Unsupervised Deep Graph Structure and Embedding Learning

Xiaobo Shen[1] , **Lei Shi**[1] , **Xiuwen Gong**[2*] and **Shirui Pan**[3]
[1]Nanjing University of Science and Technology
[2]University of Technology Sydney
[3]Griffith University
njust.shenxiaobo@gmail.com, 918106840541@njust.edu.cn, gongxiuwen@gmail.com, s.pan@griffith.edu.au

Abstract

Graph Neural Network (GNN) is powerful in graph embedding learning, but its performance has been shown to be heavily degraded under adversarial attacks. Deep graph structure learning (GSL) is proposed to defend attack by jointly learning graph structure and graph embedding, typically in node classification task. Label supervision is expensive in real-world applications, and thus unsupervised GSL is more challenging and still remains less studied. To fulfill this gap, this paper proposes a new unsupervised GSL method, i.e., unsupervised property GNN (UPGNN). UPGNN first refines graph structure by exploring properties of low rank, sparsity, feature smoothness. UPGNN employs graph mutual information loss to learn graph embedding by maximizing its correlation with refined graph. The proposed UPGNN learns graph structure and embedding without label supervision, and thus can be applied various downstream tasks. We further propose Accelerated UPGNN (AUPGNN) to reduce computational complexity, providing a efficient alternative to UPGNN. Our extensive experiments on node classification and clustering demonstrate the effectiveness of the proposed method over the state-of-the-arts especially under heavy perturbation.

1 Introduction

Graph is pervasive data structure, and its related applications have emerged in a wide range of domains, e.g., social media, finance, bioinformatics. Graph Neural Networks (GNNs) [Kipf and Welling, 2017] as powerful deep learning technique modeling graph have been successfully applied in various graph analytical tasks, e.g., node classification [Gao *et al.*, 2020], node clustering [Zhang *et al.*, 2019a], link prediction [Peng *et al.*, 2020]. GNNs aim to learn embedding of one node by recursively aggregating information from its neighbors, and their key success lies in message-passing scheme.

GNNs assume that graph is perfect, however this assumption is often violated, as graph may be attacked in real-world scenarios. Recent studies [Dai *et al.*, 2018; Wu *et al.*, 2019] have shown that performances of GNNs are heavily degraded under adversarial attacks that perform slight perturbations on graph. The lack of robustness of GNNs may lead to serious consequence in some critical applications, e.g., financial transaction. Therefore, it is of great significance to improve robustness of GNNs.

Recently deep graph structure learning (GSL) [Yang *et al.*, 2019; Zhu *et al.*, 2019] has been investigated to improve robustness of GNNs and boost performance of downstream tasks. It often first models graph adjacency matrix with learned parameters, and jointly learns adjacency matrix and GNNs under supervision of downstream node classification task [Jin *et al.*, 2020; Wan and Kokel, 2021]. Existing GSL methods parameterize adjacency matrix with different learning schemes, e.g., probabilistic model [Zhang *et al.*, 2019b; Veličković *et al.*, 2017], full parameterization [Jin *et al.*, 2020; Wan and Kokel, 2021], metric learning [Zhang and Zitnik, 2020]. Most existing GSL methods are supervised where label is used to guide the training [Jin *et al.*, 2020]. However, label is usually expensive to obtain, and enables existing GSL methods to be only applied to a few supervised scenarios. The unsupervised deep GSL that can be applied to various downstream tasks is more challenging but rarely studied.

To address the above issue, this work studies unsupervised deep GSL, and proposes unsupervised property GNN, i.e., UPGNN. Compared to existing deep GSL methods, the advantage of UPGNN lies in joint learning of graph structure and embedding without label supervision, and thus it can be applied to various downstream tasks. As shown in Figure 1, UPGNN includes two modules, where the first module refines graph structure by exploring some essential properties of clean graph, and the second module employs an unsupervised loss to guide training. UPGNN performs SVD on in each iteration, which requires time complexity of $\mathcal{O}(N^3)$ (N is number of nodes) to obtain low-rank graph, and is computationally expensive for large graph. To address efficiency issue of UPGNN, we further accelerate UPGNN by reducing the dimension of the core low-rank component to avoid performing SVD on large original graph. The main contributions of this work are as follows:

- We propose UPGNN, an unsupervised property GNN, that can jointly learn graph structure and embedding in an unsupervised manner. As UPGNN is developed in-

*Corresponding author.

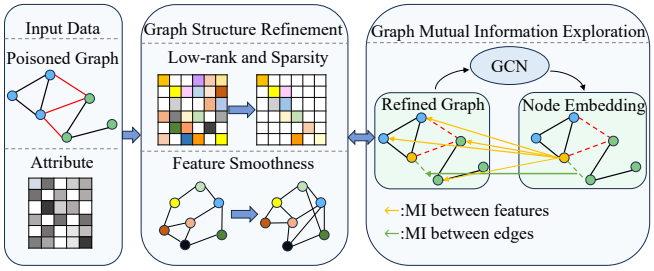


Figure 1: The overall pipeline of the proposed UPGNN. UPGNN jointly learns graph structure and embedding in an unsupervised manner. UPGNN generates a clean graph by exploring low rank, sparsity, and feature smoothness of graph. UPGNN employs graph mutual information loss on graph to maximize correlation between refined graph and embedding.

dependent of specific task, it can be freely applied to various downstream tasks.

- UPGNN refines graph structure by exploring low rank, sparse, and feature smooth properties of graph. Additionally, an unsupervised loss based on graph mutual information is employed to maximize the correlation between the refined graph and the learned graph embedding.
- We further propose Accelerated UPGNN (AUPGNN) to reduce computational complexity, providing a efficient alternative to UPGNN.
- The proposed methods are verified on node classification and clustering, and empirical results demonstrate that the proposed methods outperform the state-of-the-arts under adversarial attacks.

