Handling Sparse and Noisy Labels in Deep Graph Learning

by

Yayong Li

A thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

Under the supervision of Professor Ling Chen

Faculty of Engineering and Information Technology

University of Technology Sydney

November 2022

Declaration

I, Yayong Li declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This document has not been submitted for qualifications at any other academic institution. This research is supported by the Australian Government Research Training Program.

Signature: Production Note: Signature removed prior to publication.

Date: 11/17/2022

I would like to dedicate this thesis to my loving wife and parents ...

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Prof. Ling Chen and A/Prof. Jie Yin, who have provided me with tremendous support and guidance during my PhD study.

As my principal supervisor, Ling has provided me with the opportunity to study in the Australian Artificial Intelligence Institute (AAII) at University of Technology Sydney (UTS) where I started a unique journey. At the beginning of my PhD, Ling led me to the academic path and gave me the promising research direction on graph representation learning. From scholarship application and research guidance to thesis submission and job recommendation, Ling has been always willing to offer any form of help to support my research and career development. She has also provided many valuable opportunities that remarkably broaden my horizon and gain my experiences. When I was depressed or disappointed because of experiment frustrations during my research, her encouragement cheered up my spirit and rebuilt my confidence.

I am also extremely grateful to my co-supervisor A/Prof. Jie Yin for guiding me throughout my PhD experience. During the four years, she has generously imparted the essential knowledge and expertise to me, which allows me to gradually build up solid academic skills from scratch and get me well prepared for the independent research in the next step of my career. Jie has a strong sense of responsibility. In our weekly meetings, she has been always patient to discuss every detail of my research with me, and her valuable advice can always help me tackle my problems more effectively. Despite having busy schedules, she can still devote her time to helping me resolve my research issues, review my presentation, revise and proofread our papers. The knowledge from her advice is only a small portion of what I learned from her. Her rigorous academic attitude, integrity, and irrepressible passion for research have deeply impacted and inspired me. Her behaviours have set a great example for my future career. Words cannot express my gratitude to my supervisors, and this thesis would not have been possible without them.

I am also grateful to my senior students Dr. Daokun Zhang and Dr. Wei Wu. When I was new to the school at UTS, their generous help and useful suggestions allowed me to avoid many mistakes and accommodate to the new academic life in Sydney faster. I also learned so much from my friend Xiaowei Zhou. His clear thought process and pleasant personality impressed me, and our every discussion inspired me with new perspectives and helped me get unstuck during research. Thanks should also go to my peer students, lab mates and other friends, including Lan Wu, Wei Huang, Shaoshen Wang, Yunqiu Xu, Leijie Zhang, Lu Huo, Tianyu Liu, Tao Zhang, Ying Tian, Wei Lin, Shulin Chen, Hongyang Zhang, Xinzhu Li, Yang Xu and Mingfei Tong. We shared a lot of experiences with laughter and happiness, which enriched my life in Sydney.

I would also like to acknowledge the financial support from the CSC-UTS scholarship, which significantly relieved my burden on living expenses and allowed me to pursue my degree dedicated.

Above all, I would like to express my special thanks to my wife. My wife has kept accompany with me for almost my whole PhD journey in Sydney, where we fell in love with each other and got married witnessed by my supervisors and all our friends. She has been the one with whom I have shared the most of my emotions and feelings, regardless, laughter or tears, happiness or sadness, excitement or depression, faith or frustration. There have been countless moments when I encounter setbacks in my experiments, her understanding and encouragement relieved my pressure and kept up my spirit to stick with it. She can always believe in me and fully support my decision, which gave me much courage and confidence to cope with varieties of difficulties and challenges. I am so fortunate to have her accompanying, which made my PhD journey much easier and happier.

The special thanks should also go to my parents, my sisters and my parents-in-law, for their constant and unconditional love. Although being thousands of miles away from each other in different countries, I can still feel their care and love. Our phone chat every week was my most relaxing moment when I can put my research on hold for a while and just enjoyed the pleasing time. Their kind regards and words can always comfort me and sweep away all my depression and anxieties. They always gave me unwavering help and support with all their hearts, and no words can express my deepest gratitude for their endless love and self-sacrifice.

To them, I dedicate the dissertation.

Abstract

Nowadays, there are growing amounts of graph-structured data emerging from a broad variety of information industrial applications, such as social networks, financial networks, biomedical networks, traffic networks, and so on. The complex topological information among those graph nodes, along with their content-rich node attributes, pose a great challenge for data mining and analysis. Recently, Graph neural networks (GNNs) have been proposed as a novel learning paradigm to deal with graph-structured data, and have achieved a great success on a variety of graph-based tasks, especially on the node classification task. However, its success highly relies on the sufficient number of high-quality labels, which is often difficult to attain in the real world. On the one hand, acquiring node annotations is labour-intensive, time-consuming, and usually costs a lot of expenses for recruiting or paying annotators. This results in the label sparsity problem for GNNs learning. On the other hand, wrong labeling is almost inevitable while annotating nodes due to inter-observer variability, human annotator error, or errors in crowdsourced annotations[60]. Under this situation, GNNs are prone to overfitting to these corrupted labels, thereby leading to poor generalization abilities.

Considering these label-associated challenges, this thesis is developed to handle the label sparsity and label noise problem on graphs. Confronting the label sparsity problem on graphs, I first resort to Active Learning (AL) to improve the model performance. Within the limited labeling budget, AL can selectively construct the most informative label set for model training by querying labels for the most valuable nodes in the graph. Then I focus on the research of Pseudo-Labeling (PL) to relieve the label sparsity problem. It explores to fully exploit the unlabeled nodes to complement the severe lack of label information, and apply label augmentation techniques to enhance information propagation among graph nodes. Finally, to cope with the label noise problem, I turn to the research of label-noise representation learning in GNNs, expecting to establish a robust GNN model that can effectively detect suspicious labels and minimize their influence on model training. Therefore, in this thesis, I would specialize in the three specific research topics and make efforts to effective solutions for them correspondingly.

In terms of Active learning, it aims to boost the labeling efficiency by selecting the most informative nodes for querying their labels, such that the selected nodes can maximize the model performance. Although AL has been widely studied for alleviating label sparsity issues with the conventional independent and identically distributed (IID) data, how to make it effective over attributed graphs remains an open research question. In Chapter 4, a SEmi-supervised Adversarial active Learning (SEAL) framework is proposed on attributed graphs, which fully leverages the representation power of GNNs and designs a novel AL query strategy in an adversarial way for node classification. Extensive experiments on real-world networks validate the effectiveness of the SEAL framework with superior performance improvements to state-of-the-art baselines on node classification tasks.

Pseudo-Labeling has been proposed to explicitly address the label scarcity problem. It aims to augment the training set with pseudo-labeled nodes so as to re-train a supervised model in a self-training cycle. However, the existing pseudo-labeling approaches often suffer from two major drawbacks. First, they tend to conservatively expand the label set by selecting only high-confidence unlabeled nodes without assessing their informativeness. Unfortunately, those high-confidence nodes often convey overlapping information with given labels, leading to minor improvements for model re-training. Second, these methods incorporate pseudo-labels into the same loss function with genuine labels, ignoring their distinct contributions to the classification task. In Chapter 5, a novel informative pseudo-labeling framework is proposed to facilitate learning GNNs with extremely few labels taking both informativeness and reliability of pseudo labels into consideration. Extensive experiments on six real-world graph datasets demonstrate that the proposed approach remarkably outperforms state-of-the-art pseudo-labeling and self-supervised baseline methods on graphs.

Label-noise representation learning has also been primarily studied on the tasks with IID data, such as the image classification task, but very little research effort has been on how to improve the robustness of GNNs in the presence of label noise. Furthermore, the graph topological information poses unique challenges when dealing with label noise - label sparsity and label dependency. To tackle these challenges, a unified framework is proposed to robustly train GNN models against label noise under the semi-supervised setting in Chapter 6. The key idea is to perform label aggregation to estimate node-level class probability distributions, and then use them to guide sample reweighting and label correction simultaneously, so as to reduce model sensitivity towards noisy labels. Experimental results on real-world datasets have been conducted to demonstrate the effectiveness of proposed algorithm with regard to different levels and types of label noise.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Background and Motivation

Recent years have witnessed an enormous growth of content-rich networked data generated from various domains, in forms of social networks, citation networks, financial networks, etc.. These networked data are often represented to be graphs, where nodes denote instances (e.g., users or documents) that are often characterized by rich content features, and edges denote relationships or interactions between nodes (e.g., friendship or citation relationships). Ingesting useful information lying in the graph-structured data has shed light on various applications across different disciplines, such as the traffic forecasting [44, 145], recommendation systems [27, 138, 142], protein interface predictions [30, 140], and many other applications [157]. To enable effective knowledge discovery from graph-structured data, graph representation learning has received increasing attention, which aims to encode graphs into compact, low-dimensional graph node, edge or subgraph embeddings that can preserve both the graph topology information and node content information. In the past decades, there has emerged a large amount of graph representation learning approaches, but they tend to rely on the traditional machine learning approaches, such as summary graph statistics[12], kernel methods[132], or hand-engineered features[76], which suffer from significant restrictions because of their inflexibility and expensive costs. In contrast, inspired by the skipgram method [90], the recently proposed DeepWalk-based graph embedding methods [39, 99, 126] has attained a significant improvement. They propose to learn the node embedding by predicting its local neighborhood, which can well preserve the structure proximity into node embeddings. However, these methods tend to purely rely on network structure information but ignore node features, thus resulting in suboptimal performances.

More recently, motivated by the great success of deep learning in the domain of computer vision (CV) and natural language processing (NLP), massive efforts have been dedicated to investigate how to extend deep learning models to graph representation learning. Under this circumstance, the classic Deep Neural Network (DNN) models, such as convolutional neural network (CNN), recurrent neural network (RNN) etc., are extensively explored to adapt to graph-structured data, which has gradually developed and evolved into a variety of Graph Neural Networks (GNNs).

Graph Convolutional Neural Networks (GCNNs) are the most representative GNN models, which can be roughly grouped into two categories according to the definition of convolution operation: spectral GCNNs and spatial GCNNs. To adapt CNNs to learning hierarchical representations, spectral GCNNs define the convolution operation via the Graph Fourier Transform. They generally transform graph signals into the Fourier domain for conducting convolution by means of specific spectral graph filters, and then revert them back to the spatial domain to obtain node representations. Following this convolution paradigm, abundant spectral graph filters have been designed to instantiate GCNNs. Bruna et al. [16] and Henaff et al. [50] defined a learnable matrix as the filter, which, however, was non-spatially localized and incurred huge computational costs. To address this problem, Defferrard et al. [24] proposed a K-localized graph filter based on the Chebyshev polynomials. Kipf et al. [63] put it forward a further step and proposed the Graph Concolution Network (GCN) method using the first-order Cheybyshev polynomial and the renormalization trick. GCN significantly reduced the computational complexity and made it more applicable for the deployment of GCNNs on various graph tasks in the real world.

In terms of the spatial GCNNs, whose major challenge lies in defining the convolutional operation on the size-varied neighborhood and maintaining the local invariance of CNNs, it directly performs convolution on graphs by propagating node information along with the spatial connections. The principle of Spatial GCNNs is to learn node embeddings by recursively aggregating and transforming continuous feature vectors from local neighborhoods [20, 21, 33, 135]. For example, Atwood et al. [7] designed a diffusion-based graph convolution network for feature aggregation which assumed that the information propagation along with the graph structure via the transition matrix can reach a stationary distribution. Monti et al. [91] utilized a parameterized kernel function to learn the relative weights of neighboring nodes for feature aggregation. Gilmer et al. [36] extracted the universality among the existing spatial GCNNs frameworks, and came up with a general framework with the differentiable message-passing functions and vertex updating functions. Hamilton et al. [46] proposed an inductive graph representation learning framework, called GraphSage, which can generalize to unseen nodes by sampling and aggregating feature information from their local neighborhoods. This approach paved the way for deploying GCNNs on large-scale graphs in many real-world applications [142]. Velivckovic et al. [130] proposed an inductive graph attention network (GAT) that learned a specific weight for each connected node to perform feature aggregation. This relieved the potential information missing problem caused by neighborhood sampling in GraphSage [46]. Following GAT, the gated attention network (GAAN)[152] and GeniePath [79] were successively proposed to further improve the attentive feature aggregation strategy of GAT based on the gating mechanism. Overall, compared with the spectral GCNNs, the spatial GCNNs are generally more flexible in the computing paradigm, and can easier accommodate inductive tasks.

Apart from GCNNs, many other deep learning models have also been studied to generalize to graphs. In [45, 57, 106, 147], the idea of RNNs was generalized to capture recursive and sequential patterns from complex graphs by converting graph nodes into sequences. [3, 66, 114] incorporated Deep Reinforcement Learning (DRL) into GNNs to extract graph-structure information and generate graph representations. In addition, Chami et al. [18] proposed to transform the Euclidean features into hyperbolic embeddings, and carried out graph convolution in the hyperbolic spaces. In [62, 96], the variational autoencoder was proposed to learn node representaions by integrating the variational inferencing model with

GNNs. All these explorations have made great contributions to improving GNNs' learning ability from different perspectives and resulted in impressive performances.

