The Pennsylvania State University
The Graduate School

TOWARDS TRUSTWORTHY GRAPH NEURAL NETWORKS

A Dissertation in
Information Sciences and Technology
by
Enyan Dai

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2024

The dissertation of Enyan Dai was reviewed and approved by the following:

Suhang Wang
Assistant Professor of Information Sciences and Technology
Dissertation Advisor
Chair of Committee

Xiang Zhang
Associate Professor of Information Sciences and Technology

Lu Lin
Assistant Professor of Information Sciences and Technology

Sencun Zhu
Associate Professor of Computer Science and Engineering

Dongwon Lee
Professor of Information Sciences and Technology
Director of Graduate Studies

Abstract

Graph is a kind of data structure which is composed of a set of nodes and their edges. Graph-structured data is very pervasive in the real-world such as social networks, protein modules, and traffic networks. Inspired by the great achievements of deep neural network on i.i.d data, *Graph Neural Networks (GNNs)* are proposed to generalize the deep learning model to process graph-structured data. The success of GNNs relies on the message-passing mechanism, which updates the node representations by aggregating the information of the neighbors. The message-passing mechanism could enhance node representations, and preserve both node feature characteristics and topological structures.

Though the utilization of graph topology with message-passing mechanism largely benefits the performance of GNNs, it can also lead to various new problems of achieving trustworthy GNNs in terms of robustness, privacy, fairness and explainablity. *First*, GNNs are vulnerable to the adversarial attacks and privacy attacks. More specifically, an attacker can control the predictions of the target GNN model by deliberately perturbing the graph structure and/or node attributes. The membership inference attack can detect whether a target sample belongs to the training set, resulting to the leakage of the private training information. Therefore, it is crucial to ensure the safety of graph neural networks in robustness and privacy. *Second*, discrimination towards protected sensitive attributes could be magnified by using the topology of graphs in node classification. Generally, in graphs such as social networks, nodes of similar sensitive attributes are more likely to connect to each other than nodes of different sensitive attributes. As a result, representations are often updated by nodes with the same sensitive attributes in GNNs, which could lead to severer bias in decision making. This will largely limit the applications of GNNs in sensitive domains such as job applicant ranking and crime detection. *Third*, GNNs make predictions based on both features of nodes and the graph topology. Thus, explanations about how the message-passing utilizes the topology and features of neighbors are crucial. However, existing studies mostly focus on explaining the neural networks' predictions on i.i.d data, explainable GNNs are rather limited.

In this dissertation, I will present the solutions to trustworthy graph neural networks in the aspects of robustness, privacy, fairness, and explainability. I will firstly present the method of defending graph topology noises followed by a unified framework that can simultaneously achieve robustness and membership privacy. Then, solutions to fair GNNs and self-explainable GNNs are presented. Extensive theoretical analysis and experimental results on the real-world datasets demonstrate that the proposed methods are effective in achieving trustworthy graph neural networks.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I wish to express my deepest gratitude to my advisor, Dr. Suhang Wang. Working with Dr. Wang on challenging problems has been an incredible experience. Under his guidance, I have gained a deep understanding of the intricacies involved in conducting research. Additionally, his professional attitude and immense passion for research have had a profound influence on me. Finally, I want to thank Dr. Wang for his support in both my academic pursuits and personal life. I sincerely appreciate his understanding and support as I build my own family.

Meanwhile, I sincerely appreciate the committee members of my dissertation, Dr. Xiang Zhang, Dr. Lu Lin, and Dr. Sencun Zhu for their constructive suggests on my research from various perspectives.

I also extend my heartfelt thanks to Junjie, Minhua, and all my other labmates and collaborators for their invaluable support throughout my research projects. Their insights and assistance have played a crucial role in my achievements. Additionally, I am grateful to my friends—Jiaqi, Xian, Jiachen, and others—who have accompanied me on the long and challenging journey of my Ph.D. Our discussions, spanning from the philosophy of research to the philosophies of life, have been a true treasure. These conversations have not only enriched my academic experience but have also been a source of personal strength and inspiration.

Finally, I would like to thank my father Tonghui Dai and mother Qing Lin for their unconditional love and supports. I also want to thank my little love Tianyue Dai, who was born at the end of the pandemic. She has became my motivation. Most importantly, I want to sincerely appreciate my wife Jiale Lin for her incredible contributions to the whole family. I have no doubt that being a mom is a much more crucial achievement than being a Ph.D. Her strength, support, and sacrifices have not only sustained our family but have also inspired me every day. I am deeply grateful for her enduring love and partnership.

This material is based upon work supported by, or in part by, the National Science

Foundation (NSF) under grant IIS-1909702, IIS1955851, the Global Research Outreach program of Samsung Advanced Institute of Technology under grant #225003, Army Research Office (ARO) under grant #W911NF-21-1-0198. The findings and conclusions in this paper do not necessarily reflect the view of the funding agency.

Chapter 1 Introduction

Graph-structured data such as bioinformatics network [1], trading network [2], and social network [3] are pervasive in the real-world. Inspired by the great success of deep learning on independent and identically distributed (i.i.d) data such as images, Graph Neural Networks (GNNs) [4–6] are investigated to generalize deep neural networks to model graph-structured data. GNNs have shown great performance for various applications across various domains including finance [7,8], healthcare [9] and social analysis [10]. The success of GNNs relies on the message-passing mechanism, where node representations are updated by aggregating the information from neighbors. With this mechanism, node representations can capture node features, information of neighbors and local graph structure.

Despite their achievements in modeling graphs, the concerns in the trustworthiness of GNNs are rising. Firstly, GNN models are vulnerable to the attacks that steal the private data information or affect the behaviors of the model. Specifically, hackers can easily infer the training set with the prediction vectors from the target model [11]. They also can easily fool the GNNs to give target prediction to a node by injecting perturbations to the graph structures and node attributes. Secondly, GNN models themselves have problems in fairness and interpretability. More specifically, GNN models can magnify the bias in the training data, resulting discrimination towards the people with certain genders, skin colors, and other protected sensitive attributes [12]. Finally, due to the high nonlinearity of the model, predictions from the GNNs are difficult to understand. The lacking of interpretability also make the GNNs untrustworthy, which largely limit the applications of GNNs. Those weaknesses significantly hinder the adoption of GNNs in real-world applications, especially those high-stake scenarios such as finance and healthcare. Therefore, how to build trustworthy GNN models has become a focal topic.

Recently, a guideline of trustworthy AI system have been proposed by the European

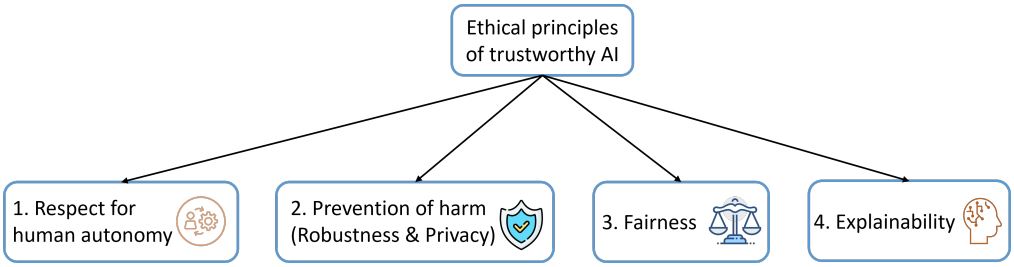


Figure 1.1: The ethical principles of trustworthy AI.

Union [13]. As shown in Figure 1.1, the guideline indicates that trustworthy AI should obey the following four ethical principles: *Respect for human autonomy*, *Prevention of harm*, *Fairness*, and *Explainability*. The principle of respect for human autonomy requires AI systems to follow human-centric design principles and leave meaningful opportunity for human choice. This generally fails in the domain of human-computer interaction. Therefore, we do not focus on this direction of trustworthy GNNs. According to the principle of prevention of harm, AI systems should be technically robust and be ensured not open to malicious use, which corresponds to the robustness and privacy aspects. The principle of fairness requires that AI systems should ensure the individuals and groups free from unfair bias, discrimination and stigmatisation. As for the explainability, it requires the decision process of AI to be transparent and explainable. Here, I first introduce the general architecture of GNNs. Next, the problems and high-level solutions of trustworthy GNNs in robustness, fairness, and explainability are presented.

1.1 Preliminaries of Graph Neural Networks

We use $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ to denote an attributed graph, where $\mathcal{V} = \{v_1, ..., v_N\}$ is the set of N nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ is the set of node attributes with $\mathbf{x}_i$ being the node attributes of node v_i. $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix of the graph $\mathcal{G}$, where $\mathbf{A}_{ij} = 1$ if nodes v_i and v_j are connected, otherwise $\mathbf{A}_{ij} = 0$. Generally, the node classification in graphs is semi-supervised. Only partial nodes $\mathcal{V}_L = \{v_1, ..., v_l\}$ are provided with labels $\mathcal{Y} = \{y_1, ..., y_l\}$.

Graph neural networks (GNNs) utilize the node attributes and edges to learn a representation $\mathbf{h}_v$ of the node $v \in \mathcal{V}$. The goal of learning representation in node classification is to predict the node v's label as $y_v = f(\mathbf{h}_v)$. Current GNNs are neighborhood aggregation approaches, which will update the representations of the nodes with the representations of the neighborhood nodes. The representations after k layers'

aggregation would capture the structural information of the k-hop network neighborhoods. The updating process of the k-th layer in GNN could be formulated as:

$$\begin{aligned}\mathbf{a}_v^{(k)} &= \text{AGGREGATE}^{(k-1)}(\{\mathbf{h}_u^{(k-1)} : u \in \mathcal{N}(v)\}),\\ \mathbf{h}_v^{(k)} &= \text{COMBINE}^{(k)}(\mathbf{h}_v^{(k-1)}, \mathbf{a}^{(k)}),\end{aligned} \tag{1.1}$$

where $\mathbf{h}_v^{(k)}$ is the representation vector of the node $v \in \mathcal{V}$ at k-th layer and $\mathcal{N}(v)$ is a set of neighborhoods of v. The representation $\mathbf{h}_v^{(0)}$ is initialized by the node attribute vector $\mathbf{x}_v$. The choice of $\text{AGGREGATE}^{(K-1)}(\cdot)$ and $\text{COMBINE}^{(k)}(\cdot)$ can be varied for different GNNs. For instance, GCN [4] apply element-wise mean pooling as the aggregator. LSTM aggregator and Pooling aggregator is also investigated in GraphSAGE [3]. For the COMBINE step, various designs such as linear transformation with a ReLU [4] and a multi layer perception [14] are also investigated.

1.2 Problems of GNNs in Trustworthiness

1.2.1 Safety of GNNs: Robustness and Privacy

To guarantee the safety of GNN models, the robustness and privacy should be simultaneously ensured. First, graph adversarial attack methods [15] can carefully revise edges and node attributes to change the predictions of GNNs on target nodes. Second, under membership inference attacks, the private training data information could be leaked from the predictions of the trained GNNs [16]. This potential attacks in adversarial robustness and membership privacy can largely hinder the application of GNNs in safety-critical domains such as healthcare. Therefore, I focused on a novel problem of simultaneously defending adversarial attacks and membership privacy attacks with a unified framework. And there are two unique challenges:

- Different from the robust models in i.i.d. data, adversarial perturbations or inherent noises in graphs happen in topology. How to eliminate the negative effects of graph-structure noise is under-explored.
- Real-world graphs are often sparsely labeled such as cell phone network for fraud detection [17]. The lack of labeled samples will challenge the existing membership privacy preserving methods, which generally require redundant labels.

To address the problem of adversarial perturbations on graph structures, I firstly propose a novel RS-GNN which adopts the noisy edges as supervision to learn a denoised and dense

graph. It can down-weight or eliminate noisy edges. The detailed design is presented in Chapter 2. I further investigate a novel information bottleneck framework named RM-GIB, which can simultaneously guarantee adversarial robustness and membership privacy. Inspired by the edge predictor in RS-GNN, RM-GIB deploys a similar module as information bottleneck to eliminate adversarial perturbations. In addition, the deployment of information regularization and privacy-preserving optimization with pseudo labels guarantee the membership privacy with the proposed RM-GIB. More details of the proposed RM-GIB are presented in Chapter 3.

1.2.2 Fairness of GNNs

Extensive studies [18–20] have revealed that historical data may include patterns of previous discrimination. Similar to other neural networks, GNNs trained on such data can inherit the bias on sensitive attributes such as ages, genders, skin color, and regions [12,18,19]. In addition, the bias of GNN can be magnified by the graph structures and message-passing mechanism of GNNs. Because nodes of similar sensitive attributes are more likely to connect to each other [21, 22]. For example, people in similar age tend to follow each other on the social network [21]. With the aggregation of neighbors' features in GNN, nodes of similar sensitive information will have similar representations, leading to predictions highly correlated with the sensitive attributes. Therefore, severe bias can in decision making, The bias would largely limit the wide adoption of GNNs in sensitive applications such as ranking of job applicants [23] and crime rate prediction [24]. Thus, it is important to investigate fair GNNs. Here I summarize the unique challenges of achieving fairness in graph neural networks:

- Generally, adundant sensitive attributes are required to constrain the machine learning models to achieve fairness. However, the sensitive attributes are often limited in real-world graphs due to privacy concern. Inadequate nodes with known sensitive attributes will challenge many existing fair models [19, 20, 25, 26].

- Apart from node features, graph structures can also lead to discrimination of the model due to the message-passing of GNNs. But the majority of existing fair models are dedicated to i.i.d data, which cannot simultaneously eliminate the bias from node features and graph structures.

Therefor, I propose to study the novel and important problem of *learning fair GNNs with limited sensitive attribute information*. FairGNN is proposed to eliminate the bias of

GNNs whilst maintaining high node classification accuracy by leveraging graph structures and limited sensitive information. Our theoretical analysis shows that FairGNN can ensure the fairness of GNNs under mild conditions given limited nodes with known sensitive attributes. Extensive experiments on real-world datasets also demonstrate the effectiveness of FairGNN in debiasing and keeping high accuracy. More details of our proposed method and experiments are presented in Chapter 4.

1.2.3 Explainability of GNNs

As an extension of deep learning for graph-structure data, GNNs also suffer from the problem of lacking explainability. What's more, apart from node features, GNNs also utilize graph topology to predict by aggregating information of neighbors. As a result, the methods [27–30] that explain the deep neural networks on i.i.d data cannot not be directly applied to explain the predictions of GNNs. The black-box characteristic of GNNs will largely restricts its use in the scenarios that demand the transparency of models. For instance, the predictions of module properties and design of drugs require explanations to make more reliable decisions. Thus, it is important to develop explainable GNNs. The potential challenges are summarized here:

- Since GNNs utilize both node attributes and graph topology to make predictions. How to explain the predictions of GNNs in terms of graph topology is required to be explored.
- Some initial efforts [31–33] have been take to generate post-hoc explanations for GNNs. However, the generated explanations have the risk of underrepresenting the true explanations of the original GNNs. How to design a explainable GNN that can provide explanations by itself is not well studied.

Therefore, I investigated a problem of *developing self-explainable GNNs*. The proposed self-explainable GNN will identify K-nearest labeled samples with interpretable similarity metric for both explaining and predicting. More details about this work are listed in Chapter 5.

1.3 Overview of the Dissertation

The rest of the dissertation is organized as follows. In Chapter 2, I introduce the initial solution to the adversarial perturbations on graph structures. Then, in Chapter 3, a

unified framework for both adversarial robustness and membership privacy is presented. The design of the fair GNN is introduced in Chapter 4. The self-explainable GNN is presented in Chapter 5. Additional discussions and potential future directions are given in Chapter 6.

Chapter 2
Safety of Graph Neural Network: Defense Against Structural Noise

2.1 Introduction

Graph neural networks (GNNs) [3, 4, 34] have made remarkable achievements in modeling graph structured data from various domains such as social networks [3], financial system [35], and recommendation system [36]. The success of GNNs relies on the message-passing mechanism [3, 4], where node representations are updated by aggregating the information from neighbors. With this mechanism, the node representations capture node features, information of neighbors and local graph structure, which facilitate various tasks, especially semi-supervised node classification.

Although GNNs have shown great ability in modeling graphs, the performance of GNNs could degrade significantly when they are trained on graphs with *noisy edges* and/or *limited labeled nodes*. *First*, due to the message passing mechanism, GNNs are vulnerable to adversarial or noisy edges. For example, as shown in Fig. 2.1, poisoning attacks [37] add/delete carefully chosen edges to the graph. These adversarial edges (shown in red) usually connect nodes of different labels or features, thus contaminating the neighborhoods of nodes, propagating noises/errors to node representations, and degrading the GNN performances significantly. In addition, inherent edge noises also exist in real-world graphs. For instance, in social networks, bots tend to build connections with normal users to spread misinformation [38], which can also harm the performance of GNNs for bot detection. *Second*, for many applications, graphs are often sparsely labeled such as cell phone network for fraud detection [17]. Label sparsity can severely reduce the involvement of unlabeled nodes during message passing, leading to poor performance. Generally, in a K-layer GNN, a labeled node aggregates its K-hop neighborhood information, thus

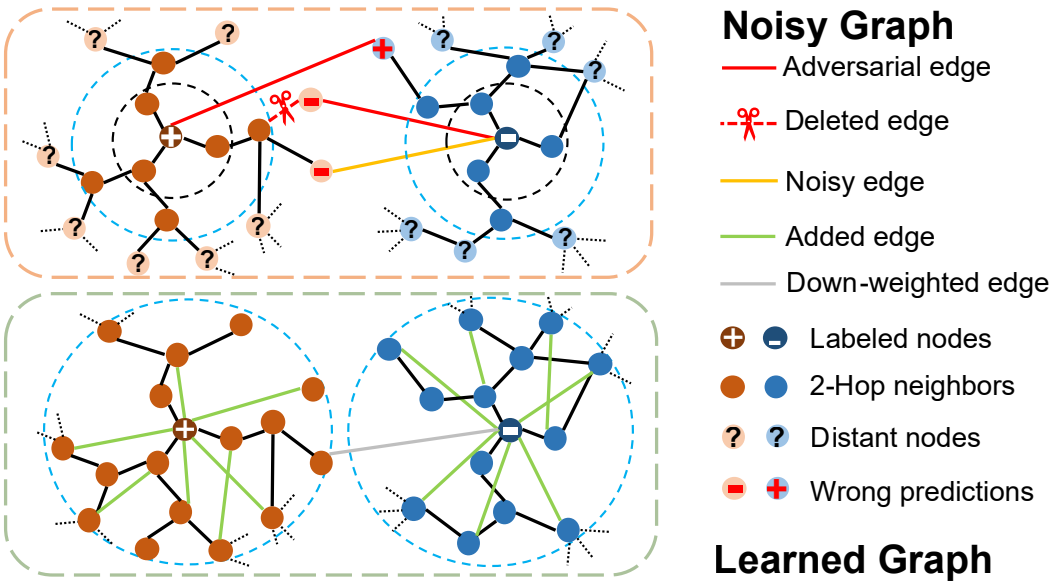


Figure 2.1: An illustration of down-weighting/removing noise edges and densifying the graph for better performance.

making many unlabeled nodes in K-hop neighborhood participate in the training, which is one major reason that GNNs can leverage unlabeled nodes for semi-supervised node classification. However, as verified in our preliminary analysis in Fig. 2.2a of Sec. 2.3.2, when the number of labeled nodes decreases, the amount of unlabeled nodes participating in training drops quickly, making message passing less effective. These shortcomings of GNNs hinder the adoption of GNNs for many real-world applications. Thus, it is important to develop robust GNNs that can simultaneously handle noisy graphs with sparse labels.

However, developing robust GNNs for graphs with noisy edges and limited labeled nodes is challenging. *First*, the training graph itself is noisy, i.e., noisy edges are mixed with the normal edges. Thus, we need supervision in down-weighting or eliminating noisy edges. *Second*, alleviating the limited label issue requires more labels, while obtaining more labeled nodes is time-consuming and expensive. Hence, we need alternative approaches to more effectively utilize the limited labels. Some initial efforts [39, 40, 40, 41] have been taken to alleviate the effects of the adversarial edges such as pruning edges by using node similarity [39], and adopting Gaussian distribution as node representations to absorb noises [42]. To address the problem of sparsely labeled graphs, some methods [43–45] propose to obtain better representations by training GNNs with self-supervised learning tasks such as pseudo label prediction [43,44] and global context predictions [45]. However, little efforts are taken for robust GNNs that can simultaneously handle noisy edges and

label sparsity.

Since both the noisy edges and limited labeled nodes harm the message passing of GNNs and message passing is directly related to the graph structure, we argue that learning a denoised and dense graph guided by the raw attributed graph is promising to facilitate message passing for robust GNNs. *First*, for many graphs such as social networks, nodes with similar features and labels tend to be linked [46], while noisy edges would link nodes of dissimilar features [39]. Thus, we can use node attributes to predict the links. For existing links, the link predictor will assign small weights to links connecting nodes of dissimilar features while large weights to links connecting nodes of similar features, thus alleviating negative issue of noisy edges during message passing. *Second*, real-world graphs are usually very sparse, containing many missing edges. With the link predictor, nodes that are potentially to be linked could be identified. Densifying the graph by linking similar nodes would induce more unlabeled nodes to become neighbors of labeled nodes with the same labels as shown in Fig. 2.1, which can alleviate label sparsity issue. In addition, since adjacent nodes tend to have the same labels, the predicted new links can be used to further regularize the label prediction of unlabeled nodes. Though promising, the work on down-weighting noisy edges and densifying graph for robust GNN on noisy graphs with sparse labels are rather limited.

Therefore, in this chapter, we investigate a novel problem of developing robust noise-resistant GNNs with limited labeled nodes by learning denoised and densified graph. In essence, we need to solve two challenges: (i) how to effectively learn a link predictor from the noisy graph which can eliminate noisy edges and densify the graph; and (ii) how to simultaneously use the learned graph to learn a structural noise-resistant GNNs with limited labeled nodes. To address these challenges, we propose a novel framework named robust structural noise-resistant GNN (RS-GNN). RS-GNN adopts the node attributes and supervision from the noisy edges to denoise and density graph, which can alleviate the negative effects of noisy edges and facilitate the message passing between unlabeled nodes and labeled nodes. The generated graph is used as input for learning a GNN. RS-GNN also adopts the predicted edges as self-supervision to further explicitly regularize the label prediction of unlabeled nodes to alleviate the label sparsity issue. In summary, the main contributions of our chapter are:

- We study a new problem of learning robust noise-resistant GNNs with limited labeled nodes;
- We propose a novel framework RS-GNN, which can simultaneously learn a denoised

and densified graph and a robust GNN on noisy graphs with limited labeled nodes; and

- We conduct extensive experiments on real-world datasets to demonstrate the robustness of RS-GNN on both noisy/clean graphs with limited labeled nodes.

2.2 Related Work

In this section, we review the related work, which includes graph neural networks, and robust graph neural networks.

2.2.1 Graph Neural Networks

Graph Neural Networks (GNNs) have shown their great power in modeling graph structured data for various applications [4, 35, 47, 48]. To generalize neural networks for graphs, two categories of GNNs are proposed, i.e., spectral-based [4, 34, 48, 49] and spatial-based [3, 5, 50, 51]. Spectral-based GNN [34] is firstly proposed by defining graph convolution with spectral graph theory. More spectral-based methods are developed for further improvements [4, 48, 49]. For instance, GCN [4] simplifies the convolutional operation by using the first order approximation. Spatial-based graph convolution is defined in spatial domain, which updates node representation by aggregating its neighbors' representations [3, 52, 53]. For example, self-attention of neighbor nodes is leveraged in graph attention network (GAT) [50]. Moreover, various spatial methods are proposed to solve the scalability issue [5, 51] and learn deeper GNNs [54]. Recently, to alleviate the problem of lacking labeled nodes, many efforts are taken to explore GNNs using self-supervision, which aims to learn better node representations with pretext tasks [43–45, 55, 56]. For instance, self-training [44] proposes to firstly train a GCN with given labels and add the confident pseudo labels to the label set for later training. Global context prediction is also propose to be applied to learn better graph representations [45].

Inspired by the great success of GNNs, methods that construct graphs and adopt GNNs for data without explicit relational structure are also explored [49, 57, 58]. Generally, a graph would be built based on certain rules [49, 57] or be learned in an end-to-end model [58]. The proposed framework is inherently different from aforementioned methods as we aim to eliminate/down-weight the noisy edges and predict the missing edges for robust GNNs on noisy graphs with limited labeled nodes.

2.2.2 Robust GNNs

Although GNNs have obtained great achievements, they are proven to be vulnerable to adversarial attacks [15, 37, 39, 59]. Based on the objective, the adversarial attacks methods on GNNs could be split into two categories, i.e., targeted attack [15, 59] and non-targeted attack [37]. Targeted attack methods aim to degrade the performance of the GNNs on target nodes. For instance, *nettack* [15] could add adversarial perturbations to a graph to attack targeted nodes. Non-targeted attack aims to reduce the overall performance of GNNs. For example, meta-learning is employed in *metattack* [37] to poison the graph globally to achieve non-targeted attack. To defend against adversarial attacks, many efforts are taken recently [39–42, 60]. RGCN [42] adopts Gaussian distributions as representations to absorb the effects of adversarial changes. And GCN-jaccard [39] prunes the perturbed edges based on Jaccard similarity of node features. Another preprocessing method by low-rank approximation of adjacent matrix is investigated [60]. Pro-GNN [40] is the most similar work to ours, which also aims to learn a clean graph structure by low-rank constraint. However, they only tackle the adversarial edges and their computational cost is very large due to the direct learning of the graph and the sparse low-rank constraint. This work is inherently different from aforementioned methods as: (i) we study a novel problem of developing robust GNN for both noisy graphs and label sparsity issues; and (ii) the proposed RS-GNN simultaneously tackles the two issues by learning an edge predictor to down-weight noisy edges and connecting nodes with high similarity to facilitate message-passing; and (iii) RS-GNN uses edge predictor instead of direct graph learning to save computational cost.

2.3 Preliminary Analysis

In this section, we discuss the inner working of GNNs, conduct preliminary analysis to show the issues of GNN with sparse labels and verify that densifying graphs by connecting similar nodes can potentially alleviate the issue.

2.3.1 Notations

In this chapter, we consider an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$. As it is mentioned in Sec 1.1, $\mathcal{V} = \{v_1, ..., v_N\}$ is the set of N nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ is the set of node attributes with $\mathbf{x}_i$ being the node attributes of node v_i. $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix of the graph $\mathcal{G}$, where $\mathbf{A}_{ij} = 1$ if nodes v_i and

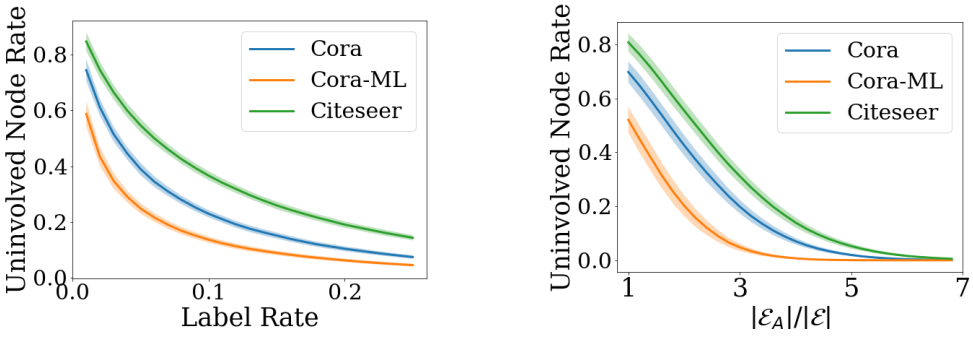
v_j are connected, otherwise $\mathbf{A}_{ij} = 0$. In our setting, only a limited number of nodes $\mathcal{V}_L = \{v_1, ..., v_l\}$ are provided with labels $\mathcal{Y} = \{\mathbf{y}_1, ..., \mathbf{y}_l\}$, where $\mathbf{y}_i \in \mathbb{R}^C$ is a one-hot vector of node v_i's label for multi-class classification. Note that the topological structure of the graph $\mathcal{G}$ could be noisy such as poisoned by adversarial edges or containing inherent noises, which would lead to poor performance of GNNs.

2.3.2 Analysis of GNNs with Sparse Labels

In this subsection, we conduct preliminary analysis on real-world graphs to show the issues of GNNs when limited labeled nodes are available for training, which paves us a way to design robust GNNs for alleviating the label sparsity issue.

During the training of node classification, the representations of labeled nodes are used to give prediction and obtain the training loss to minimize. With the message-passing mechanism, after K-layers of GNN, the node representation of v_i would capture the node features and structure information of the K-hop neighborhoods of v_i, and thus facilitating downstream tasks. In other words, in GNN, *one labeled node would make the K-hop neighborhood participate in the training of GNN*, which is one reason that GNNs have great ability in leveraging unlabeled nodes for semi-supervised node classification. Generally, GNNs, such as GCN and GAT, rely on the classification loss of the labeled nodes to learn the parameters, which is effective when we have adequate labeled nodes. However, when the size of labeled node set $\mathcal{V}_L$ is small and the graph is sparse, only a small portion of nodes would be involved in the training. This may lead to poor performance of GNNs. More specifically, for a K-layer GNN, the nodes involved in the training phase include the labeled nodes and the unlabeled nodes within K-hop distance of labeled nodes. We usually set K as 2 to 3 because deep GNNs have over-smoothing issue [44]. Since real-world graphs are usually sparse, the K-hop neighbors of the labeled nodes would be limited as well. Thus, when $\mathcal{V}_L$ is small, only a small portion of nodes would be involved in training, making GNNs less effective in leveraging unlabeled nodes.

To demonstrate that GNNs can be highly affected when the size of labeled nodes is limited, we conduct empirical analysis on three widely used datasets, i.e., Citeseer [61], Cora and Cora-ML [62]. More specifically, we analyze how the label rate affects the rates of uninvolved nodes of real-world datasets for a two layer GNN. We vary label rates from 0.01 to 0.25. The average uninvolved node rates and the standard deviations are shown in Fig. 2.2a. From the figure, we observe that (**i**) when the label rate is high, say above 0.1, most of the nodes are involved in training GNN. The benefit of further increasing label rate is marginal as the 2-hop neighbors of labeled nodes could overlap. This is one



(a) Impacts of label rate

(b) Impacts of graph density

Figure 2.2: The impacts of label rate and density of graph to uninvolved node rate in the training phase.

reason that GNNs have great ability for semi-supervised node classification with small but adequate amount of labeled nodes, and the increase of labeled nodes can marginally improve the performance; (**ii**) As the label rate decreases from 0.1, the uninvolved node rate increases significantly, i.e., the majority of nodes are not involved in the training. This indicates that GNNs would have difficulty in handling sparsely labeled graphs.

Although a higher label rate could help to reduce the uninvolved node rate, it is expensive and time-consuming to obtain more labels [63]. Thus, we need an alternative approach to effectively use the labels. From the analysis above, one potential solution is to make the graph denser so that one labeled node could have more neighbors to be involved in the training of GNN. To verify it, we randomly add different amount of edges to the three graphs. We denote the number of edges of the new graph as $|\mathcal{E}_A|$ and that of raw graph as $|\mathcal{E}|$. We fix label rate as 0.01. The impact of the graph density on the uninvolved node rate is presented in Fig. 2.2b. From the figure, we observe that when $|\mathcal{E}_A|/|\mathcal{E}|$ increase from 1 to 3, i.e., we add two times the number of original edges, the uninvoled node rate drops significantly. For example, it drops from 0.8 to less than 0.4 for Citeseer.

As real-world graphs such as social networks have many pairs of nodes who are similar but not connected together, the analysis above shows that it is promising to predict links to densify the graph, which can help the message passing of GNNs to alleviate the issue of limited labeled nodes. In addition, these predicted edges can also be directly used to regularize the predicted labels of unlabeled nodes, i.e., if two nodes are more likely to have a link, they are more likely to have the same labels.

2.3.3 Problem Definition

Our preliminary analysis shows that predicting links to densify the graph can potentially alleviate the label sparsity issue. In addition, the link prediction can potentially down-weight or eliminate noisy edges as noisy edges usually connect nodes with low node attribute similarity. Therefore, we aim to simultaneously eliminate noisy edges and densify the graph with a link predictor and train a robust GNN on the new graph for node classification. The problem is formally defined as:

Problem 1. *Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ with edge set $\mathcal{E}$ might contain a small amount of noisy edges, and a small set of labeled nodes $\mathcal{V}_L \in \mathcal{V}$ with the corresponding labels in $\mathcal{Y}$, simultaneously learn adjacency matrix $\mathbf{S} \in [0,1]^{N \times N}$ which down-weights/removes noisy edges and completes missing links by a link predictor $f_E : (v_i, v_j) \rightarrow \mathbf{S}_{ij}$, and a GNN on the learned graph for node classification, i.e., $f_{\mathcal{G}} : (\mathbf{S}, \mathbf{X}) \rightarrow \hat{\mathcal{Y}}$, where $\mathbf{S}_{ij}$ indicates the weight of edge linking v_i and v_j and $\hat{\mathcal{Y}}$ is the set of predictions for unlabeled nodes.*

2.4 Proposed Framework – RS-GNN

In this section, we present the details of the proposed RS-GNN. The main challenges are: (i) given the noisy graph, how can we learn a link predictor which can down-weight/eliminate noisy edges and densify the graph; and (ii) how to simultaneously use the learned graph for node classification. As the graph topology is noisy, we cannot directly apply a GNN on $\mathcal{G}$ to predict edges because the message passing would magnify the negative effects of the noisy edges. Generally, nodes sharing similar features tend to connect to each other; while noisy edges tend to connect nodes of dissimilar nodes. Thus, we propose to learn a MLP-based link predictor which predicts links using node attributes. The more similar the node features of two nodes are, the larger weights the link predictor will assign. Thus, the link predictor is able to down-weight or eliminate noisy edges in initial graph. Meanwhile, the edge predictor can predict missing links to alleviate label sparsity issue. We design a novel feature similarity weighted edge-reconstruction loss to train the link predictor so as to reduce the negative effects of noisy edges on the link predictor. An illustration of the framework is shown in Figure 2.3, which contains a link predictor f_E and a GCN classifier $f_{\mathcal{G}}$. The link predictor f_E takes node features as input to learn a dense adjacency matrix $\mathbf{S}$, aiming to remove adversarial edges and assign edges that benefit predictions. The GCN classifier $f_{\mathcal{G}}$ takes $\mathbf{S}$ node features $\mathbf{X}$ to predict the

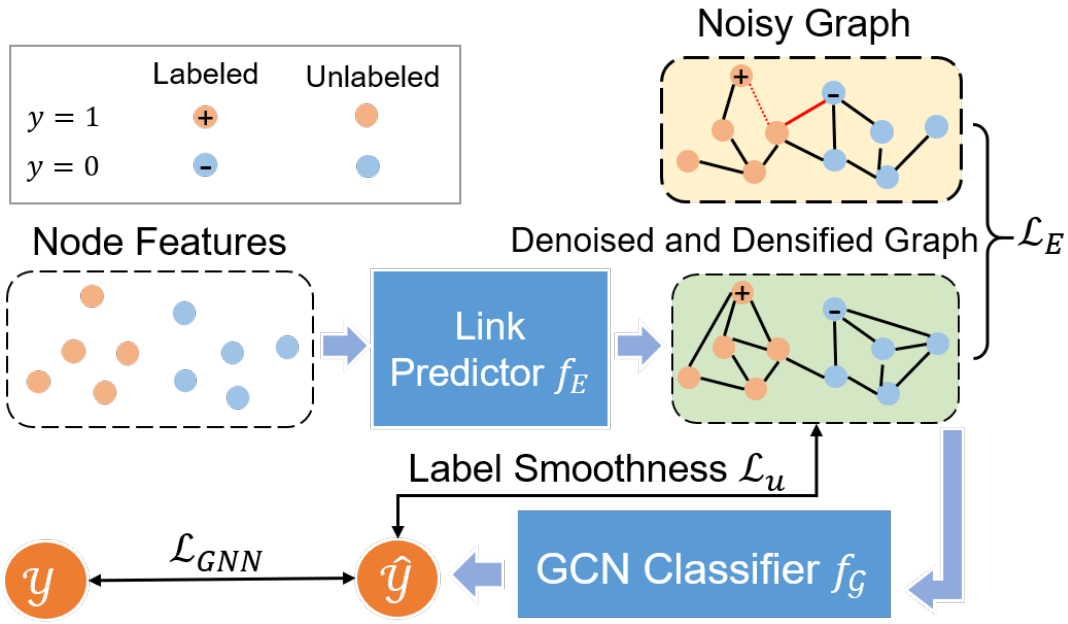


Figure 2.3: The overview of proposed RS-GNN.

node labels with the node features. Finally, label smoothness constraint based on the predicted edges will be added to the predictions of unlabeled nodes to further alleviate label sparsity issue. Next, we give the details of each component.