2 Related Work

2.1 Graph Neural Networks

Graph neural networks (GNNs) are powerful models for learning graph representations by capturing complex dependencies within graphs. GCN [Kipf and Welling, 2017] is a popular GNN model that utilizes spectral graph convolution to gather information from neighboring nodes. This mechanism enables nodes to exchange and update their representations based on the information in their local neighborhoods. Another notable GNN is the Graph Attention Network (GAT) [Veličković *et al.*, 2017], which incorporates attention mechanisms to focus on relevant neighborhood information.

One challenge faced by GNNs is their reliance on labeled data for training, which can be costly and difficult to obtain in real-world applications. To address this challenge, researchers have explored unsupervised GNN models such as the Graph Autoencoder (GAE) [Kipf and Welling, 2016]. GAE learns to encode and decode graph structures, effectively capturing efficient representations without explicit supervision. In addition, graph comparison learning models based on maximizing mutual information (MI) have achieved promising results in capturing graph similarity without relying on labeled data. [Hjelm *et al.*, 2018; Belghazi *et al.*, 2018; Peng *et al.*, 2022] For instance, GCA enhances the discriminative power of the learned representations for downstream tasks by introducing a contrastive framework and employing adaptive data augmentation techniques[Zhu *et al.*, 2021].

In summary, GNNs excel at capturing local and global structures of graphs, allowing them to learn expressive representations. Unsupervised methods like GAE and graph contrastive learning provide another possibility to learn from graph data without relying on labelled examples.

2.2 Structure Learning for GNNs

Existing studies show that GNNs are vulnerable to adversarial attacks, that is, slight or unnoticeable perturbations to the input can fool GNNs to output a wrong prediction. There are some adversarial attacks against graph. The *nettack* [Zügner *et al.*, 2018] is designed for targeted attack, and degrades performance of GNN on target nodes. Based on meta-learning, the *metattack* [Zügner and Günnemann, 2019] is proposed to generate poisoning attacks on the whole graph.

Until now some mechanisms have been designed to defend these attacks. One family is to educe the weight of potentially adversarial nodes in information aggregation. AGCN [Li *et al.*, 2018] first computes the generalized Mahalanobis distance between each pair of nodes in the latent space. Thereafter, it models the edge weight using a Gaussian kernel given the distance. SimP-GCN [Jin *et al.*, 2021] constructs a new feature graph according to node features which achieve robustness by joint learning on structure graph and feature graph.

Another family modifies the graph structure by exploring the graph prior, and they tend to use the graph structure directly as a learnable parameter. Attackers tend to connect nodes with different features, suggesting the removal of links between dissimilar nodes. Additionally, *nettack* leads to changes in the higher-order spectrum of the graph, proposing a preprocessing step with its lower-order approximation. GCN-SVD [Entezari *et al.*, 2020] is an approach for defending against adversarial attacks in graphs. It utilizes singular value decomposition to enhance the robustness of GCN. GLNN [Gao *et al.*, 2020] combines the initial graph structure, sparsity, and feature smoothing into a hybrid objective to defend against contamination. Additionally, ProGNN [Jin *et al.*, 2020] incorporates kernel norms with a low-rank prior to further improve the performance of graph neural networks. ProGNN enhances graph analysis and prediction by adaptively learning the importance of nodes and integrating structural information with graph attributes. This comprehensive approach effectively tackles challenges in graph data like attacks and contamination, resulting in improved performance.

3 Methodology

3.1 Problem Definition

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph, where $\mathcal{V}$ and $\mathcal{E}$ represent vertex and edge sets respectively. We denote node feature and initial graph structure as feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$ and adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ respectively. In this work, it is assumed that adversarial attacks are only performed on graph structure, thus edges are modified and node features remain unchanged.

Given $\mathcal{G}$ with $\mathbf{A}$ being poisoned by adversarial edges and feature matrix $\mathbf{X}$ unperturbed, the goal of this work is to jointly learn clean graph with adjacency matrix $\mathbf{S} \in \mathcal{S} = [0,1]^{N\times N}$, graph embedding $\mathbf{H}$, and GNN parameters Θ without any semantic supervision. Some downstream tasks, e.g., node classification, node clustering can be performed with the learned graph embedding.

3.2 UPGNN

As illustrated in Figure 1, the proposed UPGNN includes two modules: graph structure refinement, and graph mutual information exploration.

Graph Structure Refinement

Adversarial attacks can lead to obvious reduced performance of GNNs by generating carefully-crafted perturbation on graph [Zügner *et al.*, 2018; Zügner and Günnemann, 2019]. Thus it is highly desirable to defend attack by cleaning perturbed graph. It is well known that clean graph often share certain properties, e.g., low rank, and sparsity. However, adversarial attacks tend to add edge to link nodes of different communities [Wu *et al.*, 2019; Xu *et al.*, 2020], and significantly increase rank and singular value of adjacency matrix. Motivated by such observation, UPGNN considers properties of clean graph as the guideline to conduct graph structure refinement. Specifically, UPGNN aims to learn a clean graph with a new adjacency matrix $\mathbf{S}$ that should be close to that of poisoned graph, and share the desirable properties of low rank and sparsity. To this end, we have the following objective function:

$$\mathcal{L}_s = \|\mathbf{A}-\mathbf{S}\|_F^2 + \alpha\|\mathbf{S}\|_1 + \beta\|\mathbf{S}\|_* \tag{1}$$

where the first term ensures that learned and original adjacency matrices should be close, $\|\mathbf{S}\|_1$ and $\|\mathbf{S}\|_*$ [Richard *et al.*, 2012] enforces the learned adjacency matrix to be sparse and low-rank respectively, α and β are two regularization parameters. In addition, it is often observed that connected nodes are more likely to share similar node features. In this work, node features are assumed to be unperturbed, which motivates us to employ such feature smoothness property to refine graph. We have the following objective function:

$$\mathcal{L}_f = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbf{S}_{ij}\left(\frac{\mathbf{x}_i}{\sqrt{d_i}} - \frac{\mathbf{x}_j}{\sqrt{d_j}}\right)^2 = \mathrm{Tr}\left(\mathbf{X}^\top\hat{\mathbf{L}}\mathbf{X}\right) \tag{2}$$

where $\hat{\mathbf{L}} = \mathbf{D}^{-1/2}(\mathbf{D}-\mathbf{S})\mathbf{D}^{-1/2}$ is normalized Laplacian matrix, $\mathbf{D}$ is degree matrix of $\mathbf{S}$, and d_i denotes the degree of node i. Minimizing (2) ensures that connected nodes are required to share similar features, otherwise large loss is incurred. By such way, graph structure can be further refined based on smoothness of unperturbed node features.