Empowered by these GNNs, graph representation learning has attained substantial advancements. The learned graph representations are usually taken as inputs for various downstream graph analytical tasks, of which node classification is one of the most important tasks in graph analytics. It is aimed at predicting node labels with a partially labeled network. Based on the foundation laid by the aforementioned GNNs, there derive abundant GNN variants in dealing with the node classification task in the end-to-end manner[21, 22, 29, 33, 64, 113, 129, 135, 136, 139, 148]. For example, Xu et al. [139] proposed a jumping knowledge network by flexibly adjust the local neighborhood range for feature aggregation. Qu et al. [101] proposed a graph Markov neural network based on GCN, which applied the variational EM algorithm to model label dependency. Abu-El-Haija et al. [1] proposed to learn more expressive node representations by linearly mixing neighborhood information at various hop distances. Wang et al. [133] proposed to combine Label propagation (LP) and GCN to boost classification accuracy. Verma et al. [131] proposed a GraphMix framework based on GCN, which employed parameter sharing and interpolation-based regularization to further improve model performance. All these methods have achieved great success on node classification tasks.

However, these GNN-based methods highly depend on a sufficient number of high-quality label set to guarantee the desirable classification performance, which is usually difficult to attain in the real world. Generally, acquiring a large quantity of node labels is extravagant owing to the fact that it usually requires expert efforts, and is very costly and time-consuming. Furthermore, graphs with inter-connected nodes are arguably harder to label than individual images. Thus, very often, the graph can be only sparsely labeled, with a very small number of labels available for training. Under this situation, GNNs are unable to propagate the limited label information to entire unlabeled nodes at large distances, thus failing to learn expressive representations targeting specific tasks. Besides, wrong labeling usually happens due to inter-observer variability, human annotator's error, and errors in crowdsourced annotations[60]. Therefore, in the presence of noisy labels, since Deep Neural

Network (DNN) based methods are able to fit and memorize the whole dataset, it can lead to dramatic performance degeneracy and poor generalization ability on test sets [154].

The sparse and noisy label problems become the main obstacle that significantly limits the true success of GNNs [19, 122]. Considering this, I focus my research on alleviating the GNNs' high dependency on the amount and quality of label information. In order to address the label sparsity problem on graphs, I resort to Active Learning and Pseudo-Labeling in the context of GNNs. Active learning (AL) has been proposed to maximize the learning ability of classification models while remaining sensitive to data acquisition costs. By selectively querying the most informative node labels, AL targets at bring the most information gain to the model. Pseudo-labeling has been also proposed to explicitly address the label scarcity problem, which aims to augment the training set with pseudo-labeled nodes so as to re-train a supervised model in a self-training cycle. Moreover, to combat the label noise problem, I investigate the label-noise representation learning on graphs to improve the model robustness against the corrupted label set. Next, I will give a detailed research description on the three specific topics.

1.2 Research Descriptions

This thesis aims to relieve the pressure of demanding large amounts of high-quality labels during model training, so that the performance of graph representation learning could be driven to a higher level with imperfect label sets. I address this problem by developing machine learning methods from the following three respective topics: active learning, pseudo-labeling, and label-noise representation learning in the context of GNNs. The detailed *research descriptions* and *challenges* of the three research problems would be described in the following part.

1. **Active learning (AL) on graphs**: Given a fixed labeling budget, AL aims to selectively pick up the most informative instances for labeling so that the model could reach its best performance. An AL framework typically consists of two primary components: a *query engine* which selects an instance from the unlabeled data pool to query its label

and an *oracle* which provides a label to the queried instance. Therefore, the key is to design appropriate query strategies based on the current classifier and existing labels, according to which the most critical nodes are selected for annotation.

Existing AL algorithms on graph node classification attempt to reuse the classic AL query strategies designed for IID data. However, they suffer from two major limitations. First, different AL query strategies calculated in distinct scoring spaces are often naively combined to determine which nodes to be labeled. Second, the AL query engine and the learning of the classifier are treated as two separate processes, resulting in unsatisfactory performance.

2. **Pseudo-labeling on graphs**: Given an undirected graph, together with a small subset of labeled nodes, the pseudo-labeling method aims to design: i) a strategy for expanding the label set from unlabeled nodes, ii) a method for generating reliable pseudo labels, and iii) an exclusive loss function for pseudo labels, such that they can be combined to maximize the classification performance of GNNs.

   However, the existing pseudo-labeling approaches often suffer from two major drawbacks. First, they tend to conservatively expand the label set by selecting only high-confidence unlabeled nodes without assessing their informativeness. Unfortunately, those high-confidence nodes often convey overlapping information with given labels, leading to minor improvements for model re-training. Second, these methods incorporate pseudo-labels to the same loss function as genuine labels, ignoring their distinct contributions to the classification task.

3. **Label-noise representational learning on graphs**: When class labels in the training set are corrupted with wrong labels, GNNs are vulnerable to overfittng noisy labels, thereby leading to degraded classification performance and poor generalization ability on the test data. To address this problem, this work aims to train a robust GNN model against label noise, such that the model is less sensitive to the corrupted labels. Since the training of GNNs usually suffers from the label sparsity problem, this work further considers the label noise problem under the semi-supervised setting for graph node

classification, where only a small fraction of node labels are given accompanied with a certain rate of incorrect labels.

Classification with label noise has primarily focused on image classification but cannot be directly applied to graph data, which poses unique challenges when dealing with label noise. (1) Label sparsity: graphs with inter-connected nodes are arguably harder to label than individual images. Very often, graphs are sparsely labeled, with only a small set of labeled nodes provided for training. Hence, we cannot simply drop “bad nodes” with corrupted labels like previous methods using “small-loss trick” [47, 56]. (2) Label dependency: there exist strong dependency relationships between graph topology, node features and sparse node labels. Therefore, graph topology and sparse node labels should be fully exploited when training a robust GNN model against label noise.

1.3 Thesis Contributions

To address the limitations that sparse and noisy labels bring to GNN training, I have made comprehensive investigations on the acknowledged challenges, and proposed the solutions targeting the tasks of active learning, pseudo-labeling, and label-noise representation learning on graphs. The contributions of this thesis are summarized below:

Active learning on graphs: I propose a novel semi-supervised adversarial active learning framework to select the most informative nodes to the label on attributed graphs. The proposed framework explicitly asserts the usefulness of unlabeled nodes with regard to the existing labeled data. Inspired by adversarial learning, an *informativeness measure* is defined based on the intuition that the unlabeled nodes differing the most with the labeled ones carry the most auxiliary information on what the classifier desires the most. The main contribution of this approach is threefold:

- I propose a novel adversarial AL framework that seamlessly incorporates active learning into GNNs. Unlike previous methods that simply combine AL strategies residing at different scoring spaces, *SEAL* generates a unified informativeness score in a common latent space to enable instance selection, rendering the most desirable performance gains.

- To the best of our knowledge, this is the first work to propose a **S**emi-supervised **A**dversarial **L**earning (*SAL*) structure with multiple outputs for AL on attributed graphs. This offers an advantage that the graph embedding network and the discriminator can collaborate with each other to mutually strengthen their performance.
- I validate the *SEAL* framework through extensive experiments and ablation studies on four real-world networks, demonstrating its superior performance to state-of-the-art baselines on node classification tasks.

Pseudo-labeling on graphs: I propose a novel informative pseudo-labeling framework called InfoGNN for semi-supervised node classification with few labels. It can fully harness the power of self-training by incorporating more pseudo labels that are likely to bring more information gains. And at the same time, it can alleviate the possible negative impact caused by noisy (i.e. incorrect) pseudo labels. The main contributions can be summarized as follows,

- This study analyzes the ineffectiveness of existing pseudo-labeling strategies and proposes a novel informative pseudo-labeling method for semi-supervised node classification task with few labels;
- This approach has unique advantages to incorporate an mutual information (MI)-based informativeness measure for pseudo-label candidate selection, and to alleviate the negative impact of noisy pseudo labels via a generalized cross entropy loss.
- I validate the proposed approach on six real-world graph datasets of different types, showing its superior performance to the state-of-the-art baselines.

Label-noise representation learning on graphs: I propose a novel approach for robustly learning GNN models against noisy labels under semi-supervised settings. This approach

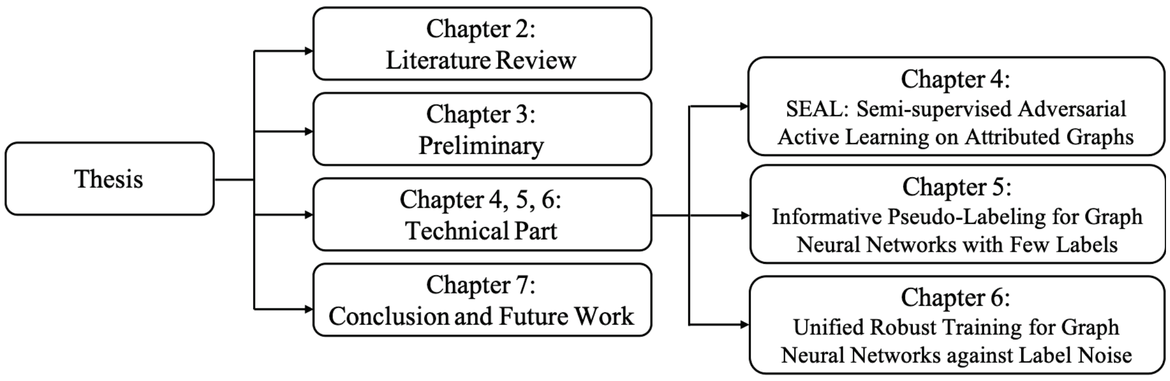

Thesis
Chapter 2: Literature Review
Chapter 3: Preliminary
Chapter 4, 5, 6: Technical Part
Chapter 7: Conclusion and Future Work
Chapter 4: SEAL: Semi-supervised Adversarial Active Learning on Attributed Graphs
Chapter 5: Informative Pseudo-Labeling for Graph Neural Networks with Few Labels
Chapter 6: Unified Robust Training for Graph Neural Networks against Label Noise

Fig. 1.1 Thesis overview

provides a unified robust training framework for GNNs (UnionNET) based on label aggregation, which performs sample reweighting and label correction simultaneously to improve robustness of the model. The main contributions of this work are threefold:

- This is the first work to study the problem of learning with noisy labels on node classification under a semi-supervised setting.
- I propose a unified learning framework to robustly train a GNN model by performing sample reweighting and label correction simultaneously.
- Experiments and ablation studies verify the robustness of the proposed approach against label noise and its superior performance over strong baselines.

1.4 Thesis Overview

My thesis is organized as depicted in Figure 1.1. Besides Chapter 1 (Introduction), it consists of four main parts, i.e. Literature Review, Preliminary, Technical Part, Conclusion and Future Work, of which the Technical Part is composed of three chapters specializing in three different research tasks. The detailed content of each chapter is summarized as follows:

- **Chapter 2:** This chapter presents the comprehensive literature review about the semi-supervised node classification and three specific research topics, i.e. active learning, pseudo-labeling and label-noise representation learning on graphs. It provides a summarization of existing related works on their main ideas, and offers inspirations for developing my works.

- **Chapter 3:** This chapter gives detailed definitions of some important terminologies and notations, and introduces the fundamental GNN architecture in the mathematical way. Summarization of the benchmark data sets is also presented in this chapter.

- **Chapter 4:** In this chapter, details of the proposed semi-supervised adversarial active learning framework are discussed to release the potential of active learning in the context of GNNs.

- **Chapter 5:** This chapter presents the investigation on learning GNNs with few labels. An informative pseudo-labeling framework is proposed to explicitly address the label scarcity problem by augmenting the training set with informative pseudo labels.

- **Chapter 6:** In this chapter, I study how label noise influences the model training as partially-observed label information propagates along with the graph structure. And a unified robust training framework has been proposed to combat corrupted label sets.

- **Chapter 7:** Finally, I conclude this thesis and outline a few directions for my future research.

1.5 Publications

Journal Publications:

1. **Yayong Li**, Jie Yin, Ling Chen. SEAL: Semi-supervised Adversarial Active Learning on Attributed Graphs. *IEEE Transactions on Neural Networks and Learning Systems*, 32(7): 3136-3147, 2021. [73]

2. **Yayong Li**, Jie Yin, Ling Chen. Informative Pseudo-Labeling for Graph Neural Networks with Few Labels. *Data Mining and Knowledge Discovery*, Accepted. [75]

Conference Publications:

1. **Yayong Li**, Jie Yin, Ling Chen. Unified Robust Training for Graph Neural Networks Against Label Noise. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*: pages 528-540. Springer, 2021. [74]

2. Wei Huang*, **Yayong Li***[1], Weitao Du, Richard Yi Da Xu, Jie Yin, Ling Chen. Towards Deepening Graph Neural Networks: A GNTK-based Optimization Perspective. *International Conference on Learning Representations*, 2022. [54]

[1]* Equal contribution

Chapter 2

Literature Review

According to the research objectives, this chapter presents a comprehensive literature review of the existing related works. I would first introduce the related works on the semi-supervised node classification, which provides a context for my specific research objectives. Then I would successively present the related works on active learning, pseudo-labeling, and label-noise representation learning. To have a more comprehensive view of the current development of the research topics, my literature review spans the works from the domains of computer vision, natural language processing and graphs, which deepens my insights and allows me to obtain valuable inspirations for my own work. I would elaborate on them as follows.

2.1 Semi-supervised Node Classification

Semi-supervised node classification has been extensively studied in the last decades, and the existing methods can be roughly grouped into four different lines. The first line is Graph Laplacian regularization based methods [9, 156, 160]. They assume that nearby nodes tend to share the same label, and impose constrains to encourage the label consistency among the connected nodes. For example, Zhou et al. [156] proposed to iteratively spread node label information to the neighboring nodes until it converged to a stationary state. And Talukdar et al. [124] proposed an improved variant of label propagation method that incorporated variance and uncertainty into the label predictions.