2.4.1 Link Prediction

As the given graph contains some adversarial noises and has missing edges, we propose to learn a new graph that down-weights noisy edges to eliminate their negative effects and completes the missing edges to facilitate GNN in dealing graphs with noisy edges and limited labeled nodes.

Building Link Predictor. Generally, noisy edges connect two nodes with dissimilar node features; while nodes of similar features are likely to have similar labels and should be connected. Therefore, we propose to predict edge weights and missing edges between nodes using nodes features. Specifically, for node v_i, a MLP takes its node attributes $\mathbf{x}_i$ to learn its node representation as

$$\mathbf{z}_i = MLP(\mathbf{x}_i) \tag{2.1}$$

With the node representations, we predict the weight $w(i,j)$ between $v_i \in \mathcal{V}$ and $v_j \in \mathcal{V}$ as:

$$w(i,j) = f(\mathbf{z}_i^T \mathbf{z}_j), \tag{2.2}$$

where f is the activation function. For the selection of f, we use ReLU instead of sigmoid as we find that when the learned adjacency matrix is used as the input of GCN, the use of sigmoid function will lead to gradient vanishing, which is consistent with previous observations [64]. Note that we use MLP instead of a GNN as the link predictor because the graph structure is noisy and the message passing of GNN could magnify the negative effects.

Learning Link Predictor. Our goal is to learn a link predictor which can (i) assign small weights to two nodes of different features so as to eliminate noisy edges; and (ii) assign larger weights to two nodes of similar node features so as to densify the graph to facilitate message passing. As for many real-world graphs, similar nodes tend to link together and linked nodes usually have high feature similarity. Thus, to learn a good link predictor f_E, we utilize the adjacency matrix reconstruction as the loss function. Since the graph is sparse, the adjacency matrix $\mathbf{A}$ contains many zero entries. Directly adopting adjacency matrix reconstruction as the loss function would (i) result in poor performance as the link predictor will be biased on predicting missing links; and (ii) require large computational cost as we need to calculate N^2 edges. To address this problem, negative sampling [65] is adopted, i.e., for each $v_j \in \mathcal{N}(v_i)$, we randomly sample Q nodes that's not connected to v_i and use them as negative samples.

However, a small portion of edges in $\mathbf{A}$ are noisy, which might have negative effects in training the predictor. To mitigate the negative effects of noisy edges and to learn a link predictor that can assign lower weights to edges that link dissimilar nodes, we propose to reweight the positive and negative samples based on the feature similarity of two nodes. Specifically, for node v_i and its positive sample $v_j \in \mathcal{N}(v_i)$, we minimize $\exp(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{\sigma^2})(w(i,j)-1)^2$, where σ is the hyperparameter to control the variance of the sample weights. Thus, if the node features of v_i and v_j are similar, A_{ij} is likely to be a clean edge and $\exp(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{\sigma^2})$ would be large. Minimizing the loss will force $w(i,j)$ to be close to 1; while if the features are dissimilar, then A_{ij} is likely to be a noisy edge and $\exp(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{\sigma^2})$ would be small, thus minimizing the loss will have little effect on $w(i,j)$. Similarly, for v_i and its negative sample v_n, we minimize $\exp(\frac{\|\mathbf{x}_i-\mathbf{x}_n\|^2}{\sigma^2})(w(i,n)-0)^2$. If the node features of v_i and v_n are dissimialr, then $\exp(\frac{\|\mathbf{x}_i-\mathbf{x}_n\|^2}{\sigma^2})$ is large, minimizing the loss would make $w(i,n)$ close to 0 as expected. With the weight defined in this way, the

loss for training the link predictor is written as:

$$\mathcal{L}_E = \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{N}(v_i)} \Big[\exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2})(w(i,j)-1)^2 + \sum_{n=1}^{Q} \cdot \mathbb{E}_{v_n \sim P_n(v_i)} \exp(\frac{\|\mathbf{x}_i - \mathbf{x}_n\|^2}{\sigma^2})(w(i,n)-0)^2 \Big], \tag{2.3}$$

where $P_n(v_i)$ is the distribution of sampling negative nodes for v_i, which is a uniform distribution. With the loss function Eq.(2.3), the link predictor would be able to downweight the noisy edges and densify the graph to facilitate the learning of robust GNN on noisy graph with limited labels.

Graph Purification and Densification. With the link predictor, we could apply the learned weights to the existing edges and drop edges whose predicted weights are small to eliminate the negative effects of noisy/adversarial edges. Moreover, to increase the involvement of unlabeled nodes to facilitate the message passing of GNNs, we also link nodes that have large weights predicted by the link predictor. However, if we predict weights of all pairs of nodes, the computation cost will be very large because we will train a link predictor and a GNN classifier end-to-end as shown in Sec. 2.4.4, which means we need to do prediction in each iteration. To save the computational cost, for each node v_i, we first construct a candidate subset $\mathcal{S}(v_i)$, which contains K nodes having the largest cosine similarities with v_i in the raw feature space $\mathbf{X}$. Note that this only needs to be done once. Since nodes not in $\mathcal{S}(v_i)$ are not likely to be connected with v_i, we only need to compute weights between v_i and $\mathcal{S}(v_i)$. The whole process of obtaining a clean and dense adjacency matrix $\mathbf{S}$ could be formally stated as:

$$\mathbf{S}_{ij} = \begin{cases} w(i,j) & \text{if } w(i,j) > T_l \text{ and } v_j \in \mathcal{N}(v_i) \cup \mathcal{S}(v_i); \\ 0 & \text{else,} \end{cases} \tag{2.4}$$

where $\mathcal{N}(v_i)$ are neighbors of v_i in the noisy graph, and T_l is a threshold to determine whether we should keep/add the edge. With the above operation, those noisy edges would be assigned smaller weights or even dropped, which mitigate the negative effects of noisy edges. Meanwhile, more edges are introduced to facilitate the message passing of GNNs during training.

2.4.2 GNN for Node Classification

With the learned adjacency matrix $\mathbf{S}$, we can apply GNNs to learn the node representation as $\mathbf{H} = GNN(\mathbf{S}, \mathbf{X})$. Note that the proposed framework is a flexible framework which

can facilitate various GNNs such as GAT [50] and GIN [14]. With the node representation, the label of node v_i can be predicted as

$$\hat{y}_i = softmax(\mathbf{z}_i) \tag{2.5}$$

where $\mathbf{z}_i$ is the representation of node v_i. The training loss is

$$\mathcal{L}_{GNN} = \sum_{v_i \in \mathcal{V}_L} l(\hat{\mathbf{y}}_i, \mathbf{y}_i) \tag{2.6}$$

Since $\mathbf{S}$ is denser than the original graph, more unlabeled nodes are involved in the training even with limited amount of labeled nodes, thus making the propagation of information more efficient.

2.4.3 Label Smoothness on Unlabeled Nodes

Though the dense graph $\mathbf{S}$ can help to include more unlabeled nodes in the loss function, their information is propagated through the message-passing mechanism instead of being directly used in the training loss. To further alleviate the issue of limited labeled nodes, we propose to adopt the predicted weighted edges for label smoothness regularization. The basic idea is the larger weights of an edge S_{ij} is, the more likely that v_i and v_j have the same label [36]. Thus, for an unlabeled node v_i, if its edge weight with node v_j is larger than a threshhold T_h, i.e., $S_{ij} > T_h$, we want their predicted labels to be similar with each other. This can be formally written as

$$\mathcal{L}_u = \sum_{v_i \in \mathcal{V}_u} \sum_{v_j \in \mathcal{V}} \mathbf{T}_{ij} \|\hat{\mathbf{y}}_i - \hat{\mathbf{y}}_j\|^2, \tag{2.7}$$

where $\mathcal{V}_u$ denotes the set of unlabeled nodes, $\hat{\mathbf{y}}_i$ and $\hat{\mathbf{y}}_j$ represent the predictions of node $v_i \in \mathcal{V}_u$ and $v_j \in \mathcal{V}$, respectively. $\mathbf{T}_{ij} = \mathbf{S}_{ij}$ if $\mathbf{S}_{ij} > T_h$; otherwise 0. In this way, we explicitly smooth the predicted labels between unlabeled nodes and nodes that are similar to them. We also take the weights of the edge into consideration by including $\mathbf{T}_{ij}$ in $\mathcal{L}_u$.

2.4.4 Final Objective Function of RS-GNN

With the link predictor down-weighting the noisy edges and densifying the graph with the supervision from $\mathbf{A}$, the GNN adopting the learned graph for label prediction and the label smoothness regularization from the generated graph, the final loss function can

be written as

$$\arg\min_{\theta_E,\theta_{\mathcal{G}}} \mathcal{L}_{GNN} + \alpha\mathcal{L}_E + \beta\mathcal{L}_u, \tag{2.8}$$

where θ_E and $\theta_{\mathcal{G}}$ are parameters of link predictor f_E and GNN classifier $f_{\mathcal{G}}$, respectively. α and β are hyperparameters to balance the contributions of reconstructing the adjacency matrix with f_E and label smoothness regularization. The proposed framework is an end-to-end framework where we simultaneously learn the link predictor and utilize the predicted edges for training a robust GNN to alleviate the noisy graph and limited labeled nodes issues.

2.5 Experiments

In this section, we evaluate the effectiveness of the proposed RS-GNN on various types of noisy graphs with limited labeled nodes. In particular, we aim to answer the following research questions:

- **RQ1** How robust is the proposed framework on various types of noisy graphs with limited labeled nodes?
- **RQ2** How does the proposed framework perform under various label rate and graph sparsity?
- **RQ3** What are the contributions of link predictor and label smoothness regularization from predicted edges on RS-GNN?

2.5.1 Experimental Settings

2.5.1.1 Datasets

For a fair comparison, we conduct experiments on four widely used benchmark datasets, i.e., Cora, Cora-ML, Citeseer and Pubmed [61]. The statistics of the datasets are presented in Table 2.1. Note that the split of validation and testing on all datasets are the same as described in the cited papers to keep consistence. For the training set, we randomly sample 1% of nodes as the labeled set for Cora, Cora-ML and Citeseer. For Pubmed, we randomly sample 10% of nodes to compose the labeled set. The training node set doesn't overlap with the validation and test sets.

Table 2.1: Statistics of datasets.

	Cora	Cora-ML	Citeseer	Pubmed
# of nodes	2,485	2,810	2,110	19,717
# of edges	5,069	7,981	3,668	44,338
# of features	1,433	2,879	3,703	500
# of classes	7	7	6	3

2.5.1.2 Noisy Graphs

To show the effectiveness of the proposed RS-GNN under various types of structural noise, we evaluate RS-GNN on the following types of noises:

- **Raw Graphs**: They are the original graphs of the benchmark datasets which may contain inherent structural noise.
- **Random Noise**: We randomly inject fake edges and remove normal edges to add random noise to graphs.
- **Non-Targeted Attack**: We adopt *metattack* [37] to poison the graph structures by adding and removing edges, which aims to reduces the overall performance of GNNs on the whole graph.
- **Targeted Attack**: It aims to lead the GNN to misclassify target nodes. Following [41], we randomly select 15% nodes as target nodes and apply *nettack* [15] to perturb the graph structure.

2.5.1.3 Baselines

We compare RS-GNN with the representative and state-of-the-art GNNs, and robust GNNs against adversarial attacks:

- **GCN** [4]: GCN is a representative GNN which defines Graph convolution with spectral analysis.
- **Self-Training** [44]: This is a self-supervised learning method. A GCN is firstly trained on given labels. Then, confident pseudo labels would be added to the label set to improve the GCN.
- **RGCN** [42]: It uses Gaussian distributions as representations to absorb the effects of adversarial edges. Attention mechanism is utilized to reduce the effects of nodes with high variance.

- **GCN-jaccard** [39]: Since noisy/adversarial edges mainly connect dissimilar nodes, GCN-Jaccard first eliminates edges that connect nodes with low Jaccard similarity, then apply GCN on the graph.
- **GCN-SVD** [60]: This preprocessing method is based on low rank assumption. It finds that *nettack* would result in a high-rank adjacency matrix. Thus, low-rank approximation of the perturbed graph is used to train GNNs against adversarial attacks.
- **Pro-GNN** [40]: It applies low-rank and sparsity constraint to directly learn a clean graph close to the noisy graph to defend against adversarial attack. Feature smoothness is also applied to regularize the learned graph. It doesn't densify the graph.

For all the baselines, we use the implementation from the repository DeepRobust [66]. All the hyperparameters of the baselines are tuned on the validation set to make a fair comparison with RS-GNN.

2.5.1.4 Implementation Details

Each experiment is conducted 5 times and average results with standard deviations are reported. The hyperparameters are tuned based on the performance of validation set. More specifically, for RS-GNN, we vary α as {0.003, 0.03, 0.3, 3, 30 }, and β as {0.01, 0.03, 0.1, 0.3, 1}. For all experiments, T_l, T_h, σ, and Q are fixed as 0.1, 0.8, 100, and 50, respectively. K is set as 100, 300, 400 and 10 for Cora, Cora-ML, Citeseer and Pubmed, respectively. More details about the hyperparameters sensitivity is discussed in Sec. 2.5.6. A one-hidden layer MLP with 64 filters is applied as the link predictor. We use GCN as the backbone of RS-GNN. Various GNNs can be used in RS-GNN and we leave it as a future work.

2.5.2 Performance on Noisy Graphs

To answer **RQ1**, we first compare RS-GNN with the baselines on various noisy graphs. We then evaluate the performance of RS-GNN on the graphs with different levels of structural noise.

2.5.2.1 Comparisons with Baselines

We conduct experiments on four types of noisy graphs, i.e., raw graphs, graphs with random noise, non-targeted attack perturbed graphs and targeted attack perturbed

Table 2.2: Node classification performance (Accuracy±Std) on different noisy graphs

Dataset	Graph	GCN	Self-Training	RGCN	GCN-jaccard	GCN-SVD	Pro-GNN	Ours
Cora	Raw Graph	65.5 ±0.5	67.9 ±0.9	63.0 ±0.7	65.7 ±0.6	62.9 ±1.1	65.9 ±1.3	**75.3 ±0.6**
	Random Noise	59.2 ±0.7	63.1 ±0.5	51.5 ±0.7	57.8 ±1.4	51.5 ±0.7	56.1 ±3.0	**71.8 ±1.5**
	Non-Targeted Attack	26.8 ±2.5	29.6 ±0.4	30.4 ±1.0	48.3 ±2.0	37.1 ±1.4	41.7 ±5.7	**70.8 ±0.7**
	Targeted Attack	45.3 ±1.2	46.7 ±2.1	40.3 ±1.0	49.5 ±1.0	44.8 ±0.7	49.7 ±0.9	**67.8 ±1.2**
Cora-ML	Raw Graph	72.4 ±0.8	72.7 ±1.4	72.9 ±0.7	71.0 ±1.2	71.1 ±1.0	62.0 ±1.5	**75.6 ±0.4**
	Random Noise	62.3 ±0.6	62.8 ±1.3	61.4 ±1.1	61.3 ±0.5	62.6 ±0.6	57.1 ±2.1	**72.9 ±0.7**
	Non-Targeted Attack	13.2 ±1.4	15.0 ±0.7	11.0 ±1.0	48.9 ±5.3	16.3 ±0.6	18.2 ±2.4	**73.2 ±1.2**
	Targeted Attack	55.7 ±0.7	57.7 ±1.2	54.6 ±0.6	61.2 ±0.9	53.0 ±0.8	55.1 ±1.6	**70.8 ±0.7**
Citeseer	Raw Graph	64.8 ±1.4	65.7 ±1.1	56.6 ±1.2	62.2 ±2.0	61.3 ±2.0	60.6 ±2.0	**71.2 ±1.4**
	Random Noise	57.0 ±1.2	58.7 ±2.1	48.2 ±1.2	61.1 ±2.8	48.3 ±1.6	54.4 ±2.6	**68.8 ±1.5**
	Non-Targeted Attack	26.6 ±2.5	28.8 ±2.7	26.6 ±1.1	57.9 ±2.7	41.7 ±1.6	41.6 ±3.1	**68.0 ±0.4**
	Targeted Attack	43.9 ±1.7	47.6 ±1.2	35.3 ±1.5	52.5 ±2.3	40.5 ±0.7	48.1 ±1.6	**67.2 ±1.3**
Pubmed	Raw Graph	85.9 ±0.1	86.1 ±0.2	85.1 ±0.1	86.0 ±0.1	83.0 ±0.1	86.1 ±0.1	**86.9 ±0.1**
	Random Noise	80.5 ±0.1	81.2 ±0.2	79.7 ±0.1	83.0 ±0.1	82.0 ±0.1	85.1 ±0.2	**86.4 ±0.1**
	Non-Targeted Attack	73.7 ±0.2	73.5 ±0.3	73.8 ±0.3	84.4 ±0.1	83.0 ±0.1	86.0 ± 0.1	**86.3 ±0.1**
	Targeted Attack	76.5 ±0.1	76.8 ±0.2	76.2 ±0.2	82.7 ±0.2	78.1 ±1.3	79.1 ±0.1	**84.3 ±0.2**

graphs. The perturbation rate of non-targeted attack and targeted attack is 0.15. The perturbation rate of random noise is set as 0.3. The setting of label rate follows the description in Sec 2.5.1.1. The average accuracy with standard deviation are reported in Table 2.2. From the table, we make the following observations:

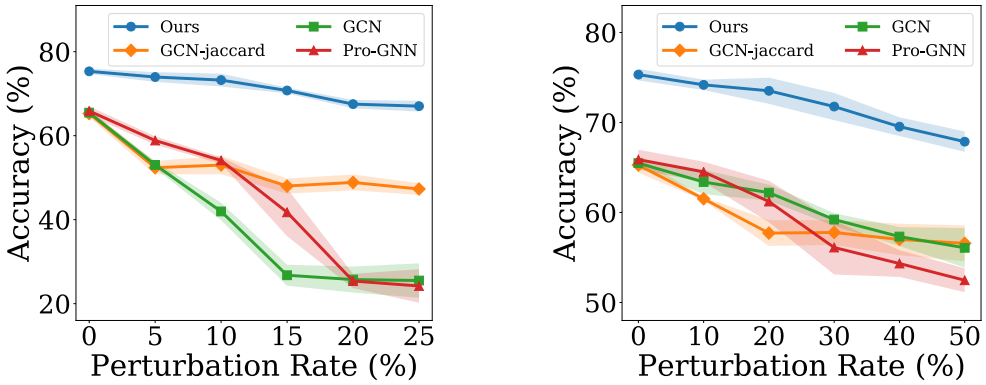
- Even when the GCN is adopted on the raw graph which is not poisoned by attack methods, the GCN hardly performs well with limited labeled nodes. By contrast, our method outperforms GCN and other robust GNNs by a large margin. This indicates the necessity of investigating method to address the challenge of sparsely labeled graphs. What's more, this shows the effectiveness of RS-GNN in dealing with graphs with limited labels.

- The random structural noise further degrades the performance of GCN, but its impact to our proposed method is negligible. This indicates RS-GNN could eliminate the effects of the noisy edges.

- Compared with the preprocessing methods and Pro-GNN, RS-GNN achieves higher accuracy on the sparsely labeled graphs perturbed by attack methods. Because RS-GNN could downweight and eliminate the adversarial edges to defend the adversarial attacks and densify the graph to facilitate the message passing for predictions of unlabeled nodes. By contrast, the baselines only focus on eliminating edges that are likely to be noisy, which will even result in less involvement of unlabeled nodes.

2.5.2.2 Robustness Under Different Ptb Rates

To show that RS-GNN is resistant to different levels of structural noise, we vary the perturbation rate as $\{0\%, 5\%, 10\%, \ldots, 25\%\}$ and compare the performance of RS-GNN



(a) Metattack (b) Random Noise

Figure 2.4: Robustness under different Ptb rates on Cora.

with the most effective baselines. The label rate is fixed as 0.01. Since we have similar observations on other datasets, we only report the average accuracy and standard deviation on Cora in Figure 2.4. From the figure, we make following observations:

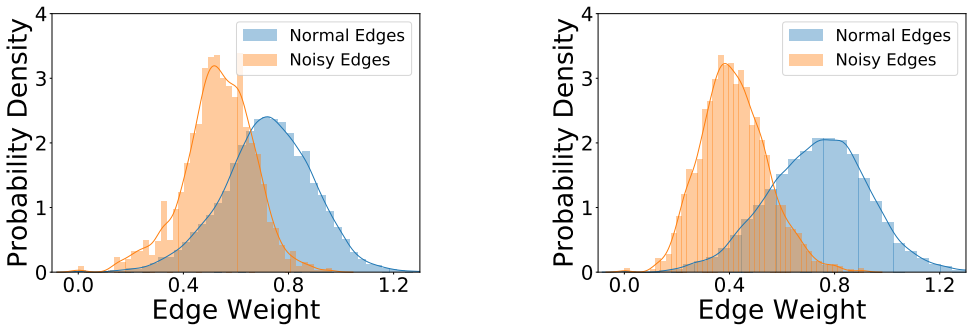
- As the perturbation rate increases, the performance of all the baselines drop significantly, which is as expected. Though the performance of RS-GNN also drops, it is much stable and consistently outperforms the baselines, which shows the robustness of RS-GNN against various levels of attacks and random noise; and
- Compared with GCN, RS-GNN uses GCN as backbone but significantly outperforms GCN, especially when the perturbation rate is large, which shows the effectiveness of eliminating the effects of noisy edges and densifying the graph to benefit the predictions given limited labels.

2.5.3 Analysis of the Generated Graph

To demonstrate that RS-GNN could alleviate negative effects of noisy edges by down-weighting the noisy edges, we investigate the distribution of the learned edge weights $\mathbf{S}_{ij}$ of normal and noisy edges in this subsection. We only present weight distributions of noisy and adversarial on graphs perturbed by random noise with 30% perturbation rate on Cora and CoraML as we have similar observations on other types of noise and datasets. The edge weight distributions are shown in Fig. 2.5. From this figure, we observe: (**i**) The weights of noisy edges are significantly lower than the weights of normal edges, which indicates RS-GNN manages to reduce the effects of noisy edges for robust GNN; and (**ii**) Although most normal edges have higher weights, some of their weights are very low, which implies inherent noise exists in the graph and RS-GNN is able to get rid of such inherent structural noise.

Table 2.3: Number of involved unlabeled nodes

Dataset	Cora	CoraML	Citeseer	Pubmed
Raw Graph	212	447	168	12,430
Generated Graph	1,383	2,161	955	18,555



(a) Cora (b) CoraML

Figure 2.5: Distributions of the weights of normal and noisy edges on the generated graph.

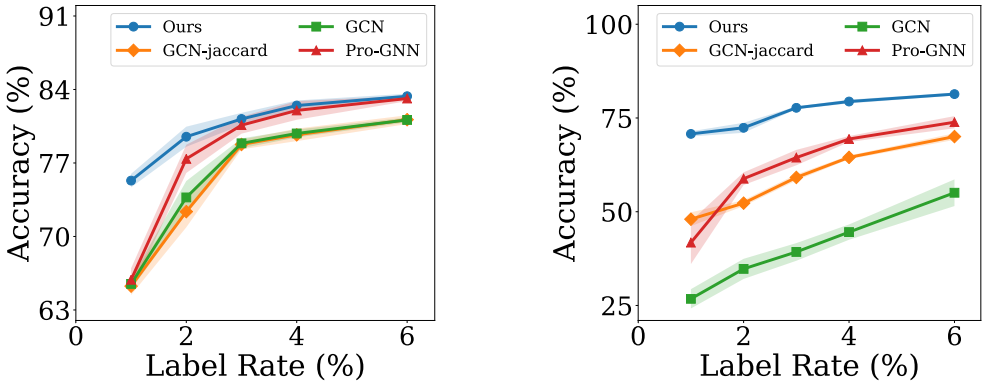
Since RS-GNN aims to densify the graphs to benefit predictions in sparsely labeled graphs, we compare the number of involved unlabeled nodes in raw and generated graphs. More specially, in a two layer GNN, the neighbors of labeled nodes within two hops will participate in the training process. The generated graphs are attained by training RS-GNN on graphs perturbed by random noise. We binarize weighted edges by setting 0.5 as the threshold. The comparisons are given in Table 2.3. We can find that more unlabeled nodes are involved in the training with the generated graphs, which implies that RS-GNN could promote predictions of unlabeled nodes by densifying graphs.

2.5.4 Impacts of Label Rate and Graph Sparsity

To answer **RQ2**, we study the impacts of the number of labeled nodes and sparsity of the graph by varying the label rate and edge rate of the graph. The hyperparameters are selected with the process described in Sec. 2.5.1.4. Each experiment is conducted 5 times and average accuracy with standard deviation are reported.

2.5.4.1 Impacts of Label Rate

We vary label rate as $\{0.01, 0.02, \ldots, 0.06\}$. Experiments are conducted on raw graphs and graphs perturbed by *mettack* to study the effectiveness of RS-GNN under various label rates. The results on Cora are shown in Fig. 2.6. We have similar observations on other datasets. From the figure, we observe:



(a) Raw Graph (b) Metattack with 15% Ptb

Figure 2.6: Performance on Cora with different label rates.

- Generally, as the increase of label rate, the performances of all the methods increase, which is as expected.
- For the raw graph, though RS-GNN consistently outperforms the baselines, as the label rate increases, the improvement of RS-GNN becomes marginal. This is because the raw graph doesn't contain much noise. Thus, as label rate increases to 6%, there are already adequate labels. Since higher label rates would result in more unlabeled nodes involving in the training, the effects of densifying graphs and label smoothness become less significant;
- For the metattack graph, as the label rate increases, RS-GNN still significantly outperforms baselines. That's because the training graph contains a lot of adversarial edges. Though we have enough training labels, the adversarial edges can still contaminate the message passing of GNNs. But RS-GNN can eliminate noisy edges and densify the graph, thus having better results.

2.5.4.2 Impacts of Graph Sparsity

As RS-GNN can generate dense graphs, it should have the ability to handle sparse graphs. Thus, we randomly select $x\%$ edges from the raw graph to build graphs of different sparsity levels. We vary edge rate $x\%$ from 20% to 100% with a step of 40%. The average results of 5 runs on Cora and Citeseer are reported in Table 2.4. From the table, we observe that: (**i**) As the edge rate decreases, the performance of all the methods decrease, which is because message-passing of GNNs becomes ineffective on very sparse graphs; (ii) RS-GNN consistently outperforms the baselines. In particular, when the graph becomes more sparse, the improvement of RS-GNN over the baselines

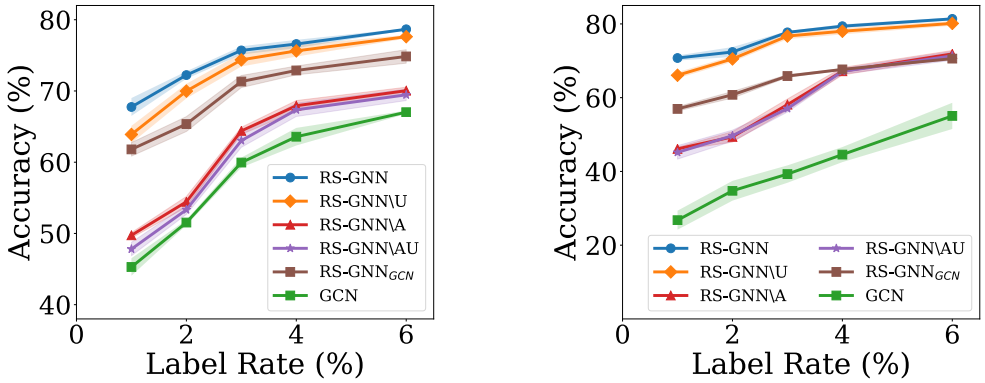
Table 2.4: Accuracy (%) on graphs in different sparsity levels.

Dataset	Edge Rate (%)	GCN	Pro-GNN	RS-GNN
Cora	20	51.9 ±2.0	49.5 ±0.8	**64.7 ±1.7**
	60	62.0 ±0.3	62.5 ±0.5	**68.5 ±1.7**
	100	65.5 ±0.5	65.9 ±1.1	**75.3 ±0.6**
Citeseer	20	54.5 ±1.2	55.2 ±1.6	**63.7 ±2.2**
	60	58.7 ±1.8	58.3 ±2.4	**69.8 ±1.1**
	100	64.8 ±1.4	60.6 ±2.0	**71.2 ±1.4**

becomes larger. For example, the improvement of RS-GNN over GCN on Citeseer is 6.4% when Edge Rate is 100%, and becomes 9.2% when Edge Rate is 20%, which shows the importance of generating edges for densifying the graph and smoothing predictions with the generated graph.

2.5.5 Ablation Study

To answer **RQ3**, we conduct ablation studies to understand the effects of graph densification, graph purification and label smoothness regularization from the generated graph. In our proposed method, the link predictor would add edges that connect similar nodes to enhance the performance on unlabeled nodes. To demonstrate the effects of adding edges with the link predictor, we remove the process of densifying edges and obtain a variant named as RS-GNN\A. RS-GNN also explicitly involves the unlabeled nodes in the training through the label smoothness regularizer. To testify the effectiveness of the regularizing predictions based on the generated graph, we eliminate the label smoothness regularization and get a variant named as RS-GNN\U. Finally, to show our link predictor could help to eliminate the effects of noisy edges, we compare a variant named as RS-GNN\AU which would neither add edges nor apply label smoothness. In RS-GNN\AU, link predictor is only utilized to the purify the noisy graph. We also implement a variant named as RS-GNN$_{GCN}$ which uses GCN as link predictor to show that the noisy edges would largely affects the GNNs in predicting edges. We follow the same process described in Sec 2.5.1.4 to select the hyperparameters. We only show the results on the Cora graph perturbed with *metattack* and random noise, because similar trends are observed on other datasets and attacked methods. From Figure 2.7, we observe that: (**i**) RS-GNN performs much better than RS-GNN\A and RS-GNN\U, which shows that dense graphs and label smoothness regularization from the generated graph could address the problem of lacking labeled nodes. (**ii**) With the increase of label rate, the gap between RS-GNN and RS-GNN\U will be narrowed. This trend is consistent with



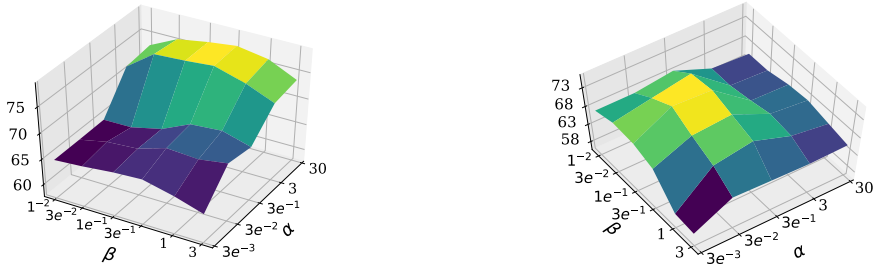
(a) Nettack (b) Metattack with 15% Ptb

Figure 2.7: Ablation studies on Cora with different label rates.

our analysis that higher label rates would involve more unlabeled nodes in the training. (**iii**) RS-GNN$_{GCN}$ performs much worse than RS-GNN, which indicates adversarial edges would impair GCN and result in a poor link predictor.

2.5.6 Parameter Sensitivity Analysis

In this subsection, we explore the sensitivity of hyperparameters α, β for RS-GNN. α and β are the hyperparameters in the final objective function Eq.(2.8), where α controls how well the link predictor reconstructs the noisy graph and β controls the contribution of label smoothness. To investigate the effects of α and β, we vary the values of α as $\{0.003, 0.03, 0.3, 3, 30\}$ and β as $\{0.01, 0.03, 0.1, 0.3, 1, 3\}$ on Cora. The results are shown in Fig. 2.8. In the raw graph, when α is large, the accuracy is stable and high. But if the α is too large in the perturbed graph, the performance would decrease. This difference is due to the noise levels of the raw graph and the perturbed graph. The structural noise in the perturbed graph is severe, faithfully reconstructing the perturbed graph with high α would lead to a poor link predictor. As for the β, a value between 0.03 to 0.3 generally gives good performance, which eases the parameter selection.



(a) Raw Graph (b) Metattack with 15% Ptb

Figure 2.8: Parameter sensitivity analysis on Cora.

Chapter 3
Safety of Graph Neural Network: A Unified Framework for Robustness and Privacy

3.1 Introduction

Graph Neural Networks (GNNs) have shown promising results in modeling graph-structured data for various applications across domains such as social network analysis [3], finance [35], and drug discovery [67]. For graphs, both graph topology and node attributes are important and can provide useful information for downstream tasks. Generally, GNNs adopt a message-passing mechanism to update a node's representation by aggregating information from its neighbors. The learned node representation can preserve both node attributes and local structural information, which facilitates various tasks, especially semi-supervised node classification.

Despite their great success in modeling graphs, GNNs are at risk of *adversarial attacks* and *privacy attacks*. *First*, GNNs are vulnerable to adversarial attacks [37, 59, 68]. An attacker can achieve various attack goals such as controlling predictions of target nodes [59] and degrading the overall performance [37] by deliberately perturbing the graph structure and/or node attributes. For example, Nettack [15] can mislead the target GNN to give wrong predictions on target nodes by poisoning the training graph with small perturbations on graph structure or node attributes. The vulnerability of GNNs largely hinders the adoption of GNNs in safety-critical domains such as finance and healthcare. *Second*, GNNs might leak private training data information under membership inference attacks (MIAs) [11, 69]. The membership inference attack can

detect whether a target sample belongs to the training set. It can effectively distinguish the training samples even with black-box access to the prediction vectors of the target GNNs. This potential membership leakage threatens the privacy of the GNN models trained on sensitive data such as clinical records. For example, an attacker can infer the patient list from GNN-based chronic disease prediction on the patient network [70].

Many efforts [40–42, 60, 71, 72] have been taken to learn robust GNNs against adversarial attacks. For instance, robust aggregation mechanisms [42, 73–75] have been investigated to reduce the negative effects of adversarial perturbations. A group of graph denoising methods [40, 60, 71, 76] is also proposed to remove/down-weight the adversarial edges injected by the attacker. Though they are effective in defending graph adversarial attacks, these methods may fail to preserve the membership privacy, which is also empirically verified in Sec. 3.5.3. For membership privacy-preserving, approaches such as adversarial regularization [77] and differential privacy [78, 79] are proposed for independent and identically distributed (i.i.d) data. However, in semi-supervised node classification, the size of labeled nodes is small and information on labeled nodes can be propagated to their neighbor nodes. These will challenge existing methods that generally process i.i.d data with sufficient labels. Work in membership privacy-preserving on GNNs is still limited [11], let alone robust and membership privacy-preserving GNNs. Therefore, in this paper, we focus on a novel problem of simultaneously defending adversarial attacks and membership privacy attacks with a unified framework.

One promising direction of simultaneously achieving robustness and membership privacy-preserving is to adopt the information bottleneck (IB) principle [80] for node classification of GNNs. The IB principle aims to learn a code that maximally expresses the target task while containing minimal redundant information. In the objective function of IB, apart from the classification loss, a regularization is applied to constrain information irrelevant to the classification task in the bottleneck code. *First*, as IB encourages filtering out information irrelevant to the classification task, the noisy information from adversarial perturbations could be reduced, resulting in robust predictions [81]. *Second*, membership inference attack is feasible because of the difference between training and test samples in posteriors. As analyzed in Sec 3.3.5, the regularization in IB can constrain the mutual information between representations and labels on the training set, which can narrow the gap between training and test sets to avoid membership privacy leakage.

Though promising, there are still two challenges in applying IB principle for robust and membership privacy-preserving predictions on graphs. *First*, in graph-structured data, adversarial perturbations can happen in both node attributes and graph structures.

However, IB for i.i.d data is only designed to extract compressed information from attributes. Simply extending the IB objective function used for i.i.d data to the GNN model may fail to filter out the structural noises. This problem is also empirically verified in Sec. 3.3.6. *Second*, in semi-supervised node classification, the size of labeled nodes is small. Without enough labels, the IB framework would have poor performance on test nodes. In this situation, the gap between labeled nodes and unlabeled test nodes can still be large even with the IB regularization term on labeled nodes, making it ineffective to defend MIA. Our empirical analysis in Sec. 3.3.5 also proves that this challenge is caused by lacking labels.