Graph Mutual Information Exploration

Some existing GSL methods employ classification loss to train GNN, and are applied in a supervised learning scenario [Elinas *et al.*, 2020; Veličković *et al.*, 2019]. However, such methods rely on label supervision that is expensive to be collected in real-world applications. This work focuses on more general unsupervised scenarios where label is unavailable, and can handle more downstream tasks other than node classification. Inspired by wide use of mutual information (MI) [Hjelm *et al.*, 2018; Belghazi *et al.*, 2018; Peng *et al.*, 2022], we employ graphical mutual information to measure correlation between refined graph and embedding, and further minimize such loss to train GNN and learn embedding.

Specifically, given node i, its support graph is defined as $\mathcal{G}_i$ that includes features of neighbors $\mathbf{X}_i$ and refined adjacency matrix of neighbors $\mathbf{S}_i$. The MI between embedding of node i and its support graph is defined as:

$$I(\mathbf{h}_i;\mathcal{G}_i) = \sum_{j=1}^{i_n} w_{ij} I(\mathbf{h}_i;\mathbf{x}_j) + I(w_{ij};\mathbf{S}_{ij}) \tag{3}$$

where $\mathbf{x}_j$ is the j-th neighbor of node i, i_n is the number of all elements in $\mathbf{X}_i$, and the weight $w_{ij} = \sigma\left(\mathbf{h}_i^\top\mathbf{h}_j\right)$ is characterized by similarity between $\mathbf{h}_i$ and $\mathbf{h}_j$ that reveals the contribution of $I(\mathbf{h}_i;\mathbf{x}_j)$, and $\sigma(\cdot)$ is a sigmoid function. The first and second terms measure MIs of both features and edges in refined graph respectively. In this work, Jensen-Shannon MI estimator (JSD) [Nowozin *et al.*, 2016] is used due to its effectiveness and efficiency, and then $I(\mathbf{h}_i;\mathbf{x}_j)$ can be defined as:

$$I(\mathbf{h}_i;\mathbf{x}_j) = -\mathrm{sp}\left(-\mathcal{D}_\Omega(\mathbf{h}_i,\mathbf{x}_j)\right) - \mathbb{E}_{\tilde{\mathbb{P}}}\left[\mathrm{sp}\left(\mathcal{D}_\Omega\left(\mathbf{h}_i,\mathbf{x}_j'\right)\right)\right] \tag{4}$$

where $\mathcal{D}_\Omega$ is a discriminator parameterized with Ω, and simply defined as $\mathcal{D}_\Omega(\mathbf{h}_i,\mathbf{x}_j) = \sigma\left(\mathbf{h}_i^\top\Omega\mathbf{x}_j\right)$ that scores the feature pairs through a simple bilinear function, $\mathrm{sp}(x) = \log(1+e^x)$ is soft-plus function, $\mathbf{x}_j'$ is a sampled negative node with respect to node i. To this end, by summarizing all nodes in graph, we have the following objective function:

$$\mathcal{L}_m = -\sum_{i=1}^{N} I(\mathbf{h}_i;\mathcal{G}_i) \tag{5}$$

Final Objective Function

By combining the above three losses, i.e., (1), (2), (5), we have the final objective function of UPGNN as follows:

$$\begin{aligned}\min_{\mathbf{S},\Theta,\Omega}\ & \mathcal{L} = \mathcal{L}_s + \lambda\mathcal{L}_f + \gamma\mathcal{L}_m \\ \text{s.t. }& \mathbf{S} = \mathbf{S}^\top, \mathbf{S}\in\mathcal{S}\end{aligned} \tag{6}$$

where λ and γ are two regularization parameters, and the constraints enforce the learned adjacency matrix to be symmetric and binary.

3.3 Optimization

It is challenging to jointly optimize $\mathbf{S}$, Θ, and Ω in (6), and an alternating optimization scheme is applied. The training procedure of UPGNN is illustrated in Algorithm 1.

Update Θ, Ω

With fixed $\mathbf{S}$, Θ and Ω are only related to $\mathcal{L}_m$. It is a typical neural network optimization problem, and stochastic gradient descent is used to optimize the two variables.

Algorithm 1 UPGNN

Input: Adjacency matrix $\mathbf{A}$, Attribute matrix $\mathbf{X}$, parameters α, β, γ, λ, τ, learning rate η, η';
Output: adjacency matrix $\mathbf{S}$, network parameters Θ, Ω.

Initialize $\mathbf{S} \leftarrow \mathbf{A}$
Randomly initialize Θ
while *Stopping condition is not met* **do**
  $\mathbf{S} \leftarrow \mathbf{S} - \eta \nabla_{\mathbf{S}} \left(\|\mathbf{A} - \mathbf{S}\|_F^2 + \lambda \mathcal{L}_f + \gamma \mathcal{L}_m \right)$
  $\mathbf{S} \leftarrow \text{prox}_{\eta\beta\|\cdot\|_*}(\mathbf{S})$
  $\mathbf{S} \leftarrow \text{prox}_{\eta\alpha\|\cdot\|_1}(\mathbf{S})$
  $\mathbf{S} \leftarrow P_{\mathcal{S}}(\mathbf{S})$
  for $i = 1$ to τ **do**
    $\Theta \leftarrow \Theta - \eta' \frac{\partial \mathcal{L}_m}{\partial \Theta}$
    $\Omega \leftarrow \Omega - \eta' \frac{\partial \mathcal{L}_m}{\partial \Omega}$
  end for
end while