The second line is the collective classification methods, which typically perform an iterative training paradigm by repeating two interactive processes, i.e. the relational feature computing and the collective inference . Generally, they first learn the relational features by training a predictive model with the given labels and node contents, and then perform the collective inference using the learned relational features and node contents. The two training processes alternate until model convergence [87, 88, 93, 108]. But this kind of method tends to separately learn the dependencies between node contents and graph structure, which ignores their interplay.

The third line is the graph embedding based methods. This kind of method often learns to embed network structure information into latent continuous node representations, and then employs them to perform model training on the predicting tasks with some specific machine learning methods. For instance, DeepWalk[99], drawing inspirations from Skip-gram model[90], proposed to learn node embeddings by predicting the node local neighborhood generated by random walks. Following DeepWalk, LINE[126] proposed to encode both the first and the second-order proximity into node representations by means of the shared local neighborhood between different nodes. And node2vec[39] devised a biased random walk procedure via interpolating between the breath-first and depth-first sampling, which enabled more diverse neighborhood exploration. However, these methods purely rely on network structure information but ignore the node features.

The last line is GNN-based methods which have achieved the most remarkable success in the field of node classification in the past few years. GNN-based methods[46, 63, 130] often rely on the message exchanging among connected nodes to embed both node features and structure information into vectors, and then transform them into the low-dimensional discriminating space for node classification using deep learning techniques. Massive amounts of excellent methods have been proposed from different perspectives, which have pushed the node classification performance to advance constantly. For instance, Abu-El-Haija et al. [1] proposed to learn more expressive node representations by linearly mixing neighborhood information at various hop distances. Zhang et al. [153] proposed a Bayesian graph neural network to incorporate the graph uncertainty information via a parameterized random graph model. Wang et al. [133] proposed to combine Label propagation (LP) and GCN to boost classification accuracy. Verma et al. [131] proposed a GraphMix framework based on GCN, which employed parameter sharing and interpolation-based regularization to further improve model performance. Wu et al. [135] decoupled the process of convolution operation and feature transformation to improve computational efficiency. Zeng et al. [148] proposed a graph sampling based inductive learning framework that were more computationally efficient and scalable. Chami et al. [18] extended GCNNs to hyperbolic geometry space so as to reduce the distortion caused by transforming graphs into Euclidean space during convolution operation. Furthermore, [21, 64, 69, 70, 103, 158] explored to deepen GNNs to capture the long-range node dependencies serving for the node classification. [100, 123, 129, 143, 144, 163] applied self-supervised learning to enable the learned node representations to be more expressive for label prediction.

Although GNN-based models have achieved great success on the semi-supervised node classification task, they suffer strict restrictions from the amount and quality of label set, i.e. the label sparsity and label noise problem. In this thesis, I will engage in active learning, pseudo-labeling, and label-noise representation learning to mitigate the two problems and further release the potential of GNN models on node classification tasks.

2.2 Active Learning

I have reviewed related works on active learning from two main branches of research studies, namely classic active learning strategies and active learning on graphs.

2.2.1 Classic Active Learning Strategies

Active learning is a machine learning framework that aims to reduce the labeling cost when learning a prediction model by selecting the most informative instances to label. In the past decade, a variety of AL algorithms have been proposed to optimize the training performance

Table 2.1 Comparison of different classic AL Strategies

Categories	Core idea	Advantages	Disadvantages
Uncertainty sampling [67, 68]	Query the instances whose labels are predicted with the least confidence based on the current labeled set	Simple and fast	Prone to selecting noisy or unrepresentative instances
Query-by-Committee (QBC) [86, 112]	Query the instances with which multiple classifiers most disagree		
Expected Model Change (EMC) [111]	Query the instances which would result in the most change to the current model parameters in the gradient of objective function	Directly optimize the model performance	Applicable only to gradient-based training methods, and incur huge computational cost when the feature or label space are large
Expected Error Reduction (EER) [59, 105]	Query the instances which most likely reduce the largest generalization error on the unlabeled pool	Minimize generalization errors by considering the entire input space, eliminating the disturbance of outliers	High computational complexity
Expected Variance Reduction (EVR) [35, 83]	Query the instances which minimize model variance		
Density-Weighted Methods (DWM) [32, 110]	Query the instances that are representative of the underlying distribution of training data	Able to avoid selecting noisy instances	Not informative enough, and often combined with other strategies

given a fixed labelling budget. These algorithms differ mostly in the query strategy that they use to specify the informativeness criterion when selecting the best instances to label. Depending on what query strategies are used, classic AL strategies can be grouped into six categories [2, 109]: uncertainty sampling [67, 68], query-by-committee (QBC) [86, 112], expected model change (EMC) [111], expected error reduction (EER) [59, 105], expected variance reduction (EVR) [35, 83], and density-weighted methods (DWM) [32, 110]. The core ideas and detailed comparisons of these algorithms are summarized in Table 2.1. In general, these classic AL strategies can be instrumented with different classification algorithms. Among others, expected error/variance reduction strategies tend to render the better empirical results because they iterate over the entire unlabeled pool to directly optimize model performance. However, they suffer from high computational overhead.

These methods have been shown to achieve good performance on i.i.d. data. However, they are not sufficiently effective for graph-structured data, where data dependency needs to be incorporated into the AL process.

2.2.2 Active Learning on Graphs

In recent years, active learning on graphs has attracted significant attention to alleviate the label sparsity issues on graphs. Early research has focused on using graph-based metrics (e.g., centrality, impact, etc.) to calculate the AL query scores when selecting the nodes to label [116]. Other attempts have been made to directly optimize an objective function over graphs, where graph structure is utilized to train a classifier, such as graph cut-based method [41, 42], Gaussian Field and Harmonic Function (GFHF) [55], Learning with Local and Global Consistency (LLGC) [40], and Label Propagation (LP) [80]. These methods aim to minimize the expected generalization error or variance of the classifier built using graph structure. However, they often suffer from high computational complexity and are difficult to scale up. To improve efficiency, Zhao et al. [155] proposed to narrow the search space by sampling structurally important nodes in advance, and Zhu et al. [161] uses uncertainty and graph centrality to prune the candidate set. However, these methods have assessed the informativeness of unlabeled nodes using only graph structure, while rich node features have not been fully explored to best inform the design of AL query strategies.

Another line of research formulates AL query strategies by integrating node features with graph topological structure [10, 11, 13, 28, 115]. Most methods design an iterative classification algorithm (ICA) classifier by combining graph structure with node-specific features, and then use various AL query strategies for instance selection. For example, ALFNET [13] adopts clustering techniques to form an initial labeled set. At each iteration, ALFNET aggregates neighboring labels with original node features to train three classifiers and computes a local disagreement score for each node. The scores are then aggregated for each cluster and the clusters with the highest scores are chosen, from which a set of nodes are selected to label. These methods, however, have focused on improving the classifiers built in the original feature space, thus leading to suboptimal prediction accuracy as compared with counterparts built in latent feature spaces by deep learning models.

Only recently, researchers have proposed to exploit deep learning to empower AL on graphs. In virtue of great representation power of GNN, AGE [17] and ANRMAB [34] propose to incorporate a GCN into traditional AL strategies, which achieves a significant improvement compared with the previous methods. Both methods combine three traditional query strategies, graph centrality, information density, and uncertainty sampling. AGE uses a naive linear combination of the three strategies, while ANRMAB further adopts the multi-armed bandit (MAB) based mechanism to adaptively adjust the weights for different strategies.

However, the combined AL query engine and the learning of graph embedding still work as two separate and independent processes, resulting in limited AL performance gains. To fill the gap, my work is proposed to fully integrate the learning of graph embedding with a novel AL query strategy via a semi-supervised discriminator. The learning of graph embedding and discriminator function as two adversarial components, which collaborate with each other to mutually strengthen their performance towards better AL performance. The detailed explanations on this method are presented in Chapter 4.

2.3 Pseudo-Labeling

To fully understand the pseudo-labeling problem on graphs, I have reviewed the related works on a few distinct but associated topics, including pseudo-labeling on graphs, graph few-shot learning, and graph self-supervised learning. Besides, I have also made an investigation on the related works of the mutual information maximization technique.

2.3.1 Graph Learning with Few Labels

GNNs have emerged as a new class of deep learning models on graphs [63, 130]. The principle of GNNs is to learn node embeddings by recursively aggregating and transforming continuous feature vectors from local neighborhoods [20, 21, 33, 135]. The generated node embeddings can then be used as input to any differentiable prediction layer, for example, a softmax layer for node classification. Recently, a series of semi-supervised GNNs such as GCNs and their variants have been proposed for node classification. The success of these models relies on a sufficient number of labeled nodes for training. How to train GNNs with a very small set of labeled nodes has remained a challenging task.

Pseudo-Labeling on Graphs. To tackle label scarcity, pseudo-labeling has been proposed as one of the prevalent *semi-supervised* methods. It refers to a specific training regime, where the model is bootstrapped with additional labeled data obtained by using a confidence-based thresholding method [65, 104]. Recently, pseudo-labeling has shown promising results on semi-supervised node classification. Li et al. [71] proposed a self-trained GCN that enlarges the training set by assigning a pseudo label to top K confidence unlabeled nodes, and then re-trains the model using both given labels and pseudo labels. A co-training method was also proposed that utilizes two models to complement each other. The pseudo labels are given by another random walk model rather than the GNN classifier itself. A similar method was also proposed in [149]. Sun et al. [121] showed that a shallow GCN is ineffective in propagating label information under few-label settings, and proposed a multi-stage self-training framework that relies on a deep clustering model to assign pseudo labels. Zhou et al. [159] proposed a dynamic pseudo-labeling approach called DSGCN that selects unlabeled nodes with prediction probabilities higher than a pre-specified threshold for pseudo-labeling, and assigns soft label confidence to them as label weight.

I argue that all of the existing pseudo-labeling methods on GNNs share two major problems: information redundancy and noisy pseudo labels. my work is proposed to explicitly overcome these limitations, with a focus on developing a robust pseudo-labeling framework that allows to expand the pseudo label set with more informative nodes, and to mitigate the negative impact of noisy pseudo labels simultaneously.

Graph Few-shot Learning. Originally designed for image classification, few-shot learning primarily focuses on the tasks where a classifier is adapted to accommodate new classes unseen during training, given only a few labeled examples for each class [119]. Several recent studies [26, 53, 134] have attempted to generalize few-shot learning to graph domains. For example, Ding et al. [26] proposed a graph prototypical network for node classification, which learns a transferable metric space via meta-learning, such that the model can extract meta-knowledge to achieve good generalization ability on target few-shot classification task.

Huang et al. [53] proposed to transfer subgraph-specific information and learn transferable knowledge via meta gradients.

Despite the fact that few-shot learning and this work both tackle the label scarcity problem, their problem settings and learning objectives are fundamentally different: in few-shot learning, the training and test sets typically reside in different class spaces. Hence, few-shot learning aims to learn transferable knowledge to enable rapid generalization to new tasks. On the contrary, this work follows the transductive GNN setting where the training and test sets share the same class space. my objective is to improve model training in face of very few labels.

2.3.2 Graph Self-supervised Learning

Our work is related to self-supervised learning on graphs [129], which also investigates how to best leverage the unlabeled data. However, there is a clear distinction in the objectives: the primary aim of self-supervised learning is to learn node/graph representations by designing pretext tasks without label-related supervision, such that the generated representations could facilitate specific classification tasks [78]. For example, You et al. [143] demonstrated that self-supervised learning can provide regularization for graph-related classification tasks. This work proposed three pretext tasks (i.e., node clustering, graph partitioning, and graph completion) based on graph properties. Other research works attempted to learn better node/graph representations through creating contrastive views, such as local node vs. global graph view in [129], or performing graph augmentation [162].

In contrast, my work resorts to explicitly augmenting label-specific supervision for node classification. This is achieved by expanding the existing label set with reliable pseudo labels to best boost model performance in a semi-supervised manner. I will elaborate more explanations on this work in Chapter 5.

2.3.3 Mutual Information Maximization

The Infomax principal was first proposed to encourage an encoder to learn effective representations that share maximized Mutual Information (MI) with the input [8, 51, 77]. Recently, this MI maximization idea has been applied to improve graph representations. Velickovic et al. [129] applied MI maximization to learn node embeddings by contrasting local subgraphs and the high-level, global graph representations. Qiu et al. [100] proposed to learn intrinsic and transferable structural representations by contrasting subgraphs from different graphs via a discriminator. Hassani et al. [49] contrasted node representations from a local view with graph representations from a global view to learn more informative node embeddings. In the context of this work, the idea of contrastive learning is leveraged to maximize the MI between each node and its neighboring context. The estimated MI enables to select more representative unlabeled nodes in local neighborhoods for pseudo-labeling so as to further advance model performance.

2.4 Label-noise Representation Learning

In terms of the label noise problem, since there are rare works specializing in label-noise learning tasks in the graph domain, I would first make a detailed review on the related works with independent and identically distributed (IID) data, such as the image classification and text classification task. Then I would turn to the review of works relating to learning GNNs with label noise.