In an attempt to address these challenges, we propose a novel Robust and Membership Privacy-Preserving Graph Information Bottleneck (RM-GIB). RM-GIB develops a novel graph information bottleneck framework that adopts an attribute bottleneck and a neighbor bottleneck, which can handle the redundant information and adversarial perturbations in both node attributes and graph topology. Moreover, a novel self-supervisor is deployed to benefit the neighbor bottleneck in alleviating noisy neighbors to further improve the robustness. Since membership privacy-preserving with IB requires a large number of labels, RM-GIB collects pseudo labels on unlabeled nodes and combines them with provided labels in the optimization to guarantee membership privacy. In summary, our main contributions are:

- We investigate a new problem of developing a robust and membership privacy-preserving framework for graphs.
- We propose a novel RM-GIB that can alleviate both attribute and structural noises with bottleneck and preserve the membership privacy through incorporating pseudo labels in the optimization.
- Extensive experiments in various real-world datasets demonstrate the effectiveness of our proposed RM-GIB in defending membership inference and adversarial attacks.

3.2 Related Works

3.2.1 Robust Graph Learning

Extensive studies [15, 37, 39, 82] have shown that GNNs are vulnerable to adversarial attacks. Attackers can inject a small number of adversarial perturbations on graph

structures and/or node attributes for their attack goals such as reducing overall performance [37, 68] or controlling predictions of target nodes [15, 82].

Recently, many efforts have been taken to defend against adversarial attacks [40–42, 60, 72], which can be roughly divided into three categories, i.e., adversarial training, robust aggregation, and graph denoising. In adversarial training [83], the GNN model is forced to give similar predictions for a clean sample and its adversarially perturbed version to achieve robustness. The robust aggregation methods [42, 74, 75] design a new message-passing mechanism to restrict the negative effects of adversarial perturbations. Some efforts in adopting Gaussian distributions as hidden representations [42], aggregating the median value of each neighbor embedding dimension [74], and incorporating l_1-based graph smoothing [75]. In graph denoising methods [39, 40, 60, 76, 84], researchers propose various methods to identify and remove/down-weight the adversarial edges injected by the attacker. For example, Wu et al. [39] propose to prune the perturbed edges based on the Jaccard similarity of node features. Pro-GNN [40] learns a clean graph structure by low-rank constraint. RS-GNN [76] introduces a feature similarity weighted edge-reconstruction loss to train the link predictor which can down-weight the noisy edges and predict the missing links. However, these methods do not consider defense against membership inference attacks; On the contrary, the proposed RM-GIB can simultaneously defend against both adversarial attacks and membership inference attacks.

3.2.2 Membership Privacy Preservation

Membership inference attack (MIA) [11, 69] is a type of privacy attack that aims to identify whether a sample belongs to the training set. The main idea of MIA is to learn a binary classifier on patterns such as posteriors that training and test samples exhibit different distributions. The membership leakage will largely threaten the privacy of the model trained on sensitive data such as medical records. Many studies [77, 78, 85–87] have been conducted to defend against the membership inference attack on models trained on i.i.d data. The overfitting on the training samples leads to the difference between training samples and test samples in terms of posteriors and other patterns, which makes the membership inference attack feasible. Hence, a group of MIA defense methods propose to reduce the generalization gap through various regularization techniques. For example, L2 regularization [85], weight normalization [86], and dropout [87, 88] have been investigated for membership privacy preservation. Adversarial regularization [77] is also explored to reduce the posterior distribution difference between training and test samples. Another type of defense [78, 79, 89] is to apply differentially private mechanisms such as

DP-SGD [78]. These mechanisms generally add noise to gradients, model parameters, or outputs to achieve membership privacy guarantee. The above membership inference attack and defense methods are mainly on i.i.d data.

Recently, several seminal works [11, 90, 91] show that GNNs also suffer from MIA. However, defending MIA on graphs is rarely explored [11]. Olatunji et al. [11] propose to inject noise to the posteriors or sample neighbors in the aggregation to protect the membership privacy on node classification. However, it will largely sacrifice the node classification performance to achieve membership privacy. On the contrary, our method combines the proposed novel graph IB and pseudo labels to give accurate and membership privacy-preserving predictions. Moreover, our framework is robust to both MIA and adversarial attacks.

3.2.3 Information Bottleneck

The Information Bottleneck (IB) principle [80] aims to learn latent representations of each sample that maximally express the target task while containing minimal redundant information. Alemi et al. [81] firstly propose the variational information bottleneck (VIB) to introduce the IB principle to deep learning. As IB filters out information irrelevant to the downstream task, it naturally leads to more robust representations, which have been investigated in [81, 92, 93] for i.i.d data. Wu et al. [94] extend the IB principle to learn robust representations on graph-structured data. IB is also applied to extract informative but compressed subgraphs for graph classification [95, 96] and graph explanation [97]. Our method is inherently different from these methods because: (i) we conduct the first attempt to design a novel IB-based framework for membership privacy-preserving on graph neural networks; (ii) we propose a unified framework that can simultaneously defend against adversarial and membership inference attacks.

3.3 Preliminaries

3.3.1 Notations

We use $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ to denote an attributed graph, where $\mathcal{V} = \{v_1, ..., v_N\}$ is the set of nodes, $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ is the set of edges, and $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ is node attribute matrix with $\mathbf{x}_i$ being the node attribute vector of v_i. $\mathbf{A} \in \mathbb{R}^{N \times N}$ denotes the adjacency matrix of $\mathcal{G}$, where $\mathbf{A}_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$ and $\mathbf{A}_{ij} = 0$ otherwise. In this work, we focus on semi-supervised node classification. Only a small set of nodes $\mathcal{V}_L$ are provided with labels

$\mathcal{Y}_L = \{y_1, \ldots, y_l\}$. $\mathcal{V}_U = \mathcal{V} - \mathcal{V}_L$ denotes the unlabeled nodes. Note that the topology and attributes of $\mathcal{G}$ could contain adversarial perturbations or inherent noises.

3.3.2 Membership Inference Attack

Attacker's Goal. The goal of MIA is to identify if a target node was used for training the target model f_T for node classification.

Attacker's Knowledge. We focus on the defense against black-box membership inference attacks as black-box MIA is a practical setting that is widely adopted in existing MIA methods. Specifically, the attacker can have black-box access to the target model f_T to obtain prediction vectors of queried samples. And a shadow graph dataset $\mathcal{G}_S$ from the same distribution of the graph for training f_T is assumed to be available for the attacker. It can be a subgraph or overlap with the training graph $\mathcal{G}$.

General Framework of MIAs. Shadow training [11, 85] is generally used to train the attack model f_A for MIA. In the shadow training, part of nodes in the shadow dataset, i.e., $\mathcal{V}_S^{in} \subset \mathcal{G}_S$, are used to train a shadow model f_S for node classification to mimic the behaviors of the target model f_T. Then, the attacker can construct a dataset by combining the prediction vectors and corresponding ground truth of membership for the attack model training. Specifically, each node $v_i \in \mathcal{V}_S^{in}$ used to train f_S is labeled as 1 (membership) and each node $v_j \in \mathcal{V}_S^{out}$ is labeled as 0 (non-membership), where $\mathcal{V}_S^{out} = \mathcal{V}_S - \mathcal{V}_S^{out}$. Then, the training process of f_A is formally written as follows:

$$\min_{\theta_A} - \sum_{v_i \in \mathcal{V}_S^{in}} \log(f_A(\hat{\mathbf{y}}_i^S)) - \sum_{v_i \in \mathcal{V}_S^{out}} \log(1 - f_A(\hat{\mathbf{y}}_i^S)) \tag{3.1}$$

where f_A denotes the attack model, which is a binary classifier to judge if a node is in the training set or not. θ_A represents the parameters of f_A. $\hat{\mathbf{y}}_i^S$ denotes the prediction vector of node v_i from the shadow model f_S. As machine learning model generally overfits on the labeled samples, it is feasible to have a well-trained attack model. With the trained attack model f_A, the membership of a target node v_t can be inferred by $f_A(\hat{\mathbf{y}}_t^T)$, where $\hat{\mathbf{y}}_t^T$ denotes the prediction vector of v_t given by the target model f_T.

3.3.3 Problem Definition

With the notations in Sec. 3.3.1 and the description of membership inference attacks in Sec. 3.3.2, the problem of learning a robust and membership privacy-preserving GNN can be formally defined as:

Problem 2. *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ with a small set of nodes $\mathcal{V}_L$ labeled, and edge set $\mathcal{E}$ and attributes $\mathbf{X}$ may be poisoned by adversarial perturbations, we aim to learn a robust and membership privacy-preserving GNN $f_{\mathcal{G}} : \mathcal{G} \to \mathcal{Y}$ that maintains high prediction accuracy on the unlabeled set $\mathcal{V}_U$ and is resistant to membership inference attacks.*

3.3.4 Preliminaries of Information Bottleneck

The objective of information bottleneck on i.i.d data is to learn a bottleneck representation $\mathbf{z} = f_\theta(\mathbf{x})$ that (i) maximizes the mutual information with label y; and (ii) filters out information not related to the label y. Various functions can be adopted for f_θ such as neural networks. Formally, the objective function of IB can be written as:

$$\min_{\theta} -I(\mathbf{z}; y) + \beta I(\mathbf{z}; \mathbf{x}), \tag{3.2}$$

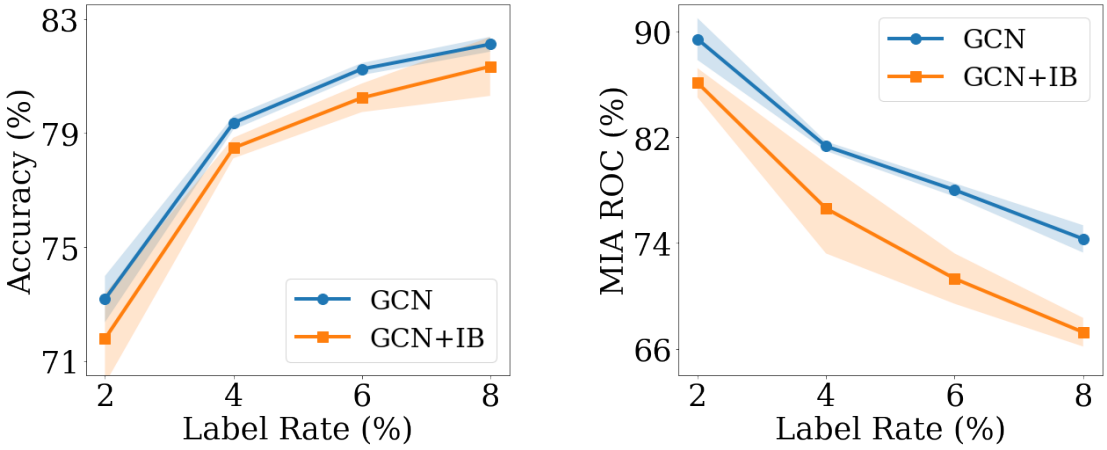
where the former term aims to maximize the mutual information between the bottleneck $\mathbf{z}$ and the label y. The latter term constrains the mutual information between $\mathbf{z}$ and input $\mathbf{x}$ to help filter out the redundant information for the classification task. β is the Lagrangian parameter that balances two terms.

3.3.5 Impacts of IB to Membership Privacy

As shown in Eq.(3.2), IB will constrain $I(\mathbf{z}; \mathbf{x})$ on the training set. Based on mutual information properties and the fact that $\mathbf{z}$ is only obtained from $\mathbf{x}$, we can derive the following equation:

$$\begin{aligned} I(\mathbf{z}; \mathbf{x}) &= I(\mathbf{z}; y) + I(\mathbf{z}; \mathbf{x}|y) - I(\mathbf{z}; y|\mathbf{x}) \\ &= I(\mathbf{z}; y) + I(\mathbf{z}; \mathbf{x}|y) \geq I(\mathbf{z}; y) \end{aligned} \tag{3.3}$$

The details of the derivation can be found in the Appendix A.4. The constraint on $I(\mathbf{z}; \mathbf{x})$ in the IB objective will simultaneously bound the mutual information $I(\mathbf{z}; y)$ on the training set $\mathcal{V}_L$. On the contrary, classifier without using IB will maximize $I(\mathbf{z}; y)$ on the training set $\mathcal{V}_L$ without any constraint. Hence, compared to classifier without using IB regularization, classifier using IB objective is expected to exhibit a smaller gap between the training set and test set. As a result, the member inference attack on classifier trained with IB regularization will be less effective. However, in semi-supervised node classification, only a small portion of nodes are labeled. $I(\mathbf{z}; y)$ will be only maximized on the small set of labeled nodes $\mathcal{V}_L$. Due to the lack of labels, the performance on



(a) Accuracy (b) MIA-F ROC

Figure 3.1: Results of classification and MIA on Cora.

unlabeled nodes could be poor. And $I(\mathbf{z}; y)$ on unlabeled nodes can still be very low. As a result, even with a constraint on $I(\mathbf{z}; y)$, the gap between labeled nodes and unlabeled nodes can still be large when the size of labeled nodes is small, resulting in membership privacy leakage.

To verify the above analysis, we directly apply the objective function of VIB [81] to GCN and denote the model as **GCN+IB**. We investigate the performance of GCN+IB against membership inference attacks by varying the number of training labeled nodes. Specifically, we vary the label rates on Cora [4] by $\{2\%, 4\%, 6\%, 8\%\}$. The ROC score of MIA-F [11] is used to evaluate the ability to preserve membership privacy. Note that a *lower* MIA-F ROC score indicates better performance in preserving membership privacy. The experimental settings of MIA and the hyperparameter tuning follow the description in Sec. 3.5.1. The results are presented in Fig. 3.1, where we can observe that (i) MIA-F ROC of GCN+IB is consistently lower than GCN, which verifies that adopting IB can benefit membership privacy preserving; (ii) membership inference attack can still be very effective on GCN+IB when the label rate is small. With the increase in label rate, the MIA-F ROC score of GCN+IB significantly decreases and the gap between GCN and GCN+IB becomes larger. This empirically shows that abundant labeled samples are required for applying IB to defend MIA effectively.

3.3.6 Impacts of IB to Adversarial Robustness

Intuitively, the negative effects of adversarial perturbations can be reduced with IB, as IB aims to learn representations that only contain information about the label of

Table 3.1: Results (Accuracy(%)+std) on perturbed graphs.

Dataset	Model	Clean	Metattack	Netattack
Cora	GCN	73.2 ±0.8	61.9 ±1.4	54.6 ±0.8
	GCN+IB	73.1 ±0.5	66.3 ±0.3	58.0 ±1.6
Citeseer	GCN	72.1 ±0.2	64.1 ±0.5	62.3 ±0.7
	GCN+IB	71.5 ±0.3	66.8 ±1.1	63.1 ±1.3

the classification task. This has been verified by VIB [81], which incorporates IB to deep neural networks on i.i.d data. However, GNNs generally explicitly combine the information of center nodes and their neighbors to obtain node representations. For example, in each layer of GCN, the center node representations are updated by averaging with neighbor representations. Directly using the IB objective function to a GNN encoder may not be sufficient to bottleneck the minimal sufficient neighbor information. As a result, adversarial perturbations on graph structures can still degrade the performance. To empirically verify this, we compare the performance of GCN+IB with GCN on graphs perturbed by Metattack [37] and Nettack [15]. The experimental settings follow the description in Sec. 3.5.1. The results are shown in Tab. 3.1. We can observe that the GCN model trained with IB objective function achieves better performance on perturbed graphs, which indicates the potential of giving robust node classification with IB. However, compared with the performance on clean graphs, the accuracy of GCN+IB on perturbed graphs is still relatively poor. This empirically verifies that simply applying IB objective function to the GNN model cannot properly eliminate the noisy information from adversarial edges and there is still a large space to improve IB for robust GNN.

3.4 Methodology

As analyzed in Sec. 3.3.5 and Sec. 3.3.6, information bottleneck can benefit both robustness and membership privacy. However, there are two challenges to be addressed for achieving better robust and membership privacy-preserving predictions: (i) how to design a graph information bottleneck framework that can handle adversarial edges? and (ii) how to ensure membership privacy with IB given a small set of labels? To address these challenges, we propose a novel framework RM-GIB, which is illustrated in Fig. 3.2. In RM-GIB, the attribute information and neighbor information are separately bottlenecked. The attribute bottleneck aims to extract node attribute information relevant to the classification. The neighbor bottleneck aims to control the information flow from neighbors to the center

node, and to filter out noisy or useless neighbors for the prediction on the center node. Hence, the influence of adversarial edges can be reduced. Moreover, a novel self-supervisor is proposed to guide the training of the neighbor bottleneck to benefit the noisy neighbor elimination. To address the challenge of lacking plenty of labels for membership privacy-preserving, we propose to obtain pseudo labels and combine them with provided labels in the training phase. Specifically, RM-GIB will be trained with the IB objective function with both labels on labeled nodes and pseudo labels on unlabeled nodes to guarantee membership privacy. More details of the design are presented in the following sections.

3.4.1 Graph Information Bottleneck

In this section, we give the objective of the proposed graph information bottleneck. For graph-structured data, both node attributes and neighbors contain crucial information for node classification. Therefore, for each node v, RM-GIB will extract bottleneck code from both node attributes $\mathbf{x}$ and its neighbor set $\mathcal{N}$, which is shown in Fig. 3.2. More specifically, the bottleneck code is separated into two parts: (i) $\mathbf{z}_x = f_x(\mathbf{x})$, encoding the node attribute information; (ii) $\mathcal{N}_S = f_n(\mathcal{N}, \mathbf{x})$, a subset of v's neighbors that bottleneck the neighborhood information for prediction. Note that $\mathcal{N}$ can be multi-hop neighbors of a node. With the explicit bottleneck mechanisms on both attributes and neighbors, the noisy information from adversarial perturbations can be suppressed. The objective function of the graph information bottleneck is given as:

$$\min_{\theta} -I(\mathbf{z}_x, \mathcal{N}_S; y) + \beta I(\mathbf{z}_x, \mathcal{N}_S; \mathbf{x}, \mathcal{N}) \tag{3.4}$$

where θ denotes the learnable parameters of attribute bottleneck and neighbor bottleneck. However, it is challenging to directly optimize Eq.(3.4) due to the difficulty in computing the mutual information. Thus, we derive tractable variational upper bounds of the two terms in Eq.(3.4).

Following [81], we introduce $q(y|\mathbf{z}_x, \mathcal{N}_S)$ as the parameterized variational approximation of $p(y|\mathbf{z}_x, \mathcal{N}_S)$. Note that $q(y|\mathbf{z}_x, \mathcal{N}_S)$ also can be viewed as a predictor, which can be flexible to various GNNs. Then, the upper bound of $-I(\mathbf{z}_x, \mathcal{N}_S; y)$ can be derived as:

$$\begin{aligned} -I(\mathbf{z}_x, \mathcal{N}_S; y) &\leq \mathbb{E}_{p(\mathbf{z}_x, \mathcal{N}_S, y)}[-\log q(y|\mathbf{z}_x, \mathcal{N}_S)] - H(y) \\ &\leq \mathbb{E}_{p(\mathbf{z}_x, \mathcal{N}_S, y)}[-\log q(y|\mathbf{z}_x, \mathcal{N}_S)] = \mathcal{L}_C \end{aligned} \tag{3.5}$$

Next, we give the upper bound of the second term in Eq.(3.4). Since the attribute

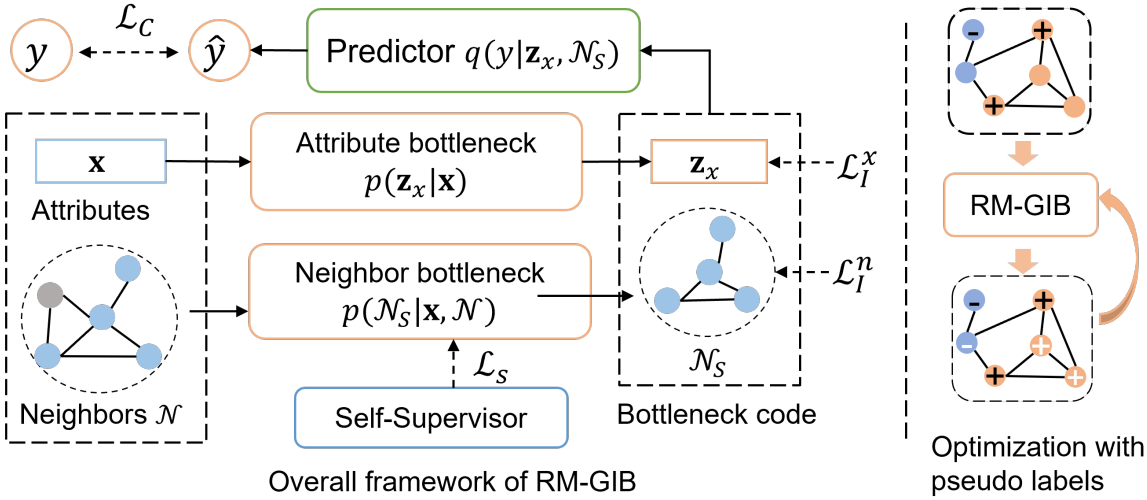


Figure 3.2: The overall framework of our method and the illustration of optimization with pseudo labels.

code $\mathbf{z}_x$ is given by $f_x(\mathbf{x})$ which only takes node attributes as input, we can infer that $p(\mathbf{z}_x|\mathbf{x}, \mathcal{N}) = p(\mathbf{z}_x|\mathbf{x})$. Then, we can get $p(\mathbf{z}_x, \mathcal{N}_S|\mathbf{x}, \mathcal{N}) = p(\mathbf{z}_x|\mathbf{x})p(\mathcal{N}_S|\mathbf{x}, \mathcal{N})$, which indicates $I(\mathbf{z}_x, \mathcal{N}_S|\mathbf{x}, \mathcal{N}) = 0$. As a result, $I(\mathbf{z}_x, \mathcal{N}_S; \mathbf{x}, \mathcal{N})$ can be derived to:

$$\begin{aligned} &I(\mathbf{z}_x, \mathcal{N}_S; \mathbf{x}, \mathcal{N}) = I(\mathbf{z}_x; \mathbf{x}, \mathcal{N}) + I(\mathcal{N}_S; \mathbf{x}, \mathcal{N}|\mathbf{z}_x) \\ =&I(\mathbf{z}_x; \mathbf{x}) + I(\mathcal{N}_S; \mathbf{x}, \mathcal{N}) - I(\mathbf{z}_x; \mathcal{N}_S) + I(\mathbf{z}_x, \mathcal{N}_S|\mathbf{x}, \mathcal{N}) \\ \leq&I(\mathbf{z}_x; \mathbf{x}) + I(\mathcal{N}_S; \mathbf{x}, \mathcal{N}) \end{aligned} \tag{3.6}$$

The term $I(\mathbf{z}_x; \mathbf{x})$ in Eq.(3.6) can be upper bounded as:

$$I(\mathbf{z}_x; \mathbf{x}) \leq \mathbb{E}_{p(\mathbf{x})}[KL(p(\mathbf{z}_x|\mathbf{x})||q(\mathbf{z}_x))] = \mathcal{L}_I^x \tag{3.7}$$

where $q(\mathbf{z}_x)$ is the variational approximation to the marginal $p(\mathbf{z}_x)$ KL denotes the KL divergence. $q(\mathbf{z}_x)$ is flexible to various distributions such as normal distribution. Similarly, let $q(\mathcal{N}_S)$ be the variational approximation to the marginal $p(\mathcal{N}_S)$, the upper bound of $I(\mathcal{N}_S; \mathbf{x}, \mathcal{N})$ is given as:

$$I(\mathcal{N}_S; \mathbf{x}, \mathcal{N}) \leq \mathbb{E}_{p(\mathbf{x}, \mathcal{N})}[KL(p(\mathcal{N}_S|\mathbf{x}, \mathcal{N})||q(\mathcal{N}_S))] = \mathcal{L}_I^n \tag{3.8}$$

With the above derivations, we obtain a variational upper bound of Eq.(3.4) as the objective function of graph information bottleneck:

$$\min_{\theta} \mathcal{L}_C + \beta(\mathcal{L}_I^x + \mathcal{L}_I^n) \tag{3.9}$$

where θ denotes the parameters to be optimized in the graph information bottleneck.

3.4.2 Neural Network Parameterization

With the objective function of graph information bottleneck given above, we specify the neural network parameterization of the attribute bottleneck $p(\mathbf{z}_x|\mathbf{x})$, neighbor bottleneck $p(\mathcal{N}_S|\mathbf{x},\mathcal{N})$ and the predictor $q(y|\mathbf{z}_x,\mathcal{N}_S)$ in this subsection.

3.4.2.1 Attribute Bottleneck

The attribute bottleneck aims to learn a code $\mathbf{z}_x$ that contains minimal and sufficient information for classification from node attributes $\mathbf{x}$. Inspired by [81], a MLP model and reparameterization trick is adopted to model $p(\mathbf{z}_x|\mathbf{x})$ for attribute bottleneck. Specifically, we assume $p(\mathbf{z}_x|\mathbf{x})$ follows Gaussian distribution with the mean and variance as the output of a MLP:

$$p(\mathbf{z}_x|\mathbf{x}) = N(\mathbf{z}_x; \boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I}), \quad \boldsymbol{\mu}, \boldsymbol{\sigma} = f_x(\mathbf{x}) \tag{3.10}$$

where f_x is a MLP which outputs $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ as the mean and standard deviation. $\mathbf{z}_x$ can be sampled by $\mathbf{z}_x = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is sampled from the normal distribution $N(\mathbf{0}, \mathbf{I})$. As $q(\mathbf{z}_x)$ is set as normal distribution, $KL(p(\mathbf{z}_x|\mathbf{x})||q(\mathbf{z}_x))$ can be easily computed for $\mathcal{L}_I^x$.

3.4.2.2 Neighbor Bottleneck

For the neighbor bottleneck, it will extract a subset of neighbors that are useful for the target classification task. With an ideal neighbor bottleneck, noisy neighbors caused by adversarial edges and inherent structural noise can be eliminated. Here, we propose a parameterized neighbor bottleneck to model $p(\mathcal{N}_S|\mathbf{x},\mathcal{N})$. To ease the difficulty of computation, we decompose $p(\mathcal{N}_S|\mathbf{x},\mathcal{N})$ into a multivariate Bernoulli distribution as

$$p(\mathcal{N}_S|\mathbf{x},\mathcal{N}) = \prod\nolimits_{u\in\mathcal{N}_S} p_u \prod\nolimits_{u\in\mathcal{N}\setminus\mathcal{N}_S} (1-p_u) \tag{3.11}$$

where p_u is the probability of $p(u|\mathbf{x},\mathcal{N})$ that follows Bernoulli distribution. To ensure the gradients can be propagated from the classifier to the neighbor bottleneck module during the optimization, Gumbel-Softmax trick [98] with the temperature set as 1 is applied in the sampling phase. Each p_u will be estimated by a MLP which takes the center node attributes $\mathbf{x}$ and the attributes of the neighbor $\mathbf{x}_u$ as input by:

$$p_u = \sigma(\mathbf{h}_u^T\mathbf{h}) \text{ with } \quad \mathbf{h} = f_n(\mathbf{x}), \quad \mathbf{h_u} = f_n(\mathbf{x}_u), \tag{3.12}$$

where σ denotes the sigmoid function, and f_n denotes a MLP model. As for the variational approximation of marginal distribution $q(\mathcal{N}_S)$, we also use a multivariate Bernoulli distribution $q(\mathcal{N}_S) = r^{|\mathcal{N}_S|}(1-r)^{|\mathcal{N}|-|\mathcal{N}_S|}$ where $r \in [0,1]$ is the probability of a predefined Bernoulli distribution. Then, the information loss on neighbor bottleneck $\mathcal{L}_I^n$ in Eq.(3.9) can be computed as:

$$\mathcal{L}_I^n = \mathbb{E}_{p(\mathbf{x},\mathcal{N})}\Big[\sum_{u\in\mathcal{N}} p_u \log\frac{p_u}{r} + (1-p_u)\log\frac{1-p_u}{1-r}\Big]. \tag{3.13}$$

3.4.2.3 Predictor

The predictor $q(y|\mathbf{z}_x, \mathcal{N}_S)$ will give predictions based on the bottleneck code of attributes and the extracted subset of neighbors. To fully utilize the rich information from bottlenecked neighbors, a GNN model is deployed as the predictor in RM-GIB. It is flexible to adopt various GNN models such as GCN [4] and SGC [99]. Note that if $\mathcal{N}_S$ contains neighbors in K hops, a K layer GNN will be adopted in this situation. In addition, to avoid the influence of noises in attributes, we also use the attribute bottleneck code $\mathbf{z}_u$ for each neighbor $\mathbf{z}_u \in \mathcal{N}_S$. Let $\mathbf{A}_S$ denote the local adjacency matrix that connects nodes in $\mathcal{N}_S$ and the center node, the prediction can be formally defined as:

$$\hat{y} = f_c(\mathbf{z}_x, \{\mathbf{z}_u\}_{u\in\mathcal{N}_S}, \mathbf{A}_S), \tag{3.14}$$

where f_c is the GNN-based classifier. As the prediction is given on bottlenecked attributes and neighbors, it can give robust predictions against adversarial perturbations on attributes and graph structures.

3.4.3 Self-supervision for Neighbor Bottleneck

The objective function in Eq.(3.9) will force the neighbor bottleneck to extract minimal sufficient neighbors that achieve good classification performance. However, the training of neighbor bottleneck will only rely on the implicit supervision from the small set of labels in semi-supervised node classification, which may not be sufficient to train a neighbor bottleneck to handle various structural noises. Therefore, we propose a novel self-supervisor to explicitly guide the training of the neighbor bottleneck. The major intuition is that the neighbor nodes with low mutual information with the center node are likely to be the noisy neighbors that are not helpful for the prediction on the center nodes. Hence, we can first estimate the mutual information of each pair of linked nodes. Then,

neighbors with low mutual information scores with the center node can be viewed as negative samples and others as positive samples. Next, we give the details of the mutual information estimation followed by the self-supervision loss on the neighbor bottleneck.

Following [100], a neural network f_M is used to estimate the mutual information between node v and u by:

$$s_{vu} = \sigma(\mathbf{h}_v^{mT}\mathbf{h}_u^m), \quad \mathbf{h}_v^m = f_M(\mathbf{x}_v), \quad \mathbf{h}_u^m = f_M(\mathbf{x}_u), \tag{3.15}$$

where σ is the sigmoid activation function and f_M is an MLP instead of a GNN model to avoid the negative effects of inherent and adversarial structural noises. A larger s_{vu} indicates higher point-wise mutual information between v and u. The mutual information estimator f_M can be trained with the following objective [100]:

$$\min_{\theta_M} -\frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}}\sum_{u\in\mathcal{N}_v}[-\log(s_{vu}) - \mathbb{E}_{n\sim p(v)}\log(1-s_{vn})], \tag{3.16}$$

where θ_M represents parameters of f_M and $\mathcal{N}_v$ is the set of neighbors of v. $p(v)$ is the distribution of sampling negative samples for v, which is set as a uniform distribution. With Eq.(3.16), the mutual information estimator can be trained. Then, we can select the neighbors with a mutual information score lower than the threshold as the negative pairs for neighbor bottleneck. Specifically, for each node v, the negative neighbors can be obtained by:

$$\mathcal{N}_v^- = \{u \in \mathcal{N}_v; s_{vu} < T\}, \tag{3.17}$$

where T is the predefined threshold. With the negative neighbors, the self-supervision on neighbor bottleneck can be given by:

$$\min_{\theta} \mathcal{L}_S = \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}}\Big[\sum_{u\in\mathcal{N}_v^+} -\log(p_u^v) - \sum_{u\in\mathcal{N}_v^-}\log(1-p_u^v)\Big], \tag{3.18}$$

where θ denotes parameters of RM-GIB, $\mathcal{N}_v^+ = \mathcal{N}_v - \mathcal{N}_v^-$ and p_u^v corresponds to the probability value of $p(u|\mathbf{x}_v, \mathcal{N}_v)$ given by neighbor bottleneck thorough Eq.(3.12). With Eq.(3.18), the neighbors who are likely to be noisy will be given lower probability scores in the neighbor bottleneck.

3.4.4 Privacy-Preserving Optimization with Pseudo Labels

As empirically verified in Sec. 3.3.5, a large number of labels are required to preserve membership privacy with IB. Thus, we propose to obtain pseudo labels of unlabeled nodes to enlarge the training set to further improve membership privacy. In particular, the adoption of pseudo labels in RM-GIB can benefit the membership privacy in two aspects: (i) classification loss will also be optimized with unlabeled nodes, which increases the confidence scores of prediction on unlabeled nodes. This will make it more difficult to distinguish the prediction vectors of labeled and unlabeled nodes. (ii) involving a large number of unlabeled nodes in the training can improve the generalization ability of attribute and neighbor bottleneck, which can help narrow the gap between the predictions on training samples and test samples. Moreover, the improvement of bottleneck code can also benefit the classification performance. Next, we give the details of the pseudo label collection and the optimization with pseudo labels.

To obtain pseudo labels that are robust to noises in graphs, we can train RM-GIB with the IB objective function combined with the self-supervision on neighbor bottleneck. Let $\mathcal{L}_C(\mathcal{V}_L, \mathcal{Y}_L)$, $\mathcal{L}_I^x(\mathcal{V}_L)$, and $\mathcal{L}_I^n(\mathcal{V}_L)$ denote the three terms in the IB objective function in Eq.(3.9) on the labeled set $\mathcal{V}_L$. Then, the process of training RM-GIB for pseudo label collection can be formulated as:

$$\min_{\theta} \mathcal{L}_C(\mathcal{V}_L, \mathcal{Y}_L) + \beta\Big(\mathcal{L}_I^x(\mathcal{V}_L) + \mathcal{L}_I^n(\mathcal{V}_L)\Big) + \gamma\mathcal{L}_S, \tag{3.19}$$

where β and γ are hyperparameters to control the contributions of regularization on bottleneck code and the self-supervision on neighbor bottleneck. θ denotes the learnable parameters in RM-GIB. With the RM-GIB trained on Eq.(3.19), we can collect high-quality pseudo labels $\hat{\mathcal{Y}}_U$ of the unlabeled set $\mathcal{V}_U$. Then, we combine pseudo labels $\hat{\mathcal{Y}}_U$ with provided labels $\mathcal{Y}_L$ and retrain RM-GIB for membership privacy-preserving. Let $\mathcal{V}_P = \mathcal{V}_L \cup \mathcal{V}_U$ and $\hat{\mathcal{Y}}_P = \hat{\mathcal{Y}}_U \cup \mathcal{Y}_L$ denote the enlarged labeled node set and labels, the membership privacy-preserving optimization can be formally written as:

$$\min_{\theta} \mathcal{L}_C(\mathcal{V}_P, \hat{\mathcal{Y}}_P) + \beta(\mathcal{L}_I^x(\mathcal{V}_P) + \mathcal{L}_I^n(\mathcal{V}_P)) + \gamma\mathcal{L}_S \tag{3.20}$$

The hyperparameters β and γ are set the same as Eq.(3.19).

Table 3.2: Statistics of datasets.

	Cora	Citeseer	Pubmed	Flickr
#classes	7	6	3	7
#features	1,433	3,703	500	500
#nodes	2,485	2,110	19,717	89,250
#edges	5,069	3,668	44,338	899,756

3.5 Experiments

In this subsection, we evaluate the proposed RM-GIB on various real-world datasets to answer the following research questions:

- **RQ1** Can our proposed RM-GIB preserve the membership privacy in node classification given a small set of labeled nodes?
- **RQ2** Is RM-GIB robust to adversarial perturbations on graphs and can membership privacy be simultaneously guaranteed?
- **RQ3** How does each component of RM-GIB contribute to the robustness and membership privacy?

3.5.1 Experimental Settings

3.5.1.1 Datasets

We conduct experiments on widely used publicly available benchmark datasets, i.e., Cora, Citeseer, Pubmed [4], and Flickr [101]. The key statistics of these datasets can be found in Tab. 3.2. Details of the dataset settings can be found in Appendix A.1

3.5.1.2 Baselines

To evaluate the performance in preserving membership privacy, we compare RM-GIB with the representative graph neural network **GCN** [4] and an existing work of graph information bottleneck **GIB** [94]. We also incorporate a state-of-the-art regularization method, i.e., adversarial regularization [77] (**Adv-Reg**). A differential privacy-based method **DP-SGD** [78] is also compared. Additionally, we compare two recent methods for defending membership inference attacks on GNNs, which are **LBP** [11] and **NSD** [11]. LBP adds noise to the posterior before it is released to end users. NSD randomly chooses

neighbors of the queried node to limit the amount of information used in the target model for membership privacy protection.