Update S

By removing irrelevant terms with respect to $\mathbf{S}$, we have:

$$\begin{aligned} \min_{\mathbf{S}} \quad & \mathcal{L}(\mathbf{S}, \mathbf{A}) + \alpha \|\mathbf{S}\|_1 + \beta \|\mathbf{S}\|_* \\ \text{s.t.} \quad & \mathbf{S} = \mathbf{S}^\top, \mathbf{S} \in \mathcal{S} \end{aligned} \tag{7}$$

where $\mathcal{L}(\mathbf{S}, \mathbf{A}) = \|\mathbf{A} - \mathbf{S}\|_F^2 + \lambda \mathcal{L}_f + \gamma \mathcal{L}_m$. It is clear that both ℓ_1 norm and nuclear norm are non-differentiable. The Forward-Backward splitting method is employed that alternates gradient descent and proximal steps as:

$$\mathbf{S}^{(k)} = \text{Prox}_{\eta R} \left(\mathbf{S}^{(k-1)} - \eta \nabla_{\mathbf{S}} \left(\mathcal{L} \left(\mathbf{S}, \mathbf{A} \right) \right) \right) \tag{8}$$

where η is learning rate, and proximal operator with respect to non-differentiable function $R(\mathbf{S})$ is defined as:

$$\text{Prox}_R(\mathbf{Z}) = \arg\min_{\mathbf{S}} \|\mathbf{S} - \mathbf{Z}\|_F^2 + R(\mathbf{S}). \tag{9}$$

In particular, the proximal operator of ℓ_1 norm and nuclear norm are defined as follows [Savalle *et al.*, 2012]:

$$\text{prox}_{\alpha\|.\|_1}(\mathbf{S}) = \text{sgn}(\mathbf{S}) \odot (|\mathbf{Z}| - \alpha)_+ \tag{10}$$

$$\text{prox}_{\beta\|\|_*}(\mathbf{S}) = \mathbf{U} \, \text{diag} \left((\sigma_i - \beta)_+ \right) \mathbf{V}^\top \tag{11}$$

where sgn denotes sign function, and $\mathbf{S} = \mathbf{U} \, \text{diag} \left(\sigma_1, \ldots, \sigma_n \right) \mathbf{V}^\top$ is singular value decomposition (SVD) of $\mathbf{S}$. To this end, in each iteration, $\mathbf{S}$ can be updated as follows [Raguet *et al.*, 2013]:

$$\begin{cases} \mathbf{S} = \mathbf{S} - \eta \nabla_{\mathbf{S}} (\|\mathbf{A} - \mathbf{S}\|_F^2 + \lambda \mathcal{L}_f + \gamma \mathcal{L}_m) \\ \mathbf{S} = \text{prox}_{\eta\beta\|\cdot\|_*} (\mathbf{S}) \\ \mathbf{S} = \text{prox}_{\eta\alpha\|\cdot\|_1} (\mathbf{S}) \end{cases} \tag{12}$$

We let $\mathbf{S} = \frac{\mathbf{S} + \mathbf{S}^\top}{2}$ to satisfy symmetric constraint. We further assign $\mathbf{S}_{ij}$ to 0 if $\mathbf{S}_{ij} < 0$, and assign $\mathbf{S}_{ij}$ to 1 if $\mathbf{S}_{ij} > 1$.

After UPGNN is trained, the learned graph embedding can be used for some downstream tasks. Specifically, in this work, for node classification, learned embedding is fed into logistic regression; for node clustering, learned embedding is fed into k-means.

3.4 Accelerated UPGNN

As shown in (12), in each iteration, proximal operator of nuclear norm in UPGNN requires applying SVD on $\mathbf{S}$, whose computational complexity is $\mathcal{O}(N^3)$. It is computationally expensive on large graph, which limits the capability of UPGNN on dealing with large-scale applications. This section studies accelerating UPGNN, and provides its efficient alternative.

The proposed Accelerated UPGNN (AUPGNN) is inspired by recent advances of robust principal component analysis (RPCA) [Feng and Wang, 2022]. The key idea is to reduce the dimension of the core low-rank component, and to avoid performing SVD on original large matrices. Specifically, instead of directly employing $\mathbf{S}$ to approximate given $\mathbf{A}$, we propose to decompose $\mathbf{A}$ as a low rank component $\mathbf{ZWZ}^\top$ and a sparse component $\mathbf{E}$, where $\mathbf{Z} \in \mathbb{R}^{N \times r}$ is a orthogonal matrix with full-column rank, $\mathbf{W} \in \mathbb{R}^{r \times r}$ is the learned low-rank matrix, where r is much smaller than N, $\mathbf{E} \in \mathbb{R}^{N \times N}$ is the learned sparse matrix that is used to separate perturbation from original graph. To this end, we have a new graph structure refinement with properties of low rank and sparsity:

$$\min_{\mathbf{W}, \mathbf{E}} \frac{1}{2} \left\| \mathbf{A} - \mathbf{ZWZ}^\top - \mathbf{E} \right\|_F^2 + \alpha \|\mathbf{E}\|_1 + \beta \|\mathbf{W}\|_* \tag{13}$$

where $\mathbf{Z}$ is considered as mostly a tall matrix, and minimizing (13) allows us to find a smaller low-rank matrix of dimension $r \times r$, rather than a matrix of original dimension $N \times N$. Similar to optimizing $\mathbf{S}$, optimizing $\mathbf{E}$ and $\mathbf{W}$ also involves proximal operator of ℓ_1 norm and nuclear norm, and is detailed as follows:

$$\begin{cases} \mathbf{W} = \mathbf{W} - \eta \nabla_{\mathbf{W}} \left(\frac{1}{2} \left\| \mathbf{A} - \mathbf{ZWZ}^\top - \mathbf{E} \right\|_F^2 + \lambda \mathcal{L}_f + \gamma \mathcal{L}_m \right) \\ \mathbf{W} = \text{prox}_{\eta\beta\|\cdot\|_*} (\mathbf{W}) \end{cases} \tag{14}$$

and

$$\begin{cases} \mathbf{E} = \mathbf{E} - \eta \nabla_{\mathbf{E}} \left(\frac{1}{2} \left\| \mathbf{A} - \mathbf{ZWZ}^\top - \mathbf{E} \right\|_F^2 \right) \\ \mathbf{E} = \text{prox}_{\eta\alpha\|\cdot\|_1} (\mathbf{E}) \end{cases} \tag{15}$$