2.4.1 Learning with Noisy Labels

In the past decade, learning with label noise have been extensively studied in the domain of both computer vision and natural language processing, such as the image classification and test classification tasks. Those methods can be categorized into three different lines. The first line of research focuses on correcting the loss function. The loss correction methods can be categorized in two types. The first type focuses on estimating the noise transition matrix between noisy labels and ground true labels. Patrini et al. [97] introduced the noise transition matrix to the loss function, and proposed a two-level estimation approach when the matrix is not known a priori. Other methods [127, 137] used a graphical model to capture the relationship between noise labels and ground-truth labels. An EM-like algorithm was proposed to infer the true labels. Both methods yet require the access to a set of clean samples. The second type proposed to add an extra softmax layer to estimate the probability of the ground-truth label flapping to noisy labels [37, 120]. However, the noise transition matrix is difficult to be accurately estimated, in particular, when there exist a large number of classes. Ardehaly et al. [5] utilized an intermediate model to fit noisy labels, and then applied the learned model to adjust the label proportion for social media text classification. Jindal et al. [58] introduced a specialized layer to incorporate statistics of the label noise into a CNN architecture to prevent overfitting to noisy labels on text classification. Zhang et al. [154] proposed to use the negative Box-Cox transformation as the loss function, which improves the robustness of standard cross entropy loss but with worse converging capability.

The second line of studies focus on separating clean samples from noisy samples, and using only most likely clean samples to update the network. MentorNet [56] pre-trains an extra network on a clean set to select clean samples to guide the network training based on their training losses. When the clean set is unavailable, MentorNet has to reply on a predifined curriculum (e.g., self-paced curriculum), which would yield suboptimal results. Guo et al.[43] proposed a curriculum learning network, which designs a distance density-based curriculum to prioritize low-complexity samples. The curriculum, however, is designed using features extracted from a pre-trained network trained on all noisy samples. Thus, the results become unreliable when noise rates are high. Malik et al. [85] proposed to automatically validate and correct the noisy labels according to the confidence of class predictions for the text classification. Co-teaching [47] trains two networks to select small-loss samples within each mini-batch to train each other. Yu et al. [146] improve Co-teaching by updating the network only with small-loss samples that two networks disagree with. Both methods heavily rely on the estimated noise rate to choose the threshold for selecting small-loss samples. Decoupling [84] updates the two networks only using samples with which two networks disagree. In our setting with very few labeled nodes, we should not simply drop "bad nodes" with corrupted labels, which can still be informative to infer labels of nearby nodes.

The third line of research takes a reweighting approach based on model prediction probabilities. Arazo et al. [4] utilized a two-component Beta Mixture Model to estimate the probability of a sample being mislabelled, based on which the loss function is modified by reweighting these samples in the gradient update. This method is further improved by combining with *mixup augmentation* method [151]. Ren et al. [102] proposed a meta-learning algorithm which allows the network to put more emphasis on the samples that have the closest gradient directions with the clean data. Unlike these reweighting methods that reply on the predicted probabilities by the trained model, the proposed reweighting scheme assigns weights to each node by leveraging topology structure in its neighborhood, which is less prone to label noise.

There are some other methods concerned with the problem of label correction. Han et al. [48] chose class prototypes based on sample distance density in a feature space, which afterwards, is used for label correction. However, it incurs extra computational overhead to the training process. Tanaka et al. [125] proposed a self-training approach to correct the labels by alternatively updating the network parameters and labeling towards the gradient descending direction of the training loss. However, this method is completely dependent on predicted labels, discarding the original given labels. This may result in degraded performance with a high noise rate. Similarly, Yi et al. [141] modeled the true labels as a probability distribution and update it together with network parameters. This method faces the same problem with Tanaka et al. [125], and moreover, it needs to fine-tune its hyper-parameters for different noise rates. This work, in contrast, proposes a unified solution for seamlessly integrating robust training with label correction in an end-to-end framework, thereby yielding remarkable gains even with high noise rates.

2.4.2 Graph Neural Networks with Noisy Labels

GNNs have emerged as a new type of deep learning models on graphs [63, 130]. The principle of GNNs is to learn node representations by recursively aggregating and compressing the continuous feature vectors from local neighborhoods. The generated node representations can then be used as input to any differentiable prediction layer, for example, a softmax layer for node classification. Various types of GNN models, such as GCN [63], GAT [130], GraphSAGE [46] and their variants, have been proposed in recent year. These models differ mainly in the way how features are aggregated in local neighborhood. It has been shown that these models have achieved competitive results on graph related tasks, such as node classification, on the assumption that genuine node labels are always provided for training. So far, very little attention has been put on robustly training a GNN model in the presence of label noise. NT et al. [95] introduced the problem of learning GNNs with symmetric label noise. Their method first estimates the noise transition matrix on the noisy data and then uses it to correct the standard cross entropy loss for graph classification. Thus, it shares the same limitation with similar methods on image data that the noise transition matrix is not easy to be accurately estimated.

Following [63], in this work, I investigate node classification with both symmetric and asymmetric label noise under the semi-supervised setting. The proposed method provides a unified solution to robust training and label correction for GNNs, which required no extra clean supervision nor explicit estimation of the noise transition matrix. More details about this work are provided in Chapter 6.

2.5 Conclusion

This chapter provides a comprehensive review of the related works regarding handling the label sparsity and label noise problems with an emphasis on that in the context of GNNs. Particularly, the related works on semi-supervised node classification are firstly presented as the elementary background for the specific label-associated tasks. Then, the related works on active learning, pseudo-labeling and label-noise representation learning are successively presented. These related works well demonstrate the key knowledge in the specific areas, and give a summary of the current methods in terms of both their advantages and disadvantages. In the next chapter, I will further describe the associated preliminaries that are commonly shared throughout the thesis to help the understanding of specific proposed methods in the technical chapters.

Chapter 3

Preliminary

In order to facilitate formulating the specific research problem in the technical chapters, I would provide some preliminary knowledge and concepts in this chapter with three subsections, including definitions and notions, fundamental architectures, and benchmark datasets. These concepts are shared by the whole thesis and they will provide the technical context for the latter chapters.

3.1 Definitions and Notations

Definition 1 (Attributed Graph) *An undirected attributed graph is presented as* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$*, where* $\mathcal{V}$ *denotes the node set,* $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ *denotes the edge set,* $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T \in \mathbb{R}^{N \times M}$ *denotes the node feature matrix, and* $\mathbf{x}_i \in \mathbb{R}^M$ *is the* M*-dimensional feature vector of node* $v_i \in \mathcal{V}$*.* $e_{ij} = (v_i, v_j) \in \mathcal{E}$ *denotes the edge between node* v_i *and* v_j*. I use the adjacent matrix* $A \in \mathbb{R}^{N \times N}$ *to denote the connections among nodes, where* $A_{ij} = 1$ *if* $e_{ij} \in \mathcal{E}$ *otherwise* $A_{ij} = 0$*.*

Definition 2 (Transductive Learning on Attributed Graphs) *Given an undirected attributed graph* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$ *with its adjacent matrix* $A \in \mathbb{R}^{N \times N}$*. Let* $\mathcal{L} = \{(\mathbf{x}_l, \mathbf{y}_l)\}_{l=1}^{|L|}$ *denote a set of labeled nodes, where* $\mathbf{y}_l$ *is the one-hot encoding of node* v_l*'s class label. The rest of nodes belong to the unlabeled set* $\mathcal{U} = \{\mathbf{x}_{l+u}\}_{u=1}^{|U|}$*. The goal is to find among an* *admissible set class labels* $Y = \{\mathbf{y}_{l+u}\}_{u=1}^{|U|}$ *determined by the indicator functions* $f(\mathscr{G};\theta)$*, the one that has smallest classification errors on the unlabeled set* $\mathscr{U} = \{\mathbf{x}_{l+u}\}_{u=1}^{|U|}$.

Transductive learning is a particular setting of semi-supervised learning. It generally assumes that all the data contents, including both training and test data, can be observed in advance and exploited at the same time, but only labels of the training data are available for model updating. Therefore, under the setting of transductive learning on graphs, we can usually access the train set features and their supervision information in conjunction with the whole graph topology information and test node features to facilitate model training, and then minimize the suitable loss to generate label predictions for the test data.

Notations Table 3.1 summarizes a list of frequently used symbols and notations throughout this paper.

Table 3.1 Table of Symbols

Symbol	Description
$\mathscr{G} = \{\mathscr{V}, \mathscr{E}, \mathbf{X}\}$	an undirected graph $\mathscr{G}$ with node set $\mathscr{V}$, edge set $\mathscr{E}$ and node feature matrix X
$\mathbf{x}_i$	feature vector of node v_i
e_{ij}	the edge between v_i and v_j
$\mathscr{L}$, $\mathscr{U}$	the label set and unlabeled set
Y	the class set
$\mathbf{y}_i$	the one-hot label of node v_i
$\mathbf{h}_v^k$	the embedding of node v at *k-th* layer's propagation
A,$\tilde{A}$	the adjacent matrix and the adjacent matrix with added self-connection
D, $\tilde{D}$	the degree matrix of A, $\tilde{A}$
$\mathscr{N}_v$	the neighbors of node v
$H_{\mathscr{N}_v}$	the neighboring node embedding matrix of node v

3.2 Fundamental Architectures

Graph Convolutional Network (GCN) GCN[63] is one of the most representative and influential GNN models for graph representation learning, which has also achieved remarkable performances on the node classification task under the semi-supervised setting. From the methodological point of view, it dramatically reduces the computational complexity to be deployment-friendly on the large scale graph datasets by simplifying the spectral graph filter to be the first-order Chebyshev polynomial with the renormalization technique. This allows GCN to bridge the gap between the spectral GCNNs and spatial GCNNs. Specifically, the layer-wise propagation rule of GCN is defined as Eq.3.1:

$$H^{(l+1)} = \sigma(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{3.1}$$

where $\widetilde{A} = A + I_N$ is the adjacent matrix with added self-connections. I_N is the identity matrix and $\widetilde{D} = \sum_j \widetilde{A}_{ij}$. $H^{(l)}$ is the node representation matrix of l-th layer. When l is 0, $H^{(0)}$ is the input feature matrix X. $W^{(l)}$ is the layer-specific trainable weight matrix in l-th layer. $\sigma(\cdot)$ is the activation function.

Under the semi-supervised node classification setting, a few propagation layers would be stacked together with the softmax function appended in the last layer to obtain the final predicting probability distribution. Generally, for example, a widely-used two-layer GCN can be represented as Eq.3.2:

$$\begin{aligned} Z = f(A, \mathbf{X}) =& softmax(\hat{A}\sigma(\hat{A}\mathbf{X}W^{(0)})W^{(1)}), \\ & with\, \hat{A} = \widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}} \end{aligned} \tag{3.2}$$

Finally, the cross-entropy loss over the labeled nodes is usually utilized for model parameters updating as Eq. 3.3:

$$J = -\sum_{l=1}^{|L|}\sum_{k=1}^{K} y_{lk} log z_{lk} \tag{3.3}$$

where $\mathbf{z}_l = \{z_{l1}, z_{l2}, ..., z_{lK}\} \in Z$ is the class predictions of node v_l, and K is the number of classes.

3.3 Benchmark Datasets

In this section, I summarize the statistics of datasets used in my thesis as listed in Table 3.2. On the whole, I adopt three types of graph datasets that are collected and constructed from various domains, including citation networks, webpage networks and coauthor networks. The detailed dataset information is introduced below.

Table 3.2 Details of Datasets

Dataset	Nodes	Edges	Classes	Features
Citeseer	3,327	4,732	6	3,703
Cora	2,708	5,429	7	1,433
Pubmed	19,717	44,338	3	500
Dblp	17,716	105,734	4	1,639
Wikics	11,701	216,123	10	300
Coauthor-CS	18,333	81,894	15	6,805
Coauthor-Phy	34,493	247,962	5	8,415

Citation networks: Cora, Citeseer, Pubmed[1] [63] and Dblp[2][14]. The four networks are constructed based on the citation relationships among papers, which are collected from four different digital libraries for scientific papers. In these datasets, each node represents a paper, and they are classified into different classes according to their subjects. The node feature is the bag-of-words vector of the paper, which is formed with a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. Furthermore, each edge represents the citation links between two papers.

- **Cora**: The Cora dataset consists of 2,708 scientific publications pertaining to seven different research topics in machine learning. Each node has a 1,433-dimensional feature vector, and there exist 5,429 citation links among these nodes.
- **Citeseer**: The CiteSeer dataset contains 3,312 scientific publications pertaining to six different research topics in machine learning. Each node has a 3,703-dimensional feature vector, and there exist 4,732 citation links among these nodes.
- **Pubmed**: The Pubmed dataset contains 19,717 papers from PubMed database w.r.t. diabetes, which can be categorized into three classes. Each node has a 500-dimensional feature vector, and there exist 44,338 citation links among those publications.

[1]https://linqs.soe.ucsc.edu/data

[2]https://github.com/abojchevski/graph2gauss

- **Dblp**[3]: The Dblp dataset contains 17,716 scientific publication pertaining to four different research topics in machine learning. Each node has a 1,639-dimensional feature vector, and there exist 105,734 citation links among those nodes.

Webpage networks: Wikics[4][89] is a computer science related webpage network in Wikipedia. Nodes represent articles about computer science, and edges represent hyperlinks between articles. The features of nodes are mean vectors of GloVe word embeddings of articles. It contains 11701 articles in 10 different classes. Each article is represented by a 300-dimensional feature vector, and those articles are connected with 216123 edges.

Coauthor networks: Coauthor-CS and Coauthor-Phy[5][113]. They are coauthor networks in the domain of computer science and Physics based on the Microsoft Academic Graph from the KDD Cup 2016 challenge[6]. Nodes represent authors, and the keywords from authors' papers are regarded as node features. Edges indicate coauthor relationships among these authors. According to the most active research fields, authors are categorized into different classes.

- **Coauthor-CS**: The Coauthor_CS dataset consists of 18,333 authors from 15 different research fields in the domain of Computer Science. 81,894 links are built up among these authors by co-authoring papers. Each author has a 6,805-dimensional feature vector extracted from paper keywords.