To evaluate the robustness of RM-GIB against adversarial attacks on graphs, apart from GCN and GIB, we also compare representative and state-of-the-art robust GNNs. Specifically, we compare two classical preprocessing methods, i.e., **GCN-jaccard** [39] and **GCN-SVD** [60]. Two state-of-the-art robust GNNs are also incorporated in the comparison, which are **Elastic** [75] and **RSGNN** [76]. For more detailed descriptions about the above baselines, please refer to Appendix A.2. To make a fair comparison, the hyperparameters of all baselines are tuned based on the validation set. For our RM-GIB, hyperparameter sensitivity analysis is given in Sec. 3.5.5. More implementation details of RM-GIB can be found in Appendix A.3.

3.5.1.3 Evaluation Protocol

In this subsection, we provide details of experimental settings and metrics to evaluate the performance in defending membership inference attacks and adversarial attacks.

Membership Privacy. We adopt the state-of-the-art MIA on GNNs in [11] for membership privacy-preserving evaluation. The shadow training [11] described in Sec. 3.3.2 is adopted. Here, GCN is applied as the shadow model. The attack setting is set as black-box, i.e., the attacker can only obtain the predictive vectors and cannot access model parameters. As for the shadow dataset, we use two settings:

- **MIA-F**: The attacker has the complete graph used for training along with a small set of labels;
- **MIA-S**: The attacker has a subgraph of the dataset with a small set of labels; In all experiments, we randomly sample 50% nodes as the subgraph that is available for the attacker.

In both settings, *the labeled nodes used in the attack have no overlap with the training set of target model*. The number of labeled nodes used in the attack is the same as the training set. The attack ROC score is used as a metric for membership privacy-preserving evaluation. And a GNN model with a lower attack ROC score indicates better performance in defending MIAs.

Robustness. To evaluate the robustness against adversarial attacks, we evaluate RM-GIB on graphs perturbed by following methods:

Table 3.3: Comparison with baselines in defending membership inference attack on various clean graphs.

Dataset	Metrics	GCN	GCN+PL	Adv-Reg	DP-SGD	GIB	LBP	NSD	RM-GIB
Cora	Accuracy (%) ↑	73.2±0.8	74.7±0.2	75.5 ±0.8	57.9 ±0.2	72.5 ±0.7	69.7 ±0.7	65.4 ±0.3	**78.1 ±0.4**
	MIA-F ROC (%) ↓	90.6 ±0.8	61.6±0.2	70.6 ±0.4	73.8 ±3.3	86.6 ±0.8	71.0 ±1.7	81.8 ±0.8	**57.4 ±0.2**
	MIA-S ROC (%) ↓	88.8 ±0.2	63.8 ±0.8	70.6 ±0.3	75.3 ±1.2	87.3 ±0.7	71.1 ±1.5	81.2 ±0.6	**59.5 ±1.2**
Citeseer	Accuracy (%)↑	72.1 ±0.2	73.1 ±0.2	72.4 ±1.0	57.9 ±0.2	71.0 ±0.2	66.5 ±0.8	65.6 ±0.2	**73.9 ±0.6**
	MIA-F ROC (%)↓	88.5 ±1.8	65.2 ±0.6	60.9 ±0.6	73.8 ±3.3	85.8 ±0.5	66.6 ±0.4	84.4 ±0.1	**55.2 ±0.8**
	MIA-S ROC (%)↓	84.9 ±1.5	65.8 ±0.5	61.2 ±1.1	75.3 ±1.2	80.3 ±0.4	67.3 ±0.7	88.3 ±0.1	**55.9 ±1.7**
Pubmed	Accuracy (%)↑	79.9 ±0.1	79.9 ±0.1	79.4 ±1.1	69.3 ±3.2	78.1 ±0.4	78.3 ±0.1	75.5 ±0.1	**81.4 ±0.2**
	MIA-F ROC (%)↓	75.1 ±0.2	60.8 ±0.2	60.6 ±1.8	56.3 ±1.8	68.5 ±1.6	67.4 ±1.6	68.4 ±0.2	**53.9 ±0.3**
	MIA-S ROC (%)↓	73.4 ±0.1	63.4 ±0.2	62.8 ±2.0	58.3 ±2.1	67.0 ±1.8	65.7 ±2.0	72.1 ±0.1	**57.2 ±0.2**
Flickr	Accuracy (%)↑	**52.5 ±0.2**	51.8 ±0.8	48.2 ±1.8	46.2 ±0.1	45.2 ±2.0	44.6 ±0.5	41.6 ±0.5	52.2 ±0.2
	MIA-F ROC (%)↓	87.9 ±0.7	72.9 ±1.5	64.3 ±3.9	66.5 ±0.7	79.9 ±4.4	67.9 ±0.8	59.0 ±1.5	**58.2 ±0.1**
	MIA-S ROC (%)↓	84.2 ±0.7	69.7 ±1.2	66.4 ±1.2	65.1 ±0.6	76.5 ±0.7	71.3 ±0.9	63.5 ±1.3	**57.6 ±0.3**

- **Mettack** [37]: It aims to reduce the overall performance of the target GNN by perturbing attributes and graph structures. The perturbation rate is set as 0.2 in all experiments.
- **Nettack** [15]: It aims to lead the GNN to misclassify target nodes. Following [76], 15% nodes are randomly selected as target nodes.

As the cited papers do, both Mettack and Nettack can access the whole graph. Similar to MIA, the adversarial attacker is assumed to have nodes with labels that do not overlap with the training set.

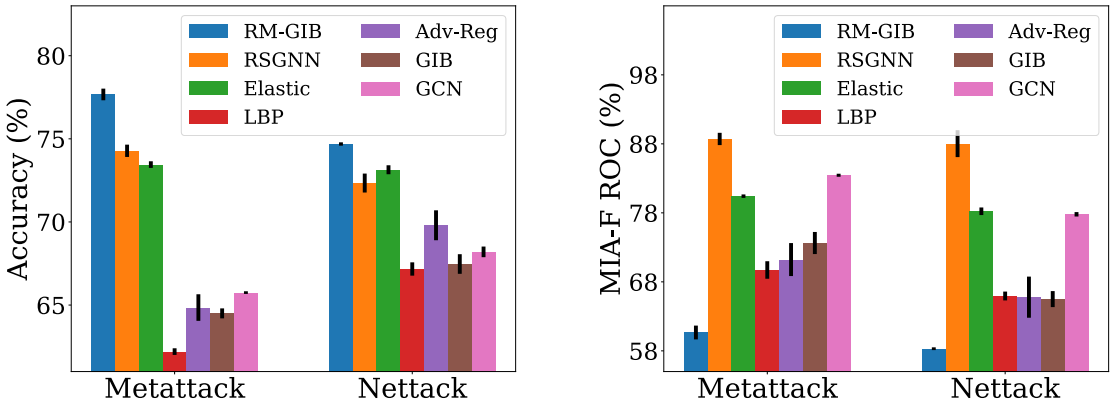
3.5.2 Privacy Preserving on Clean Graphs

To answer **RQ1**, we compare RM-GIB with baselines in defending membership inference attacks on various real-world graphs. The prediction accuracy of each method is reported. As described in Sec. 3.5.1.3, for membership privacy-preserving evaluation, we report the membership attack ROC score on two different settings, i.e., MIA-F and MIA-S, which correspond to the MIA-F ROC and MIA-S ROC in the evaluation metrics. Note that lower attack ROC score indicates better performance in preserving privacy. The results on the default dataset split setting described in Appendix A.1 are reported in Tab. 3.3. Results on different sizes of training set can be found in Appendix A.6. From the Tab. 3.3, we can observe:

- GCN can be easily attacked by membership inference attacks. This demonstrates the necessity of developing membership privacy-preserving methods for node classification on graphs.
- RM-GIB gives significantly lower scores in MIA-F ROC and MIA-S ROC than baselines.

Table 3.4: Comparison with Robust GNNs in node classification (Accuracy(%)±Std) on various adversarially perturbed graphs.

Dataset	Graph	GCN	GIB	GCN-jaccard	GCN-SVD	Elastic	RSGNN	RM-GIB
Cora	Clean	73.2 ±0.8	72.5 ±0.7	68.9 ±0.6	65.1 ±0.6	77.9 ±0.9	74.6 ±1.0	**78.5 ±0.6**
	Metattack	61.9 ±1.4	65.6 ±0.1	64.4 ±0.2	60.5 ±1.3	70.2 ±0.4	65.3 ±2.5	**71.1 ±0.6**
	Nettack	54.6 ±0.8	60.1 ±3.2	58.6 ±0.5	54.8 ±0.7	64.8 ±1.1	**66.9 ±0.4**	65.6 ±1.3
Citeseer	Clean	72.1 ±0.2	71.0 ±0.2	72.2 ±0.1	63.0 ±0.4	73.7 ±0.3	73.7 ±1.3	**73.9 ±0.6**
	Metattack	64.1 ±0.5	66.8 ±0.7	70.5 ±0.1	59.7 ±1.1	71.5 ±0.4	**73.0 ±0.3**	72.1 ±0.9
	Nettack	62.3 ±0.7	63.8 ±1.6	68.9 ±0.2	55.6 ±1.1	68.5 ±0.2	69.0 ±0.9	**69.9 ±0.8**
Pubmed	Clean	79.8 ±0.1	78.1 ±0.4	79.5 ±0.1	75.1 ±0.1	80.6 ±0.2	75.6 ±0.3	**81.4 ±0.1**
	Metattack	67.5 ±0.1	61.5 ±0.4	74.1 ±0.6	74.5 ±0.1	73.5 ±0.2	74.4 ±0.2	**77.3 ±0.1**
	Nettack	68.2 ±0.1	67.5 ±0.6	74.0 ±0.7	67.9 ±0.2	73.2 ±0.3	72.8 ±0.6	**75.0 ±0.2**



(a) Accuracy on Pubmed

(b) MIA-F ROC on Pubmed

Figure 3.3: Results on perturbed Cora and Pubmed graphs.

The attack ROC scores can be even close to 0.5, indicating invalid privacy attacks. This demonstrates the effectiveness of RM-GIB in preserving membership privacy.

- The baseline methods often improve membership privacy with a significant decline in accuracy. By contrast, our RM-GIB can simultaneously maintain high prediction accuracy and preserve membership privacy. This is because baselines generally need to either largely regularize the model or inject strong noises. RM-GIB does not only rely on the regularization in the IB objective function. Pseudo labels are further incorporated in training RM-GIB, which helps to bottleneck redundant information to improve performance and narrow the gap between training and test samples for preserving membership privacy.

3.5.3 Results on Adverarially Perturbed Graphs

To answer **RQ2**, we first compare RM-GIB with Robust GNNs on various perturbed graphs. Then, the performance of membership privacy-preserving on perturbed graphs is also evaluated.

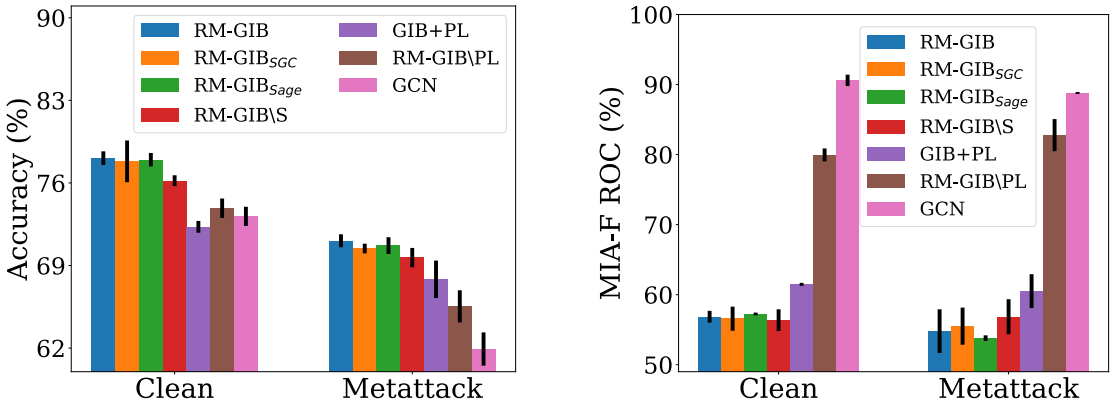
3.5.3.1 Robust Classification

Two types of adversarial attacks, i.e., Metattack and Nettack, are considered for all datasets. Metattack and Nettack will result in out of memory in attacking the large-scale dataset Flickr. Therefore, we only conduct experiments on Cora, Citeseer, and Pubmed. The detailed settings of attacks follow the description in Sec. 3.5.1. The average results and standard deviations of 5 runs are reported in Tab. 3.4, where we can observe:

- Our proposed RM-GIB achieves comparable/better results compared with the state-of-the-art robust GNNs on perturbed graphs, which indicates RM-GIB can mitigate the attribute noises and structural noises with the attribute and neighbor bottleneck.
- Our RM-GIB performs much better than GIB, which also applies IB on graphs to filter out noises in attributes and structures. This is because self-supervision on neighbor bottleneck is adopted in RM-GIB to eliminate noisy neighbors irrelevant to label information. Meanwhile, incorporating pseudo labels of unlabeled nodes also benefits bottleneck code learning.
- On clean graphs, RM-GIB can also consistently outperform baselines including GCN. This is because clean graphs can contain superfluous information and inherent noises, which can be alleviated with the bottleneck in RM-GIB.

3.5.3.2 Membership Privacy Preserving

We also evaluate RM-GIB on perturbed graphs in terms of membership privacy-preserving. The most effective privacy-preserving baselines in Tab. 3.3 and robust GNNs in Tab. 3.4 are selected for comparison. The accuracy and MIA-F ROC on Pubmed and Cora that are perturbed by Metattack and Nettack are shown in Fig. 3.3 and Fig. A.1, respectively. From this figure, we can find that robust GNNs generally fail in preserving privacy. And privacy-preserving baselines give poor classification performance on perturbed graphs. In contrast, RM-GIB can simultaneously preserve membership privacy and give robust predictions in a unified framework.



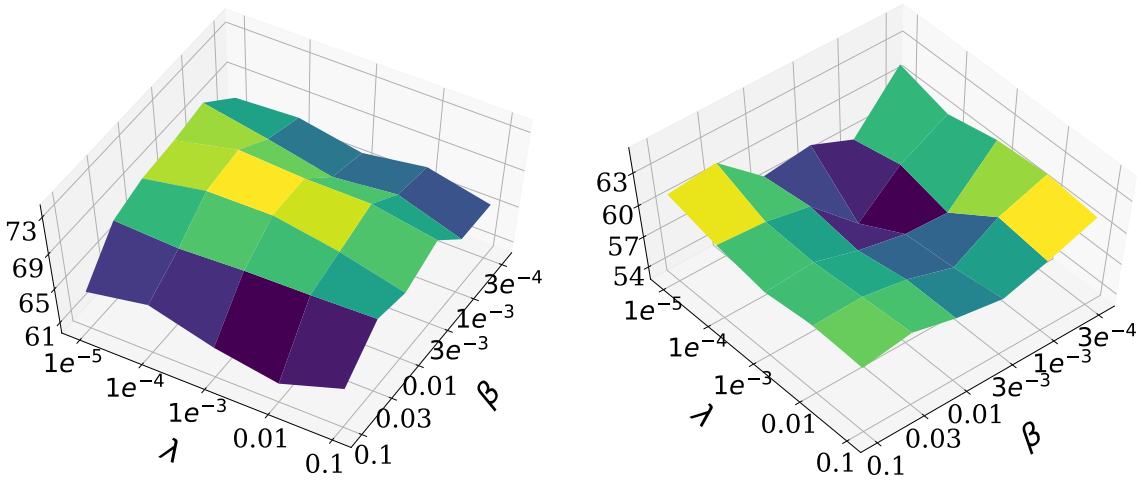
(a) Accuracy (b) MIA-F ROC

Figure 3.4: Ablation Studies on the Cora graph.

3.5.4 Ablation Study

To answer **RQ3**, we conduct an ablation study to understand the effects of the proposed graph information bottleneck, self-supervision on the neighbor bottleneck, and adoption of pseudo labels. To demonstrate the effectiveness of the self-supervision on the neighbor bottleneck, we set γ as 0 when we train RM-GIB and denote this variant as RM-GIB\S. Moreover, to show our RM-GIB can better bottleneck noisy neighbors, a GIB+PL model which trains GIB [94] with pseudo labeling is adopted as a reference. We train a variant RM-GIB\PL that does not incorporate any pseudo labels of unlabeled nodes in the optimization to show the benefits of using pseudo labels in the training. To prove the flexibility of RM-GIB, we train two variants of RM-GIB that use SGC and GraphSage as the predictor, which correspond to RM-GIB$_{SGC}$ and RM-GIB$_{Sage}$. Results of classification and membership privacy-preserving on clean graphs and Metattack perturbed graphs are reported in Fig. 3.4. We only show results on Cora as we have similar observations on other datasets. Concretely, we observe that:

- RM-GIB$_{SGC}$ and RM-GIB$_{Sage}$ achieve comparable results in both robustness and membership privacy-preserving, which shows the flexibility of our proposed RM-GIB.
- The accuracy of RM-GIB\S and GIB+PL is worse than RM-GIB especially on perturbed graphs, which verifies self-supervision on neighbor bottleneck can benefit filtering out noisy neighbors.
- RM-GIB outperforms RM-GIB\PL in both accuracy and membership privacy preserving. This shows the effectiveness of adopting pseudo labels to IB for preserving



(a) Accuracy (b) MIA-F ROC

Figure 3.5: Hyperparameter Analysis on Cora under Metattack

membership privacy. Pseudo labels on unlabeled nodes also improve the quality of the bottleneck code, resulting in better classification performance.

3.5.5 Hyperparameter Sensitivity Analysis

In this subsection, we conduct hyperparameter sensitivity analysis to investigate how β and γ affect the RM-GIB, where β controls the regularization on the bottleneck code and γ controls the contributions of self-supervision on the neighbor bottleneck. More specifically, we vary β and γ as $\{0.0003, 0.0001, 0.003, 0.001, 0.03, 0.1\}$ and $\{0.00001, 0.0001, 0.001, 0.01, 0.1\}$, respectively. We report the accuracy and MIA-F ROC on Cora graph perturbed by Metattack. Similar trends are also observed on other datasets and attack methods. The results are shown in Fig. 3.5. We find that: (i) With the increase of β, the performance of classification and membership privacy-preserving both become better. This is because with very small β, the regularization will be too weak, which can cause overfitting and failure in filtering out noisy information. When β is very large, the strong constraint will lead to poor generalization ability of bottleneck code, resulting in worse performance of both classification and membership privacy; (ii) With the increment of γ, the classification accuracy on perturbed graphs tends to first increase and decrease. And its effects on preserving membership privacy is negligible. When γ is in $[0.0001, 0.001]$, RM-GIB generally gives good classification performance.

Chapter 4
Fairness of Graph Neural Network

4.1 Introduction

Graph neural networks (GNNs) [3, 4, 34, 50] have achieved remarkable performance on various domains such as knowledge graph [47, 102], social media mining [3], nature language processing [4, 103], and recommendation system [104, 105]. Generally, message-passing process is adopted in GNNs [3, 4], where information from neighbors is aggregated for every node in each layer. This process enriches node representations, and preserves both node feature characteristics and topological structures.

Despite the success in modeling graph data, GNNs trained on graphs may inherit the societal bias in data, which limits the adoption of GNNs in many real-world applications. *First*, extensive studies [18–20] have revealed that historical data may include patterns of previous discrimination and societal bias. Machine learning models trained on such data can inherit the bias on sensitive attributes such as ages, genders, skin color, and regions [18, 19], which implies that GNNs could also exhibit the bias. *Second*, the topology of graphs and the message-passing of GNNs could magnify the bias. Generally, in graphs such as social networks, nodes of similar sensitive attributes are more likely to connect to each other than nodes of different sensitive attributes [21, 22]. For example, young people tend to build friendship with people of similar age on the social network [21]. This makes the aggregation of neighbors' features in GNN have similar representations for nodes of similar sensitive information while different representations for nodes of different sensitive features, leading to severe bias in decision making, i.e., the predictions are highly correlated with the sensitive attributes of the nodes. Our preliminary experiments in Sec. 4.3.4 indicate that GNNs have a larger bias due to the adoption of graph structure than models which only use node attributes, which verifies our intuition. The bias would largely limit the wide adoption of GNNs in domains such as ranking of job applicants [23]

and crime rate prediction [24]. Thus, it is important to investigate fair GNNs.

However, developing fair GNNs is a non-trivial task. *First*, to achieve fairness, we need to obtain abundant nodes with known sensitive attributes so that we can either revise the data or regularize the model; whereas people are unwilling to share their sensitive information in the real-world, and resulting in inadequate nodes with sensitive attributes known for fair model learning. For example, only 14% teen users public their complete profiles on Facebook [106]. The lacking of sensitive information challenges many existing work on fair models [19, 20, 25, 26]. *Second*, though extensive efforts have been made to establish fair models by revising features [107–109], disentanglement [20, 26], adversarial debiasing [19, 110] and fairness constraints [111, 112], they are overwhelmingly dedicated to independently and identically distributed (i.i.d) data, which cannot be directly applied on graph data for the absence of simultaneous consideration of the bias from node attributes and graph structures. Recently, [22, 113] aim to learn fair node representations from graphs. These methods merely deal with plain graphs without any node attributes, and focus on fair node representations instead of fair node classifications.

Therefore, in this chapter, we study a novel problem of learning fair graph neural networks with limited sensitive information. In essence, we need to solve two challenges: (i) how to overcome the shortage of sensitive attributes for eliminating discrimination; and (ii) how to ensure the fairness of the GNN classifier. In an attempt to address these challenges, we propose a novel framework named as **FairGNN** for fair node classification. A GNN sensitive attribute estimator is adopted in FairGNN to predict plenty of sensitive attributes with noise for fair classification. Inspired by existing works of fair classification on i.i.d data with adversarial learning [19, 110, 114, 115], we deploy an adversary to ensure the GNN classifier make predictions independent with the estimated sensitive attributes. To further stabilize the training process and performance in fairness, we introduce a fairness constraint to make the predictions invariant with the estimated sensitive attributes. Our main contributions are:

- We study a novel problem of fair graph neutral networks learning with limited sensitive information;
- A new framework, FairGNN, is proposed to settle the shortage of sensitive attributes for adversarial debiasing and fairness constraint by estimating users' sensitive attributes;
- We conduct theoretical analysis showing fairness achieves at the global minimum even with estimated sensitive attributes;

- Extensive experiments on different datasets demonstrate the effectiveness of our methods in eliminating discrimination while keeping high accuracy of GNNs.

The rest of the chapter is organized as follows. In Sec. 4.2, we review related work. In Sec. 4.3, we conduct preliminary analysis to understand the bias issue of GNNs. In Sec. 4.4, we give the details of FairGNN. In Sec. 4.5, we conduct experiments to show the effectiveness of FairGNN.

4.2 Related Work

In this section, we will review related work including graph neural networks and fairness in machine learning.

4.2.1 Graph Neural Networks

Graph neural networks (GNNs), which generalize neural networks for graph structured data, have shown great success for various applications [41, 102–104, 116–118]. Generally, GNNs can be categorized into two categories, i.e., spectral-based [4, 34, 48, 49, 119] and spatial-based [3, 50, 51, 104]. Spectral-based GNNs define graph convolution based on spectral graph theory, which is first explored in [34]. Since then, more spectral-based methods are developed for further improvements and extensions [4, 48, 49, 119]. Graph Convolutional Network (GCN) [4] is a particularly popular method which simplifies the convolutional operation on the graph. Spatial-based graph convolution directly updates the node representation by aggregating its neighborhoods' representations [3, 52, 53, 104]. Graph Attention Network (GAT) [50] introduces the self-attention into the aggregation of spatial graph convolution by assigning higher weights to the more important nodes. Various spatial methods are proposed to solve the scalability issue of GCN [3, 51]. For example, a neighbor sampling method to train GNN with nodes in mini-batch instead of the whole graph is developed in GraphSAGE [3]. Moreover, spatial-based methods have already been successfully deployed to deal with extremely large industrial datasets [104].

The essential idea of GNNs is to propagate the information of nodes through the graph to get better representations. However, people tend to build relationships with those sharing the same sensitive attributes. Then, representations in GNNs are nearly propagated within the subgroup, which highly increases the risk of discrimination towards sensitive attributes. Despite the risk of discrimination in GNNs, there is no existing work

to address this problem. Thus, we study the novel problem of learning fair GNNs to eliminate the potential discrimination.

4.2.2 Fairness in Machine Learning

Many works have been conducted to deal with the bias in the training data to achieve fairness in machine learning [18,19,25,107–109,120]. Based on which stage of the machine learning training process is revised, algorithms could be split into three categories: the pre-processing approaches, the in-processing approaches, and the post-processing approaches. The pre-processing approaches are applied before training machine learning models. They could reduce the bias by modifying the training data through correcting labels [107, 108], revising attributes of data [109, 121], generating non-discriminatory labeled data [122–124], and obtaining fair data representations [19,20,25,26,110,125]. The in-processing approaches are designed to revise the training of the state-of-the-art models. Typically the machine learning models are trained with additional regularization terms or a new objective function. [18, 111, 114, 126]. Finally, the post-processing approaches directly change the predictive labels to ensure fairness [120, 127]. Recently, several works explore the learning of fair graph embeddings for recommendation [22, 113]. Fairwalk [22] modifies the random walk procedure of node2vec [128] to obtain a more diverse network neighborhood representations. The sensitive attributes of all the nodes are required in the sampling procedure of FairWalk. Discriminators is applied to eliminate the sensitive information in the graph embeddings [113]. Similar to Fairwalk, the training process of the discriminators is in need of the sensitive attributes of all the nodes.

Our work is inherently different from existing works: (i) we focus on learning fair GNNs for node classification instead of fair graph embeddings; (ii) we address the problem that only a limited number of nodes are provided with sensitive attributes in practice.

4.3 Preliminaries Analysis

In this section, we first conduct preliminary analysis on real-world datasets to show that GNNs could exhibit more serve bias due to the graph structure and the message-passing. Sequentially, We formally give the problem definition of fair node classification.

Table 4.1: The statistics of datasets.

Dataset	Pokec-z	Pokec-n	NBA
# of nodes	67,797	66,569	403
# of node attributes	59	59	39
# of edges	882,765	729,129	16,570
Size of $\mathcal{V}_L$	500	500	100
Size of $\mathcal{V}_S$	200	200	50
Group ratio	1.84	2.46	2.77
# of inter-group edges	39,804	31,515	4,401
# of intra-group edges	842,961	697,614	12,169

4.3.1 Notations

In this chapter, we consider an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$. As it is mentioned in Sec. 1.1, $\mathcal{V} = \{v_1, ..., v_N\}$ is the set of N nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ is the set of node features. $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix of the graph $\mathcal{G}$, where $\mathbf{A}_{ij} = 1$ if nodes v_i and v_j are connected; otherwise, $\mathbf{A}_{ij} = 0$. In the semi-supervised setting, part of nodes $v \in \mathcal{V}_L$ are provided with labels $y_v \in \mathcal{Y}$, where $\mathcal{V}_L \subseteq \mathcal{V}$ denotes nodes with labels, and $\mathcal{Y}$ is the set of labels. Sensitive attributes of training nodes are required to achieve fairness of machine learning algorithms. In our setting, only a small set of nodes $\mathcal{V}_S \subset \mathcal{V}$ are provided with the sensitive attribute $s \in \{0, 1\}$. The set of provided sensitive attributes is denoted by $\mathcal{S}$.

4.3.2 Datasets

For the purpose of this study, we collect and sample datasets from Pokec and NBA. The details are described as below.

Pokec [**129**]: It is the most popular social network in Slovakia, which is very similar to Facebook and Twitter. This dataset contains anonymized data of the whole social network in 2012. User profiles of Pokec contain gender, age, hobbies, interest, education, working field and etc. The original Pokec dataset contains millions of users. Based on the provinces that users belong to, we sampled two datasets named as: **Pokec-z** and **Pokec-n**. Both Pokec-z and Pokec-n consist of users belonging to two major regions of the corresponding provinces. We treat the region as the sensitive attribute. The classification task is to predict the working field of the users.

NBA: This is extended from a Kaggle dataset [1] containing around 400 NBA basketball players. The performance statistics of players in the 2016-2017 season and other various

[1]https://www.kaggle.com/noahgift/social-power-nba

information e.g., nationality, age, and salary are provided. To obtain the graph that links the NBA players together, we collect the relationships of the NBA basketball players on Twitter with its official crawling API [2]. We binarize the nationality to two categories, i.e., U.S. players and oversea players, which is used as sensitive attribute. The classification task is to predict whether the salary of the player is over median.

For all the datasets, we eliminate nodes without any links with others. We randomly sample labels and sensitive attributes separately to get $\mathcal{V}_L$ and $\mathcal{V}_S$. We randomly sample 25% and 50% of nodes containing both sensitive attributes and labels in Pokec-z, Pokec-n and NBA as validation sets and test sets. Note that the validation sets and test sets have no overlap with $\mathcal{V}_L$ and $\mathcal{V}_S$. The key statistics of the datasets are given in Table 4.1. Apart from the basic statistics, we also report the ratio of the majority and minority group and the number of edges linking the same group and different groups. It is evident from the table that: (i) skew exists in sensitive attributes; (ii) most of relationships are between users who share the same sensitive attribute.

4.3.3 Fairness Evaluation Metrics

In this subsection, we will present two definitions of fairness for the binary label $y \in \{0, 1\}$ and the sensitive attribute $s \in \{0, 1\}$. $\hat{y} \in \{0, 1\}$ denotes the prediction of the classifier η: $\mathbf{x} \to y$.

Definition 1. *(Statistical Parity [18]). Statistical parity requires the predictions to be independent with the sensitive attribute s, i.e., $\hat{y} \perp s$. It could be formally written as:*

$$P(\hat{y}|s = 0) = P(\hat{y}|s = 1). \tag{4.1}$$

Definition 2. *(Equal Opportunity [120]). Equal opportunity requires the probability of an instance in a positive class being assigned to a positive outcome should be equal for both subgroup members. The property of equal opportunity is defined as:*

$$P(\hat{y} = 1|y = 1, s = 0) = P(\hat{y} = 1|y = 1, s = 1). \tag{4.2}$$

The equal opportunity expects the classifier to give equal true positive rates across the subgroups. According to [19, 26], we apply the following metrics to quantitatively evaluate

[2] https://developer.twitter.com/en

Table 4.2: Results of models w/ and w/o utilizing graph.

Dataset	Metrics	MLP	MLP-e	GCN	GAT
Pokec-z	ACC (%)	65.3 ±0.5	68.6 ±0.3	70.2 ±0.1	70.4 ±0.1
	AUC (%)	71.3 ±0.3	74.8 ±0.3	77.2 ±0.1	76.7 ±0.1
	Δ_{SP} (%)	3.8 ±1.3	6.9 ±1.0	9.9 ±1.1	9.1 ±0.9
	Δ_{EO} (%)	2.2 ±0.7	4.0 ±1.5	9.1 ±0.6	8.4 ±0.6
Pokec-n	ACC (%)	63.1 ±0.4	66.3 ±0.6	70.5 ±0.2	70.3 ±0.1
	AUC (%)	68.2 ±0.3	72.4 ±0.6	75.1 ±0.2	75.1 ±0.2
	Δ_{SP} (%)	3.3 ±0.6	8.7 ±1.0	9.6 ±0.9	9.4 ±0.7
	Δ_{EO} (%)	7.1 ±0.9	9.9 ±0.6	12.8 ±1.3	12.0 ±1.5
NBA	ACC (%)	63.6 ±0.9	66.1 ±1.1	71.2 ±0.5	71.9 ±1.1
	AUC (%)	73.5 ±0.3	74.4 ±1.2	78.3 ±0.3	78.2 ±0.6
	Δ_{SP} (%)	6.0 ±1.5	10.9 ±1.9	7.9 ±1.3	10.2 ±2.5
	Δ_{EO} (%)	6.1 ±1.8	8.8 ±3.0	17.8 ±2.6	15.9 ±4.0

statistical parity and equal opportunity:

$$\Delta_{SP} = |P(\hat{y} = 1|s = 0) - P(\hat{y} = 1|s = 1)|, \tag{4.3}$$

$$\Delta_{EO} = |P(\hat{y} = 1|y = 1, s = 0) - P(\hat{y} = 1|y = 1, s = 1)|, \tag{4.4}$$

where the probabilities are evaluated on the test set.

4.3.4 Discrimination in Graph Neural Networks

Various machine learning algorithms such as logistic regression [111], SVM [111], and MLP [110] have been reported to have discrimination. The features of the instances may contain proxy variables of the sensitive attribute. It could result in biased predictions. For GNNs, edges in graph can bring linking bias, i.e., the misrepresentation due to the connections of users [23]. It has been proven that the embeddings of nodes within the connected component will be closer after one aggregation in GCN [44, 130]. Since most of edges are intra-group as Table 4.1 shows, embeddings of nodes sharing the same sensitive attribute will be closer after k-layer information aggregation. As a result, representations of the nodes may exhibit bias. Intuitively, similar discrimination also exists in other GNNs that aggregate information of neighborhoods.

To empirically demonstrate the existence of discrimination in GNNs, we make comparisons between the following models:

- **MLP**: A multi-layer perception model trained on $\mathcal{V}_L$.
- **MLP-e**: A MLP model utilizes graph structure by adding embeddings learned by deepwalk to the features.

- **GCN** [4]: A state-of-the-art spectral graph neural network.
- **GAT** [50]: A spatial graph neural network which utilizes attention to assign higher weights to more important edges.

For each model, we run the experiment 5 times. The classification results and discrimination scores on the test set are reported in Table 4.2. From the table, we observe that (i) both performance of GCN and GAT are much better than MLP, which is as expected because GCN and GAT adopt both node attributes and the graph structure for classification; (ii) Compared with MLP, models utilizing graph structure, i.e., GCN and GAT, perform significantly worse in terms of fairness, which verifies that *bias exists in GNNs and the graph structure could further aggravate the discrimination.*

4.3.5 Problem definition

Our preliminary analysis verifies that GNNs have severe bias issue. Thus, it is important to develop fair GNNs. Following existing work of fair models [19, 26, 121, 122], we focus on the binary class and binary sensitive attribute setting, i.e., both y and s can either be 0 or 1. We leave the extension to multi-class and multi-sensitive attribute setting as a future work. With the notations given in Section 4.3.1, the fair GNN problem is formally defined as:

Problem 3. *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, small labeled node set $\mathcal{V}_L \in \mathcal{V}$ with the corresponding labels in $\mathcal{Y}$, and a small set of nodes $\mathcal{V}_S \in \mathcal{V}$ with corresponding sensitive attributes in $\mathcal{S}$, learn a fair GNN for fair node classification, i.e.,*

$$f(\mathcal{G}, \mathcal{Y}, \mathcal{S}) \rightarrow \hat{\mathcal{Y}} \tag{4.5}$$

where f is the function we aim to learn and $\hat{\mathcal{Y}}$ is the set of predicted labels for unlabeled nodes. $\hat{\mathcal{Y}}$ should maintain high accuracy whilst satisfying the fairness criteria such as statistical parity.