As can be observed, (14) performs proximal operator of $\mathbf{W}$, and requires performing SVD on a smaller matrix with dimension of $r \times r$, whose computational complexity is $\mathcal{O}(r^3)$. Therefore, AUPGNN has computational advantage over UPGNN. The training procedure of AUPGNN is similar to that of UPGNN, and involves iterative optimization of $\mathbf{W}$, $\mathbf{E}$, network parameters Θ, Ω.

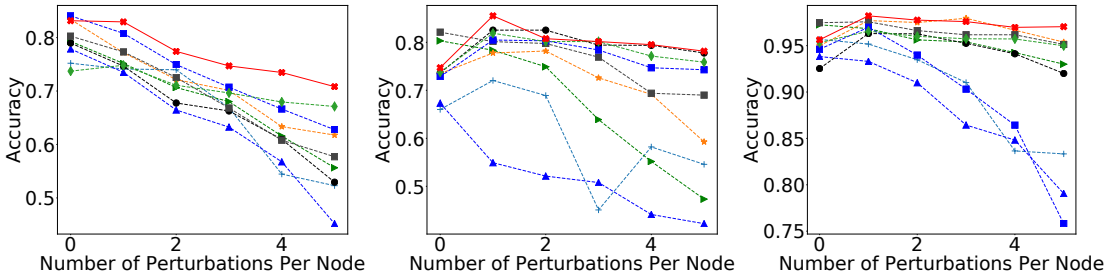
4 Experiments

4.1 Datasets

Three benchmark datasets, i.e., Cora, Citeseer, Polblogs are employed for evaluation, and the largest connected component is considered for each graph. For each graph, we randomly choose 10% of nodes for training, 10% of nodes for validation, and the remaining 80% of nodes for testing. The average performance of 10 runs is reported for each experiment.

Dataset	Ptb Rate (%)	Node2Vec	Deep Walk	GCN	DGI	GMI	GCN-SVD	DGC	SLAPS-2S	UPGNN
Cora	0	75.57±0.78	78.42±0.46	83.50±0.44	78.98±0.11	**83.60±0.12**	80.63±0.45	83.07±0.42	75.49±0.58	83.58±0.09
	5	75.96±0.40	70.68±0.12	76.55±0.79	74.15±0.12	78.34±0.87	78.39±0.54	75.00±1.03	**78.96±0.53**	78.07±0.02
	10	70.58±0.93	68.77±1.10	70.39±1.28	69.69±0.15	72.22±0.68	71.47±0.83	75.54±1.01	73.29±1.31	**76.20±0.06**
	15	65.76±1.74	61.27±1.39	65.10±0.71	57.53±0.09	66.82±0.83	66.69±1.18	73.16±1.37	68.25±1.79	**74.16±0.04**
	20	51.65±1.87	50.54±0.96	59.56±2.72	51.23±0.13	53.73±1.01	58.94±1.13	65.65±1.29	65.37±0.16	**72.57±0.39**
	25	48.08±2.17	46.08±1.05	47.53±1.96	45.60±0.11	49.68±1.21	52.06±1.19	59.65±1.29	63.62±1.06	**71.41±0.13**
Citeseer	0	64.01±0.49	67.32±0.48	71.96±0.55	73.63±0.08	73.83±0.09	70.65±0.32	72.85±1.39	74.27±0.60	**74.38±0.04**
	5	65.59±0.91	67.25±0.56	70.88±0.62	**73.49±0.08**	72.12±0.17	68.84±0.72	72.29±0.94	69.85±1.08	72.25±0.03
	10	67.63±0.80	63.96±0.99	67.55±0.89	69.63±0.09	70.02±0.36	68.87±0.62	71.87±0.58	68.07±0.30	**72.17±0.15**
	15	62.86±1.31	60.34±0.83	64.52±1.11	65.38±0.07	68.51±0.19	63.26±0.96	68.82±0.75	68.08±0.93	**71.22±0.13**
	20	56.49±0.97	54.02±0.88	62.03±3.49	62.75±0.16	59.18±0.44	58.55±1.09	65.77±0.70	65.70±1.12	**69.32±0.15**
	25	54.19±2.15	51.50±0.64	56.94±2.09	62.92±0.09	57.08±0.80	57.18±1.87	62.16±0.28	66.59±0.63	**68.33±0.29**
Polblogs	0	93.69±0.53	93.72±0.25	**95.69±0.38**	91.00±0.13	95.40±0.07	95.31±0.18	94.61±0.44	95.08±0.62	95.62±0.04
	5	77.62±2.62	69.90±0.26	73.07±0.80	79.12±0.24	90.62±0.19	89.09±0.22	83.75±0.14	82.88±0.95	**94.40±0.71**
	10	73.91±0.27	73.39±0.55	70.72±1.13	75.47±0.19	76.56±0.10	81.24±0.49	69.52±0.17	73.87±0.69	**82.98±0.20**
	15	70.31±0.17	66.94±0.52	64.96±1.91	71.44±0.14	71.60±0.14	68.10±3.73	64.75±0.07	67.38±0.84	**78.03±0.65**
	20	59.60±0.96	49.55±0.29	51.27±1.23	51.95±0.21	63.89±0.32	57.33±3.15	64.75±0.23	58.39±1.54	**74.84±0.04**
	25	46.09±1.58	38.56±0.93	49.23±1.36	42.57±0.22	62.03±0.80	48.66±9.93	61.07±0.80	50.75±0.13	**68.22±0.42**

Table 1: Node classification performance (Accuracy±Std) under non-targeted adversarial attack, i.e., *metattack*.



