

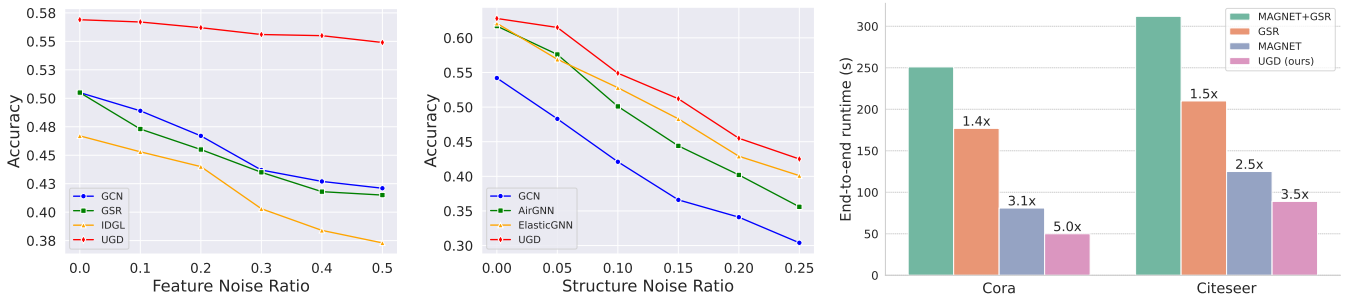
Methods	Cora	Citeseer	Pubmed	AComp	Coauthor
GCN [10]	56.5±2.0	42.1±2.9	65.5±1.1	68.4±2.7	81.4±2.5
MAGNET [23]	69.7±2.4	52.6±1.9	OOM	OOM	OOM
AirGNN [12]	68.0±2.8	50.1±2.2	69.1±2.9	57.1±4.3	80.7±2.8
ElasticGNN [13]	68.3±3.3	52.8±3.0	66.7±1.3	58.0±4.3	75.7±2.9
Pro-GNN [8]	62.9±3.1	44.8±1.0	65.6±2.3	58.8±2.6	80.7±3.1
IDGL [1]	55.2±2.4	37.3±2.4	65.1±0.7	37.3±17.5	OOM
GSR [22]	58.5±3.8	41.5±1.0	64.4±1.0	26.1±8.0	77.3±6.3
MAGNET+GSR	69.4±2.4	51.6±1.5	OOM	OOM	OOM
AirGNN+GSR	57.1±2.8	40.2±2.1	66.3±1.7	57.0±3.5	77.1±4.1
MAGNET+IDGL	67.5±2.8	50.7±1.9	OOM	OOM	OOM
GSR+MAGNET	68.6±2.2	51.8±2.2	OOM	OOM	OOM
GSR+AirGNN	69.0±1.9	50.1±2.6	67.2±2.8	52.2±7.8	78.4±3.8
UGD (ours)	**70.3**±1.2	**54.9**±3.2	**70.0**±1.4	**73.0**±1.3	**83.2**±1.8
w/o HNP	69.7±2.8	54.4±4.4	69.4±1.8	72.8±1.5	83.1±2.6
w/o FR	58.4±2.1	43.0±2.5	66.1±1.5	69.3±2.3	81.6±2.3
w/o IU (F+S)	69.7±2.3	54.8±3.8	69.3±1.2	72.4±1.5	82.9±2.4
w/o IU (S+F)	68.9±2.5	53.7±3.1	69.0±2.1	72.2±2.3	83.0±2.3

3.2 Numerical Comparison

We evaluate the performance of all methods against two types of noise, with an injection of 10% structure noise and 50% feature noise into the data. The ratio settings are referenced from prior works [13, 23], but we are pioneering in simultaneously considering both of them. Tabel 2 summarizes the results of downstream node classification tasks after denoising. We report the mean test accuracy, together with a standard deviation over five different random seeds (OOM means out of memory). We can make the following observations from the results:

(1) The performance of feature denoising methods can be heavily affected by structure noise. Specifically, on the AComp and Coauthor datasets, the elaborate approaches perform even worse than vanilla GCN. Besides, structure learning methods have shown large fluctuations in performance with feature noise. GSR performs much worse than GCN on most datasets (except for Cora). These results demonstrate the necessity of considering both types of noise.

(2) Simply connecting the methods cannot achieve a satisfying result. For example, AirGNN+GSR leads to performance degradation compared with a single AirGNN on all graphs, which confirms our motivation to develop a unified denoising framework.



(a) Fixed structure noise **(b) Fixed feature noise** **(c) Efficiency comparison**

Figure 2: Model analysis.

(3) With the iterative updating algorithm, UGD jointly optimizes structure and feature denoising, and significantly improves the performance of GNNs. Concretely, UGD consistently outperforms the best baselines across five datasets, obtaining improvements of 0.9%, 4.0%, 1.3%, 6.7%, and 2.2%, respectively.

3.3 Ablation Study

To examine the contributions of different components in UGD, we also consider variants of UGD that serve as ablation: (1)*w/o HNP*, which focuses solely on features denoising without refining the graph structure, and (2)*w/o FR*, which applies structure denoising without feature reconstruction. (3)*w/o IU*, which removes the iterative updating algorithm and performs graph denoising in order of feature first (*F+S*) or structure first (*S+F*).

In Table 2, the bottom part summarizes the results, from which we have two observations. First, the performance of all UGD variants drops distinctly when compared to the full model, demonstrating that each designed component contributes to the success of UGD. Second, UGD without high-order proximity still outperforms most feature denoising methods, and UGD without feature reconstruction is still comparable to most structure learning methods, verifying the effectiveness of our designs under mixed noises.