4.4 Methodology

In this section, we give the details of FairGNN. An illustration of the proposed framework is shown in Figure 4.1, which is composed of a GNN classifier $f_{\mathcal{G}}$, a GCN based sensitive attribute estimator f_E and an adversary f_A. The classifier $f_{\mathcal{G}}$ takes $\mathcal{G}$ as input for node

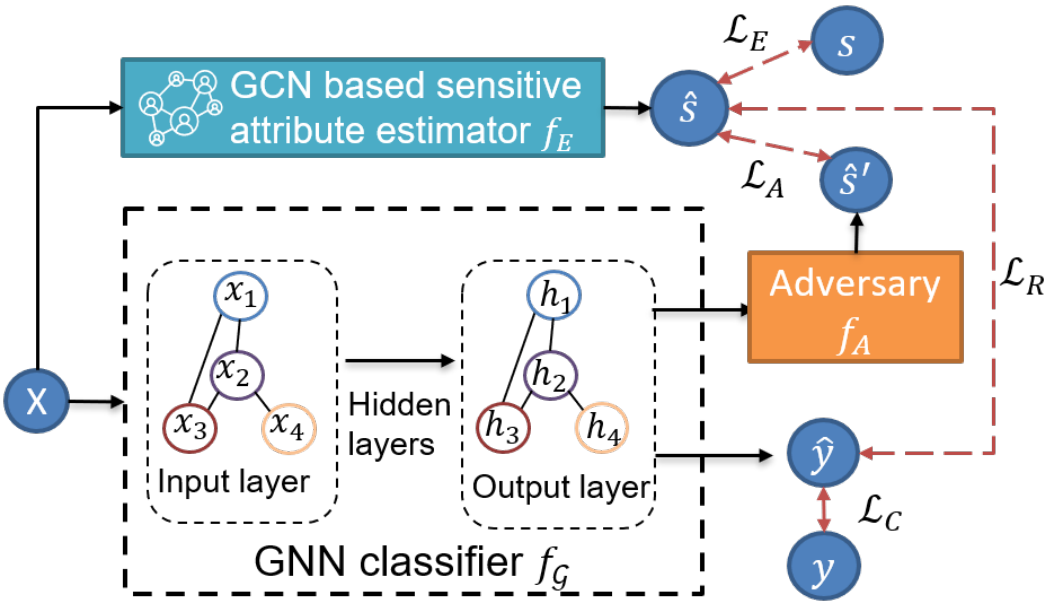


Figure 4.1: The overall framework of FairGNN.

classification. The sensitive attribute estimator f_E is to predict the sensitive attributes for nodes whose sensitive attributes are unknown, which paves us a way to adopt adversarial learning to learn fair node representations and to regularize the predictions of $f_{\mathcal{G}}$. Specifically, the adversary f_A aims to predict the known or estimated sensitive attributes by f_E from the node representation learned by $f_{\mathcal{G}}$; while $f_{\mathcal{G}}$ aims to learn fair node representations that can fool the adversary f_A to make wrong predictions. We theoretically prove that under mild conditions, such minmax game can guarantee that learned representations are fair. In addition to make the representations fair, we directly add a regularizer on the predictions of $f_{\mathcal{G}}$ to guarantee that $f_{\mathcal{G}}$ gives fair predictions. Next, we introduce each component in detail along with theoretical proof.

4.4.1 The GNN Classifier $f_{\mathcal{G}}$

The GNN classifier $f_{\mathcal{G}}$ takes $\mathcal{G}$ as input and predicts node labels. The proposed framework FairGNN is flexible. Any GNNs that follow the structure of Eq.(1.1) can be used such as GCN [4] and GAT [50]. Let $f_{\mathcal{G}}^{(k)}$ denote the operation of aggregating and combining the information of node v and its k-hop neighborhoods through k layers' iterations in GNN classifier $f_{\mathcal{G}}$. For a GNN with K layers, the representation of node v of the final layer could be written as:

$$\mathbf{h}_v = f_{\mathcal{G}}^{(K)}(\mathbf{x}_v, \mathcal{N}_v^{(K)}), \tag{4.6}$$

where $\mathcal{N}_v^{(K)}$ represents the K-hop neighborhoods of v. To get the $\hat{y}_v$, i.e., the prediction of node v, a linear classification layer is applied to $\mathbf{h}_v$ as:

$$\hat{y}_v = \sigma(\mathbf{h}_v \cdot \mathbf{w}), \tag{4.7}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the weights of the linear classification layer and σ is the sigmoid function. The loss function for training $f_{\mathcal{G}}$ is

$$\min_{\theta_{\mathcal{G}}} \mathcal{L}_C = -\frac{1}{|\mathcal{V}_L|} \sum_{v \in \mathcal{V}_L} [y_v \log \hat{y}_v + (1 - y_v) \log (1 - \hat{y}_v)], \tag{4.8}$$

where $|\mathcal{V}_L|$ denotes the size of $\mathcal{V}_L$, $\theta_{f_{\mathcal{G}}}$ represents the parameters of $f_{\mathcal{G}}$ and y_v is the groundtruth label of node v.

4.4.2 Adversarial Debiasing with Estimator f_E

The GNN classifier $f_{\mathcal{G}}$ can make biased predictions because the learned representations of $f_{\mathcal{G}}$ exhibit bias due to the node features, graph structure and aggregation mechanism of GNN. One way to make $f_{\mathcal{G}}$ fair is to eliminate the bias of the final layer representations $\mathbf{h}_v$. Recently, adversarial debiasing has been proven to be effective in alleviating the bias of representations [19, 110, 115, 131]. In the general process of adversarial debiasing, an adversary is used to predict sensitive attributes from the representations of the classifier; while the classifier is trained to learn representations to make the adversary unable to predict the sensitive attributes while keep high accuracy in the classification task. Such process requires *abundant data samples with known sensitive attributes* so that we can judge if the adversary can make accurate predictions or not.

However, in practice people are reluctant to share their sensitive attributes, which leads to small size $\mathcal{V}_S$. Lacking of data with labeled sensitive attributes would result in poor improvement in fairness even with adversarial debiasing. Though we have limited nodes with sensitive attributes, i.e., small $\mathcal{V}_S$, generally, nodes with similar sensitive attributes are more likely connected to each other, which makes it possible to accurately predict the sensitive attributes for nodes in $\mathcal{V} - \mathcal{V}_S$ using the graph $\mathcal{G}$ and $\mathcal{V}_S$. Thus, we deploy a graph convolutional network $f_E : \mathcal{G} \to \mathcal{S}$ to estimate the sensitive attribute of node whose sensitive attribute is unavailable. The large amount of estimated sensitive attributes would greatly benefit the adversarial debiasing. Note that it is important to use two separate GNNs for node label prediction and sensitive attribute prediction because we aim to learn fair representations $\mathbf{h}_v$ for $f_{\mathcal{G}}$, i.e., $\mathbf{h}_v$ does not contain the

sensitive information. The objective function of training f_E is

$$\min_{\theta_E} \mathcal{L}_E = -\frac{1}{|\mathcal{V}_S|} \sum_{v \in \mathcal{V}_S} [s_v \log \hat{s}_v + (1 - s_v) \log (1 - \hat{s}_v)], \tag{4.9}$$

where $\hat{s}_v$ is the predicted sensitive attribute of node $v \in \mathcal{V}_S$ by f_E and θ_E is the set of parameters of f_E.

With f_E, we could get the estimation of the sensitive attributes $\hat{\mathcal{S}}_u$ of the nodes $u \in (\mathcal{V} - \mathcal{V}_S)$. We use $\hat{\mathcal{S}}$ to denote the set of sensitive attributes by combining $\mathcal{S}$ and $\hat{\mathcal{S}}_u$, i.e., $\hat{\mathcal{S}} = \mathcal{S} \cup \hat{\mathcal{S}}_u$. During the training process, for each node $v \in \mathcal{V}$, the adversary f_A tries to predict v's sensitive attribute $\hat{s}_v$ given the representation $\mathbf{h}_v$ as $f_A(\mathbf{h}_v)$; while $f_\mathcal{G}$ aims to learn node representation $\mathbf{h}_v$ that makes the adversary f_A unable to distinguish which sensitive group the node v belong to. This min max game can be written as

$$\begin{aligned} \min_{\theta_\mathcal{G}} \max_{\theta_A} \mathcal{L}_A = &\mathbb{E}_{\mathbf{h} \sim p(\mathbf{h}|\hat{s}=1)}[\log(f_A(\mathbf{h}))] \\ &+ \mathbb{E}_{\mathbf{h} \sim p(\mathbf{h}|\hat{s}=0)}[\log(1 - f_A(\mathbf{h}))], \end{aligned} \tag{4.10}$$

where $\mathbf{h} \sim p(\mathbf{h}|\hat{s} = 1)$ means sampling a node with sensitive attribute as 1 from $\mathcal{G}$. θ_A is the parameters of f_A.

Theoretical Analysis. Since the size of $\mathcal{V}_S$ is small, the estimation of sensitive attributes will introduce nonnegligible noise. The noise of the sensitive attributes may influence the adversarial debiasing. Thus, we conduct theoretical analysis to show that sensitive attributes containing noise could help to achieve statistical parity under mild conditions. Next, we give the details of the proof.

Proposition 1. *The global minimum of Eq.(4.10) is achieved if and only if* $p(\mathbf{h}|\hat{s} = 1) = p(\mathbf{h}|\hat{s} = 0)$*, where* $\hat{s} \in \hat{\mathcal{S}}$ *and* $\mathbf{h}$ *is final layer representation learned by the* K*-layer GNN classifier* $f_\mathcal{G}$*.*

Proof. According to Proposition 1. in [132], the optimal adversary is $f_A^*(\mathbf{h}) = \frac{p(\mathbf{h}|\hat{s}=1)}{p(\mathbf{h}|\hat{s}=1)+p(\mathbf{h}|\hat{s}=0)}$. Then the min max game in Eq.(4.10) could be reformulated as minimizing this function:

$$\begin{aligned} C^s &= \mathbb{E}_{\mathbf{h} \sim p(\mathbf{h}|\hat{s}=1)}\Big[\log \frac{p(\mathbf{h}|\hat{s}=1)}{p(\mathbf{h}|\hat{s}=1) + p(\mathbf{h}|\hat{s}=0)}\Big] \\ &+ \mathbb{E}_{\mathbf{h} \sim p(\mathbf{h}|\hat{s}=0)}\Big[\log \frac{p(\mathbf{h}|\hat{s}=0)}{p(\mathbf{h}|\hat{s}=1) + p(\mathbf{h}|\hat{s}=0)}\Big] \\ &= -\log(4) + 2 \cdot JSD(p(\mathbf{h}|\hat{s}=1)||p(\mathbf{h}|\hat{s}=0). \end{aligned} \tag{4.11}$$

The Jensen-Shannon divergence between two distributions is non-negative, and become

zero if the two distributions are equal. Thus, only if $p(\mathbf{h}|\hat{s}=1) = p(\mathbf{h}|\hat{s}=0)$, the objective function C^s will reach the minimum, which completes our proof. □

Theorem 1. *Let $\hat{y}$ denote the prediction of $f_{\mathcal{G}}$. Suppose:*

1. *The estimated sensitive attribute $\hat{s}$ and $\mathbf{h}$ are independent conditioned on true sensitive attribute s, i.e., $p(\hat{s}, \mathbf{h}|s) = p(\hat{s}|s)p(\mathbf{h}|s)$;*

2. *$p(s=1|\hat{s}=1) \neq p(s=1|\hat{s}=0)$.*

If Eq.(4.10) reaches the global minimum, the GNN classifier $f_{\mathcal{G}}$ will achieve statistical parity, i.e., $p(\hat{y}|s=0) = p(\hat{y}|s=1)$.

Proof. Under the assumption that $p(\hat{s}, \mathbf{h}|s) = p(\hat{s}|s)p(\mathbf{h}|s)$, we could obtain $p(\mathbf{h}|s, \hat{s}) = p(\mathbf{h}|s)$. From Proposition 1, we have $p(\mathbf{h}|\hat{s}=1) = p(\mathbf{h}|\hat{s}=0)$ when the algorithm converges, which is equivalent to $\sum_s p(\mathbf{h}, s|\hat{s}=1) = \sum_s p(\mathbf{h}, s|\hat{s}=0)$. Together with $p(\mathbf{h}|s, \hat{s}) = p(\mathbf{h}|s)$, we arrive at

$$\sum_s p(\mathbf{h}|s)p(s|\hat{s}=1) = \sum_s p(\mathbf{h}|s)p(s|\hat{s}=0) \tag{4.12}$$

Reordering the terms in Eq.(4.12), we can get

$$\begin{aligned} \frac{p(\mathbf{h}|s=1)}{p(\mathbf{h}|s=0)} &= \frac{p(s=0|\hat{s}=1) - p(s=0|\hat{s}=0)}{p(s=1|\hat{s}=0) - p(s=1|\hat{s}=1)} \\ &= \frac{(1-p(s=1|\hat{s}=1)) - (1-p(s=1|\hat{s}=0))}{p(s=1|\hat{s}=0) - p(s=1|\hat{s}=1)} \\ &= 1 \end{aligned} \tag{4.13}$$

Eq.(4.13) shows that at the global minimum $p(\mathbf{h}|s=1) = p(\mathbf{h}|s=1)$ under the assumption $p(s=1|\hat{s}=1) \neq p(s=1|\hat{s}=0)$. Since $\hat{y} = \sigma(\mathbf{h} \cdot \mathbf{w})$, we could get $p(\hat{y}|s=1) = p(\hat{y}|s=0)$. Thus, the statistical parity is achieved when Eq.(4.10) converges. □

In our proof, two assumptions are made. For the first assumption, since we use f_E to predict the sensitive attributes $\hat{s}$ and $f_{\mathcal{G}}$ to get the latent representation $\mathbf{h}$, and f_E and $f_{\mathcal{G}}$ doesn't share any parameters, it is generally true that $\hat{s}$ is independent with the representation $\mathbf{h}$, i.e., $p(\hat{s}, \mathbf{h}|s) = p(\hat{s}|s)p(\mathbf{h}|s)$. As for the second assumption, it will be satisfied when we have a reasonable estimator f_E, i.e., f_E doesn't give random predictions.

4.4.3 Covariance Constraint

The instability of the training process of adversarial learning is well known [133]. In adversarial debiasing, failure to coverage may result in a classifier with discrimination. To alleviate this issue, we add a covariance constraint [111, 112] on the output of $f_{\mathcal{G}}$ to help the model achieve fairness. The covariance constraint has been explored in [111, 112] by minimizing the absolute covariance between users' sensitive attributes and the signed distance from the users' features to the decision boundary for fair linear classifiers. In our problem, only a small portion of users' sensitive attributes are known and the decision boundary of GNN is hard to obtain. Thus, we propose to minimize the absolute covariance between the noisy sensitive attribute $\hat{s} \in \hat{\mathcal{S}}$ and prediction $\hat{y}$ as

$$\mathcal{L}_R = |\mathrm{Cov}(\hat{s}, \hat{y})| = |\mathbb{E}[(\hat{s} - \mathbb{E}(\hat{s}))(\hat{y} - \mathbb{E}(\hat{y}))]|, \tag{4.14}$$

where $|\cdot|$ indicates the absolute value.

Theoretical Analysis. Since $\mathcal{L}_R$ is the absolute value of covariance between $\hat{y}$ and $\hat{s}$, $\mathcal{L}_R = 0$, i.e., the global minimum of $\mathcal{L}_R$, is the prerequisite that $\hat{y}$ and $\hat{s}$ are independent. Thus, we will show that $\mathcal{L}_R = 0$ is the prerequisite of the statistical parity under mild assumption with the following theorem.

Theorem 2. *Suppose that $p(\hat{s}, \mathbf{h}|s) = p(\hat{s}|s)p(\mathbf{h}|s)$, when $f_{\mathcal{G}}$ satisfies statistical parity, i.e. $\hat{y} \perp s$, $\hat{y}$ is independent with $\hat{s}$ and $\mathcal{L}_R = 0$.*

Proof. Through $p(\hat{s}, \mathbf{h}|s) = p(\hat{s}|s)p(\mathbf{h}|s)$, we could get $p(\mathbf{h}|s, \hat{s}) = p(\mathbf{h}|s)$. Then, $p(\hat{y}|s, \hat{s}) = p(\hat{y}|s)$ could be derived. When $\hat{y} \perp s$, the distribution $p(\hat{y}, \hat{s})$ would be:

$$p(\hat{y}, \hat{s}) = \sum_s p(\hat{y}|s)p(\hat{s}, s) = \sum_s p(\hat{y})p(\hat{s}, s) = p(\hat{y})p(\hat{s}). \tag{4.15}$$

Thus, $\hat{y}$ is independent with $\hat{s}$ when the statistical parity is achieved. Then, we can get $\mathcal{L}_R = |\mathrm{Cov}(\hat{s}, \hat{y})| = |\mathbb{E}(\hat{s}, \hat{y}) - \mathbb{E}(\hat{s})\mathbb{E}(\hat{y})| = 0$. □

In the proof, we use the first assumption in Theorem 2, which is generally valid as discussed previously.

4.4.4 Final Objective Function of FairGNN

We now have $f_{\mathcal{G}}$ for label prediction, f_E for sensitive attribute estimation, f_A with adversarial debiasing to force the node representations learned by $f_{\mathcal{G}}$ are fair, and

Algorithm 1 Training Algorithm of FairGNN.

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, $\mathcal{Y}$, $\mathcal{S}$, α and β.
Output: $f_{\mathcal{G}}$, f_A, and f_E
1: Initialize f_E by optimizing Eq.(4.9) w.r.t θ_E
2: **repeat**
3: Obtain the estimated sensitive attributes with f_E
4: Optimize the GNN classifier parameters $\theta_{\mathcal{G}}$, the adversary parameters θ_A, and the estimator parameters θ_E by Eq.(4.16).
5: **until** convergence
6: **return** $f_{\mathcal{G}}$, f_A, and f_E

covariance constraint to further ensure that the prediction of $f_{\mathcal{G}}$ is fair. Combining all these together, the final objective function could be formulated as:

$$\min_{\theta_{\mathcal{G}}, \theta_E} \max_{\theta_A} \mathcal{L}_C + \mathcal{L}_E + \alpha \mathcal{L}_R - \beta \mathcal{L}_A, \tag{4.16}$$

where $\theta_{\mathcal{G}}$, θ_E, and θ_A are the parameters of classifier, estimator, and adversary, respectively. α and β are scalars to control the contributions of the covariance constraint and adversarial debiasing.

4.4.5 An Training Algorithm of FairGNN

The training algorithm of FairGNN is presented in Algorithm 1. Specially, we first pretrain f_E to ensure it meets the second assumption in Theorem 1. Sequentially, we optimize the whole model with Eq.(4.16) through the ADAM optimizer [134]. In the training process, we replace the hard labels in $\mathcal{L}_A$ with soft labels, i.e., the probability produced by f_E, to stabilize the adversarial learning [135].

4.5 Experiments

In this section, we conduct experiments to show the effectiveness of FairGNN for fair node classification. In particular, we aim to answer the following questions:

- **RQ1** Can the proposed FairGNN reduce the bias of GNNs while maintaining high accuracy?
- **RQ2** How do the sensitive attribute estimator, adversarial loss, and covariance constraint affect FairGNN?

- **RQ3** Is FairGNN effective when different amount of sensitive attributes or labels are provided in the training set?

We use the same datasets introduced in Sec. 4.3.2 for all the experiments. Next, we will begin by introducing compared methods.

4.5.1 Compared Methods

We compare our proposed framework with GCN, GAT, and the following representative and state-of-the-art methods for fair classification and fair graph embedding learning:

- **ALFR** [110]: This is a pre-processing method. A discriminator is applied to remove the sensitive information in the representations produced by a MLP-based autoencoder. Then, linear classifier is trained on the debiased representations.
- **ALFR-e**: To utilize the graph structure information, ALFR-e concatenates the graph embeddings learned by deepwalk [136] with the user features in the ALFR.
- **Debias** [114]: This is an in-processing fair classification method. It directly applies an discriminator on the estimated probability of classifier $\eta : \mathbf{x} \to \mathbb{R}$. It would make the probability distribution $p(\eta(\mathbf{x})|s = 0)$ closer to $p(\eta(\mathbf{x})|s = 1)$.
- **Debias-e**: Similar to the ALFR-e, we also add the deepwalk embeddings to the features used in Debias.
- **FCGE** [113]: FCGE is proposed to learn fair node embeddings in graph without node features through edge prediction. The sensitive information in the embeddings is filtered by discriminators.

ALFR and ALFR-e are trained with features of all the users $\mathcal{V}$, labels of $\mathcal{V}_L$, and the sensitive attributes of $\mathcal{V}_S$ for fair classification. Debis and Debias-e require the sensitive attributes of labeled nodes, which is on contrary with our setting that $\mathcal{V}_L$ could have no overlap with $\mathcal{V}_S$. Thus, we use the estimated labels of $\mathcal{V}_S$, features of $\mathcal{V}_L$, and labels of $\mathcal{V}_L$ to train Debias and Debias-e. FCGE utilizes $\mathcal{G}$, labels of $\mathcal{V}_L$, and sensitive attributes of $\mathcal{V}_S$.

For FairGNN, we deploy a one hidden layer GCN for f_E. The hidden dimension is set as 128. We use a linear classifier for f_A. To verify that our framework is useful for various GNNs, we adopt both GCN and GAT as the backbone of the FairGNN classifier $f_{\mathcal{G}}$, which are named as **FairGCN** and **FairGAT**. In FairGCN, the GCN classifier contains

one hidden layer with dimension 128. The GAT classifier in FairGAT also contains two layers in total. We set the number of heads as 1. The dimensions of the GAT classifiers' hidden layer for Pokec-z, Pokec-n and NBA are 64, 64 and 32, respectively.

4.5.2 Fair Classification on Graph

To answer **RQ1**, we evaluate our proposed FairGNN in terms of fairness and classification performance. Δ_{SP} and Δ_{EO} are used to show the discrimination level, which are introduced in Section 4.3.3. The smaller Δ_{SP} and Δ_{EO} are, the more fair the classifier is. Accuracy (ACC) and ROC AUC score are used to evaluate the classification performance. For all the models, we tune the hyperparameters on the training set via cross validation. For FairGCN, we set α to 100 and β to 1. For FairGAT, α is 2 and β is 0.1. More details about hyperparameter selection will be discussed in Sec 4.5.5. All the experiments are conducted 5 times. The mean and standard deviations for all the models on the three datasets are reported in Table 4.3. From the table, we make the following observations:

- Compared with GCN and GAT, the general fair classification methods and graph embeddings learning method show poor performance in classification even with the help of graph information, while FairGCN and FairGAT perform very close to the based GNNs. This suggests the necessity of investigating fair classification algorithms on GNNs for accurate predictions;
- Under the condition of limited sensitive information, baselines show obvious bias and the ones utilizing graph information are even worse. On the contrary, our proposed models obtain Δ_{SP} and Δ_{EO} that are close to 0, which indicates that the discrimination is basically eliminated; and
- FairGAT is slightly better than FairGCN in Fairness. This is reasonable because the learnable edge coefficients in GAT could be helpful to reduce the weights of the edges that bring bias.

These observations demonstrate the effectiveness of our proposed framework in making fair and accurate predictions.

4.5.3 Ablation Study

To answer **RQ2**, we conduct ablation studies to understand the impacts of f_E, adversarial loss, and covariance constraint.

Table 4.3: The comparisons of our proposed methods with the baselines.

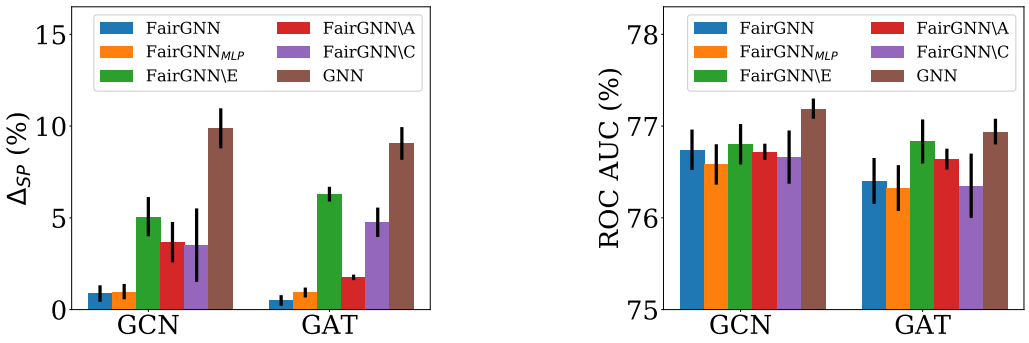
Dataset	Metrics	GCN	GAT	ALFR	ALFR-e	Debias	Debias-e	FCGE	FairGCN	FairGAT
Pokec-z	ACC (%)	70.2 ±0.1	70.4 ±0.1	65.4 ±0.3	68.0 ±0.6	65.2 ±0.7	67.5 ±0.7	65.9 ±0.2	**70.0 ±0.3**	**70.1 ±0.1**
	AUC (%)	77.2 ±0.1	76.7 ±0.1	71.3 ±0.3	74.0 ±0.7	71.4 ±0.6	74.2 ±0.7	71.0 ±0.2	**76.7 ±0.2**	**76.5 ±0.2**
	Δ_{SP} (%)	9.9 ±1.1	9.1 ±0.9	2.8 ±0.5	5.8 ±0.4	1.9 ±0.6	4.7 ±1.0	3.1 ±0.5	**0.9 ±0.5**	**0.5 ±0.3**
	Δ_{EO} (%)	9.1 ±0.6	8.4 ±0.6	1.1 ±0.4	2.8 ±0.8	1.9 ±0.4	3.0 ±1.4	1.7 ±0.6	**1.7 ±0.2**	**0.8 ±0.3**
Pokec-n	ACC (%)	70.5 ±0.2	70.3 ±0.1	63.1 ±0.6	66.2 ±0.5	62.6 ±0.9	65.6 ±0.8	64.8 ±0.5	**70.1 ±0.2**	**70.0 ±0.2**
	AUC (%)	75.1 ±0.2	75.1 ±0.2	67.7 ±0.5	71.9 ±0.3	67.9 ±0.7	71.7 ±0.7	69.5 ±0.4	**74.9 ±0.4**	**74.9 ±0.4**
	Δ_{SP} (%)	9.6 ±0.9	9.4 ±0.7	3.05 ±0.5	4.1 ±0.5	2.4 ±0.7	3.6 ±0.2	4.1 ±0.8	**0.8 ±0.2**	**0.6 ±0.3**
	Δ_{EO} (%)	12.8 ±1.3	12.0 ±1.5	3.9 ±0.6	4.6 ±1.6	2.6 ±1.0	4.4 ±1.2	5.5 ±0.9	**1.1 ±0.5**	**0.8 ±0.2**
NBA	ACC (%)	71.2 ±0.5	71.9 ±1.1	64.3 ±1.3	66.0 ±0.4	63.1 ±1.1	65.6 ±2.4	66.0 ±1.5	**71.1 ±1.0**	**71.5 ±0.8**
	AUC (%)	78.3 ±0.3	78.2 ±0.6	71.5 ±0.3	72.9 ±1.0	71.3 ±0.7	72.9 ±1.2	73.6 ±1.5	**77.0 ±0.3**	**77.5 ±0.7**
	Δ_{SP} (%)	7.9 ±1.3	10.2 ±2.5	2.3 ±0.9	4.7 ±1.8	2.5 ±1.5	5.3 ±0.9	2.9 ±1.0	**1.0 ±0.5**	**0.7 ±0.5**
	Δ_{EO} (%)	17.8 ±2.6	15.9 ±4.0	3.2 ±1.5	4.7 ±1.7	3.1 ±1.9	3.1 ±1.3	3.0 ±1.2	**1.2 ±0.4**	**0.7 ±0.3**

4.5.3.1 Impact of f_E

In our proposed framework, a GCN estimator is deployed to predict sensitive attributes for adversarial debiasing. To show the importance of the GCN estimator, we analyze it from two aspects. Firstly, to demonstrate the effectiveness of the noisy sensitive attributes, we eliminate the estimator and only use the provided sensitive attributes $\mathcal{S}$ to get a variant denoted as FairGNN\E. Secondly, to investigate how a weaker estimator would influence the fair classification, we train a variant FairGNN$_{MLP}$ by using MLP as the estimator. Hyperparameters of these variants are determined by cross validation with gird search. Specifically, we vary α and β among $\{0.0001, 0.001, 0.1, 1\}$ and $\{1, 2, 5, 10, 20, 50, 100\}$, respectively. For each variant, the experiments are conducted 5 times. The average performance of fairness in terms of Δ_{SP} and node classificaiton in terms of AUC on Pockec-z are presented in Fig. 4.2(a) and (b), respectively. We only show the results on Pockec-z as we have similar observations on the other datasets. From the figures, we make the following observations:

- The Δ_{SP} score of FairGNN\E is much larger than that of FairGNN. which is because the provided sensitive attributes are inadequate. This shows that f_E plays an important role in FairGNN; and

- The performance of sensitive attribute prediction in terms of AUC for MLP estimator is 0.69, which is much lower than that of GCN estimator, which is 0.80. Though FairGNN$_{MLP}$ adopts a much weaker estimator than FairGNN, the performance in terms of fairness is slightly worse than FairGNN. This aligns with our theoretical analysis that f_E doesn't need to be very accurate. However, the marginal differences still indicate that too much noise in sensitive attributes may still slightly affect the fairness.



(a) Δ_{SP} (b) ROC AUC

Figure 4.2: Comparisons between FairGNN and its variants.

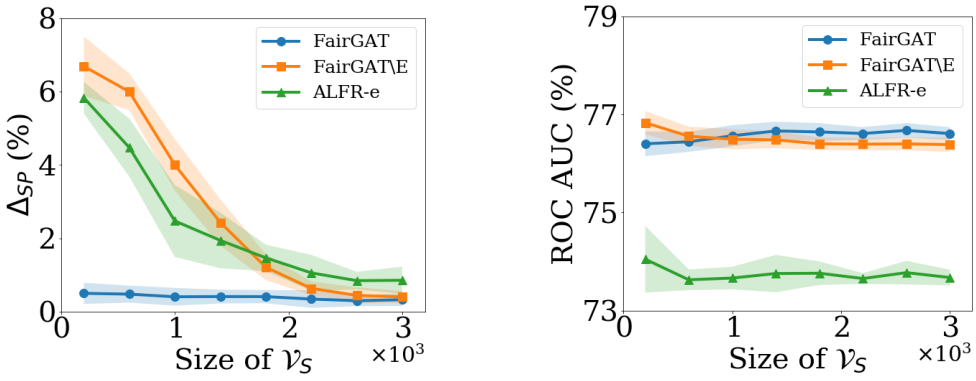
4.5.3.2 Impacts of the Adversarial debiasing and Covariance Constraint

To demonstrate the effects of the adversarial loss and covariance constraint, we train two variants of FairGNN, i.e., FairGNN\A and FairGNN\C, where FairGNN\A means FairGNN without the adversarial loss, and FairGNN\C means FiarGNN without covariance constraint. Similarly, for each variant, we run the experiment 5 times on Pokec-z and the average performances are shown in Figure 4.2. From the figure, we observe:

- The Δ_{SP} scores for both FairGNN\C and FairGNN\A are much smaller than that of GNNs in Figure 4.2, which shows that both covariance constraint and adversarial debiasing can improve fairness; and
- The Δ_{SP} scores for both FairGNN\C and FairGNN\A are much larger than that of FairGNN, which implies that using both covariance constraint and adversarial debiasing can achieve better fairness. This is because they regularize the GNN from two different perspectives, i.e., adversarial debiasing regularizes on the node representations while covariance cosntraint is directly on the predictions for fair classification.

4.5.4 Impacts of Sizes of $\mathcal{V}_S$ and $\mathcal{V}_L$

To answer **RQ3**, we study the impacts of the sizes of $\mathcal{V}_S$ and $\mathcal{V}_L$ on FairGAT. We set $\alpha = 0.1$ and $\beta = 2$ based on cross validation. We vary $|\mathcal{V}_S|$ as from 200 to 3000 with a step of 400. Each experiment is conducted 5 times and the average results on Pokec-z with comparison to FairGAT\E and ALFR-e are shown in Fig. 4.3. From the figure, we observe that: (i) Generally, both FairGAT\E and ALFR-e have high discrimination scores when $|\mathcal{V}_S|$ is small. They need plenty of data with sensitive attributes to become



(a) Δ_{SP} (b) ROC AUC

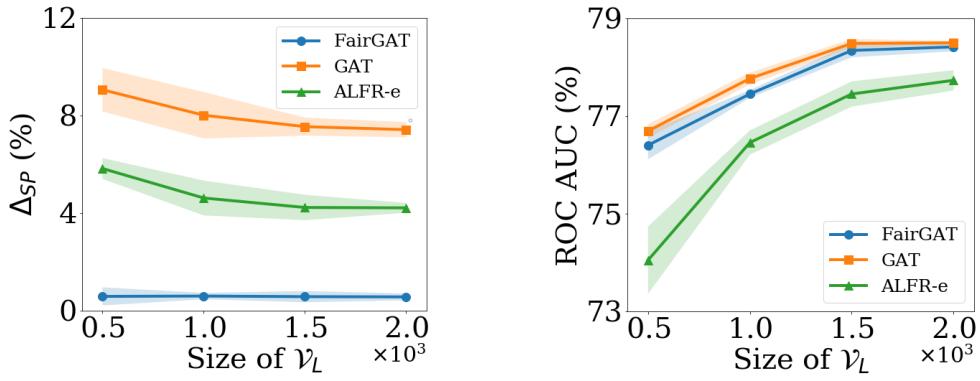
Figure 4.3: Impacts of the size of $\mathcal{V}_S$ to FairGAT.

effective. FairGAT could get very low Δ_{SP} even when $|\mathcal{V}_S|$ is as small as 200. This implies that FairGAT is insensitive to the size of data with sensitive attributes, which is because we have f_E to estimate the sensitive attributes. Though extremely small $|\mathcal{V}_S|$ would lead to a weak f_E, we still have similar Δ_{SP} score as that when $\mathcal{V}_S$ is large. This verifies our theoretical analysis that we can achieve good fairness with a reasonable f_E; (ii) FairGAT\E and ALFR-e decrease slightly in classification performance with the increasing of the size of $\mathcal{V}_S$, which is because more data with sensitive attribute would lead to a stricter regularization. In the contrary, FairGAT keeps high classification performance and even perform slightly better with more sensitive attributes. This is because the size of sensitive attributes $\hat{\mathcal{S}}$ used for training FairGAT are fixed to the size of $\mathcal{V}$, and less noise in the estimation of the sensitive attributes is helpful to better learn representations for classification.

Similarly, we vary $|\mathcal{V}_L|$ as $\{500, 1000, 1500, 2000\}$ and each experiment is run for 5 times. The average results on Pokec-z are reported in Figure 4.4. We only report the results on Pokec-z as we have simialr observations on other datasets. From the figure, we observe that: FairGAT consistently shows effectiveness in eliminating discrimination. The drop in classification performance is marginal. This demonstrates that our proposed method could achieve fairness while keep high accuracy in general scenarios which correspond to various sizes of $\mathcal{V}_S$ and $\mathcal{V}_L$.

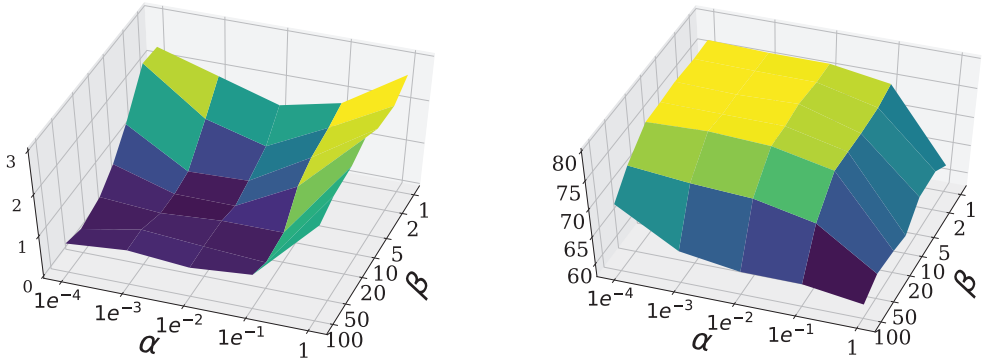
4.5.5 Parameter Sensitivity

There are two important hyperparameters in our proposed model, .i.e., α controlling the influence of the adversary to the GNN classifier, while β controlling the contribution of the covariance constraint to ensure fairness. To investigate the parameter sensitivity



(a) Δ_{SP} (b) ROC AUC

Figure 4.4: Impacts of the size of $\mathcal{V}_L$ to FairGAT.