(a) Cora (b) Citeseer (c) Polblogs

Figure 2: Node classification performance under target attack, i.e., *nettack*.

4.2 Experimental Setup

Downstream Tasks

Two tasks, i.e., node classification and clustering are used to evaluate the quality of learned graph and embedding. For evaluation, classification accuracy is employed as metric for classification task, and clustering accuracy, Normalized Mutual Information (NMI), F1-scores are employed as metrics for clustering task.

Baselines

For node classification, three categories of methods are compared, including raph embedding methods, i.e., DeepWalk [Perozzi *et al.*, 2014], Node2Vec [Grover and Leskovec, 2016], graph structure-fixed GNNs, i.e., GCN [Kipf and Welling, 2017], DGI [Veličković *et al.*, 2019], GMI [Peng *et al.*, 2022], and deep GSL methods, i.e., GCN-SVD [Entezari *et al.*, 2020], DGC [Gasteiger *et al.*, 2019], SLAPS-2S [Fatemi *et al.*, 2021]. For node clustering, three categories of methods are compared, including feature-based clustering methods, i.e., k-means, spectral clustering (SC), structure-based clustering methods, i.e., DeepWalk [Perozzi *et al.*, 2014], DNGR [Cao *et al.*, 2016], M-NMF [Wang *et al.*, 2017b], and attributed graph based clustering methods, i.e., TADW [Yang *et al.*, 2015], VGAE [Kipf and Welling, 2016], MGAE [Wang *et al.*, 2017a]. For the proposed method, α, β, γ, and λ are searched in a wide range. For GNNs, we set dimensions of hidden and output layers to 512 and 64 respectively. For the baselines, we use official codes or borrow results reported in their papers.

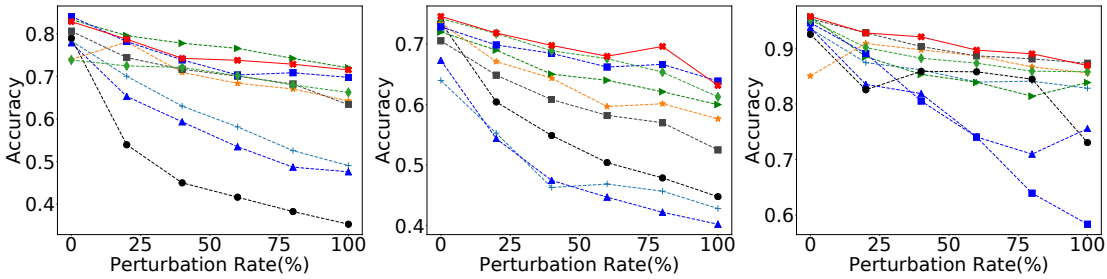
4.3 Node Classification

We evaluate node classification performance against two types of attacks, i.e., non-target attack, target attack. We first use the attack method to poison graph, then train multiple models on the poisoned graph, and finally evaluate node classification performance.

Against Non-targeted Adversarial Attack

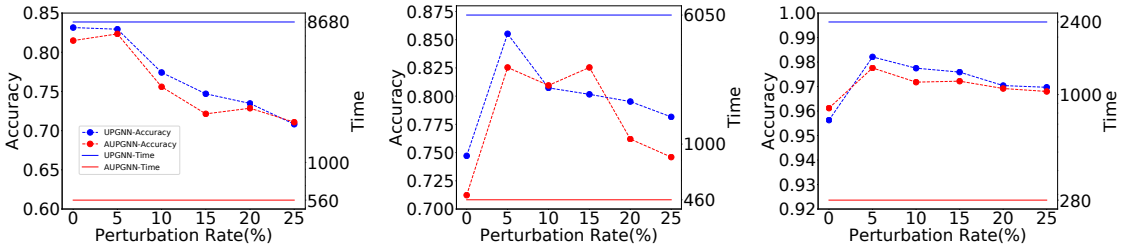
Non-targeted adversarial attack degrades the overall performance on the whole graph, and a representative one, i.e., *metattack* is employed. The perturbation rate, i.e., the ratio of changed edges is varied from 0 to 25% with a step of 5%. Table 1 reports average accuracy with standard deviation of all the methods, and the bold indicates the best. We find the following observations:

- The proposed UPGNN outperforms all the baselines on 14 out of 18 cases. The advantage of the proposed UPGNN is more obvious under large perturbation where it outperforms most baselines by a large margin. For instance, at 25% perturbation rate, the proposed UPGNN improves conventional GCN by 23.88%, 11.39%, 18.99% on Cora, Citeseer, Polblogs respectively.



(a) Cora (b) Citeseer (c) Polblogs

Figure 3: Node classification performance under random attack.



(a) Cora (b) Citeseer (c) Polblogs

Figure 4: UPGNN versus AUPGNN under target attack, i.e., *nettack* on three datasets.

- Deep GSL methods generally outperform graph structure-fixed GNNs, which verifies effectiveness of GSL. Among three GSL methods, DGC generally outperforms SLAPS-2S, and GCN-SVD performs worst. Among three structure-fixed GNNs, GMI outperforms DGI, and GCN underperforms.
- GNNs outperform conventional graph embedding methods, indicating advantage of GNN on graph modeling. The above empirical results verify effectiveness of GSL, and demonstrate that the proposed UPGNN well defends non-targeted adversarial attack.

Against Targeted Adversarial Attack

Targeted adversarial attack generates attacks on specific nodes, and one of state-of-the-arts, i.e., *nettack* [Zügner *et al.*, 2018] is employed. The node classification accuracy on target attack is shown in Figure 2. From this figure, we see that the proposed UPGNN achieves the best performance in most cases. As the number of perturbations increases, the performance drop of the proposed UPGNN is not as heavy as the baselines. It indicates that the proposed UPGNN is more capable of resisting targeted adversarial attack.