- **Coauthor-Phy**: The Coauthor_Phy dataset consists of 34,493 authors from 5 different research fields in the domain of Physics. Those authors build up 347,962 links among them by co-authoring papers. Each author has an 8,415-dimensional feature vector extracted from paper keywords.

[3]https://github.com/abojchevski/graph2gauss
[4]https://github.com/pmernyei/wiki-cs-dataset/raw/master/dataset
[5]https://github.com/shchur/gnn-benchmark
[6]https://kddcup2016.azurewebsites.net/

3.4 Conclusion

In this chapter, I have provided some basic knowledge and concepts as the preparation for the technical chapters. Detailed definitions of some important terminologies and notations are firstly introduced. Then, a fundamental GNN architecture, i.e. GCN, is introduced in a mathematical manner, which provides the technical context for the three specific research problems in the latter chapters. After that, the details and summarization of benchmark data sets are presented to offer reference convenience. In the next chapter, I will introduce the proposed semi-supervised adversarial active learning method, and demonstrate its effectiveness through extensive experiments on real-world datasets.

Chapter 4

SEAL: Semi-supervised Adversarial Active Learning on Attributed Graphs[1]

In recent years, the GNN classification models have made remarkable achievements on the node classification task. However, their success exhibits a high dependency on the abundant training data, which poses great burdens due to the expensive consumption of acquiring labels. To cope with this problem, a semi-supervised adversarial active learning framework (SEAL) is proposed on attributed graphs in this chapter. The SEAL framework is able to selectively query labels for the nodes that are most likely leading to performance gains to the model. Extensive experiments demonstrate its effectiveness on four real-world datasets. Next, I would give the detailed description on the proposed SEAL framework.

4.1 Introduction

Recent years have witnessed a great success of Graph Neural Networks (GNNs) in dealing with networked data over various tasks. Node classification is one of the most important tasks in analyzing such content-rich networks, which aims to predict the labels of unlabeled nodes given a partially labeled network. Although GNNs[63] have been demonstrated to be

[1]Yayong Li, Jie Yin, Ling Chen. SEAL: Semi-supervised Adversarial Active Learning on Attributed Graphs. *IEEE Transactions on Neural Networks and Learning Systems*, 32(7): 3136-3147, 2021. [73]

effective in classifying many real-world networks, they rely on a sufficient number of labeled nodes provided to ensure desirable classification accuracy. Very often, however, acquiring a large quantity of node labels requires expert efforts, and is very costly and time-consuming, which significantly limits the true success of these algorithms [19, 122].

In response, active learning (AL) has been proposed to alleviate the label sparsity issue in classifying sparsely labeled networks. It aims at maximizing the learning ability of classification models with the least labeling costs. An AL framework consists of two primary components: a *query engine* which selects an instance from the unlabeled pool to query its label and an *oracle* which provides a label to the queried instance. Different AL algorithms have been proposed in the last decade, which measure the informativeness of instances with certain criteria, and selectively label instances that most potentially lead to performance improvements. For example, Lewis et al. [67] chose to label the instances for which the classifier predicts with the least confidence, and Roy et al. [105] proposed to label the instances which are most likely to bring the model error reduction on the unlabeled pool. These active instance selection strategies allow to construct a more accurate model with fewer labels. However, when AL meets graph-structured data, these classic AL strategies that are effective for non-relational data, such as query-by-committee [86, 112] or uncertainty sampling [67, 68], fail to achieve satisfactory results because of their inability to exploit topological structure of graphs. Accordingly, a series of graph-aware AL strategies have been proposed [40, 55, 82]. These methods directly minimize the expected generalization error or variance of the classifiers and often incur high computational costs. They are designed based on graph structure only, and thus lack the ability to exert the already labeled nodes and rich node features that indeed provide leverage to node classification. By contrast, ICA based algorithms [10, 13, 28] leverage label dependency among neighboring nodes to choose the most informative nodes that can best improve the classifier built on the original node feature space, and thus have limited classification performance.

Recently, graph neural networks (GNNs) [52, 63, 130] have achieved remarkable success by exploiting the power of deep learning in dealing with graph-structured data. Compared with traditional methods like Iterative Classification Algorithm (ICA) [81, 93] or DeepWalk based embedding methods [99], GNN offers its notable advantages through representation learning in terms of capturing graph structure and aggregating neighboring information. Thus, GNN and its variants have achieved state-of-the-art results on both node and graph classification tasks. Therefore, it is of great potential to further advance AL performances on attributed graphs by leveraging the great representation learning ability of GNNs to empower AL, which, however, is largely unexplored yet. AGE [17] and ANRMAB [34] are two AL algorithms that attempt to integrate graph convolutional networks (GCNs) with three AL query strategies, namely graph centrality, information density, and information entropy. While the former uses a linear combination of these strategies, the latter utilizes a multi-armed bandit (MAB) mechanism to dynamically adjust the weights on the respective strategies according to the MAB reward. However, the two algorithms share common weaknesses. First, they use a naive combination of three AL criteria as the informativeness measure. The AL criteria combined still operate separately on different scoring spaces, failing to capture interaction and inter-relatedness between different factors. Second, they use the output of a GCN in a post-processing way to determine the AL query strategy. This means that the AL query engine and the learning of graph embedding still work as two separating pieces; although the newly labeled node can improve the learning of GCN, the capacity of the query engine remains unchanged, leading to unsatisfactory performance improvements.

In this chapter, I propose a novel **SE**mi-supervised **A**dversarial active **L**earning (***SEAL***) framework that seamlessly integrates active learning with deep neural networks to select the most informative nodes to label on attributed graphs. The proposed framework explicitly asserts the usefulness of unlabeled nodes with regard to the existing labeled data. Inspired by adversarial learning, I define an *informativeness measure* based on the intuition that the unlabeled nodes differing the most with the labeled ones carry the most auxiliary information on what the classifier desires the most. Our *SEAL* framework comprises two adversarial components, a *graph embedding network* and a *semi-supervised discriminator network*, which form a closed loop to actively collaborate with each other. The graph embedding network is trained to embed both the labeled and unlabeled nodes into the same latent space that encodes both graph structure and node features, expecting to fool the discriminator to regard all nodes as already labeled. The discriminator learns how to differentiate the unlabeled from the already labeled nodes. Instead of using a binary discriminator, we design a semi-supervised discriminator with multiple outputs. The outputs, on the one hand receiving supervision from the existing labels, serve as class predictions, and on the other hand, produce a unified informativeness score in a common latent space. This score measures the divergence between the unlabeled and already labeled nodes. The unlabeled node with the highest score is selected to query its label. At the same time, the loss of the discriminator is backpropagated to the graph embedding network. The two adversarial components mutually reinforce each other in an iterative way to boost the AL performance.

4.2 Problem Statement and Preliminaries

This section gives a formal problem definition and reviews the preliminaries of GCN and adversarial learning.

4.2.1 Problem Statement

Given an undirected attributed graph $\mathscr{G} = \{\mathscr{V}, \mathscr{E}, \mathbf{X}\}$ with its adjacent matrix $A \in \mathbb{R}^{N\times N}$. We assume there are a small label set $\mathscr{L} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{|L|}$ and an unlabeled set $\mathscr{U}$ in the initial stage, where the number of labeled nodes $|L|$ is far less that the unlabeled set$|U|$. This work focuses on pool-based active learning for node classification. Given a pool of unlabeled nodes, it works sequentially to select one unlabeled node to label at a time while retraining the classifier at each iteration.

Given a fixed labeling budget B and an initial labeled set $\mathscr{L}$, our active learning problem aims to design a good query strategy $Q(v_i;\Theta)$ parameterized with Θ that specifies which unlabeled node should be selected to label at each iteration. $Q(v_i;\Theta)$ can also be considered as a utility function that assigns a score to each unlabeled node indicating its informativeness to the current classifier. The unlabeled node that maximizes $Q(v_i;\Theta)$ is selected to label, so that the retrained classifier can achieve the maximum accuracy [31]. Formally, this can be expressed as follows:

$$v^* = \underset{v_i \in \mathscr{U}}{\operatorname{argmax}}\, Q(v_i; \Theta), \tag{4.1}$$

where v^* represents the most informative node that is selected to query its label at each iteration, and Θ represents the parameters of the learned query strategy.

4.2.2 Preliminaries on Adversarial Learning

Generative Adversarial Networks (GANs) [38] have emerged as a powerful framework for learning deep representations of arbitrarily complex data distributions via an adversarial process. A regular GAN sets up an adversarial platform for a generator $g_\theta(\cdot)$ and a discriminator $d_\phi(\cdot)$, where $g_\theta(\cdot)$ intends to produce samples as close to the real data as possible, while $d_\phi(\cdot)$ tries to tell apart samples either from the real data ($\mathbf{x} \sim \mathscr{P}_{data}(\mathbf{x})$), or from the generator ($\mathbf{s} \sim \mathscr{P}_g(\mathbf{s})$) as accurately as possible. This adversarial process is formulated as a min-max game with the following loss function:

$$\min_\theta \max_\phi E_{\mathbf{x} \sim \mathscr{P}_{data}(\mathbf{x})} log d_\phi(\mathbf{x}) + E_{\mathbf{s} \sim \mathscr{P}_g(\mathbf{s})} log(1 - d_\phi(g_\theta(\mathbf{s}))) \tag{4.2}$$

The conventional GAN framework finally converges at the state where $g_\theta(\cdot)$ recovers the training data perfectly and $d_\phi(\cdot)$ predicts 0.5 everywhere. Recently, this adversarial idea is adopted to solve active learning problems on sequence and image classification. Deng et al. [25] designed a sequence-based AL algorithm that utilizes a binary adversarial network to shrink the search space of candidate samples, while an uncertainty strategy has to be applied to determine which sample should be chosen. Sinha et al. [118] used a similar adversarial active learning method to discriminate labeled and unlabeled images. However, both methods use a fully supervised binary discriminator and ignore the original class probability distribution. Our work was developed independently from these works, which yet focuses on graph-structured data where graph topological structure needs to be properly exploited. Our adversarial discriminator works in a semi-supervised manner, which allows to distinguish labeled from unlabeled nodes and to model the class probability distribution.

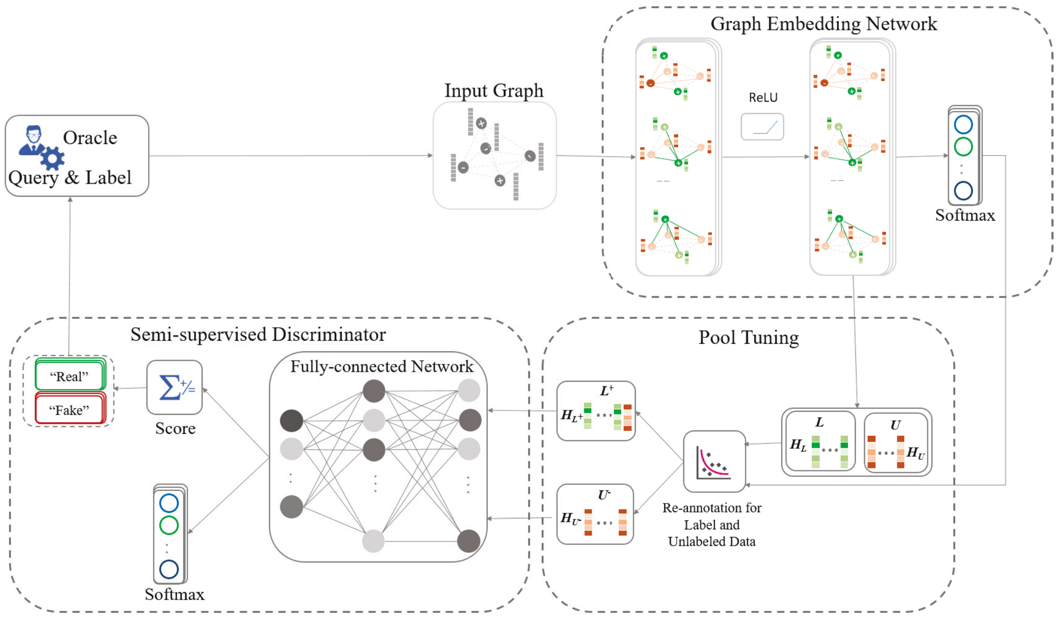


Fig. 4.1 The *SEAL* Framework is composed of three main components: a graph embedding network, a pool tuning (*PT*), and a semi-supervised discriminator network.

4.3 The SEAL Framework

This section presents the overview of the *SEAL* framework, followed by detailed description of the main components.

4.3.1 Framework Overview

Our active learning objective is to select the nodes differing the most with the existing labeled nodes so as to improve the node classification performance with the minimal labeling cost. To measure the discrepancy between the unlabeled and labeled nodes, we incorporate the adversarial learning method into active learning, and formulate it to be a GAN-like framework with one graph embedding network $G(\cdot)$ with a semi-supervised discriminator network $D(\cdot)$. $G(\cdot)$ embeds both the labeled and unlabeled nodes into a common latent space with the uniform distribution to maximally confuse the discriminator, whereas $D(\cdot)$ intends to distinguish the unlabeled from labeled data as much as possible. The discriminator $D(\cdot)$ is iteratively reinforced through this competitive process, which provides a unified quantitative criterion to measure the divergence of the unlabeled nodes with respect to the existing labeled nodes. This criterion ideally enables the selection of the most informative unlabeled node to be labeled by an oracle.

Our *SEAL* framework comprises three main components, namely a graph embedding network $G(\cdot)$, a pool tuning (*PT*), and a discriminator network $D(\cdot)$. As illustrated in Fig. 4.1, its workflow operates as follows:

1) Taking graph $\mathscr{G}$ as input, the graph embedding network $G(\cdot)$ encodes both the labeled and unlabeled nodes into low-dimensional, latent node representations, H_L and H_U, respectively, with the aim to characterize their class attributes and fool the discriminator *D(·)*, simultaneously.