(a) Δ_{SP} (%) (b) ROC AUC (%)

Figure 4.5: Parameter sensitivity analysis.

and find the ranges that achieve high accuracy with low discrimination score, we train FairGAT models on Pokec-z with various hyperparameters. More specifically, we alter the values of α and β among $\{0.0001, 0.001, 0.01, 0.1, 1\}$ and $\{1, 2, 5, 10, 20, 50, 100\}$. The results are presented in Figure 4.5. From Figure 4.5 (b), we can find that when $\alpha \leq 0.01$ and $\beta \leq 20$ the classification performance is almost unaffected. Once α and β are too large, the classifier's performance will decay rapidly. The impacts of the hyperparameters to the discrimination score are presented in Figure 4.5 (a). When we increase the value of α, Δ_{SP} will firstly decrease as expected. Then, it would increase when the value of α is too large. Because it would be difficult to optimize the GNN classifier to the global minimum when the contribution of the adversary is extremely high. As for β, the discrimination score would consistently reduce when we increase its value. Combining the two figures, we could determine that when $\alpha \in [0.001, 0.01]$ and $\beta \in [5, 20]$, the GNN classifier achieves fairness and maintains high node classification accuracy.

Chapter 5
Self-Explainable Graph Neural Networks

Graph neural networks (GNNs) [3,4,34] have made remarkable achievements in modeling graph structured data from various domains such as social networks [3, 12], financial system [35], and recommendation system [36]. The success of GNNs relies on the message-passing. More specifically, the node representations in GNNs will aggregate the information from the neighbors. As a result, the learned representations will capture the node attributes and local topology information to facilitate various tasks, especially the semi-supervised node classification.

Despite the success in modeling graph data, predictions of GNNs are not interpretable for human, which limits the adoption of GNNs in various domains such as credit approval in finance. *First*, as an extension of deep learning on graphs, the high non-linearity of GNNs makes the predictions difficult to understand. *Second*, GNNs utilize both node attributes and graph topology to give predictions, which leads to additional challenges to interpret the predictions. Though various approaches [137, 138] have been studied to give interpretable predictions or explain trained neural networks, most of them are designed for i.i.d data such as images and cannot handle the relational information. Thus, they cannot be directly applied for GNNs which highly rely on relational information in graphs. Some initial efforts [31, 33, 139] have been taken to address this problem. For instance, GNNExplainer [31] takes a trained GNN and its predictions as inputs and output crucial node attributes and subgraphs to explain the GNN's predictions. However, all the aforementioned GNN explainers only focus on the post-hoc explanations, i.e., learning an explainer to explain the outputs of a trained GNN. Since the post-hoc explanations are not directly obtained from the GNNs, they can be biased and misrepresent the true explanations of the GNNs. Therefore, it is crucial to develop a self-explainable GNN,

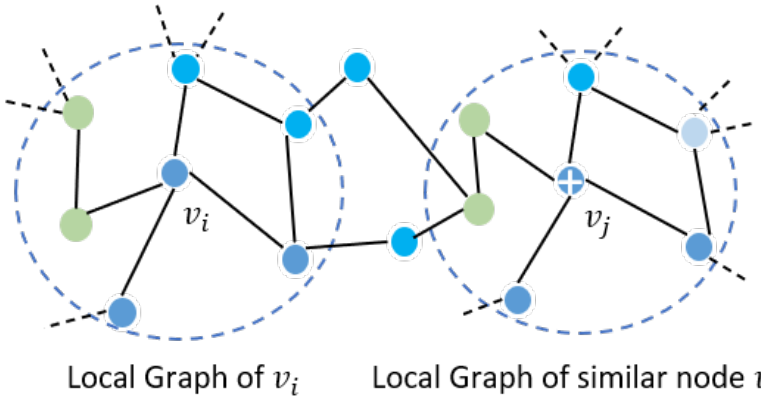


Figure 5.1: Example of interpretable K-nearest labeled nodes

which can simultaneously give predictions and explanations.

One perspective to obtain the self-explanation for node classification is to identify *interpretable K-nearest labeled nodes* for each node and utilize the K-nearest labeled nodes to simultaneously give label prediction and explain why such prediction is given. An example of the interpretable K-nearest labeled is shown in Figure 5.1. As it shows in Figure 5.1, the topology and node attributes in local graph of node v_i are well matched with that of the node v_j which is labeled as positive. Thus, v_i should be predicted as positive. Then the explaintions can be: (i) "node v_i is classified as class X because most of the K-nearest labeled nodes of v_i belong to class X"; and (ii) "node v_i is similar to node v_j because the n-hop subgraphs centered at these two nodes are similar. For example, this part of the structure and attributes of v_i's n-hop subgraph matches to that of v_j's". Though being promising, the work on exploring K-nearest neighbors for self-explainable GNNs is rather limited.

Therefore, in this paper, we investigate a novel problem of self-explainable GNN by exploring K-nearest labeled nodes. However, developing self-explainable GNN which gives interpretable K-nearest labeled nodes is non-trivial. In essense, we are faced with the following challenges: (**i**) how to incorporate the node similarity and structure similarity to identify K-nearest labeled nodes? For graph-structured data, the similarity should take both node attributes and local topology into consideration. Existing works of interpretable deep KNN [138] mostly focus on i.i.d data and cannot handle the structure similarity. One straightforward solution is using the cosine similarity of node representations learned by GNNs [3, 4] to identify the K-nearest labeled nodes. But the selection of the K-nearest labeled nodes is not interpretable, because the local graph topology and node attributes are implicitly embedded in latent vectors. In addition,

ground-truth of node and structure similarity is generally not available. Due to the lack of explicit supervision, it is challenging to identify the true K-nearest labeled nodes; and (**ii**) how to simultaneously give accurate predictions and correct corresponding explanations. The supervision we have is only the labels for classification accuracy. How to leverage label information to help learn the explanations is required to be investigated.

In an attempt to solve the challenges, we propose a model named as Self-Explainable GNN (SEGNN)[1]. SEGNN adopts a novel mechanism that can explicitly evaluate the node similarity and local structure similarity. For the structure similarity estimation, similarity scores of edges across the local graphs of two nodes will be evaluated to provide self-explanations. The prediction can be supervised by provided labels. And labels can also provide implicit supervision to guide the similarity modeling, because nodes with same labels are more likely to be a similar pair. Therefore, We propose a novel classification loss to simultaneously ensure the accuracy of the prediction and facilitate the interpretable similarity modeling. Furthermore, SEGNN adopts contrastive learning on node and edge representations to supervise node similarity and edge matching explanations. The main contributions are:

- We study a novel problem of self-explainable GNN by exploring K-nearest labeled nodes for predictions and explanations;
- We develop a novel framework SEGNN, which adopts an interpretable similarity modeling to identify K-nearest labeled nodes and enhance the explanations with self-supervision;
- We generate a synthetic dataset to quantitatively evaluate the quality of explanations, i.e., interpretable K-nearest labeled nodes, which can facilitate future research in this direction; and
- We conduct extensive experiments on both real-world datasets and synthetic datasets and demonstrate that SEGNN can give accurate predictions and explanations.

[1]https://github.com/EnyanDai/SEGNN

5.1 Related Work

5.1.1 Graph Neural Networks

Graph Neural Networks (GNNs) have shown their great power in modeling graph-structure data for various applications such as traffic analysis [117] and drug generation [140]. GNNs can be generally split into two categories, i.e., spectral-based [4, 34, 48] and spatial-based [3, 5, 50, 51, 141]. Spectral-based methods learn the node representations based on spectral graph theory. For example, [34] first generalize the convolution to graph-structure data with spectral graph theory. To remove the computationally expensive Laplacian eigendecomposition, ChebyNets [119] define graph convolutions with Chebyshev polynomials. GCN [4] further simplifies the convolutional operation. Spatial-based graph convolution is defined in spatial domain, which updates node representations by aggregating their neighbors' representations [3,52]. For example, GAT [50] updates the representations of the nodes from the neighbors with an attention mechanism. Moreover, various spatial methods are proposed for further improvements [5,41,54,142]. For instance, FastGCN [5] is proposed to solve the scalability issue.

Recently, some works employ self-supervised learning to graph structured data for better node representations. Various pretext tasks have been explored to benefit the training of GNNs [43, 55, 56]. For example, SuperGAT [143] deploys the edge prediction as self-supervised task to guide the learning of attention. Furthermore, contrastive learning is also widely adopted as the self-supervised task [144–147]. For instance, [146] design a subgraph instance discrimination to better capture the topology information. The aforementioned methods are inherently different from our proposed SEGNN as we focus on self-explainable GNNs and the self-supervision is applied to obtain high-quality explanations.

5.1.2 Explainability of Graph Neural Networks

To address the problem of lacking interpretability, extensive works have been proposed. Explanations in the existing works generally fall into two categories, i.e., self-explainable and post-hoc explanations. Self-explainable models are self-intrinsic, which can simultaneously make predictions and explain the predictions [148, 149]. For example, Alvarez et al. [148] propose a rich class of interpretable models where the explanations are intrinsic to the model. Post-hoc explanations use another model or strategy to explain the behavior of a trained model, such as by learning a local approximation with an interpretable

model [27], computing saliency map [150], identifying feature importance [28, 151, 152], obtaining prototypes [153] and exploring meaning of hidden neurons [154].

Despite the great success, the aforementioned approaches are overwhelmingly developed for i.id. data such as images and texts; while the work on interpretable GNNs for graph structured data are rather limited [31, 33, 139, 155]. GNNExplainer [31] learns soft masks for edges and node features to find the crucial subgraphs and features to explain the predictions. PGExplainer [33] applies a parameterized explainer to generate the edge masks from a global view to identify the important subgraphs. GraphLime [139] extends the LIME [27] to graph neural networks and investigates the importance of different node features for node classification. However, the graph structure information is ignored in GraphLime.

Our work is inherently different from the aforementioned explainable GNN methods: (i) we focus on learning a self-explainable GNN which can simultaneously give predictions and explanations while all the aforementioned are post-hoc explanations. We don't need additional explainer, which reduces the risk of misrepresenting the true decision reasons of the model; and (ii) we study a novel approach of finding the K-nearest labeled nodes in terms of node and structure similarity for classification and explanation.

5.2 Problem Definition

We use $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ to denote an attributed graph, where $\mathcal{V} = \{v_1, ..., v_N\}$ is the set of N nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ is the set of node attributes with $\mathbf{x}_i$ being the node attributes of node v_i. $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix of the graph $\mathcal{G}$, where $\mathbf{A}_{ij} = 1$ if nodes v_i and v_j are connected; otherwise $\mathbf{A}_{ij} = 0$. In the semi-supervised setting, part of the nodes $\mathcal{V}_L = \{v_1, ..., v_l\} \subset \mathcal{V}$ are labeled. We use $\mathcal{Y} = \{\mathbf{y}_1, ..., \mathbf{y}_l\}$ to denote the corresponding labels, where $\mathbf{y}_i \in \mathbb{R}^C$ is a one-hot vector of node v_i's label. $\mathcal{V}_U = \mathcal{V} - \mathcal{V}_L$ is a set of unlabeled nodes. Most of existing GNNs focus on label prediction of unlabeled nodes in $\mathcal{V}_U$ [156]. They usually lack interpretability on why GNN classifiers give such predictions. There are few attempts of post-hoc explainers [31, 33, 139, 155] to provide explanations for trained GNNs; while the work on self-explainable GNNs is rather limited.

We aim to develop self-explainable GNNs. In particular, for each unlabeled node v_i, we want to find K most similar labeled nodes with v_i, and simultaneously utilize these similar nodes to predict v_i's label and provide explanation of the prediction. Since both node features and local structure are important for classificaiton, the similarity should

be measured in terms of both node features and the n-hop subgraph centered at each node. Sample *explanations can be*: (*i*) "node v_i is classified as class B because these K nodes have the most similar node features and n-hop subgrph with that of v_i, and most of these K nodes belong to class B"; (*ii*) "node v_i is similar to node v_j because the n-hop subgraphs centered at these two nodes are similar. For example, this part of the structure and attributes of v_i's n-hop subgraph matches to that of v_j's".

Let $\mathcal{G}_s^{(n)}(v_i) = (v_i \cup \mathcal{N}^{(n)}(v_i), \mathcal{E}_s^{(n)}(v_i))$ be the n-hop subgraph centered at node v_i, where $\mathcal{N}^{(n)}(v_i)$ is the set of nodes within n-hop of v_i and $\mathcal{E}_s^{(n)}(v_i)$ is the set of edges that linked the nodes in $\mathcal{G}_s^{(n)}(v_i)$. Let $\mathcal{P}^{(n)}(v_i, v_j) \subseteq \mathcal{E}_s^{(n)}(v_i) \times \mathcal{E}_s^{(n)}(v_j)$ be the edge matching result between the local graphs of node v_i and v_j. With these notations, the problem of learning self-explainable GNN by predicting with interpretable K-nearest neighbors is defined as:

Problem 4. *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with labeled node set $\mathcal{V}_L$ and correspoding label set $\mathcal{Y}$, learn a self-explainable GNN $f_{\mathcal{G}} : \mathcal{G} \to \mathcal{Y}$ which could give an accurate prediction for each unlabeled node $v_i \in \mathcal{V}_U$ and simultaneously generate explanation from the K-nearest labeled node $\{(v_{ik}, \mathcal{G}_s^{(n)}(v_{ik}), \mathcal{P}^{(k)}(v_i, v_{ik}))\}_{k=1}^K$, where $v_{ik} \in \mathcal{V}_L$ and $\mathcal{G}_s^{(n)}(v_{ik})$ are the identified top K nearest labeled nodes and the corresponding n-hop subgraph, respectively. $\mathcal{P}^{(n)}(v_i, v_{ik})$ is the edge matching results between local graphs of v_i and v_{ik} to explain the similarity in structure.*

5.3 Methodology

In this section, we present the details of the proposed framework SEGNN. The basic idea of SEGNN is: for each node v_i, it identifies K most similar labeled nodes with v_i, and utilize the labels of these K nodes to predict v_i's label. Meanwhile, the K most similar labeled nodes provide explanations on why such prediction is made in terms of both the structure and feature similarity. There are mainly two challenges: (i) how to obtain interpretable K-nearest labeled nodes that consider both node and structure similarity; and (ii) how to simultaneously give accurate predictions and correct corresponding explanations. To address these challenges, SEGNN explicitly models the node similarity and local structure similarity with explanations. An illustration of the proposed framework is shown in Figure 5.2. It is mainly composed of an interpretable similarity module and a self-supervisor to enhance the explanations. With the novel similarity modeling process, K-nearest labeled nodes of the target node and the similarity explanations can be obtained. Then, prediction of the target node can be given based

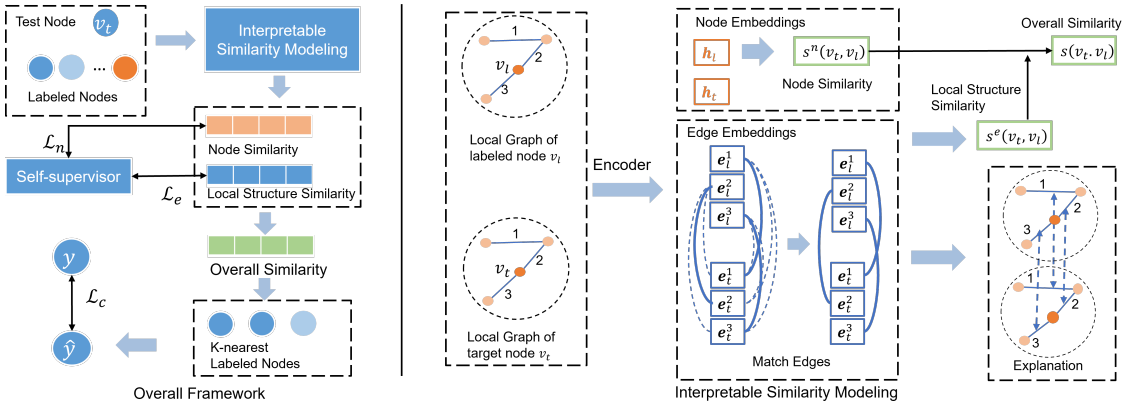


Figure 5.2: An overview of the proposed SEGNN.

on the identified K-nearest labeled nodes. And, a novel loss function is designed to ensure the accuracy of predictions and facilitate the similarity modeling. Furthermore, self-supervision for explanations is applied to further benefit the accurate explanation generation.

5.3.1 Interpretable Similarity Modeling

Since for each node, SEGNN relies on interpretable K-nearest labeled nodes for predictions and explanations, we need to design an interpretable similarity measurements to measure the similarity of nodes. Unlike i.i.d data, which only needs to measure the similarity from the feature perspective, for graph-structure data, both node attributes and the local graph structures of nodes contain crucial information for node classification. Thus, we propose to explicitly model the node similarity and structure similarity to obtain an interpretable overall similarity.

5.3.1.1 Node Similarity

The node similarity is to evaluate how similar the target node is with the labeled nodes in the node level. Since node features could be noisy and sparse, directly measuring the node similarity in the raw feature space will result in noisy similarity. Following existing work on similarity metric learning [146], we first learn node representations followed by a similarity function, which could better measure node similarity. One straightforward way is to adopt a deep GNN such as GCN [4] and GAT [50] to learn powerful node embeddings. However, current GNNs are found to experience the over-smoothing issue [44]. This may lead to indistinguishable similarity scores for node pairs that actually differ a lot. And a deep GNN may implicitly model the structure information, which could reduce the interpretability of the node similarity. Therefore, we propose to firstly encode the node

features with a MLP. Then, the node embeddings are further updated by aggregating the representations from their neighbors with one residual GCN layer [4]. This process can be mathematically written as:

$$\mathbf{H}^m = MLP(\mathbf{X}), \quad \mathbf{H} = \sigma(\tilde{\mathbf{A}}\mathbf{H}^m\mathbf{W}) + \mathbf{H}^m, \tag{5.1}$$

where $\mathbf{X}$ denotes the node attributes, $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix , and $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sum_i A_{ij}$. $\mathbf{I}$ is the identity matrix. σ is the activation function such as ReLU. With the learned node embeddings, the node similarity between a target node v_t and a labeled node v_l can be obtained as:

$$s^n(v_t, v_l) = sim(\mathbf{h}_t, \mathbf{h}_l), \tag{5.2}$$

where $\mathbf{h}_t$ and $\mathbf{h}_l$ are the learned embeddings of node v_t and v_l, respectively. sim is flexible to be various similarity metrics such as cosine similarity [147] and distance-based similarity [157].

5.3.1.2 Local Structure Similarity

Generally, the content information, i.e., the n-hop graph structure, is very important for node classification. If two nodes v_i and v_j have similar n-hop subgraphs, their labels are likely to be similar. Thus, in addition to the node level similarity, we also measure the local structure similarity, which can (i) explicitly consider the local structure information to facilitate the identification of K-nearest labeled nodes and (ii) provide explanations about the similarity in a structure level. Specifically, we propose to measure how well the edges between the local graphs of two nodes match to evaluate the similarity in structure level. The edge matching results can explain the similarity of two nodes in the aspect of local structure. To match edges for local structure similarity, we need to first learn representations of edges. Since an edge is determined by the two nodes linked by it, for an edge $e = (v_i, v_j)$ which links node v_i and v_j, we get the edge representation $\mathbf{e}_{ij}$ as:

$$\mathbf{e}_{ij} = f_e(\mathbf{h}_i, \mathbf{h}_j), \tag{5.3}$$

where $\mathbf{h}_i$ and $\mathbf{h}_j$ are embeddings of node v_i and v_j, respectively. f_e is the function of aggregating the node information to get the edge embedding, which is flexible to various functions such as average pooling and LSTM. In our implementation, we apply average pooling function as f_e. With Eq.(5.3), we can get two sets of edge representations

$\mathcal{R}_t = \{\mathbf{e}_t^1, ..., \mathbf{e}_t^M\}$ and $\mathcal{R}_l = \{\mathbf{e}_l^1, ..., \mathbf{e}_l^N\}$ for $\mathcal{E}_s^{(n)}(v_t)$ and $\mathcal{E}_s^{(n)}(v_l)$, where $\mathcal{E}_s^{(n)}(v_t)$ and $\mathcal{E}_s^{(n)}(v_l)$ are the edges in the local graphs of v_t and v_l, respectively. For an edge $e_t^i \in \mathcal{E}_s^{(n)}(v_t)$, we will find the edge in $\mathcal{E}_s^{(n)}(v_l)$ that matches e_t^i best. The process of identifying the paired edge for e_t^i can be formally written as:

$$e_p^i = \arg \max_{e_l^j \in \mathcal{E}_s^{(n)}(v_l)} sim(\mathbf{e}_t^i, \mathbf{e}_l^j), \tag{5.4}$$

where e_p^i is the found edge in local graph of v_l that matches edge e_t^i. The whole edge matching results, which can explain the local structure similarity, can be obtained by:

$$\mathcal{P}^{(n)}(v_t, v_l) = \{(e_t^i, e_p^i)\}_{i=1}^M. \tag{5.5}$$

Then, the local structure similarity between v_t and v_l can be obtained by averaging the similarity scores of all the paired edges:

$$s^e(v_t, v_l) = \frac{1}{M} \sum\nolimits_{i=1}^{M} sim(\mathbf{e}_t^i, \mathbf{e}_p^i), \tag{5.6}$$

where $\mathbf{e}_p^i$ is the representation of edge e_p^i.

5.3.1.3 Overall Similarity

With the node similarity and local structure similarity between node v_t and v_l, the overall similarity is:

$$s(v_t, v_l) = \lambda s^n(v_t, v_l) + (1 - \lambda) s^e(v_t, v_l), \tag{5.7}$$

where λ is a positive scalar to balance the contributions of node similarity and structure similarity.

5.3.2 Self-Explainable Classification

With the similarity metric described in Section 5.3.1, we are able to identify the interpretable K-nearest labeled nodes to predict and explain the label of the target node. Next, we introduce the details of the prediction process, discussions about the explanations, and the loss function that ensures classification accuracy.

5.3.2.1 Prediction with K-nearest Labeled Nodes

Let $\mathcal{K}_t = \{v_t^1, ..., v_t^K\}$ be the set of K-nearest labeled nodes of the target node v_t based on Eq.(5.7). Intuitively, the more similar v_t and v_t^i are, i.e., the larger $s(v_t, v_t^i)$ is, the more likely v_t has the same label as v_t^i. Thus, following deep KNN [157, 158], we predict the label of node v_t as weighted average of the labels of the K-nearest neighbors. Specifically, the weight a_{ti} of the i-th nearest labeled node, i.e., v_t^i, is calculated as

$$a_{ti} = \frac{\exp(s(v_t, v_t^i)/\tau)}{\sum_{i=1}^{K} \exp(s(v_t, v_t^i)/\tau)}, \tag{5.8}$$

where τ is the temperature parameter. With the weight a_{ti}, the label of v_t is predicted as

$$\hat{\mathbf{y}}_t = \sum\nolimits_{i=1}^{K} a_{ti} \cdot \mathbf{y}_t^i, \tag{5.9}$$

where $\mathbf{y}_t^i$ is the one-hot label vector of v_t^i.

5.3.2.2 Explanation

For a target node v_t, the found K-nearest labeled nodes $\mathcal{K}_t = \{v_t^1, ..., v_t^K\}$ and the corresponding similarity scores $\{s(v_t, v_t^i)\}_{i=1}^K$ can clearly explain the predictive label. For the overall similarity $s(v_t, v_t^i)$, the contributions of node and structure similarity can be given through $s^n(v_t, v_t^i)$ and $s^e(v_t, v_t^i)$. Moreover, we can track back the identified edge pairs $\mathcal{P}^{(n)}(v_t, v_t^i)$ between local graph of v_t and its i-th nearest labeled node v_t^i to explain the local structure similarity. Though our SEGNN focuses on explaining the predictions with K-nearest labeled nodes, it also can extract the crucial subgraph for explanation. For a crucial edge, the local graphs of K-nearest neighbors should also contain similar one. Thus, the importance of an edge $e_t^i \in \mathcal{E}_s^{(n)}(v_t)$ can be evaluated by its average similarity with the identified pair edges $\{e_p^{ij}\}_{j=1}^K$ in the local graphs of the K-nearest labeled nodes as

$$p(e_t^i) = \frac{1}{K} \sum\nolimits_{j=1}^{K} sim(\mathbf{e}_t^i, \mathbf{e}_p^{ij}), \tag{5.10}$$

where $\mathbf{e}_t^i$ and $\mathbf{e}_p^{ij}$ are edge representations of e_t^i and e_p^{ij}, respectively. Then, a threshold can be set to filter out the unimportant edges in the local graph of node v_t.

5.3.2.3 Classification Loss

SEGNN is expected to give accurate predictions. Therefore, we utilize the supervision from the given labels to ensure the accuracy of the self-explainable GNN. One straightforward way is to optimize the predictions of nodes with labels in a leave-one-out manner. More specifically, for a labeled node $v_i \in \mathcal{V}_L$, we identify K-nearest labeled nodes from other labeled nodes $\mathcal{V}_L - \{v_i\}$. Then loss such as cross entropy loss can be applied to optimize the model. However, in this way, the optimization only involves the searched K-nearest neighbors which are generally similar nodes. Lacking of various negative samples may negatively affect similarity modeling. In addition, the computational cost to identify K-nearest neighbors will be large when numerous labeled nodes are given. To address these issues, we propose a loss function that adopts negative sampling [65] and approximately select K-nearest labeled nodes as positive samples. Specifically, for a target node $v_t \in \mathcal{V}_L$, we randomly sample Q labeled nodes $\mathcal{V}_n^t$ which do not share the same label with v_t as negative samples. And we randomly sample N ($N > K$) labeled nodes sharing the same label with v_t as support set to obtain approximate K-nearest labeled nodes $\tilde{\mathcal{K}}_t$. The objective function can be formally written as:

$$\min_{\theta} \mathcal{L}_c = \frac{1}{|\mathcal{V}_L|} \sum_{v_t \in \mathcal{V}_L} -\log \frac{\sum_{\tilde{v}_t^i \in \tilde{\mathcal{K}}_t} \exp(s(v_t, \tilde{v}_t^i)/\tau)}{\sum_{v_n^i \in \tilde{\mathcal{K}}_t \cup \mathcal{V}_n^t} \exp(s(v_t, v_n^i)/\tau)}, \tag{5.11}$$

where θ denotes the parameters of SEGNN, and τ is the temperature hyperparameter. With loss function Eq.(5.11), the similarity scores of node pairs with different labels will be minimized, and the similarity scores between a node and its approximate k-nearest neighbor with the same labels will be maximized. As a result, accurate predictions can be given. Moreover, it provides supervision to guide the similarity modeling. *Note that this sampling strategy to obtain approximate K-nearest labeled nodes is only applied during the training phase.* For testing phase, K-nearest labeled nodes are identified from the whole labeled node set $\mathcal{V}_L$ and the label is predicted with Eq.(5.9).

5.3.3 Enhance Explanation with Self-Supervision

Although the labels can provide the supervision to facilitate the similarity modeling with objective function in Eq.(5.11), they do not explicitly supervise the node similarity and local structure similarity. In addition, the edge matching results and identification of nearest labeled nodes may not generalize well to unlabeled nodes, because unlabeled nodes are not involved in Eq.(5.11). To further benefit the explanation generation and

similarity metric learning, we adopt a contrastive pretext task to provide self-supervision for node similarity and local structural similarity learning.

Recently, contrastive learning has shown to be effective for unsupervised representation learning on graphs [145–147]. Essentially, contrastive learning aims to maximize representation consistency under differently augmented views. In other words, with contrastive learning, similar nodes/graphs will be given similar representations. Therefore, we adopt contrastive learning on node representations to facilitate the node similarity modeling. Moreover, the explanation for local structure similarity relies on accurate edge machining. To guide the edge matching on unlabeled nodes, a contrastive task on edge representations is deployed as well. In detail, we maximize the the agreement between two augmented views of the graphs via a contrastive loss in node and edge representations. Following GraphCL [147], two augmentations, i.e., attribute masking and edge perturbation, are applied to obtain representations from different views. We apply infoNCE loss [159] as the contrastive loss. The infoNCE loss transfers the mutual information maximization to a classification task which requires positive pairs and negative pairs. Representations of the same node/edge from these two views will compose positive pairs for contrastive learning. And representations of different nodes in both views compose negative pairs. The contrastive learning is trained in a minibatch manner. For a query representation $\mathbf{h}_i$, a dictionary $\{\tilde{\mathbf{h}}_0, ..., \tilde{\mathbf{h}}_{Q_N}\}$ which contains one positive sample $\tilde{\mathbf{h}}_+$ will be built. The objective function of contrastive learning on node representations can be formulated as:

$$\min_{\theta} \mathcal{L}_n = \frac{1}{|\mathcal{V}_B|} \sum_{v_i \in \mathcal{V}_B} -\log \frac{\exp(sim(\mathbf{h}_i, \tilde{\mathbf{h}}_+)/\tau)}{\sum_{j=0}^{Q_N} \exp(sim(\mathbf{h}_i, \tilde{\mathbf{h}}_j)/\tau)}. \tag{5.12}$$

Similarly, contrastive learning is adopted on the edge representations to benefit the edge matching explanations. Let $\mathcal{E}_B$ denotes a minibatch of edges. The objective function can be written as:

$$\min_{\theta} \mathcal{L}_e = \frac{1}{|\mathcal{E}_B|} \sum_{e_i \in \mathcal{E}_B} -\log \frac{\exp(sim(\mathbf{e}_i, \tilde{\mathbf{e}}_+)/\tau)}{\sum_{j=0}^{Q_E} \exp(sim(\mathbf{e}_i, \tilde{\mathbf{e}}_j)/\tau)}, \tag{5.13}$$

where $\mathbf{e}_i$ denotes a query edge representation, and $\tilde{\mathbf{e}}_+$ is the positive sample from dictionary $\{\tilde{\mathbf{e}}_0, ..., \tilde{\mathbf{e}}_{Q_E}\}$. With the self-supervision, the representations of similar edges and nodes will be similar, which can facilitate the similarity modeling for explanations. Moreover, since edge perturbation is used to augment the graph in contrastive learning, the representations from the perturbed graph will be enforced to be consistent with the

representations from the clean graph. This will lead to robustness against structure noise.

5.3.4 Overall Objective Function

With the supervision from the labels for accurate prediction, and the contrastive learning on node and edge representations to facilitate the modeling of similarity, the final loss function of SEGNN is:

$$\min_{\theta} \mathcal{L}_c + \alpha\mathcal{L}_n + \beta\mathcal{L}_e, \tag{5.14}$$

where θ represents the parameters of SEGNN. α and β are hyperparameters that control the contributions of self-supervision on node similarity modeling and edge matching in local similarity evaluation, respectively.

5.3.5 Training Algorithm and Time Complexity

5.3.5.1 Training Algorithm

The training algorithm of SEGNN is given in Algorithm 2. In line 1, the parameters of SEGNN are randomly initialized with Xavier initialization [160]. In line 3, the supervised loss from the labeled nodes is calculated. Graph augmentation is conducted in line 4, which gives graphs in different views for contrastive learning. From line 5 to line 9, contrastive losses for nodes and edges are computed. For the size of negative samples in Eq.(5.11), it is fixed as 20. The sizes of negative samples in Eq.(5.12) and Eq.(5.13) are both set as 100 during the training phase.

5.3.5.2 Time Complexity

During the test phase, the main time complexity comes from the similarity scores calculation between the test node v_t and all the labeled nodes $\mathcal{V}_L$. For each node $v_l \in \mathcal{V}_L$, the cost for calculating the edge matching with v_t is $\mathcal{O}(d \cdot |\mathcal{E}_t| \cdot |\mathcal{E}_l|)$, where d is the embedding dimension. Thus, the time complexity for one test node is approximately $\mathcal{O}(d \cdot |\mathcal{E}_t| \cdot \sum_{v_l \in \mathcal{V}_L} |\mathcal{E}_l|)$. As for the training phase, we adopt a sampling strategy in Eq.(5.11) to reduce the pairs of similarity to be computed. For each node $v_t \in \mathcal{V}_L$, let $\mathcal{S}_t = \mathcal{K}_t \cup \mathcal{V}_n^t$ denotes the sampled positive and negative nodes in Eq.(5.11), the cost of calculating classification loss for v_t is $\mathcal{O}(d \cdot |\mathcal{E}_t| \cdot \sum_{v_l \in \mathcal{S}_t} \cdot |\mathcal{E}_l|)$. Thus, the cost for classification loss of all labeled nodes is $\mathcal{O}(\sum_{v_t \in \mathcal{V}_t} \sum_{v_l \in \mathcal{S}_t} d \cdot |\mathcal{E}_t| \cdot |\mathcal{E}_l|)$.With the computation cost on contrastive learning, the overall time complexity for an iteration in the training phase is $\mathcal{O}(d \cdot (Q_n|\mathcal{V}_B| + Q_e|\mathcal{E}_B| + \sum_{v_t \in \mathcal{V}_t} \sum_{v_l \in \mathcal{S}_t} |\mathcal{E}_t| \cdot |\mathcal{E}_l|))$.

Algorithm 2 Training Algorithm of SEGNN.

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, $\mathcal{Y}$, K, α, β, λ, τ.
Output: Self-explainable GNN $f_{\mathcal{G}}$.

```
Randomly initialize the parameters of f_G.
repeat
    Obtain the classification loss by Eq.(5.11)
    Augment the G with attribute masking and edge perturbation to receive graphs in
    different views
    Sample a node batch V_B with positive and negative pairs
    Calculate contrastive loss on nodes by Eq.(5.12)
    Sample an edge batch E_B with positive and negative pairs
    Get contrastive loss on edges by Eq.(5.13)
    Optimize the parameter of f_G by Eq.(5.14)
until convergence
return f_G
```

5.4 Experiments

In this section, we conduct extensive experiments on real-world and synthetic datasets to demonstrate the effectiveness of SEGNN. In particular, we aim to answer the following research questions:

- **RQ1** Can our proposed method simultaneously provide accurate predictions and corresponding reasonable explanations?
- **RQ2** Is SEGNN robust to the structure noises in the datasets?
- **RQ3** How does each component of our proposed SEGNN contribute to the classification performance and explainability?

5.4.1 Datasets

To quantitatively and qualitatively evaluate SEGNN in predictions and explanations, we conduct extensive experiments on three real-world datasets and two synthetic datasets. The statistics of the datasets are presented in Table 5.1.

5.4.1.1 Real-World Datasets

To demonstrate the effectiveness of our proposed methods, for real-world datasets, we choose three widely used benchmark networks, i.e., Cora, Citeseer, and Pubmed [61]. For

Cora and Pubmed, the standard dataset splits as in the cited paper are applied. As for Citeseer, it contains isolated nodes which are not applicable to our method. Therefore, following the pre-processing strategy in [40], we select the largest connected component in the Citeseer graph and apply the same dataset splits.

5.4.1.2 Synthetic Datasets

We construct two synthetic datasets, i.e., Syn-Cora and BA-Shapes, which provide ground truth of explanations for quantitative analysis. The details are described below.

Syn-Cora: This dataset is synthesized from the Cora graph which provides ground-truth of explanations, i.e., K-nearest labeled nodes and edge machining results. To construct the graph, motifs are obtained by sampling local graphs of nodes from Cora. Various levels of noises are applied to the motifs in attributes and structures to generate similar local graphs. For a motif and it's corresponding perturbed versions provide the groundtruth K-nearest neighbors and the corresponding edge-matching. Specifically, we sample three motifs for each class, resulting in 21 unique motifs in total. To link the synthetic local graphs together, a subgraph of Cora that have no overlap with the motifs is sampled as the basis graph. These synthetic local graphs are attached to the basis graph by randomly linking three nodes. To simulate a realistic training scenario, we randomly select 30% nodes from the motifs and basis graph as the training set. Testing is conducted on the remaining nodes in the motifs for explanation accuracy evaluation.

BA-Shapes: To compare with the state-of-the-art GNN explainers [31,33] which identify crucial subgraphs for predictions, we construct BA-Shapes following the setting in GNNExplainer [31]. BA-Shapes is a single graph consisting of a base Barabasi-Albert (BA) graph with 300 nodes and 80 "house"-structured motifs. These motifs are attached to the BA graph. And random edges are added to perturb the graph. Node features are not assigned in BA-Shapes. Nodes in the base graph are labeled with 0. Nodes locating at the top/middle/bottom of the "house" are labeled with 1,2,3, respectively. Dataset split is the same as that in [31].

5.4.2 Experimental Settings

5.4.2.1 Baselines

To evaluate the performance and robustness of SEGNN in node classification on real-world datasets, we first compare with the following representative and state-of-the-art GNNs, self-supervised GNN, and robust GNN.

Table 5.1: Statistics of datasets.

	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,433	7
Citeseer	2,110	3,668	3,703	6
Pubmed	19,171	44,338	500	3
Syn-Cora	1,677	4,610	1,433	7
BA-Shapes	700	4,421	-	4

Table 5.2: Node classification accuracy (%) on real-world datasets.