Against Random Adversarial Attack

Random attack randomly injects fake edges into the graph, and can be viewed as adding random noise to graph. The results are reported in 3 The figure shows that UPGNN consistently outperforms all other baselines and successfully resists random attack. Together with observations from 4.3 and 4.3, we can conclude that UPGNN is able to defend various types of adversarial attacks

Dataset	Cora			Citeseer		
	ACC	NMI	F1	ACC	NMI	F1
k-means	50.0	31.7	37.6	54.4	31.2	41.3
SC	39.8	29.7	33.2	30.8	9.0	25.7
DeepWalk	52.9	38,4	43.5	39.0	13.1	30.5
DNGR	41.9	31.8	34.0	32.6	8.0	30.0
M-NMF	42.3	25.6	32.0	33.6	9.9	25.5
TADW	53.6	36.6	40.1	52.9	32.0	43.6
VGAE	59.2	40.8	45.6	39.2	16.3	27.8
MGAE	68.1	48.9	53.1	**66.9**	**41.6**	52.6
UPGNN	**70.0**	**54.1**	**67.4**	65.6	41.2	**57.3**

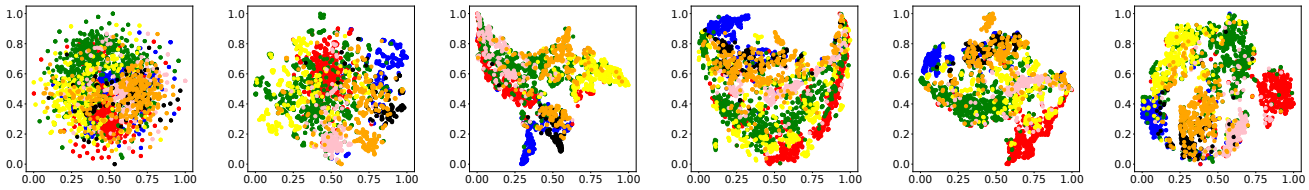
Table 2: Node clustering performance (in percentage).

4.4 Node Clustering

Table 2 reports node clustering performance of all the methods. From this table, we see that the proposed UPGNN outperforms most baselines in most cases. The performance gain benefits from refinement of graph structure in the proposed method. The above empirical results indicate good performance of the proposed method on more downstream tasks other than node classification.

4.5 UPGNN Versus AUPGNN

This section empirically compares UPGNN and AUPGNN in terms of effectiveness of efficiency. Figure 4 illustrates accuracy and training time of the two methods on three datasets. As can be observed, UPGNN generally have higher accuaries than AUPGNN in most cases. However, AUPGNN is around 10 times faster than UPGNN, verifying our theoretical findings on computational compelity of AUPGNN.



(a) Original Feature (b) Deepwalk (c) GCN (d) GMI (e) DGC (f) UPGNN

Figure 5: Visualization of the learned graph embedding by six representative methods on Cora under 20% perturbation by *metattack* [Zügner and Günnemann, 2019].

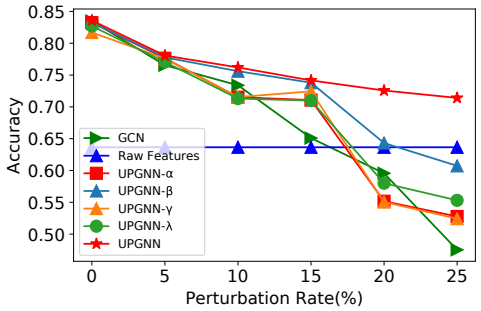
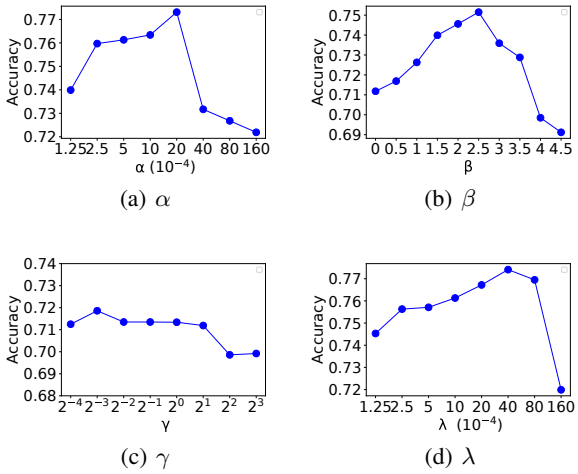


Figure 6: Ablation study of the proposed UPGNN on Cora.



(a) α (b) β (c) γ (d) λ

Figure 7: Parameter analysis of the proposed UPGNN on Cora.

4.6 Ablation Study

This section conducts ablation study on each loss in the proposed UPGNN. Specifically, we compare the proposed UPGNN with four variants, including (1) UPGNN-α that only includes sparsity loss, (2) UPGNN-β that only includes low rank loss, (3) UPGNN-γ that only includes GNN related loss, (4) UPGNN-λ that only includes feature smoothing loss. Figure 6 illustrates accuracies of all the methods with respect to different perturbation rates on Cora. It can be observed that UPGNN outperforms all the variants, confirming the importance of each loss in defending against adversarial attacks Among these variants, UPGNN-β outperforms the other variants, demonstrating the crucial role of low-rank property. Notably, even with substantial graph perturbations, UPGNN maintains competitive performance, surpassing methods that rely solely on raw features, while the variants and GCN perform worse.

4.7 Parameter Analysis

This section emprically investigates the sensitivity of some parameters, i.e., α, β, γ, and λ in the proposed UPGNN. Figure 7 reports accuracies with respect to varying parameters using *metattack* at a 15% perturbation rate on Cora. As can be observed, with the increase of these parameters, accuracy of UPGNN first improves and then decreases. The performance remains relatively stable with varying γ, compared to the other parameters. UPGNN can achieve the good performance when α, β, γ, and λ are set to 2×10^{-3}, 2.5, 2^{-3}, and 4×10^{-3} respectively.