2) The latent node representations and their prediction probabilities are then passed to pool tuning (*PT*). *PT* picks a portion of nodes with high prediction certainty from the unlabeled pool U and moves them to the labeled pool $\mathscr{L}$. The two tuned pools are named *pseudo* labeled (*p-labeled*) pool $\mathscr{L}^+$, and *pseudo* unlabeled (*p-unlabeled*) pool $\mathscr{U}^-$, respectively. Correspondingly, their latent representations are denoted as H_{L^+} and H_{U^-}.

3) The discriminator network $D(\cdot)$ takes H_{L^+} and H_{U^-} as input and maps them into a latent space to generate multiple outputs. These outputs not only produce the probabilities of nodes belonging to K classes, but also generate a scoring function to quantify the informativeness of unlabeled nodes with respect to the existing labeled data. The unlabeled node with the highest score from the *p-unlabeled* pool $\mathscr{U}^-$ is selected to be labeled. After that, the original labeled pool $\mathscr{L}$ and unlabeled pool $\mathscr{U}$ are updated and re-input to $G(\cdot)$.

Using this adversarial learning approach, the graph embedding network and the discriminator network form a closed loop to collaborate and reinforce each other.

4.3.2 Graph Embedding Network

Our graph embedding network $G(\cdot)$ encodes all nodes into a low-dimensional, latent embedding space with better representation and discrimination power for learning an AL classifier.

For this purpose, we use a specifically modified GCN as the classifier and the representation learner in the *SEAL* framework, which is actually generic in nature and can be directly replaced by any other graph embedding algorithms. On the one hand, $G(\cdot)$ takes the same responsibility as GCN to learn the latent feature representations of both labeled and unlabeled nodes by using both graph structure and node attributes. The latent node representations are used to learn a classifier for prediction in a latent space. Thus, the cross entropy loss as Eq.(4.3) is incorporated into our loss function for predicting the labels of graph nodes:

$$J_{GCN} = -\sum_{l=1}^{|L|}\sum_{k=1}^{K} y_{lk} log z_{lk} \tag{4.3}$$

where $\mathbf{z}_l = \{z_{l1}, z_{l2}, ..., z_{lK}\} \in Z$ is the node v_l's class predictions generated by $G(\cdot)$, and K is the number of classes.

On the other hand, $G(\cdot)$ is expected to provide guidance for the downstream discriminator to improve its capability of measuring the divergence of unlabeled nodes w.r.t. the existing labeled nodes. Inspired by the adversarial idea of GAN, the loss derived from distinguishing the labeled and unlabeled nodes by the discriminator $D(\cdot)$, is backpropagated to $G(\cdot)$ to subsequently reinforce the generalization and discrimination capability of the discriminator. In this way, $G(\cdot)$ intends to fool the discriminator $D(\cdot)$ to believe all nodes are from the labeled pool, while the discriminator tries to learn how to differentiate the unlabeled from the existing labeled nodes. Concretely, instead of having the discriminator $D(\cdot)$ output a score "1" for both labeled and unlabeled nodes as the cross-entropy loss does, we employ *feature matching* method to rectify their feature distributions, which was empirically proved to be more effective in situations where traditional cross-entropy based supervised methods are volatile [107]. This method is aimed at minimizing the mean discrepancy of feature distributions between *p-labeled* and *p-unlabeled* data obtained from the intermediate layer of the discriminator, so that the *p-unlabeled* data can match the statistics of *p-labeled* data. We assume that $G^{(m)}$ and $D^{(n)}$ represent the hidden representation of the m-layer of graph embedding network and n-layer of the discriminator, respectively.

Finally, the loss function of our graph embedding network is formulated as:

$$J_G = \|\mathbb{E}_{\mathbf{x}\sim\mathscr{L}^+} D^{(n)}(G^{(m)}(\mathbf{x})) - \mathbb{E}_{\mathbf{x}\sim\mathscr{U}^-} D^{(n)}(G^{(m)}(\mathbf{x}))\|^2 + J_{GCN}, \tag{4.4}$$

where the first term measures the mean feature discrepancies of nodes in the *p-labeled pool* $\mathscr{L}^+$ and the *p-unlabeled pool* $\mathscr{U}^-$, backpropagated from the discriminator, and the second term is the cross-entropy loss that is calculated using the existing labeled data as Eq. (4.3).

By minimizing this loss function, $G(\cdot)$ tries to not only minimize the classification loss of predicting the labels of nodes, but also to force the distribution $\mathbf{x} \sim \mathscr{P}_{U^-}$ to approximate the distribution of $\mathbf{x} \sim \mathscr{P}_{L^+}$. As the training proceeds, by descending mean feature discrepancies sent from the discriminator, $G(\cdot)$ is able to push the two distributions closer. The indistinguishable distributions, reversely, drives $D(\cdot)$ to improve its classification ability by descending its labeled-unlabeled classification errors. Thus, $G(\cdot)$ and $D(\cdot)$ form a GAN-like framework, where $G(\cdot)$ embeds both the labeled and unlabeled nodes into a common latent space with the uniform distribution to fool the discriminator, while $D(\cdot)$ intends to distinguish the unlabeled nodes from the labeled pool as accurately as possible. By appropriately parameterizing and optimizing $G(\cdot)$ and $D(\cdot)$, the adversarial process can iteratively strengthen the discriminator with high generalization and discrimination capability.

4.3.3 Pool Tuning

Our objective is to select unlabeled nodes that can provide auxiliary information that has not been captured by the classifier from the labeled data yet. After several epoches of training, it can be assumed that the information of labeled data has been sufficiently acquired by GCN. As for the unlabeled data, many unlabeled nodes, especially those for which the current classifier can predict with high certainty, carry similar information that the current classifier has already captured, due to local dependencies between neighboring nodes. Thus, we tune the distribution of the labeled and unlabeled data according to the uncertainty predicted by the current classifier.

According to the GCN's estimated probability distribution, we re-annotate a portion of unlabeled nodes with high predicting confidence as the *pseudo* labeled data, and exclude them from the unlabeled pool. The tuned labeled and unlabeled pool are denoted as $\mathscr{L}^+$ and $\mathscr{U}^-$, respectively. We use a threshold δ, whose value will be empirically determined, to decide which unlabeled nodes should be moved to the labeled pool. Specifically, for any node whose predicting probability on any class exceeds the threshold δ, it would be re-annotated and put into the labeled pool $\mathscr{L}^+$, using Eq.(4.5) and Eq.(4.6).

$$\mathscr{L}^+ = \mathscr{L} \cup \{\mathbf{x}_i \in \mathscr{U} | P(\hat{\mathbf{y}}|\mathbf{x}_i) > \delta\}, \tag{4.5}$$

$$\mathscr{U}^- = \{\mathbf{x}_i \in \mathscr{U} | P(\hat{\mathbf{y}}|\mathbf{x}_i) <= \delta\}, \tag{4.6}$$

where $\hat{\mathbf{y}}$ denotes the most probable class that $\mathbf{x}_i$ belongs to. $\mathscr{L}^+$ and $\mathscr{U}^-$ are then fed to our discriminator that decides the most informative node to be selected from $\mathscr{U}^-$.

4.3.4 Semi-supervised Adversarial Learning

Following pool tuning, we design a discriminator network $D(\cdot)$ that approximates a divergence measure to gauge the discrepancy between the *p-unlabeled* and *p-labeled* data distribution. In other words, the discriminator tries to tell apart *p-labeled* nodes from *p-unlabeled* nodes by minimizing an appropriate loss function.

Instead of using a simple cross-entropy based binary discriminator, we design a semi-supervised discriminator $D(\cdot)$ that outputs $K+1$ probabilities, where K probabilities correspond to probabilities of the node belonging to the K specific classes, and one probability corresponds to the probability of the node be from the unlabeled pool [107]. It has the advantage of being able to distinguish the *p-unlabeled* from *p-labeled* nodes, but also being capable to predict the probabilities of nodes belonging to specific classes. Generally speaking, the appropriately trained discriminator could lie its decision boundary between data manifolds of different classes, which would in turn improves the generalization performance of the discriminator [72]. Thus, in our AL problem, this objective naturally achieves a good trade-off between the *exploitation* that finds the most informative nodes to improve the prediction task, and the *exploration* in the latent feature space.

For a K-class problem, we assume that $D(\cdot)$ takes $G^{(m)}(\mathbf{x})$ as input and outputs $(K+1)$-dimensional logits $l_1(\mathbf{x}), l_2(\mathbf{x}), \cdots, l_{K+1}(\mathbf{x})$. These logits, by applying a softmax function, are then turned into class probabilities $P(\mathbf{y}=j \mid \mathbf{x})(j \in 1,2,..,K+1)$, of which $P(\mathbf{y}=K+1 \mid \mathbf{x})$ represents the probability of $\mathbf{x}$ being unlabeled. Thus, the loss function of this discriminator can be formulated as follows:

$$J_D = \alpha \cdot J_{sup} + J_{unsup}, \tag{4.7}$$

$$J_{sup} = -\mathbb{E}_{\mathbf{x}\sim\mathscr{L}} log P(\mathbf{y} \mid \mathbf{x}, \mathbf{y} < K+1), \tag{4.8}$$

$$J_{unsup} = -\{\mathbb{E}_{\mathbf{x}\sim\mathscr{L}^+} log(D(G^{(m)}(\mathbf{x}))) + \mathbb{E}_{\mathbf{x}\sim\mathscr{U}^-} log(1 - D(G^{(m)}(\mathbf{x})))\}, \tag{4.9}$$

$$D(G^{(m)}(\mathbf{x})) = 1 - P(\mathbf{y} = K+1 \mid \mathbf{x}), \tag{4.10}$$

where J_{sup} and J_{unsup} denote the supervised loss and unsupervised loss, respectively. The two components are balanced by a hyper-parameter α. J_{sup} is calculated with only the original labeled data $\mathscr{L}$ using cross entropy, while J_{unsup} is calculated using both *p-labeled* nodes $\mathscr{L}^+$ and *p-unlabeled* nodes $\mathscr{U}^-$ via the adversarial training method. $D(G^{(m)}(\mathbf{x}))$ represents the likelihood of node $\mathbf{x}$ being *p-labeled*. The optimal solution to minimizing J_{sup} and J_{unsup} is to have $e^{l_j(\mathbf{x})} = c(\mathbf{x})p(\mathbf{y}=j,\mathbf{x}), \forall j < K+1$ and $e^{l_{K+1}(\mathbf{x})} = c(\mathbf{x})p(\mathbf{y}=K+1,\mathbf{x})$ for some undetermined scaling function $c(\mathbf{x})$. In other words, that means a perfect solution to minimizing J_{unsup} is also perfect to minimizing J_{sup}. Thus, the consistence of J_{sup} and J_{unsup} could guarantee that the optimization of J_{unsup} also helps improve the supervised performance [6]. As such, we expect to better estimate the optimal solution by minimizing the two loss functions jointly.

Furthermore, because $(K+1)$-output classification tends to have the over-parameterized problem, we adopt the following strategy: Given that subtracting a term $f(\mathbf{x})$ would not change the softmax distribution, we fix the last output logit $l_{K+1}(\mathbf{x})$ as zero by operating Eq. (4.11):

$$\hat{l}_j(\mathbf{x}) = l_j(\mathbf{x}) - f(\mathbf{x}),\ \forall j \leq K+1, \tag{4.11}$$

Therefore J_{sup} is recast into a standard supervised loss function with K classes. The probability of nodes being labeled is thus given by:

$$D(\mathbf{x}) = \frac{\sum_{k=1}^{K} e^{\hat{l}_k(\mathbf{x})}}{\sum_{k=1}^{K} e^{\hat{l}_k(\mathbf{x})} + 1}. \tag{4.12}$$

4.3.5 Active Scoring

For active learning, we define an *informativeness measure* based on the divergence between the *p-labeled* nodes and *p-unlabeled* nodes. Intuitively, the more divergent an unlabeled node is from the existing labeled data, the more likely it would contribute useful information to the current classifier. As described in Section 4.3.4, the output of the discriminator can provide a divergence measure. Thus, we devise an active scoring function as

$$div(\mathbf{x}_{U^-}, \mathscr{L}^+) = 1 - D(\mathbf{x}_{U^-}). \tag{4.13}$$

Intuitively, the higher the score is, the more informative the *p-unlabeled* node is with respect to the existing labeled data. Consequently, we select node x^* from the *p-unlabeled* pool $\mathscr{U}^-$ such that $div(x^*, \mathscr{L}^+)$ is maximized and query its label.

4.3.6 Model Training and Complexity Analysis

The model training of *SEAL* is given in Algorithm 1. In the main training loop, we iteratively train the graph embedding network $G(\cdot)$ and the discriminator $D(\cdot)$ (lines 3-7) and then proceed with the instance selection process (lines 8-11). The major computational cost of Algorithm 1 lies in training $G(\cdot)$ and $D(\cdot)$. The computational complexity of a two-layer GCN is linear with the number of edges $|\mathscr{E}|$, i.e., $\mathscr{O}(|\mathscr{E}|HKM)$, where M denotes the dimension of node features, H denotes number of hidden layer units, and K denotes number of classes. The computational complexity of the three-layer discriminator is linear with the number of nodes, i.e., $\mathscr{O}(NKH_1H_2|M'|)$, where $|M'|$ denotes the dimension of the input node representations, H_1, H_2 denote the number of the two hidden layer units, respectively. Therefore, in the training process, the overall computational complexity of *SEAL* is $\mathscr{O}(|\mathscr{E}|HKM + NKH_1H_2|M'|)$, which is linear with number of edges $|\mathscr{E}|$ and number of nodes N.