Dataset	GCN	GIN	SuperGAT	Pro-GNN	MLP-K	GCN-K	GIN-K	Ours
Cora	80.8±1.1	80.5±0.8	**82.4±0.7**	79.1±0.1	53.9±1.8	78.8±1.2	78.8±0.3	80.4±0.3
Citeseer	71.9±1.0	72.5±0.8	73.6±0.2	73.3±0.7	61.6±1.6	71.4±0.8	69.2±1.2	**73.8±0.6**
Pubmed	78.4±0.4	78.9±0.2	79.2±0.4	79.4 ±0.4	73.2±0.2	77.4±0.2	78.8±0.3	**80.0±0.2**

- **GCN** [4]: GCN is a popular spectral-based GNN which defines graph convolution with spectral theory.
- **GIN** [14]: Compared with GCN, multi-layer perception is used in GIN to process the aggregated information from the neighbors in each layer to learn more powerful representations.
- **SuperGAT** [143]: This is a self-supervised graph neural network. Edge prediction is deployed as the pretext task to directly guide the learning of attention to facilitate the information aggregation.
- **Pro-GNN** [40]: This is state-of-the-art GNN against noisy edges in graphs. It applies low-rank and sparsity constraint to directly learn a clean graph close to the noisy graph to defend against adversarial attacks.

Existing work that identifies the K-nearest labeled nodes is rather limited. Therefore, we compare with the following baselines based on the deep KNN [138] on i.i.d data to evaluate our explanations.

- **MLP-K**: Following [138], we first train a MLP with the node features and labels. After training, the node representations from the final layer of MLP is used to find the K-nearest labeled nodes. The final prediction of an unlabeled node is obtained using weighted average of the its K-nearest labeled nodes.
- **GCN-K**: Similar to MLP-K, we use the representations from the final layer of GCN to get K-nearest neighbors and predictions. Post-hoc explanations in structure

similarity can be obtained by edge matching through the edge representations. Edge representations are average representations of the linked nodes .

- **GIN-K**: It replaces the backbone in GCN-K with GIN to obtain the predictions and explanations.

Finally, we compare with the state-of-the-art GNN explainers in extracting important subgraphs to explain predictions:

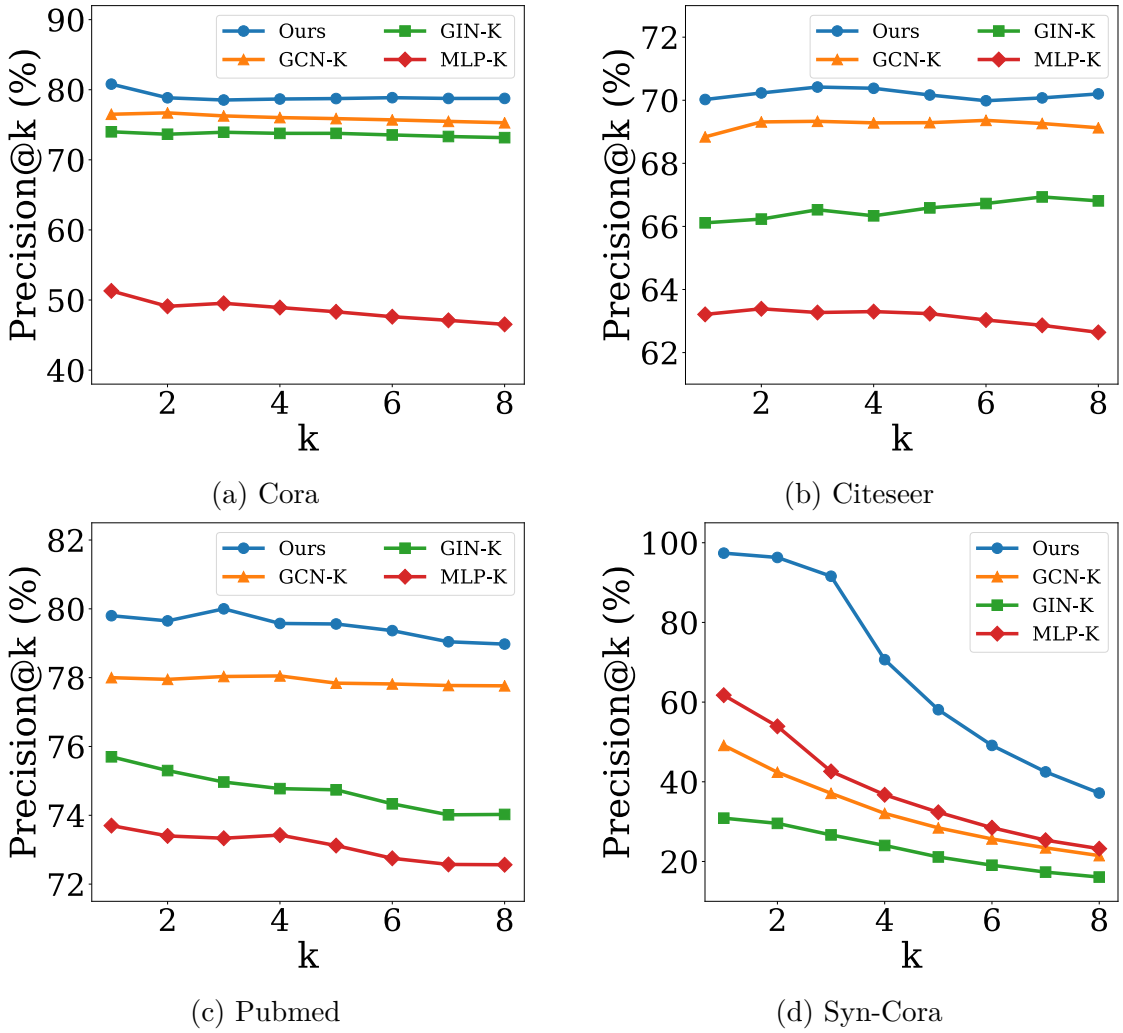
- GNNExplainer [31]: GNNExplainer takes a trained GNN and the predictions as input to obtain post-hoc explanations. It learns a soft edge mask for each instance to identify the crucial subgraph.
- PGExplainer [33]: It adopts a MLP-based explainer to obtain the important subgraphs from a global view to reduce the computation cost and obtain better explanations.

5.4.2.2 Implementation Details

The local structure similarity is based on 2-hop local graphs of nodes in all the experiments. For SEGNN, the encoder consists of two MLP layers and one GCN layer with residual connection. The hidden dimension is set as 64 in these MLP and GCN layers. The rate of attribute masking in the contrastive learning is set as 0.2. We replace 10% edges in the graph to noisy edges for edge perturbation. For experiments on real-world datasets, all hyperparameters are tuned based on the prediction results on the validation set. We vary α and β among $\{0.0001, 0.001, 0.01, 0.1, 1\}$. The λ which balance the node similarity and structure similarity is set as 0.5 for all datasets. The number of nearest labeled nodes used for prediction, i.e., K, is set as searched as $\{25, 40, 50\}$ for all the datasets. The temperature hyperparameter τ is fixed as 1 in all experiments. For the hyperparameters on the Syn-Cora and BA-Shapes, we reuse the setting on the Cora graph. The BA-shapes dataset does not provide features, we initialize the node features with node degree and number of involved triangles. The hyparameters for the baselines are also tuned on the validation set. All the experiments are conducted 5 times and the average results with standard deviations are reported.

5.4.3 Classification and Explanation Quality

To answer **RQ1**, we compare SEGNN with baselines on real-world datasets and synthetic datasets in terms of classification performance and explanation quality.



(a) Cora (b) Citeseer

(c) Pubmed (d) Syn-Cora

Figure 5.3: The precision@k of the K-nearest labeled nodes.

5.4.3.1 Results on Real-World Datasets

To demonstrate that our SEGNN can give accurate predictions, we compare with the state-of-the-art GNNs on real-world datasets. Each experiment is conducted 5 times. The average node classification accuracy and standard deviations are reported in Table 5.2. From the table, we observe:

- Our method outperforms GCN and GIN on various real-world datasets especially on the large dataset. This is because information from numerous unlabeled nodes is leveraged in SEGNN through the similarity modeling with contrastive learning.
- SEGNN achieves comparable performance with self-supervised method SuperGAT. Note that encoder in SEGNN only involves 1-hop neighbors to learn representations.

Table 5.3: Average ratings of human evaluation.

Dataset	MLP-K	GCN-K	GIN-K	Ours
Cora	0.030	0.405	0.207	**0.763**
Citeseer	0.030	0.311	0.326	**0.733**
Pubmed	0.089	0.348	0.252	**0.674**

This indicates that the designed local structure similarity manages to capture the complex structure information for node classification.

- Though GCN-K and GIN-K also utilize GNN to learn representation and K-nearest neighbors for prediction, SEGNN performs much better than them, which shows the effectiveness of SEGNN in utilizing the label information and contrastive learning for learning better representations and similarity metrics.

Intuitively, for each test node $v_t \in \mathcal{V}_U$, SEGNN should assign higher similarity score to labeled nodes of the same class as v_t. To analyze this, for each v_t, we first rank the labeled nodes based on similarity scores. Then We treat the label of v_t as the groundtruth and calculate the precision@k for the ranked list. We average the results for all $v_t \in \mathcal{V}_U$. Generally, if a method assigns higher similarity scores to labeled nodes of the same class as v_t, it would have large precision@k. We vary k as $\{1, 2, \ldots, 8\}$. The hyperparameters are set as described in Section 5.4.2.2. The results on the three real-world datasets are presented in Figure 5.3a, 5.3b and 5.3c, respectively. We can observe that SEGNN consistently outperforms other baselines by a large margin, which indicates that SEGNN can retrieve reliable K-nearest labeled nodes for prediction and explanation.

We also conduct qualitative evaluation on explanations on real-world datasets. The explanations of an instance from Citeseer are presented in Figure 5.4. Specifically, the local graphs of nearest labeled nodes identified by different methods are presented. And we apply t-SNE to node features to obtain the positions of nodes in the visualized graph for node similarity comparison. The shapes of the graphs can help to assess the similarity of local structures. From the Figure 5.4, we can observe that SE-GNN can correctly identify the labeled node whose features and local topology are both similar with the target node. And the given edge matching results well explain the local structure similarity. On the other hand, baselines fail to identify the similar labeled nodes and provide poor explanations in structure similarity.

To further testify the quality of our explanations, 30 annotators are asked to rate the model's explanations on three real-world datasets. The explanations are presented in the same way as Fig. 5.4. Each annotator rates explanations at least 15 instances

Table 5.4: Results on Syn-Cora.

Metric (%)	MLP-K	GCN-K	GIN-K	Ours
Accuracy	93.8±2.3	94.8±0.7	94.6±0.7	**97.7±1.6**
Edge ACC	-	25.1±0.4	18.2±1.8	**81.1**±1.1

Table 5.5: Structure explanation AUC on BA-Shapes.

GNNExplainer	PGExplainer	Ours
95.6±3.7	98.7±2.1	98.1±0.5

from the three real-world datasets. The rating score is either 0 (similar) or 1 (disimialr). The average ratings are presented in Table 5.3. From the table, we can find that the nearest neighbors identified by our method receive the highest ratings, which shows that our explanations are in line with the human's decisions. It verifies the quality of our explanations on real-world datasets.

5.4.3.2 Results on Syn-Cora

We compare with baselines on Syn-Cora which provides the ground-truth explanations to quantitatively evaluate the two-level explanations, i.e., K-nearest labeled nodes and the edge matching results for similarity explanation. The prediction performance is evaluated by accuracy. Precision@k is used to show the quality of K-nearest labeled nodes. The accuracy of matching edges (Edge ACC) is used to demonstrate the quality of local structure similarity explanation. The results are presented in Table 5.4 and Figure 5.3d. Note that edge matching is not applicable for MLP-K, because it cannot capture structure information. From the table and figure, we observe:

- Though GCN-K and GIN-K achieve good performance in classification, they fail to identify the true similar nodes and explain the struck similarity. This is due to the over-smoothing issue in deep GNNs, which leads representations poorly persevere similarity information. By contrast, SEGNN achieves good performance in all explanation metrics, which shows node similarity and local structure similarity are well modeled in SEGNN.

- Compared with MLP-K which does not experience over-smoothing issue, SEGNN can give more accurate explanations. This is because we apply the supervision from labels and self-supervision to guide the learning of two-level explanations.

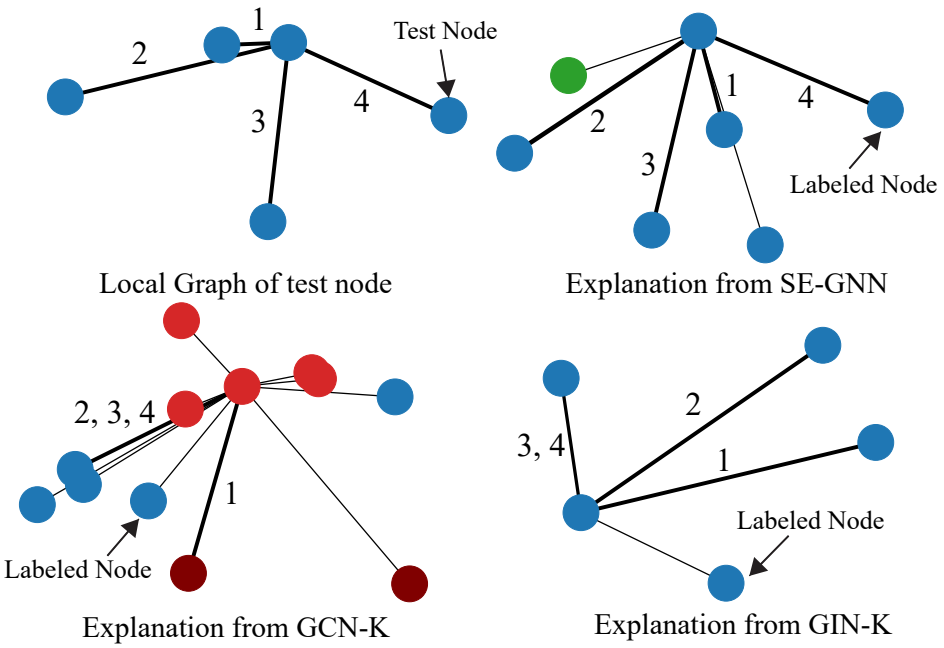


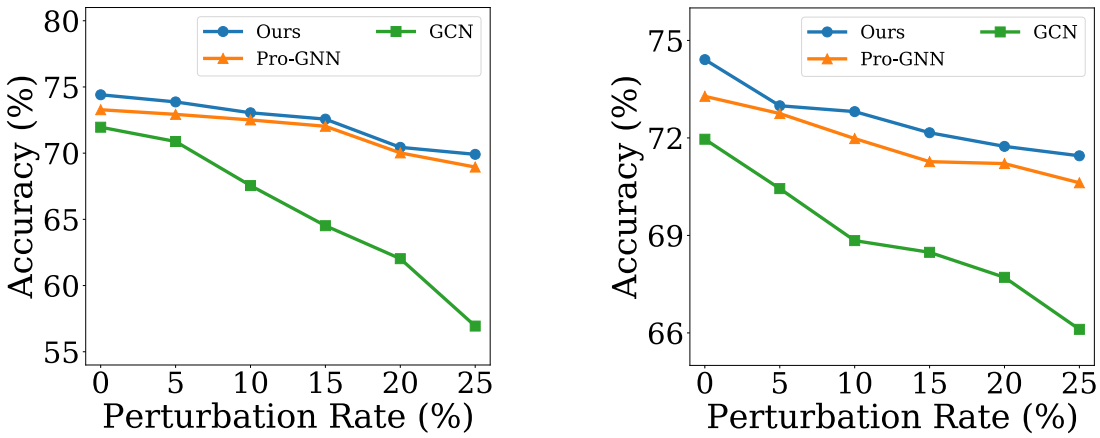
Figure 5.4: Illustration of the explanation from SEGNN and other baselines. Node colors denote the label of nodes. Edges with the same number denote that they are matched.

5.4.3.3 Results on BA-Shapes

As it is discussed in Section 5.3.2.2, our SEGNN can be extended to extract a crucial subgraph of the test node's local graph to explain the prediction. To demonstrate the effectiveness of extracting crucial structures as explanations, we compare SEGNN with state-of-the-art GNN explainers on a commonly used synthetic dataset BA-Shapes. Following [31], crucial structure explanation AUC is used to assess the performance in explanation. The average results of 5 runs are reported in Table 5.5. From this table, we can observe that, though SEGNN is not developed for extracting crucial subgraph for providing explanations, our SEGNN achieves comparable explanation performance with state-of-the-art methods. This implies that accurate crucial structure can be derived from the SEGNN's explanations in local structure similarity, which further demonstrates that our SEGNN could give high-quality explanations.

5.4.4 Robustness

Structure noises widely exist in the real world and can significantly degrade the performance of GNNs [15, 37]. SEGNN adopts graph topology in representations learning and local similarity evaluation, which could be affected by noisy edges. Therefore, we



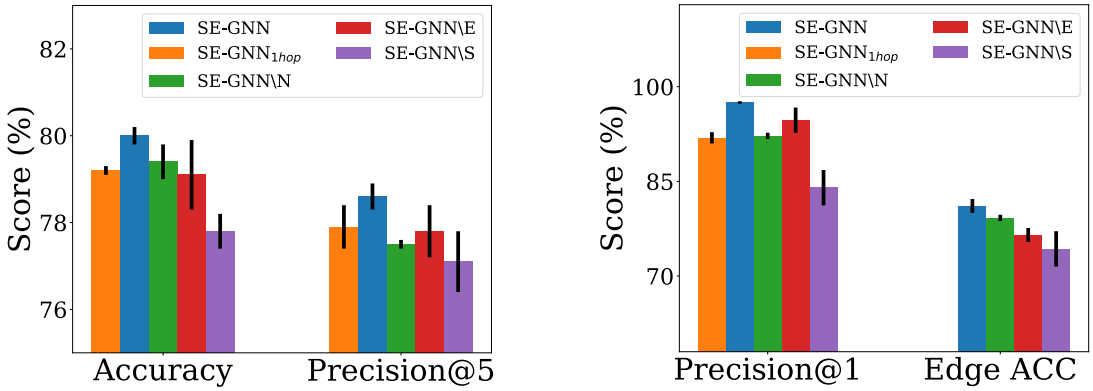
(a) Metattack (b) Random Noise

Figure 5.5: Robustness under different Ptb rates on Citeseer.

conduct experiments on noisy graphs to evaluate the robustness of SEGNN to answer **RQ2**. Experiments are conducted on two types of noisy graphs, i.e., graphs with random noise and non-targeted attack perturbed graphs. For non-targeted attack, we apply *metattack* [15], which poisons the structure of the graphs via meta-learning. The perturbation rate of non-targeted attack and random noise is varied as $\{0\%, 5\%, \ldots, 25\%\}$. The results on Citeseer are shown in Figure 5.5. From this figure, we observe that SEGNN outperforms GCN by a large margin when the perturbation rates are higher. For example, SEGNN achieves over 10% improvements when the perturbation rate of metattack is 25%. And SEGNN even performs better than Pro-GNN which is one of the state-of-the-art robust GNNs against structure noise. This is because:(i) The contrastive learning in SEGNN encourages the representations consistency between the clean graph and randomly perturbed graph. Thus, the learned encoder will not be largely affected by structure noises; (ii) Noisy edges link the nodes that are rarely linked together. Thus, the noise edges generally receive low similarity scores and would not be selected to compute local structure similarity.

5.4.5 Ablation Study

To answer **RQ3**, we conduct ablation study to explore the effects of local structure similarity modeling and self-supervision for explanations. SEGNN utilizes 2-hop local graphs to obtain local structure similarity. To investigate how the similarity modeling will be influenced by the hop of local graphs, we train a variant SEGNN_{1hop} which calculates local structure similarity based on 1-hop local graph. To demonstrate the effectiveness of



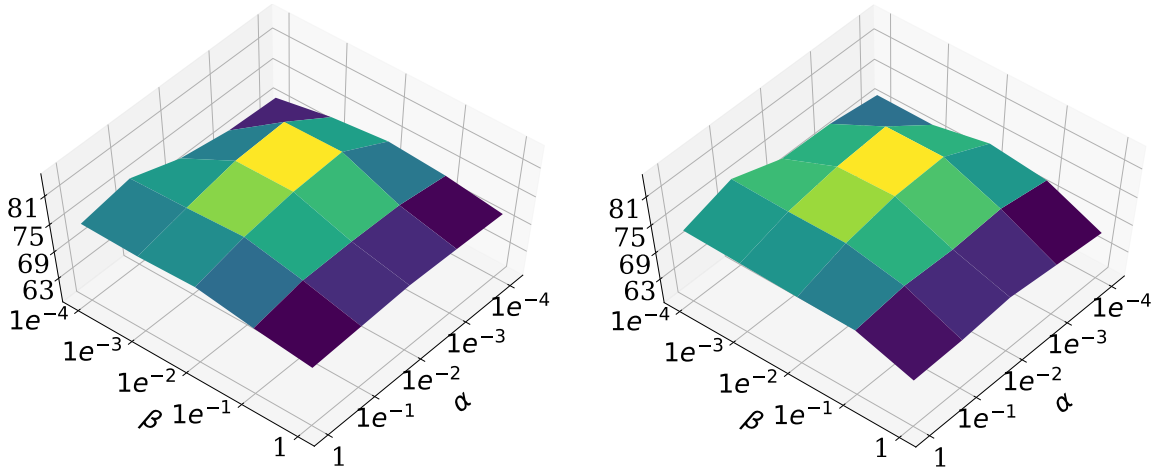
(a) Pubmed (b) Syn-Cora

Figure 5.6: Comparisons with SEGNN and its variants.

self-supervision on node similarity, we set α in objective function Eq.(5.14) as 0 to obtain SEGNN\N. Similarly, we remove the self-supervision on local structure similarity and obtain a variant named as SEGNN\E. We also train a variant SEGNN\S which does not incorporate any self-supervision as the reference. Results on Pubmed and Syn-Cora are presented in Figure 5.6. Since the edge matching in SEGNN$_{1hop}$ only considers 1-hop local graph, Edge ACC on 2-hop local graph is not applicable to SEGNN$_{1hop}$. From the Figure 5.6, we can observe that: (i) The performance of SEGNN$_{1hop}$ is significantly lower than SEGNN in both prediction and explanation. This indicates the importance of incorporating more rich structure information for similarity modeling; and (ii) SEGNN outperforms SEGNN\E and SEGNN\N by a large margin, which implies that the self-supervision on node similarity and local structure similarity is helpful for identifying interpretable K-nearest labeled nodes.

5.4.6 Parameter Sensitivity Analysis

In this subsection, we investigate how the hyperparameter α and β affect the performance of SEGNN, where α and β control the contribution of self-supervision in node similarity modeling and local structure similarity modeling, respectively. We vary α and β as $\{0.0001, 0.001, 0.01, 0.11\}$. We report the classification accuracy and precision@5 of K-nearest labeled nodes on Pubmed to show the effects of α and β on predictions and explanations, respectively. The results are shown in Fig. 5.7. We find that: (i) with the increase of α, the performance in prediction and explanation will first increase and then decrease. When α is too small, little self-supervision is received for node similarity modeling. Low-quality K-nearest labeled nodes are obtained, which results in poor performance in classification and explanation. When α is too large, the overall loss



(a) Accuracy (%)

(b) Precision@5 (%)

Figure 5.7: Parameter sensitive analysis on Pubmed.

function will be dominated by node similarity modeling, which can also lead to a poor overall similarity metric. When α is between 0.001 to 0.01, the performance of SEGNN in prediction and explanation is generally good. (ii) Similarly, with the increasing of β, the performance of SEGNN tends to first increase then decrease. For β, a value between 0.001 to 0.01 generally gives good performance.

Chapter 6 Discussion

6.1 Conclusion

In Chapter 2, we study a novel problem of learning robust GNNs on noisy graphs with limited labeled nodes. Our analysis and experiments demonstrate that noisy edges and limited labeled nodes would largely impair the performance of GNNs. A novel framework named RS-GNN is proposed to mitigate these issues. More specially, we adopt the edges in the noisy graph as supervision to obtain a denoised and densified graph to facilitate the message passing for predictions of unlabeled nodes. Moreover, we also utilize the supervision from the generated graph to explicitly involve the unlabeled nodes. Extensive experiments on real-world datasets demonstrate the robustness of the proposed framework on noisy graphs with limited labeled nodes.

In Chapter 3, we study a novel problem of developing a unified framework that can simultaneously achieve robustness and preserve membership privacy. We verify that IB has potential to eliminate the noises and adversarial perturbations in the data. In addition, IB regularizes the predictions on labeled samples, which can benefit membership privacy. However, the deployment of IB on graphstructured data is challenged by structural noises and shortage of labels in node classification on graphs. To address these issues, we propose a novel graph information bottleneck framework that separately bottlenecks the attribute and neighbor information to handle attribute and structural noises. A self-supervision loss is applied to neighbor bottleneck to further help to filter out adversarial edges and inherent structural noises.

In Chapter 4, we study a novel problem of fair GNN learning with limited sensitive information. We empirically demonstrate that GNNs exhibit severe bias. We propose a novel and flexible framework FairGNN which is able to significantly alleviate the bias issue of GNNs meanwhile maintain high performance on node classification. FairGNN adopts a

sensitive attribute estimator to alleviate the issue of lacking sensitive attribute information. With the estimated sensitive attributes, FairGNN designs adversarial debiasing and covariance constraint to regularize the GNN to have fair node representations and predictions, respectively. We theoretically show that FairGNN can reduce the bias. Experiment results on real-world datasets demonstrate the effectiveness of the proposed framework in terms of both fairness and classification performance.

In Chapter 5, we study a novel problem of self-explainable GNNs by exploring K-nearest labeled nodes. We propose a new framework, which designs intepretable similarity module for finding K-nearest labeled nodes and simultaneously utilizes these nodes for label prediction and explanations. SEGNN also adopts the contrastive learning to benefit the similarity module. Extensive experiments on real-world and synthetic datasets demonstrate the effectiveness of the proposed SEGNN for explainable node classification.

6.2 Future Research Directions

Unified Framework for Trustworthy GNNs. High-stake applications, such as financial analysis, often require multiple dimensions of trustworthiness. However, existing research in trustworthy GNNs mainly focuses on a single dimension of trustworthiness. Methods designed for distinct trustworthy aspects can be conflict with each other in model design, resulting in poor performance. Consequently, a unified framework for trustworthy GNNs is necessary. I have already conducted a preliminary work in unifying adversarial robustness and membership privacy within the graph information bottleneck (GIB) framework [161]. As the principle of GIB is to extract minimal sufficient information, it is promising to further utilize GIB to remove sensitive attribute information for fairness. The extracted minimal sufficient information can be analyzed to interpret the model's decision-making processes, promoting the explainability of graph learning. Another potential direction is to enhance the robustness and fairness with model explanations. I have already worked on self-explainable GNNs [162]. Moving forward, my aspiration is to align the self-explanations generated by models with the rationale provided by humans when making decisions that are both robust and unbiased. To achieve this, research in data construction, techniques of alignment, and self-explainable frameworks for various applications would be investigated.

Trustworthy Model for Science. Graph-structured data is also very pervasive in scientific research such as the studies on molecules, proteins, and gene graphs. Capturing

useful information from such complex topology is a crucial challenge. For instance, the 3D topology of a drug molecule plays a vital role in drug analysis. Hence, adopting trustworthy graph models to process these data could bring breakthroughs to scientific problems in critical areas. I have conducted works in explaining the predictions on molecule graphs with prototype molecules which capture key patterns of molecules with the target property [163]. These explanations could assist researchers in designing chemical compounds. In addition, an explainable GNN that aligns biomedical prior knowledge with the model logic is developed. The preprint is in submission. In the future, I plan to further collaborate with researchers in biomedical, chemistry, physics, and other fields. Leveraging the intersection of these disciplines with my expertise in trustworthy GNNs, I envision significant contributions to pivotal areas such as drug discovery and personalized medicine.

Appendix A
More Details of RM-GIB

A.1 Dataset

Cora, Citeseer, and Pubmed are citation networks, where nodes in the graphs represent the papers and edges denote citation relationship. The attributes of the nodes are the bag-of-words of these papers. For small citation graphs, i.e., Cora and Citeseer, we randomly sample 2% nodes as the training set. For the large citation graph Pubmed, we randomly sample 0.5% nodes as the training set. As for Flickr [101], it is a large-scale graph to categorize the type of images. Each node represents an image and the image description is used as a node attribute. Edges are formed between nodes sharing common properties. We randomly sample 2% nodes from Flickr as the training set. Splits of validation and test sets on all datasets follow the cited papers for consistency. Note that the training node set doesn't overlap with the validation and test sets.

A.2 Baselines

To evaluate the performance in preserving membership privacy, we compare RM-GIB with the following representative and state-of-the-art methods in defending membership inference attacks:

- GCN [4]: This is a representative graph convolutional network which defines graph convolution with spectral analysis.
- GCN+PL [164]: A GCN is firstly trained to obtain pseudo labels. Then, pseudo labels of unlabeled nodes and labels of labeled nodes are used to retrain the GCN.

- GIB [94]: It proposes a graph information bottleneck that regularizes the structural and attribute information in GAT [50].
- Adv-Reg [77]: Min-max game between the training model and the membership inference attacker is introduced as regularization for membership privacy-preserving.
- DP-SGD [78]: This is a differentially private mechanism that adds noises to gradients during optimization for preserving privacy.
- LBP [11]: This is an output perturbation method by adding noise to the posterior before it is released to end users.
- NSD [11]: It randomly chooses neighbors of the queried node in inference to limit the amount of information used in the target model for membership privacy protection.

Apart from GCN and GIB, we also compare the following representative and state-of-the-art robust GNNs to evaluate the robustness of RM-GIB against adversarial attacks on graphs:

- GCN-jaccard [39]: It preprocesses a graph by removing edges linking nodes with low Jaccard feature similarity, then trains a GCN on the preprocessed graph.
- GCN-SVD [60]: It uses a low-rank approximation of the perturbed graph to defend against graph adversarial attacks with the observation that adversarial edges often result in a high-rank adjacency matrix.
- Elastic [75]: Elastic designs a robust message-passing mechanism which incorporates l_1-based graph smoothing in GNNs.
- RSGNN [76]: This is a state-of-the-art robust GNN that denoises and densifies the noisy graph to give robust predictions.

A.3 Implementation Details

A 2-layer MLP is deployed as the attribute bottleneck. The neighbor bottleneck also uses a 2-layer MLP. As for the predictor, we use a 2-layer GCN without on default. The mutual information estimator used for self-supervision on neighbor bottleneck also deploys a 2-layer MLP. All the hidden dimensions of the neural networks are set as 256. For the hyperparameter T which is the threshold to determine the negative neighbors

Table A.1: Results of defending membership inference attack (Accuracy(%)↑ | MIA-F ROC(%) ↓) with various label rates.

Dataset	Method	2%	4%	6%	8%
Cora	GCN	73.2±0.8 | 89.4±0.5	79.4±0.2 | 81.3±1.4	81.2±0.2 | 78.0±0.4	**82.1±0.3** | 74.3±0.1
	GIB	72.5±0.7 | 86.6±0.8	78.8±0.5 | 78.6±0.7	80.6±1.5 | 71.4±1.8	80.9±0.8 | 67.8±0.6
	RM-GIB	**78.5±0.6** | **56.4±0.2**	**79.6±0.6** | **56.9±0.3**	**81.9±0.4** | **55.9±0.6**	81.9±0.3 | **54.4±1.0**
Citeseer	GCN	70.2±0.2 | 88.5±1.8	71.3±0.4 | 83.1±0.3	73.6±0.1 | 76.0±0.3	73.9±0.1 | 73.2±0.1
	GIB	70.1±1.1 | 87.4±0.6	72.1±0.6 | 80.4±2.1	74.8±0.5 | 70.9±0.3	74.6±0.7 | 69.9±0.9
	RM-GIB	**73.9±0.6** | **55.2±0.8**	**73.6±0.8** | **53.0±0.1**	**76.1±0.3** | **50.3±0.7**	**76.4±0.7** | **50.2±1.8**
Pubmed	GCN	81.0±0.1 | 56.6±0.1	82.8±0.4 | 56.6±0.1	83.9±0.1 | 54.9±0.1	85.3±0.1 | 53.0±0.1
	GIB	81.9±0.1 | 56.1±0.2	84.0±0.2 | 53.7±0.4	85.1±0.3 | 52.0±0.1	85.5±0.8 | 51.3±0.1
	RM-GIB	**84.0±0.1** | **50.3±0.5**	**85.2±0.4** | **49.8±0.7**	**85.9±0.3** | **50.1±0.3**	**86.4±0.2** | **50.1±0.1**

for self-supervision, it is set as 0.5 for all experiments. As for the hyperparameters β and γ used in the final objective function Eq.(3.20), they are selected based on accuracy on the validation set with grid search. Specifically, we vary β and γ as $\{0.0003, 0.0001, 0.003, 0.001, 0.03, 0.1\}$ and $\{0.00001, 0.0001, 0.001, 0.01, 0.1\}$, respectively.

A.4 Proof Details

Recall that in IB, for a given training sample $(\mathbf{x}_n, y_n)$, its distribution of $\mathbf{z}$ is obtained by $P(\mathbf{z}|\mathbf{x}_n, y_n; \theta) = f_\theta(\mathbf{z}, \mathbf{x}_n)$, where $f_\theta(\mathbf{z}, \mathbf{x}_n)$ is the probability density function modeled by the nerual network with parameters θ. In the practice of computing mutual information, $P(\mathbf{x}, y, \mathbf{z}; \theta)$ is approximated with the empirical data distribution $P(\mathbf{x}, y, \mathbf{z}; \theta) = \frac{1}{N}\sum_{n=1}^{N} \delta_{\mathbf{x}_n}(\mathbf{x})\delta_{y_n}(y) f_\theta(\mathbf{z}, \mathbf{x}_n)$, where $\delta()$ is the Dirac function. Then, we can have the following equations:

$$P(\mathbf{x}, \mathbf{z}; \theta) = \int_y P(\mathbf{x}, y, \mathbf{z}; \theta) dY = \frac{1}{N}\sum_{n=1}^{N} \delta_{\mathbf{x}_n}(\mathbf{x}) f_\theta(\mathbf{z}, \mathbf{x}_n) \tag{A.1}$$

$$P(\mathbf{x}, y; \theta) = \int_{\mathbf{z}} P(\mathbf{x}, y, \mathbf{z}; \theta) d\mathbf{z} = \frac{1}{N}\sum_{n=1}^{N} \delta_{\mathbf{x}_n}(\mathbf{x})\delta_{y_n}(y) \tag{A.2}$$

$$P(\mathbf{x}; \theta) = \int_y P(\mathbf{x}, y; \theta) d\mathbf{x} = \frac{1}{N}\sum_{n=1}^{N} \delta_{\mathbf{x}_n}(\mathbf{x}) \tag{A.3}$$

The $I_\theta(\mathbf{z}; y|\mathbf{x})$ can be computed by:

$$
\begin{aligned}
&I_\theta(\mathbf{z}; y|\mathbf{x}) \\
&= \int_{\mathbf{x}} \int_y \int_{\mathbf{z}} p(\mathbf{x}, y, \mathbf{z}; \theta) \log \frac{P(\mathbf{x}; \theta) P(\mathbf{x}, y, \mathbf{z}; \theta)}{P(\mathbf{x}, y; \theta) P(\mathbf{x}, \mathbf{z}; \theta)} d\mathbf{x} dy d\mathbf{z} \\
&= \frac{1}{N} \int_{\mathbf{x}} \int_y \int_{\mathbf{z}} \Big(\sum_{n=1}^{N} \delta_{\mathbf{x}_n}(\mathbf{x}) \delta_{y_n}(y) f_\theta(\mathbf{z}, \mathbf{x}_n) \Big) \\
&\cdot \log \frac{\Big(\sum_{n=1}^{N} \delta_{\mathbf{x}_n}(\mathbf{x}) \Big) \cdot \Big(\sum_{n=1}^{N} \delta_{\mathbf{x}_n}(\mathbf{x}) \delta_{y_n}(y) f_\theta(\mathbf{z}, \mathbf{x}_n) \Big)}{\Big(\sum_{n=1}^{N} \delta_{\mathbf{x}_n}(\mathbf{x}) \delta_{y_n}(y) \Big) \cdot \Big(\sum_{n=1}^{N} \delta_{\mathbf{x}_n}(\mathbf{x}) f_\theta(\mathbf{z}, \mathbf{x}_n) \Big)} d\mathbf{x} dy d\mathbf{z} \\
&= \frac{1}{N} \int_{\mathbf{z}} \sum_{n=1}^{N} f_\theta(\mathbf{z}, \mathbf{x}_n) \log \frac{f_\theta(\mathbf{z}, \mathbf{x}_n)}{f_\theta(\mathbf{z}, \mathbf{x}_n)} = 0
\end{aligned}
\tag{A.4}
$$

Based on the above proof, we verify that $I_\theta(\mathbf{z}; y|\mathbf{x}) = 0$ regardless the value of model parameters. Thus, we can derive the first line of Eq.(3.3) in our paper.