4.8 Visualization

Figure 5 shows the visualization of graph embedding learned by six representative methods on Cora under 20% perturbation by *metattack*. As can be observed, compared to baselines, UPGNN can generate small clusters, and separate clusters better. The qualitative empirical results are consistent with previous quantitative empirical results.

5 Conclusion

This work focuses on the challenging and less studied area of unsupervised graph structure and learning without label supervision. We propose a new unsupervised deep GSL method, i.e., UPGNN that refines graph structure by exploring desirable properties of clean graph, and employs unsupervised graph mutual information loss to maximize correlation between refined graph and graph embedding. We further propose AUPGNN as a efficient alternative of UPGNN. Experiments on node classification and clustering validate the effectiveness of the proposed methods.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No. 62176126, 62101268,

the Natural Science Foundation of Jiangsu Province, China under Grant No. BK20230095.

References

[Belghazi *et al.*, 2018] Mohamed I. Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Hjelm R D. Mine: mutual information neural estimation. In *ICML*, pages 531–540, 2018.

[Cao *et al.*, 2016] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *AAAI*, pages 1145–1152, 2016.

[Dai *et al.*, 2018] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *ICML*, pages 1115–1124, 2018.

[Elinas *et al.*, 2020] Pantelis Elinas, Edwin V Bonilla, and Louis Tiao. Variational inference for graph convolutional networks in the absence of graph data and adversarial settings. *NIPS*, 33:18648–18660, 2020.

[Entezari *et al.*, 2020] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. All you need is low (rank) defending against adversarial attacks on graphs. In *WSDM*, pages 169–177, 2020.

[Fatemi *et al.*, 2021] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. Slaps: Self-supervision improves structure learning for graph neural networks. *NIPS*, 34:22667–22681, 2021.

[Feng and Wang, 2022] Long Feng and Junhui Wang. Projected robust PCA with application to smooth image recovery. *JMLR*, 23:249:1–249:41, 2022.

[Gao *et al.*, 2020] Xiang Gao, Wei Hu, and Zongming Guo. Exploring structure-adaptive graph learning for robust semi-supervised classification. In *ICME*, pages 1–6. IEEE, 2020.

[Gasteiger *et al.*, 2019] Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. *NIPS*, 32, 2019.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *SIGKDD*, pages 855–864, 2016.

[Hjelm *et al.*, 2018] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[Jin *et al.*, 2020] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *SIGKDD*, pages 66–74, 2020.

[Jin *et al.*, 2021] Wei Jin, Tyler Derr, Yiqi Wang, Yao Ma, Zitao Liu, and Jiliang Tang. Node similarity preserving graph convolutional networks. In *WSDM*, pages 148–156, 2021.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NeurIPS*, 2016.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Li *et al.*, 2018] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *AAAI*, volume 32, 2018.

[Nowozin *et al.*, 2016] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *NIPS*, 29, 2016.

[Peng *et al.*, 2020] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. Graph representation learning via graphical mutual information maximization. In *Proceedings of The Web Conference 2020*, pages 259–270, 2020.

[Peng *et al.*, 2022] Zhen Peng, Minnan Luo, Wenbing Huang, Jundong Li, Qinghua Zheng, Fuchun Sun, and Junzhou Huang. Learning representations by graphical mutual information estimation and maximization. *TPAMI*, 2022.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *SIGKDD*, pages 701–710, 2014.

[Raguet *et al.*, 2013] Hugo Raguet, Jalal Fadili, and Gabriel Peyré. A generalized forward-backward splitting. *SIAM Journal on Imaging Sciences*, 6(3):1199–1226, 2013.

[Richard *et al.*, 2012] Emile Richard, Pierre-André Savalle, and Nicolas Vayatis. Estimation of simultaneously sparse and low rank matrices. *ICML*, 2012.

[Savalle *et al.*, 2012] Pierre-André Savalle, Emile Richard, and Nicolas Vayatis. Estimation of simultaneously sparse and low rank matrices. In *ICML*, 2012.

[Veličković *et al.*, 2017] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *ICLR*, 2017.

[Veličković *et al.*, 2019] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *ICLR*, 2019.

[Wan and Kokel, 2021] Guihong Wan and Harsha Kokel. Graph sparsification via meta-learning. *AAAI*, 2021.

[Wang *et al.*, 2017a] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. MGAE: marginalized graph autoencoder for graph clustering. In *CIKM*, pages 889–898, 2017.

[Wang *et al.*, 2017b] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *AAAI*, pages 203–209, 2017.

[Wu *et al.*, 2019] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples on graph data: Deep insights into attack and defense. *IJCAI*, 2019.

[Xu *et al.*, 2020] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. Adversarial attacks and defenses in images, graphs and text: A review. *IJAC*, 17:151–178, 2020.

[Yang *et al.*, 2015] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. Network representation learning with rich text information. In *IJCAI*, pages 2111–2117, 2015.

[Yang *et al.*, 2019] Liang Yang, Zesheng Kang, Xiaochun Cao, Di Jin, Bo Yang, and Yuanfang Guo. Topology optimization based graph convolutional network. In *IJCAI*, pages 4054–4061, 2019.

[Zhang and Zitnik, 2020] Xiang Zhang and Marinka Zitnik. Gnnguard: Defending graph neural networks against adversarial attacks. *NeurIPS*, 33:9263–9275, 2020.

[Zhang *et al.*, 2019a] Xiaotong Zhang, Han Liu, Qimai Li, and Xiao-Ming Wu. Attributed graph clustering via adaptive graph convolution. *IJCAI*, 2019.

[Zhang *et al.*, 2019b] Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Ustebay. Bayesian graph convolutional neural networks for semi-supervised classification. In *AAAI*, volume 33, pages 5829–5836, 2019.

[Zhu *et al.*, 2019] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *SIGKDD*, pages 1399–1407, 2019.

[Zhu *et al.*, 2021] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *WWW*, pages 2069–2080, 2021.

[Zügner and Günnemann, 2019] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *ICLR*, 2019.

[Zügner *et al.*, 2018] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *SIGKDD*, pages 2847–2856, 2018.