Algorithm 1 *SEAL* Model Training

```
Input: Graph $\mathscr{G}(\mathscr{V},\mathscr{E},\mathbf{X})$, node sets $\mathscr{L},\mathscr{U}$, labeling budget $B$, pre-training epoches $n_p$,
    training epoches $n_G$ and $n_D$ for $G(\cdot)$ and $D(\cdot)$
Output: a set of selected nodes $\mathscr{L}_t$ Initialize the parameter of $G(\cdot)$ and $D(\cdot)$ network
 1: while not converged do
 2:     for $t_G = 0; t_G < n_G; t_G = t_G + 1$ do
 3:         Update $G(\cdot)$ by descending gradients of Eq.(4.4)
 4:     end for
 5:     Tune and generate the candidate pools $\mathscr{L}_t^+$ and $U_t^-$ based on Eq.(4.5) and Eq.(4.6)
 6:     for $t_D = 0; t_D < n_D; t_D = t_D + 1$ do
 7:         Update $D(\cdot)$ by descending gradients of Eq.(4.7)
 8:     end for
 9:     if $t > n_p$ and $|L_t| - |L| < B$ then
10:         Calculate scores for nodes in $\mathscr{U}_t^-$ using Eq.(4.13)
11:         Select node $\mathbf{x}^* \leftarrow \mathrm{argmax}\, div(\mathbf{x}_{U_t^-}, \mathscr{L}_t^+)$
12:         Update pools: $\mathscr{L}_t \leftarrow \mathscr{L}_t \cup \mathbf{x}^*$, $\mathscr{U}_t \leftarrow \mathscr{U}_t \setminus \mathbf{x}^*$
13:     end if
14: end while
15: return a set of selected nodes $\mathscr{L}_t$
```

4.3.7 Discussion: Differences from GAN

Our *SEAL* framework is designed as a GAN-like architecture. However, its training objective is essentially different from a regular GAN. As mentioned in Section 4.2.2, a regular GAN generally aims to obtain a perfect generator, and it finally converges at the state where the generator exactly recovers the training data distribution while predicting probability of the discriminator equals to 0.5 everywhere. In contrast, our objective is to obtain a strong discriminator that can measure the divergence between the labeled and unlabeled data distribution with high confidence to enable active instance selection.

If the $G(\cdot)$ achieves the exact match between the distribution of $x \sim \mathscr{P}_{U^-}$ and $x \sim \mathscr{P}_{L^+}$, for any optimal solution $\mathscr{S}$ to the supervised loss J_{sup}, there exists an optimal solution $\mathscr{S}^*$ to the semi-supervised $(K+1)$-class objective J_D such that $\mathscr{S}$ and $\mathscr{S}^*$ share the same generalization error. Therefore, under the semi-supervised setting in *SEAL*, a perfect $G(\cdot)$ that can exactly matches the two distribution of $x \sim \mathscr{P}_{L^+}$ and $x \sim \mathscr{P}_{U^-}$ would not be able to improve the generalization capability of the discriminator over the supervised setting. Consequently, a weaker $G(\cdot)$ would be necessary to guarantee a stronger discriminator. Thus, it is required to appropriately optimize Eq.(4.4) and Eq.(4.7) using an alternating optimization switching between updates to the $G(\cdot)$ and $D(\cdot)$. While this optimization is not guaranteed to converge, empirically it provides us a strong discriminator if $G(\cdot)$ and $D(\cdot)$ are well balanced. This is consistent with existing findings in [23].

4.4 Experimental Analysis

To validate the effectiveness of our *SEAL* framework, a series of experiments are conducted on node classification tasks under a transductive, pool-based AL setting: Given the initial labeled nodes and a certain labeling budget, unlabeled nodes are iteratively selected to label and train a classifier, whose performance is tested from different perspectives.

4.4.1 Datasets

Four benchmark citation networks, including Citeseer[2], Cora[2], Pubmed[2] [108], and DBLP[3] [150], are used in our experiments. In these networks, each node represents a document with a certain label and each edge represents the citation links between two documents. We treat

[2]https://linqs.soe.ucsc.edu/data
[3]https://aminer.org/citation

Table 4.1 Statistics of Data Sets

Dataset	Nodes	Edges	Classes	Features	$\lvert L_{init}\rvert / \lvert L_{max}\rvert$
Citeseer	3327	4732	6	3703	24 / 120
Cora	2708	5429	7	1433	28 / 140
DBLP	18447	91052	4	2476	16 / 80
Pubmed	19717	44338	3	500	12 / 60

Table 4.2 The Micro-F1 and Macro-F1 performance comparison with L_{max} labeled nodes for training

Method	Citeseer		Cora		DBLP	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
GCN-Random	0.685	0.615	0.812	0.794	0.720	0.702
AGE	0.717	0.666	0.813	0.800	0.772	0.719
ANRMAB	<u>0.721</u>●	<u>0.672</u>●	<u>0.819</u>●	<u>0.807</u>●	<u>0.778</u>●	<u>0.739</u>●
ALFNET	0.650	0.613	0.765	0.750	0.656	0.628
SEAL-ad	0.712	0.660	0.819	0.809	0.757	0.706
SEAL-fm	0.721	0.665	0.825	0.816	0.760	0.705
SEAL-sal	0.717	0.662	0.817	0.806	0.762	0.716
SEAL-pt	0.733	0.675	0.829	0.815	0.782	0.711
SEAL	**0.734**	**0.676**	**0.831**	**0.822**	**0.802**	**0.747**

these networks as undirected and unweighted graphs, and each node is characterized by a sparse bag-of-words feature vector according to word occurrence. DBLP is a subgraph of the DBLP bibliographic network, including publications from four research areas. Cora, Citeseer and DBLP are used to evaluate the classification-related performance, and Pubmed is used to test the computational complexity. Details of the four datasets are summarized in Table 4.1.

At the beginning of the training process, only $|L_{init}|$ labels are accessible, and as AL algorithms proceed, one unlabeled node is selected to label at each iteration. A maximum number of $|L_{max}|$ labels can be queried. In our implementation, for each dataset, we start with 4 labeled nodes per class $|L_{init}| = 4 \times K$, as the initial state, and the labeling budget $B = 20 \times K - |L_{init}|$, where K denotes the number of classes.

4.4.2 Experimental Set-up

Our experiments closely follow the settings as in [17, 63]. For each dataset, we randomly choose 1000 nodes for testing, 500 nodes for validation. To quantify the performance difference induced only by different AL query strategies, instead of randomly sampling 500 nodes as the validation set for each run of experiments, we generate 10 different validation sets by randomly sampling from the non-testing unlabeled pool and repeat experiments for 10 times on each validation set. This setting is designed to ensure that the same unlabeled pool is used when running different AL query strategies. Finally, the reported results are averaged over 10 (validation sets) $\times 10$ (initial labeled sets).

We utilize a two-layer GCN network, and its hidden layer has 16 units. ReLU and $L2$ regularization (5×10^{-4}) are applied for the first layer only. It is optimized using Adam, where the learning rate is 0.005 for the instance selection period, and 0.01 for the subsequent node prediction period, and the dropout rate is 0.5. The discriminator is a three-layer fully-connected neural network with $(128, 128, K)$ units, respectively, and each layer is followed by Leaky ReLU activation. It is also optimized using Adam, where the learning rate is 0.01, and the dropout rate is 0.5. Before the AL query process starts, the whole network is pre-trained for 300 epoches, i.e. $n_p = 300$ in Algorithm 1, to ensure that $G(\cdot)$ has adequate representation learning capacity and $D(\cdot)$ has adequate discrimination ability.

4.4.3 Baselines

We compare our *SEAL* framework against four state-of-the-art active learning methods, with details as follows:

- **AGE** [17] and **ANRMAB** [34]: They are two state-of-the-art methods that combine GCN with classic AL strategies, via a linear combination of three AL query strategies (graph centrality, information density, and uncertainty sampling). ANRMAB improves AGE by dynamically adjusting the weights of different strategies based on the MAB reward. They differ from *SEAL* in terms of different AL query strategies designed, which are used to validate the efficacy of *SEAL*'s unified AL query strategy.

- **ALFNET** [13]: It is a traditional AL strategy that uses ICA and QBC ensemble to make instance selection. This method is used to evaluate the advantages of GNN-based AL methods over traditional graph-based AL methods. In our settings, we adapt it as a transductive semi-supervised version and allow it to select only one node at each iteration for fair comparison.

- **GCN-Random** [63]: It uses GCN as the classifier but randomly chooses one unlabeled node to querty its label.

To assess the importance of different aspects of *SEAL*, we also compare with four variants of *SEAL* via ablation studies:

- **SEAL-ad**: It is a variant of *SEAL* with adversarial learning obliterated. Specifically, it changes $G(\cdot)$'s loss function in Eq.(4.4) as $J_G = J_{GCN}$. $D(\cdot)$ is still used to discriminate unlabeled from labeled nodes with loss function J_D as Eq.(4.7), but loss of $D(\cdot)$ is not back propagated to $G(\cdot)$. It is designed to validate effectiveness of the adversarial learning mechanism.

- **SEAL-fm**: This method is a variant of *SEAL* to test the effectiveness of feature matching loss for generator $G(\cdot)$. Instead of minimizing feature matching loss as Eq.(4.4), it maximizes the log-likelihood of both labeled and unlabeled nodes to confuse $D(\cdot)$, given by

$$J_G = -\mathbb{E}_{\mathbf{x} \sim \mathscr{L} \bigcup \mathscr{U}} D(G^{(m)}(\mathbf{x})) + J_{GCN}. \tag{4.14}$$

- **SEAL-sal**: This method is another variant of *SEAL* that uses a cross-entropy based binary p-labeled/p-unlabeled discriminator. It is equivalent to setting α as 0 in Eq.(4.7), while other parameters remain the same as with *SEAL*.

- **SEAL-pt**: This method differs from *SEAL* in that it removes the operation of *PT*, and the generated nodes representations are directly sent to $D(\cdot)$. This is equivalent to setting δ to 1 in Eq.(4.5) and (4.6). Other parameters remain the same with *SEAL*.

For classification, ALFNET uses the ICA with logistic regression as the base classifier. All other algorithms use GCN as the classifier for fair comparison.

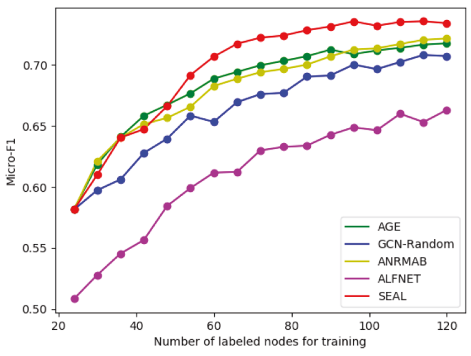
4.4.4 Overall Performance Comparison

We use Micro-F1 and Macro-F1 scores as the evaluation criteria to validate node classification performance. Table 4.2 compares the performance of different algorithms on Citeseer, Cora, and DBLP. For *SEAL*, we set parameters α and δ to 0.6 for this experiment. The best performer is highlighted by **bold**, and the second best performer is highlighted by underline on each setting. Overall, as can be seen, *SEAL* exhibits evident advantages over other baselines by designing a new AL query strategy on a unified scoring space. In terms of Micro-F1 score, *SEAL* improves upon ANRMAB by a margin of 1.3%, 1.2% and 2.4%, respectively, on the three datasets. Similar results can also be seen on the Macro-F1 score results. Moreover, we have also performed paired t-test between the Micro-F1 scores achieved by SEAL and the best baseline methods, where we use $\bullet(\circ)$ to indicate that the SEAL is significantly better (worse) than the compared baseline methods at 95% significance level. This significance test further verify the advantage of our SEAL. This is in accordance with our expectation that the discriminator allows to further exploit the GNN-generated representations in a new latent space, where dependencies among these latent representations can be better captured to support instance selection. Furthermore, our mechanism could select nodes with least redundant information. The two reasons lead to superior classification performance to weighted combination methods like AGE and ANRMAB. It is also worth noting that ALFNET performs consistently worse than any other GNN-based method, including GCN-Random. This proves the advantageous representation power of GNN over traditional ICA-based methods.

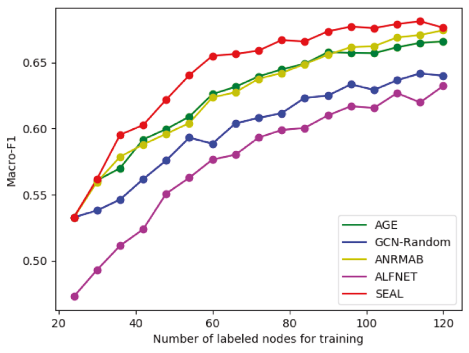
4.4.5 Ablation Study

We also conduct a ablation study to assess the importance of different components of *SEAL*. Our analysis is reported in the bottom section of Table 4.2.

Overall, *SEAL* has stable performance gains over its degraded counterparts, owing to the contributions of different components. The adversarial leaning mechanism (SEAL-ad) allows $G(\cdot)$ and $D(\cdot)$ to reinforce each other and thus boost the classification ability of $D(\cdot)$.