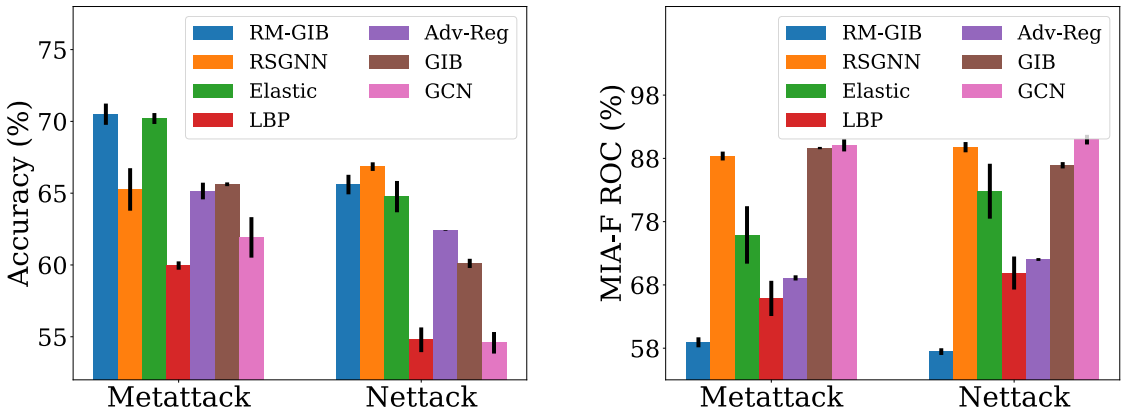
A.5 Time Complexity Analysis

We analyze the time complexity of the proposed RM-GIB in the following. The time complexity mainly comes from the pretraining of self-supervisor for the neighbor bottleneck, and the training of RM-GIB. Let h and K denote the embedding dimension and training epochs, respectively. The cost of training the self-supervisor is approximately $O(Khd|\mathcal{V}|)$, where d is the average degree of nodes and $|\mathcal{V}|$ is the number of nodes in the graph. Next, we analyze the time complexity of the optimization of RM-GIB. The time complexity of attribute bottleneck and neighbor bottleneck in each epoch are $O(h|\mathcal{V}|)$ and $O(hd|\mathcal{V}|)$, respectively. As for the computation cost of the predictor is approximately $O(hd|\mathcal{V}|)$ in each epoch. The privacy-preserving optimization requires firstly training RM-GIB for pseudo label collection followed by the optimization on the enlarged label set. Hence, the time complexity of optimizing RM-GIB is $O(2Kh(2d+1)|\mathcal{V}|)$. Combining the training of self-supervisor, the overall time complexity for training is $O(Kh(4d+3)|\mathcal{V}|)$. Our RM-GIB is linear to the size of the graph, which proves its scalability.

A.6 Additional Experimental Results

The additional results on the perturbed Cora graph are shown in Fig. A.1, which have the same observations as Fig. 3.3.

Impacts of Label Rates. We add the experiments that vary label rates by {2%, 4%, 6%, 8%} to verify our motivation and the effectiveness of our RM-GIB. All the



(a) Accuracy on Cora

(b) MIA-F ROC on Cora

Figure A.1: Additional results on the perturbed Cora.

Table A.2: Impacts of labels rates in defending metattack.

		2%	4%	6%	8%
Cora	GCN	62.7±0.6	71.9±0.2	76.0±0.2	77.7±0.3
	GIB	65.6±0.1	74.0±0.7	77.5±1.0	78.4±0.5
	RM-GIB	**71.1±0.6**	**75.7±0.6**	**78.4±0.5**	**79.6±0.6**
Citeseer	GCN	66.1±0.5	67.9±1.9	68.3±0.8	71.1±0.4
	GIB	66.8±0.7	68.90.7	69.4±0.2	72.2±0.5
	RM-GIB	**72.1±0.9**	**71.9±0.9**	**74.5±0.9**	**74.6±0.3**
Pubmed	GCN	70.3±0.1	72.1±0.1	72.9±0.1	74.1±0.3
	GIB	70.8±0.2	73.4±0.2	74.4±0.2	75.2±0.3
	RM-GIB	**81.2±0.2**	**81.9±0.3**	**83.1±0.5**	**84.4±0.6**

hyperparameters of GCN, GIB, and our RM-GIB are tuned on the validation set for a fair comparison. The results are presented in Table A.1. We can observe that:

- When the label rates are small, GIB gives high MIA-F ROC scores and marginally outperforms GCN in privacy preservation. This verifies that GIB is vulnerable to membership inference attack under a semi-supervised learning setting.

- Our method RM-GIB can consistently achieve a very low MIA-F ROC score (close to 50%) with different sizes of labeled nodes. This demonstrates the effectiveness of our RM-GIB in membership privacy preservation under different data settings.

We also show the accuracy (%)) of defending metattack (20% perturbation rate) under different label rates in Tab. A.2. Our RM-GIB consistently performs better than

Table A.3: Results (%) of varying pseudo label Sizes.

	5%	10%	20%	50%	100%
MIA-F ROC	68.0±0.9	63.2±3.3	62.0±4.3	58.1±2.1	54.8±3.1
Accuracy	68.1±0.5	69.8±1.2	70.5±0.3	71.1±0.6	71.7±0.4

Table A.4: Accuracy on attribute-perturbed only graphs.

Dataset	GCN	GIB	RM-GIB
Cora	70.3±1.3	74.3±0.2	**78.2±0.7**
Citeseer	70.7±0.5	71.6±0.2	**73.9±0.9**

GIB by a large margin in defending graph adversarial attacks given different sizes of labels.

Varying Sizes of Pseudo Labels. We vary the rates of unlabeled nodes used for the pseudo-label generation by {5%, 10%, 20%, 50%, 100%} . Experiments are conducted on the Cora graph. For the adversarial attacks, we apply metattack with 20% perturbation rate. All other settings are the same as the description in Sec. 3.5.1. The results are shown in Tab. A.3. We can observe from the results that with the increase of pseudo labels, the performance in defending membership inference attack and adversarial attacks will both increase. This demonstrates the effectiveness of incorporating pseudo labels. It justifies that we should generate pseudo labels for all the unlabeled nodes in the graph.

Results on Attribute Perturbation. we conduct experiments on attribute-perturbed only graphs to empirically verify the effectiveness of our methods in defending against noises in attributes. We apply metattack to poison the attributes of the Cora and Citeseer graphs with the perturbation rate set as 20%. The other settings are the same as the description in Sec. 3.5.1. The results are shown in Tab. A.4, where we can observe that our RM-GIB performs better than GIB on attribute-perturbed graphs. This verifies the effectiveness of our method in defending noises in node attributes.

Bibliography

[1] Kawahara, J., C. J. Brown, S. P. Miller, B. G. Booth, V. Chau, R. E. Grunau, J. G. Zwicker, and G. Hamarneh (2017) "BrainNetCNN: Convolutional neural networks for brain networks; towards predicting neurodevelopment," *NeuroImage*, **146**, pp. 1038–1049.

[2] Wang, J., S. Zhang, Y. Xiao, and R. Song (2021) "A Review on Graph Neural Network Methods in Financial Applications," *arXiv preprint arXiv:2111.15367*.

[3] Hamilton, W., Z. Ying, and J. Leskovec (2017) "Inductive representation learning on large graphs," in *NeurIPS*, pp. 1024–1034.

[4] Kipf, T. N. and M. Welling (2016) "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*.

[5] Chen, J., T. Ma, and C. Xiao (2018) "Fastgcn: fast learning with graph convolutional networks via importance sampling," *arXiv preprint arXiv:1801.10247*.

[6] Xiao, T., Z. Chen, D. Wang, and S. Wang (2021) "Learning how to propagate messages in graph neural networks," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1894–1903.

[7] Lv, L., J. Cheng, N. Peng, M. Fan, D. Zhao, and J. Zhang (2019) "Auto-encoder based graph convolutional networks for online financial anti-fraud," in *2019 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, IEEE, pp. 1–6.

[8] Harl, M., S. Weinzierl, M. Stierle, and M. Matzner (2020) "Explainable predictive business process monitoring using gated graph neural networks," *Journal of Decision Systems*, **29**(sup1), pp. 312–327.

[9] Li, X., Y. Zhou, N. Dvornek, M. Zhang, S. Gao, J. Zhuang, D. Scheinost, L. H. Staib, P. Ventola, and J. S. Duncan (2021) "Braingnn: Interpretable brain graph neural network for fmri analysis," *Medical Image Analysis*, **74**, p. 102233.

[10] Fan, W., Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin (2019) "Graph neural networks for social recommendation," in *The World Wide Web Conference*, pp. 417–426.

[11] Olatunji, I. E., W. Nejdl, and M. Khosla (2021) "Membership inference attack on graph neural networks," in *TPS-ISA*, IEEE, pp. 11–20.

[12] Dai, E. and S. Wang (2021) "Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 680–688.

[13] Smuha, N. (2019) "Ethics guidelines for trustworthy AI," in *AI & Ethics, Date: 2019/05/28-2019/05/28, Location: Brussels (Digityser), Belgium*.

[14] Xu, K., W. Hu, J. Leskovec, and S. Jegelka (2018) "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*.

[15] Zügner, D., A. Akbarnejad, and S. Günnemann (2018) "Adversarial attacks on neural networks for graph data," in *SIGKDD*, pp. 2847–2856.

[16] Hu, H., Z. Salcic, L. Sun, G. Dobbie, P. S. Yu, and X. Zhang (2022) "Membership inference attacks on machine learning: A survey," *ACM Computing Surveys (CSUR)*, **54**(11s), pp. 1–37.

[17] Gallagher, B., H. Tong, T. Eliassi-Rad, and C. Faloutsos (2008) "Using ghost edges for classification in sparsely labeled networks," in *SIGKDD*, pp. 256–264.

[18] Dwork, C., M. Hardt, T. Pitassi, O. Reingold, and R. Zemel (2012) "Fairness through awareness," in *ITCS*, pp. 214–226.

[19] Beutel, A., J. Chen, Z. Zhao, and E. H. Chi (2017) "Data decisions and theoretical implications when adversarially learning fair representations," *arXiv preprint arXiv:1707.00075*.

[20] Creager, E., D. Madras, J.-H. Jacobsen, M. A. Weis, K. Swersky, T. Pitassi, and R. Zemel (2019) "Flexibly fair representation learning by disentanglement," *arXiv preprint arXiv:1906.02589*.

[21] Dong, Y., O. Lizardo, and N. V. Chawla (2016) "Do the Young Live in a" Smaller World" Than the Old? Age-Specific Degrees of Separation in a Large-Scale Mobile Communication Network," *arXiv preprint arXiv:1606.07556*.

[22] Rahman, T. A., B. Surma, M. Backes, and Y. Zhang (2019) "Fairwalk: Towards Fair Graph Embedding." in *IJCAI*, pp. 3289–3295.

[23] Mehrabi, N., F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan (2019) "A survey on bias and fairness in machine learning," *arXiv preprint arXiv:1908.09635*.

[24] Suresh, H. and J. V. Guttag (2019) "A framework for understanding unintended consequences of machine learning," *arXiv preprint arXiv:1901.10002*.

[25] Locatello, F., G. Abbati, T. Rainforth, S. Bauer, B. Schölkopf, and O. Bachem (2019) "On the fairness of disentangled representations," in *NeurIPS*, pp. 14584–14597.

[26] Louizos, C., K. Swersky, Y. Li, M. Welling, and R. Zemel (2015) "The variational fair autoencoder," *arXiv preprint arXiv:1511.00830*.

[27] Ribeiro, M. T., S. Singh, and C. Guestrin (2016) "" Why should i trust you?" Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144.

[28] Selvaraju, R. R., M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra (2017) "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, pp. 618–626.

[29] Simonyan, K., A. Vedaldi, and A. Zisserman (2013) "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*.

[30] Yang, F., S. K. Pentyala, S. Mohseni, M. Du, H. Yuan, R. Linder, E. D. Ragan, S. Ji, and X. Hu (2019) "XFake: explainable fake news detector with visualizations," in *The World Wide Web Conference*, pp. 3600–3604.

[31] Ying, R., D. Bourgeois, J. You, M. Zitnik, and J. Leskovec (2019) "Gnnexplainer: Generating explanations for graph neural networks," *Advances in neural information processing systems*, **32**, p. 9240.

[32] Pope, P. E., S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann (2019) "Explainability methods for graph convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10772–10781.

[33] Luo, D., W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang (2020) "Parameterized explainer for graph neural network," *arXiv preprint arXiv:2011.04573*.

[34] Bruna, J., W. Zaremba, A. Szlam, and Y. LeCun (2013) "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*.

[35] Wang, D., J. Lin, P. Cui, Q. Jia, Z. Wang, Y. Fang, Q. Yu, J. Zhou, S. Yang, and Y. Qi (2019) "A Semi-supervised Graph Attentive Network for Financial Fraud Detection," in *ICDM*, IEEE, pp. 598–607.

[36] Wang, H., F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, and Z. Wang (2019) "Knowledge-aware graph neural networks with label smoothness regularization for recommender systems," in *SIGKDD*, pp. 968–977.

[37] Zügner, D. and S. Günnemann (2019) "Adversarial attacks on graph neural networks via meta learning," *arXiv preprint arXiv:1902.08412*.

[38] Ferrara, E., O. Varol, C. Davis, F. Menczer, and A. Flammini (2016) "The rise of social bots," *Communications of the ACM*, **59**(7), pp. 96–104.

[39] Wu, H., C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, and L. Zhu (2019) "Adversarial examples on graph data: Deep insights into attack and defense," *arXiv preprint arXiv:1903.01610*.

[40] Jin, W., Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang (2020) "Graph structure learning for robust graph neural networks," in *SIGKDD*, pp. 66–74.

[41] Tang, X., Y. Li, Y. Sun, H. Yao, P. Mitra, and S. Wang (2020) "Transferring Robustness for Graph Neural Network Against Poisoning Attacks," in *WWW*, pp. 600–608.

[42] Zhu, D., Z. Zhang, P. Cui, and W. Zhu (2019) "Robust graph convolutional networks against adversarial attacks," in *SIGKDD*, pp. 1399–1407.

[43] Sun, K., Z. Zhu, and Z. Lin (2020) "Multi-stage self-supervised learning for graph convolutional networks," *AAAI*.

[44] Li, Q., Z. Han, and X.-M. Wu (2018) "Deeper insights into graph convolutional networks for semi-supervised learning," in *AAAI*.

[45] Peng, Z., Y. Dong, M. Luo, X.-M. Wu, and Q. Zheng (2020) "Self-Supervised Graph Representation Learning via Global Context Prediction," *arXiv preprint arXiv:2003.01604*.

[46] Liben-Nowell, D. and J. Kleinberg (2007) "The link-prediction problem for social networks," *JASIST*, **58**(7), pp. 1019–1031.

[47] Wang, Z., Q. Lv, X. Lan, and Y. Zhang (2018) "Cross-lingual knowledge graph alignment via graph convolutional networks," in *EMNLP*, pp. 349–357.

[48] Levie, R., F. Monti, X. Bresson, and M. M. Bronstein (2018) "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, **67**(1), pp. 97–109.

[49] Henaff, M., J. Bruna, and Y. LeCun (2015) "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*.

[50] Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio (2017) "Graph attention networks," *arXiv preprint arXiv:1710.10903*.

[51] Chiang, W.-L., X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh (2019) "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks," in *SIGKDD*, pp. 257–266.

[52] Niepert, M., M. Ahmed, and K. Kutzkov (2016) "Learning convolutional neural networks for graphs," in *ICML*, pp. 2014–2023.

[53] Gilmer, J., S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl (2017) "Neural message passing for quantum chemistry," *arXiv preprint arXiv:1704.01212*.

[54] Chen, M., Z. Wei, Z. Huang, B. Ding, and Y. Li (2020) "Simple and deep graph convolutional networks," in *ICML*, PMLR, pp. 1725–1735.

[55] Zhu, Q., B. Du, and P. Yan (2020) "Self-supervised Training of Graph Convolutional Networks," *arXiv preprint arXiv:2006.02380*.

[56] Jin, W., T. Derr, H. Liu, Y. Wang, S. Wang, Z. Liu, and J. Tang (2020) "Self-supervised learning on graphs: Deep insights and new direction," *arXiv preprint arXiv:2006.10141*.

[57] Chen, Z.-M., X.-S. Wei, P. Wang, and Y. Guo (2019) "Multi-label image recognition with graph convolutional networks," in *CVPR*, pp. 5177–5186.

[58] Jiang, B., Z. Zhang, D. Lin, J. Tang, and B. Luo (2019) "Semi-supervised learning with graph learning-convolutional networks," in *CVPR*, pp. 11313–11320.

[59] Dai, H., H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song (2018) "Adversarial attack on graph structured data," *ICML*.

[60] Entezari, N., S. A. Al-Sayouri, A. Darvishzadeh, and E. E. Papalexakis (2020) "All You Need Is Low (Rank) Defending Against Adversarial Attacks on Graphs," in *WSDM*, pp. 169–177.

[61] Sen, P., G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad (2008) "Collective classification in network data," *AI magazine*, **29**(3), pp. 93–93.

[62] McCallum, A. K., K. Nigam, J. Rennie, and K. Seymore (2000) "Automating the construction of internet portals with machine learning," *Information Retrieval*, **3**(2), pp. 127–163.

[63] Gallagher, B. and T. Eliassi-Rad (2008) "Leveraging label-independent features for classification in sparsely labeled networks: An empirical study," in *International Workshop on Social Network Mining and Analysis*, Springer, pp. 1–19.

[64] He, X., L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua (2017) "Neural collaborative filtering," in *WWW*, pp. 173–182.

[65] Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013) "Distributed representations of words and phrases and their compositionality," in *NeurIPS*, pp. 3111–3119.

[66] Li, Y., W. Jin, H. Xu, and J. Tang (2020) "DeepRobust: A PyTorch Library for Adversarial Attacks and Defenses," *arXiv preprint arXiv:2005.06149*.

[67] Irwin, J. J., T. Sterling, M. M. Mysinger, E. S. Bolstad, and R. G. Coleman (2012) "ZINC: a free tool to discover chemistry for biology," *Journal of chemical information and modeling*, **52**(7), pp. 1757–1768.

[68] Zou, X., Q. Zheng, Y. Dong, X. Guan, E. Kharlamov, J. Lu, and J. Tang "Tdgia: Effective injection attacks on graph neural networks," in *SIGKDD, pages=2461–2471, year=2021*.

[69] Shokri, R. and V. Shmatikov (2015) "Privacy-preserving deep learning," in *CCS*, pp. 1310–1321.

[70] Lu, H. and S. Uddin (2021) "A weighted patient network-based framework for predicting chronic diseases using graph neural networks," *Scientific reports*, **11**(1), p. 22607.

[71] Zhang, X. and M. Zitnik (2020) "GNNGuard: Defending Graph Neural Networks against Adversarial Attacks," in *NeurIPS*, vol. 33, pp. 9263–9275.

[72] Dai, E., T. Zhao, H. Zhu, J. Xu, Z. Guo, H. Liu, J. Tang, and S. Wang (2022) "A Comprehensive Survey on Trustworthy Graph Neural Networks: Privacy, Robustness, Fairness, and Explainability," *arXiv preprint arXiv:2204.08570*.

[73] Chen, L., J. Li, Q. Peng, Y. Liu, Z. Zheng, and C. Yang (2021) "Understanding structural vulnerability in graph convolutional networks," *arXiv preprint arXiv:2108.06280*.

[74] Geisler, S., T. Schmidt, H. Şirin, D. Zügner, A. Bojchevski, and S. Günnemann (2021) "Robustness of graph neural networks at scale," *NeurIPS*, **34**, pp. 7637–7649.

[75] Liu, X., W. Jin, Y. Ma, Y. Li, H. Liu, Y. Wang, M. Yan, and J. Tang (2021) "Elastic graph neural networks," in *ICML*, PMLR, pp. 6837–6849.

[76] Dai, E., W. Jin, H. Liu, and S. Wang (2022) "Towards robust graph neural networks for noisy graphs with sparse labels," in *WSDM*, pp. 181–191.

[77] Nasr, M., R. Shokri, and A. Houmansadr (2018) "Machine learning with membership privacy using adversarial regularization," in *CCS*, pp. 634–646.

[78] Abadi, M., A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang (2016) "Deep learning with differential privacy," in *CCS*, pp. 308–318.

[79] Papernot, N., M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar (2016) "Semi-supervised knowledge transfer for deep learning from private training data," *arXiv preprint arXiv:1610.05755*.

[80] Tishby, N., F. C. Pereira, and W. Bialek (2000) "The information bottleneck method," *arXiv preprint physics/0004057*.

[81] Alemi, A. A., I. Fischer, J. V. Dillon, and K. Murphy (2016) "Deep variational information bottleneck," *arXiv preprint arXiv:1612.00410*.

[82] Dai, E., M. Lin, X. Zhang, and S. Wang (2023) "Unnoticeable Backdoor Attacks on Graph Neural Networks," in *WWW*, pp. 2263–2273.

[83] Xu, K., H. Chen, S. Liu, P.-Y. Chen, T.-W. Weng, M. Hong, and X. Lin (2019) "Topology attack and defense for graph neural networks: An optimization perspective," *arXiv preprint arXiv:1906.04214*.

[84] Li, K., Y. Liu, X. Ao, J. Chi, J. Feng, H. Yang, and Q. He (2022) "Reliable Representations Make A Stronger Defender: Unsupervised Structure Refinement for Robust GNN," in *SIGKDD*, pp. 925–935.

[85] Shokri, R., M. Stronati, C. Song, and V. Shmatikov (2017) "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*, IEEE, pp. 3–18.

[86] Hayes, J., L. Melis, G. Danezis, and E. De Cristofaro (2017) "Logan: Membership inference attacks against generative models," *arXiv preprint arXiv:1705.07663*.

[87] Choquette-Choo, C. A., F. Tramer, N. Carlini, and N. Papernot (2021) "Label-only membership inference attacks," in *ICML*, PMLR, pp. 1964–1974.

[88] Salem, A., Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes (2018) "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," *arXiv preprint arXiv:1806.01246*.

[89] Chaudhuri, K., C. Monteleoni, and A. D. Sarwate (2011) "Differentially private empirical risk minimization." *JMLR*, **12**(3).

[90] He, X., R. Wen, Y. Wu, M. Backes, Y. Shen, and Y. Zhang (2021) "Node-level membership inference attacks against graph neural networks," *arXiv preprint arXiv:2102.05429*.

[91] Wu, B., X. Yang, S. Pan, and X. Yuan (2021) "Adapting membership inference attacks to gnn for graph classification: Approaches and implications," in *ICDM*, IEEE, pp. 1421–1426.

[92] Wang, Z., T. Jian, A. Masoomi, S. Ioannidis, and J. Dy (2021) "Revisiting Hilbert-Schmidt Information Bottleneck for Adversarial Robustness," *NeurIPS*, **34**, pp. 586–597.

[93] Kim, J., B.-K. Lee, and Y. M. Ro (2021) "Distilling robust and non-robust features in adversarial examples by information bottleneck," *NeurIPS*, **34**, pp. 17148–17159.

[94] Wu, T., H. Ren, P. Li, and J. Leskovec (2020) "Graph information bottleneck," *NeurIPS*, **33**, pp. 20437–20448.

[95] Sun, Q., J. Li, H. Peng, J. Wu, X. Fu, C. Ji, and S. Y. Philip (2022) "Graph structure learning with variational information bottleneck," in *AAAI*, vol. 36, pp. 4165–4174.

[96] Yu, J., T. Xu, Y. Rong, Y. Bian, J. Huang, and R. He (2020) "Graph information bottleneck for subgraph recognition," *arXiv preprint arXiv:2010.05563*.

[97] Miao, S., M. Liu, and P. Li (2022) "Interpretable and generalizable graph learning via stochastic attention mechanism," in *ICML*, PMLR, pp. 15524–15543.

[98] Jang, E., S. Gu, and B. Poole (2016) "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*.

[99] Wu, F., A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger (2019) "Simplifying graph convolutional networks," in *ICML*, PMLR, pp. 6861–6871.

[100] Hjelm, R. D., A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio (2018) "Learning deep representations by mutual information estimation and maximization," *arXiv preprint arXiv:1808.06670*.

[101] Zeng, H., H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna (2020) "GraphSAINT: Graph Sampling Based Inductive Learning Method," in *ICLR*.

[102] Hamaguchi, T., H. Oiwa, M. Shimbo, and Y. Matsumoto (2017) "Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach," *arXiv preprint arXiv:1706.05674*.

[103] Yao, L., C. Mao, and Y. Luo (2019) "Graph convolutional networks for text classification," in *AAAI*, vol. 33, pp. 7370–7377.

[104] Ying, R., R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec (2018) "Graph convolutional neural networks for web-scale recommender systems," in *SIGKDD*, pp. 974–983.

[105] Berg, R. v. d., T. N. Kipf, and M. Welling (2017) "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*.

[106] Madden, M., A. Lenhart, S. Cortesi, U. Gasser, M. Duggan, A. Smith, and M. Beaton (2013) "Teens, social media, and privacy," *Pew Research Center*, **21**(1055), pp. 2–86.

[107] Zhang, L., Y. Wu, and X. Wu (2017) "Achieving non-discrimination in data release," in *SIGKDD*, pp. 1335–1344.

[108] Kamiran, F. and T. Calders (2009) "Classifying without discriminating," in *ICCC*, IEEE, pp. 1–6.

[109] ——— (2012) "Data preprocessing techniques for classification without discrimination," *KAIS*, **33**(1), pp. 1–33.

[110] Edwards, H. and A. Storkey (2015) "Censoring representations with an adversary," *arXiv preprint arXiv:1511.05897*.

[111] Zafar, M. B., I. Valera, M. G. Rodriguez, and K. P. Gummadi (2015) "Fairness constraints: Mechanisms for fair classification," *arXiv preprint arXiv:1507.05259*.

[112] Zafar, M. B., I. Valera, M. Gomez Rodriguez, and K. P. Gummadi (2017) "Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment," in *WWW*, pp. 1171–1180.

[113] Bose, A. J. and W. L. Hamilton (2019) "Compositional fairness constraints for graph embeddings," *arXiv preprint arXiv:1905.10674*.

[114] Zhang, B. H., B. Lemoine, and M. Mitchell (2018) "Mitigating unwanted biases with adversarial learning," in *AIES*, pp. 335–340.

[115] Madras, D., E. Creager, T. Pitassi, and R. Zemel (2018) "Learning adversarially fair and transferable representations," *arXiv preprint arXiv:1802.06309*.

[116] Tang, X., H. Yao, Y. Sun, Y. Wang, J. Tang, C. Aggarwal, P. Mitra, and S. Wang (2020) "Investigating and Mitigating Degree-Related Biases in Graph Convoltuional Networks," in *CIKM*, pp. 1435–1444.

[117] Zhao, T., X. Tang, X. Zhang, and S. Wang (2020) "Semi-Supervised Graph-to-Graph Translation," in *CIKM*, pp. 1863–1872.

[118] Sun, Y., S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar (2019) "Node injection attacks on graphs via reinforcement learning," *WWW*.

[119] Defferrard, M., X. Bresson, and P. Vandergheynst (2016) "Convolutional neural networks on graphs with fast localized spectral filtering," in *NeurIPS*, pp. 3844–3852.

[120] Hardt, M., E. Price, and N. Srebro (2016) "Equality of opportunity in supervised learning," in *NeurIPS*, pp. 3315–3323.

[121] Feldman, M., S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian (2015) "Certifying and removing disparate impact," in *SIGKDD*, pp. 259–268.

[122] Xu, D., S. Yuan, L. Zhang, and X. Wu (2018) "Fairgan: Fairness-aware generative adversarial networks," in *Big Data*, IEEE, pp. 570–575.

[123] ——— (2019) "FairGAN+: Achieving Fair Data Generation and Classification through Generative Adversarial Nets," in *Big Data*, IEEE, pp. 1401–1406.

[124] Sattigeri, P., S. C. Hoffman, V. Chenthamarakshan, and K. R. Varshney (2019) "Fairness GAN: Generating datasets with fairness properties using a generative adversarial network," *IBM Journal of Research and Development*, **63**(4/5), pp. 3–1.

[125] Zemel, R., Y. Wu, K. Swersky, T. Pitassi, and C. Dwork (2013) "Learning fair representations," in *ICML*, pp. 325–333.

[126] Kamishima, T., S. Akaho, and J. Sakuma (2011) "Fairness-aware learning through regularization approach," in *ICDMW*, IEEE, pp. 643–650.

[127] Pleiss, G., M. Raghavan, F. Wu, J. Kleinberg, and K. Q. Weinberger (2017) "On fairness and calibration," in *NeurIPS*, pp. 5680–5689.

[128] Grover, A. and J. Leskovec (2016) "node2vec: Scalable feature learning for networks," in *SIGKDD*, pp. 855–864.

[129] Takac, L. and M. Zabovsky (2012) "Data analysis in public social networks," in *International scientific conference and international workshop present day trends of innovations*, vol. 1.

[130] Wang, H. and J. Leskovec (2020) "Unifying graph convolutional neural networks and label propagation," *arXiv preprint arXiv:2002.06755*.

[131] Liao, J., C. Huang, P. Kairouz, and L. Sankar (2019) "Learning generative adversarial representations (GAP) under fairness and censoring constraints," *arXiv preprint arXiv:1910.00411*.

[132] Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014) "Generative adversarial nets," in *NeurIPS*, pp. 2672–2680.

[133] Arjovsky, M. and L. Bottou (2017) "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*.

[134] Kingma, D. P. and J. Ba (2014) "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*.

[135] Salimans, T., I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen (2016) "Improved techniques for training gans," in *NeurIPS*, pp. 2234–2242.

[136] Perozzi, B., R. Al-Rfou, and S. Skiena (2014) "Deepwalk: Online learning of social representations," in *SIGKDD*, pp. 701–710.

[137] Shu, K., L. Cui, S. Wang, D. Lee, and H. Liu (2019) "defend: Explainable fake news detection," in *SIGKDD*, pp. 395–405.

[138] Papernot, N. and P. McDaniel (2018) "Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning," *arXiv preprint arXiv:1803.04765*.

[139] Huang, Q., M. Yamada, Y. Tian, D. Singh, D. Yin, and Y. Chang (2020) "Graphlime: Local interpretable model explanations for graph neural networks," *arXiv preprint arXiv:2001.06216*.

[140] Bongini, P., M. Bianchini, and F. Scarselli (2021) "Molecular generative Graph Neural Networks for Drug Discovery," *Neurocomputing*, **450**, pp. 242–252.

[141] Zhao, T., X. Zhang, and S. Wang (2021) "GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks," in *WSDM*, pp. 833–841.

[142] Dai, E., C. Aggarwal, and S. Wang (2021) "NRGNN: Learning a Label Noise-Resistant Graph Neural Network on Sparsely and Noisily Labeled Graphs," *arXiv preprint arXiv:2106.04714*.

[143] Kim, D. and A. Oh (2021) "How to find your friendly neighborhood: Graph attention design with self-supervision," in *ICLR*.

[144] Sun, F.-Y., J. Hoffmann, V. Verma, and J. Tang (2019) "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," *arXiv preprint arXiv:1908.01000*.

[145] Veličković, P., W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm (2018) "Deep graph infomax," *arXiv preprint arXiv:1809.10341*.

[146] Qiu, J., Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang (2020) "Gcc: Graph contrastive coding for graph neural network pre-training," in *SIGKDD*, pp. 1150–1160.

[147] You, Y., T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen (2020) "Graph contrastive learning with augmentations," *NeurIPS*, **33**.

[148] Alvarez-Melis, D. and T. S. Jaakkola (2018) "Towards robust interpretability with self-explaining neural networks," *arXiv preprint arXiv:1806.07538*.

[149] Hind, M., D. Wei, M. Campbell, N. C. Codella, A. Dhurandhar, A. Mojsilović, K. Natesan Ramamurthy, and K. R. Varshney (2019) "TED: Teaching AI to explain its decisions," in *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 123–129.

[150] Zeiler, M. D. and R. Fergus (2014) "Visualizing and understanding convolutional networks," in *European conference on computer vision*, Springer, pp. 818–833.

[151] Du, M., N. Liu, Q. Song, and X. Hu (2018) "Towards explanation of dnn-based prediction with guided feature inversion," in *SIGKDD*, pp. 1358–1367.

[152] Shrikumar, A., P. Greenside, and A. Kundaje (2017) "Learning important features through propagating activation differences," in *ICML*, PMLR, pp. 3145–3153.

[153] Koh, P. W. and P. Liang (2017) "Understanding black-box predictions via influence functions," in *ICML*, PMLR, pp. 1885–1894.

[154] Yuan, H., Y. Chen, X. Hu, and S. Ji (2019) "Interpreting deep models for text analysis via optimization and regularization methods," in *AAAI*, vol. 33, pp. 5717–5724.

[155] Yuan, H., J. Tang, X. Hu, and S. Ji (2020) "Xgnn: Towards model-level explanations of graph neural networks," in *SIGKDD*, pp. 430–438.

[156] Wu, Z., S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip (2020) "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*.

[157] Plötz, T. and S. Roth (2018) "Neural nearest neighbors networks," *NeurIPS*, **31**, pp. 1087–1098.

[158] Ren, W., Y. Yu, J. Zhang, and K. Huang (2014) "Learning convolutional nonlinear features for k nearest neighbor image classification," in *2014 22nd international conference on pattern recognition*, IEEE, pp. 4358–4363.

[159] Oord, A. v. d., Y. Li, and O. Vinyals (2018) "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*.

[160] Glorot, X. and Y. Bengio (2010) "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*, pp. 249–256.

[161] Dai, E., L. Cui, Z. Wang, X. Tang, Y. Wang, M. Cheng, B. Yin, and S. Wang (2023) "A unified framework of graph information bottleneck for robustness and membership privacy," *SIGKDD*.

[162] Dai, E. and S. Wang (2021) "Towards self-explainable graph neural network," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 302–311.

[163] ——— (2022) "Towards prototype-based self-explainable graph neural network," *arXiv preprint arXiv:2210.01974*.

[164] Lee, D.-H. et al. (2013) "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, vol. 3, p. 896.

Vita

Enyan Dai

I pursue the Ph.D. degree in the College of Information Sciences and Technology at Pennsylvania State University. My advisor is Dr. Suhang Wang. I obtained my Master of AI degree from Computer Science Department at KU Leuven. I received my Bachelor Degree from the University of Science and Technology of China. My research interests lie in data mining, Trustworthy Graph Learning, AI for Social Good applications, Trustworhty AI for Science, and Graph Foundation Model. My works have been published in top journals and conference proceedings such as ICLR, NeurIPS, KDD, WWW, TKDE etc. and have earned over 900 citations. The selected publications are listed as follows:

- **Enyan Dai**, Jie Wei, Hui Liu, and Suhang Wang. "Towards Robust Graph Neural Networks for Noisy Graphs with Sparse Labels." **Oral paper** In Proceedings of 15th ACM International Conference on Web Search and Data Mining (**WSDM 2022**)
- **Enyan Dai**, Limeng Cui, Zhengyang Wang, Xianfeng Tang, Yinhan Wang, Monica Chen, Bing Yin, Suhang Wang. "A Unified Framework of Graph Information Bottleneck for Robustness and Membership Privacy." Accepted by 29th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (**KDD 2023**)
- **Enyan Dai**, and Suhang Wang. "Say No to the Discrimination: Learning Fair Graph Neural Networks with Limited Sensitive Attribute Information." In Proceedings of 14th ACM International Conference on Web Search and Data Mining (**WSDM 2021**)
- **Enyan Dai**, and Suhang Wang. "Towards Self-Explainable Graph Neural Network." In Proceedings of 30th ACM International Conference on Information and Knowledge Management (**CIKM 2021**)