3.4 Model Analysis

We have performed comprehensive analysis to verify the advantages and effectiveness of UGD. Limited by the page, we report main results on Citeseer as following:

3.4.1 Performance on different feature noise ratios. Maintaining a consistent structure noise ratio of 10%, we conduct experiments with varying feature noise ratios of 0% to 50% on Citeseer dataset. The experimental results are presented in Figure 2(a). In this study, we compare our UGD model with GCN and other GSL methods to analyze how feature noise affects the structure denoising capability of the models. It can be observed that when the feature noise ratio increases, the baselines' node classification accuracy drops fast. UGD consistently outperforms baselines with a significant margin. Concretely, the accuracy on Citeseer drops by 18% and 21% for GSR and IDGL, respectively. For UGD, the accuracy decreases from 0.57 to 0.55, resulting in a slight decay of 3.5%, indicating that UGD is more robust against feature noise.

3.4.2 Performance on different structure noise ratios. Maintaining a consistent feature noise ratio of 50%, we also compare UGD with graph denoising methods with different ratios of structure noise varying from 0% to 25% on Citeseer dataset. As shown in Fig. 2(b), structure noise can severely affect the graph denoising methods. Nevertheless, our UGD's performance is superior to baselines in most cases, showing better robustness against structure noise.

3.4.3 Efficiency. Another advantage of UGD is its high efficiency in graph denoising, which could be essential for large graphs. We report the end-to-end runtime of denoising methods with the same epochs in Fig.2(c). Compared to MAGNET+GSR, UGD can achieve a speedup of 5.0x and 3.5x on Cora and Citeseer, respectively. In fact, UGD performs faster than single MAGNET and GSR, showing potential in applications with high-efficiency requirements. Moreover, UGD is model-free and can be readily combined with scalable GNNs to handle large graphs.

4 Conclusion

In this paper, we focus on robust graph learning with challenging structure and feature perturbations and introduce a UGD framework to handle structure and feature denoising for downstream tasks simultaneously. In UGD, high-order neighborhood proximity and feature reconstruction are jointly optimized with an iterative updating algorithm. Experiments on various datasets demonstrate the significant effectiveness of UGD. Extensive analyses further verify UGD's advantages against different noise and high efficiency.

Acknowledgments

This work was supported by National Key R&D program of China under Grant No.2023YFC3807603.

References

[1] Yu Chen, Lingfei Wu, and Mohammed Zaki. 2020. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in neural information processing systems* 33 (2020), 19314–19326.
[2] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. In *International conference on machine learning*. PMLR, 1115–1124.
[3] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. 2020. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM international conference on information & knowledge management*. 315–324.

[4] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[5] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 1263–1272.

[6] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[7] Wei Jin, Yaxin Li, Han Xu, Yiqi Wang, and Jiliang Tang. 2020. Adversarial attacks and defenses on graphs: A review and empirical study. *arXiv preprint arXiv:2003.00653* 10, 3447556.3447566 (2020).

[8] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 66–74.

[9] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

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

[11] Sanjay Kumar, Abhishek Mallik, Anavi Khetarpal, and Bhawani Sankar Panda. 2022. Influence maximization in social networks using graph embedding and graph neural network. *Information Sciences* 607 (2022), 1617–1636.

[12] Xiaorui Liu, Jiayuan Ding, Wei Jin, Han Xu, Yao Ma, Zitao Liu, and Jiliang Tang. 2021. Graph neural networks with adaptive residual. *Advances in Neural Information Processing Systems* 34 (2021), 9720–9733.

[13] Xiaorui Liu, Wei Jin, Yao Ma, Yaxin Li, Hua Liu, Yiqi Wang, Ming Yan, and Jiliang Tang. 2021. Elastic graph neural networks. In *International Conference on Machine Learning*. PMLR, 6837–6849.

[14] Michael P O'Mahony, Neil J Hurley, and Guénolé CM Silvestre. 2006. Detecting noise in recommender system databases. In *Proceedings of the 11th international conference on Intelligent user interfaces*. 109–115.

[15] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[16] Xiran Song, Jianxun Lian, Hong Huang, Zihan Luo, Wei Zhou, Xue Lin, Mingqi Wu, Chaozhuo Li, Xing Xie, and Hai Jin. 2023. xgcn: An extreme graph convolutional network for large-scale social link prediction. In *Proceedings of the ACM Web Conference 2023*. 349–359.

[17] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *International Conference on Learning Representations (ICLR)* (2017).

[18] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph neural networks in recommender systems: a survey. *Comput. Surveys* 55, 5 (2022), 1–37.

[19] Zhiming Xu, Xiao Huang, Yue Zhao, Yushun Dong, and Jundong Li. 2022. Contrastive attributed network anomaly detection with data augmentation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 444–457.

[20] Tianmeng Yang, Min Zhou, Yujing Wang, Zhengjie Lin, Lujia Pan, Bin Cui, and Yunhai Tong. 2023. Mitigating Semantic Confusion from Hostile Neighborhood for Graph Active Learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 4380–4384.

[21] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 974–983.

[22] Jianan Zhao, Qianlong Wen, Mingxuan Ju, Chuxu Zhang, and Yanfang Ye. 2023. Self-supervised graph structure refinement for graph neural networks. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 159–167.

[23] Bingxin Zhou, Yuanhong Jiang, Yuguang Wang, Jingwei Liang, Junbin Gao, Shirui Pan, and Xiaoqun Zhang. 2023. Robust graph representation learning for local corruption recovery. In *Proceedings of the ACM Web Conference 2023*. 438–448.

[24] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2847–2856.

[25] Daniel Zügner and Stephan Günnemann. 2019. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Bylnx209YX