(a) Micro-F1



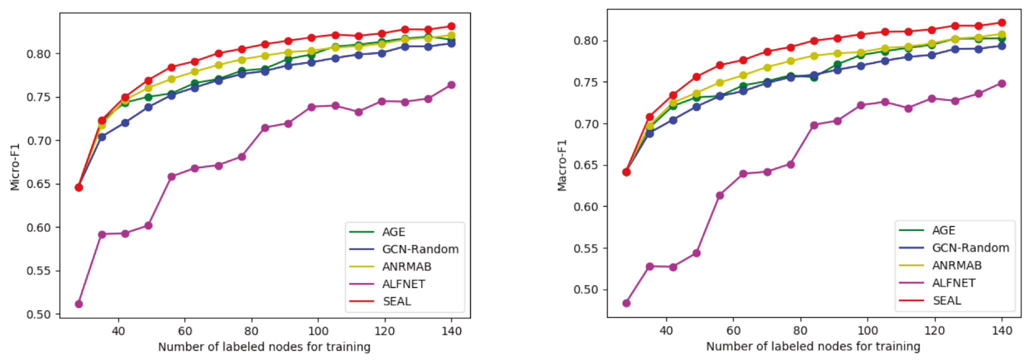
(b) Macro-F1

Fig. 4.2 Performance comparison with respect to different labeling budgets on Citeseer

This enables to select the most informative samples for improving the overall performance. Feature matching (SEAL-fm), as a method for alleviating overfitting and mode collapse problems in GAN [107], effectively stabilizes the training of *SEAL*. Pool tuning (SEAL-pt), redistributing labeled and unlabeled nodes, enables $D(\cdot)$ to better distinguish unlabeled nodes and to reduce the search space of unlabeled candidates. Semi-supervised adversarial learning mechanism (SEAL-sal) allows $D(\cdot)$ to measure the divergence between labeled and unlabeled nodes. Effectiveness of pool tuning and semi-supervised adversarial learning mechanism will be further analyzed carefully in Section 4.4.7 and 4.4.8.

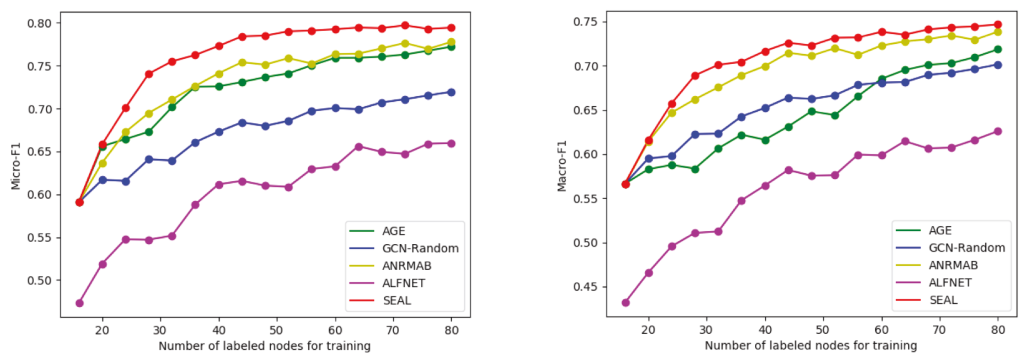
4.4.6 Performance Comparison on Different Labeling Budgets

Figures 4.2-4.4 compare classification performance of different methods with respect to different labeling budgets on Citeseer, Cora, and DBLP. Although all methods have an overall upward trend as the number of labeled nodes increases, *SEAL* offers the steepest improvement slopes with remarkable gains over other baselines. Taking Figure 4.2(a) as an example, *SEAL* reaches 72.0% of classification accuracy with only 66 labeled nodes, while ANRMAB reaches the similar accuracy until obtaining 120 labeled nodes. This indicates that *SEAL* achieves similar classification accuracy with much fewer nodes labeled, which again proves the effectiveness of our AL strategy.



(a) Micro-F1 (b) Macro-F1

Fig. 4.3 Performance comparison with respect to different labeling budgets on Cora.

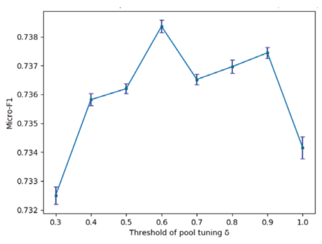


(a) Micro-F1 (b) Macro-F1

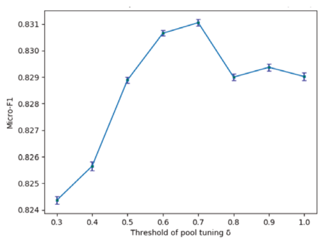
Fig. 4.4 Performance comparison with respect to different labeling budgets on DBLP

4.4.7 Effectiveness Study on *PT*

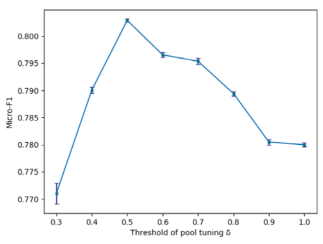
Figure 4.5 shows performance changes with varying thresholds δ for pool tuning, where a similar trend can be observed on Citeseer, Cora and DBLP. As can be seen, *SEAL* achieves the best performance when δ is equal to 0.6, and degrades with varying speeds as δ further increases. The standard variances are plotted as a vertical line at each point, which are calculated over 100 repeating tests on each threshold. The longer the vertical line is, the larger the standard variance is. Pool tuning picks a bunch of unlabeled nodes carrying similar information with already labeled nodes as pseudo labeled nodes. The redistributed node sets make it easier for $D(\cdot)$ to find the most distinct unlabeled nodes. Moreover, it helps avoid



(a) Citeseer

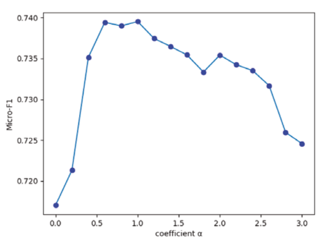


(b) Cora

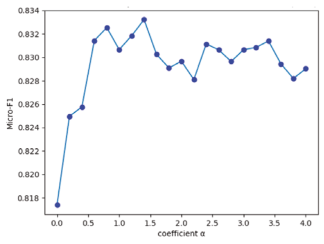


(c) DBLP

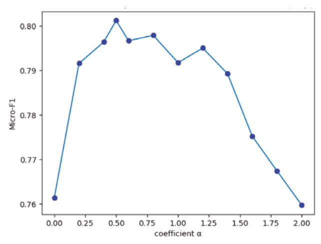
Fig. 4.5 Comparison of Micro-F1 with respect to varying δ values on Citeseer, Cora, and DBLP.



(a) Citeseer



(b) Cora



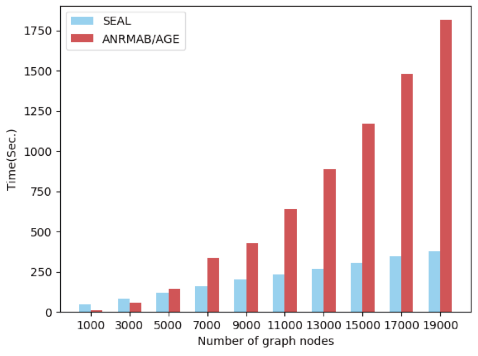
(c) DBLP

Fig. 4.6 Comparison of Micro-F1 scores with respect to varying α values on Citeseer, Cora, and DBLP.

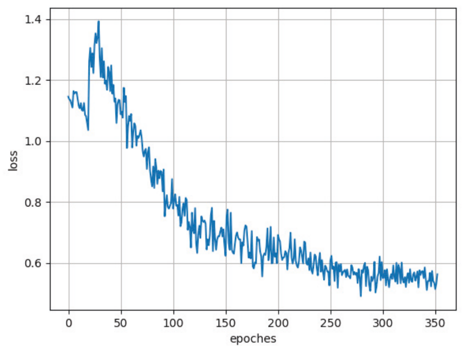
the potential overfitting caused by extremely imbalanced labeled/unlabeled samples, thus leading to more stable performance.

4.4.8 Effectiveness Study on the *SAL*

Figure 4.6 illustrates the sensitivity of *SEAL* with respect to α that balances the trade-off between J_{sup} and J_{unsup} in Eq.(4.7). Taking DBLP in Figure. 4.6(c) as an example, when α is zero, J_D degrades as an unsupervised loss function (i.e. $J_D = J_{unsup}$), where only 76.2% of predictions are correctly made. Then, the performance fluctuates with the increase of α, during which it reaches the peak at the point around α=0.6. An appropriate α allows $D(\cdot)$ to be aware of differentiation between different classes when measuring the similarity between *p-unlabeled* and *p-labeled* nodes. This awareness alleviates the risk of selecting abnormal nodes as it often occurs in uncertainty sampling methods, thereby resulting in better classification accuracy.



(a) Comparison of Training Time



(b) Convergence of Training Loss

Fig. 4.7 Training Time and Convergence Analysis on Pubmed

4.4.9 Training Time Comparison and Convergence Rate

We also conduct experiments to compare the training time (in seconds) of three GCN-based AL methods, *SEAL*, AGE, and ANRMAB. All methods are implemented in tensorflow on a Linux system with Inter(R) Xeon CPU E5-2690 @3.4GHz*8 and 32G memory. We compare their training time on Pubmed, where the number of nodes increases from 1000 to 19000 with an increment of 2000. We record the total training time for selecting 48 unlabeled nodes for labeling, and the results are shown in Figure 4.7(a). As we can see, ANRMAB and AGE take almost the same amount of training time, which is less than *SEAL* at the beginning, when network size is small. However, as network size increases, the training time of ANRMAB/AGE rapidly grows at a non-linear rate, which is several times more than that of *SEAL*. This is because the frequent sorting operation in ANRMAB/AGE incurs a computational overhead of $\mathscr{O}(N^2)$, whereas *SEAL* maintains a linear growth rate with respect to number of nodes N. This indicates that *SEAL*'s computational overhead for instance selection is acceptable and it is reasonably efficient on large graphs.

Figure 4.7(b) shows changes in training loss J_G as the training proceeds. At the beginning, as new unlabeled nodes start to be added into the labeled set, the training loss exhibits some fluctuations. This is probably due to that conspicuously distinct information carried by the newly added nodes disturbs parameter updates of the classifier trained by existing labeled data. As more nodes are labeled, the training loss quickly decreases and stabilizes at a lower level. The overall trend of loss J_G exhibits good convergence.

4.5 Conclusion

In this chapter, I address the active learning problem on attributed graphs. I argued that, the existing AL frameworks are ineffective in (1) they use a naive weighted combination of different AL strategies to select the nodes to label, and (2) they treat the learning of graph embedding and the AL query engine as two separate and independent processes, leading to limited AL performance gains. To fill the research gap, I propose a novel semi-supervised AL framework called *SEAL*, which fully exploits the representation power of GNNs and devises a novel AL query strategy for graphs. *SEAL* comprises two adversarial components; a graph embedding network is trained to embed both the labeled and unlabeled nodes into a common latent space, and to trick the discriminator to believe all nodes are from the labeled pool, while a discriminator network learns how to tell them apart using a semi-supervised structure with multiple outputs. The divergence score, produced by the discriminator, serves as the informativeness measure to select the most useful node to be labeled by the oracle. The two adversarial components form a closed loop to mutually and simultaneously reinforce each other towards enhancing the active learning performance. Extensive experiments and ablation studies prove that the *SEAL* renders remarkable performance gains compared with state-of-the-art AL methods on node classification tasks.

In conclusion, the proposed *SEAL* framework can effectively relieve the label sparsity problem by selectively constructing the train set with the most informative nodes. In the next chapter, I will investigate the graph pseudo labeling scenario to further mitigate the stress from inadequate label supervision, where it mainly focuses on fully exploiting the unlabeled data using the pseudo labeling technique.

Chapter 5

Informative Pseudo-Labeling for Graph Neural Networks with Few Labels[1]

In this chapter, I would introduce the proposed informative pseudo-labeling framework for learning GNNs with few labels. Being aware of the neglected problem of information redundancy and noisy pseudo-labeling in the existing related works, the proposed framework is designed to be able to robustly pseudo-label, instead of the most confident nodes, the most informative nodes while mitigating the negative effect caused by unreliable pseudo labels. This allows the model to intake more useful information and achieve better performances. Next, I will elaborate on the details of this informative pseudo-labeling framework.

5.1 Introduction

Graph neural networks have emerged as state-of-the-art models for undertaking semi-supervised node classification tasks on graphs [99], [63], [130], [135], [46]. However, under extreme cases where very few labels are available (e.g., only a handful of labeled nodes per class), popular GNN models, which rely on iteratively aggregating and transforming the neighboring node features to learn node embeddings, are ineffective in propagating the lim-

[1]Yayong Li, Jie Yin, Ling Chen. Informative Pseudo-Labeling for Graph Neural Networks with Few Labels. *Data Mining and Knowledge Discovery*, Accepted. [75]

ited label information to learn discriminative node embeddings within a shadow architecture. This tends to result in inferior classification performance. Recently, a central theme of latest studies has attempted to improve classification accuracy by designing deeper GNNs or new network architectures [101, 131]. However, the challenge of how to effectively learn GNNs with few labels is still under-explored.

Recently, pseudo-labeling, also called self-training, has been proposed as one prevalent semi-supervised method to explicitly tackle the label scarcity problem on graphs. Pseudo-labeling expands the label set by assigning a pseudo label to high-confidence unlabeled nodes, and iteratively re-trains the model with both given labels and pseudo labels. Li et al. [71] first proposed a self-trained GCN that chooses top-K high-confidence unlabeled nodes to enlarge the training set for model re-training. Sun et al. [121] pointed out a shallow GCN's ineffectivenss in propagating label information under few-label settings. A multi-stage approach was then proposed, which applies deep clustering techniques to assign pseudo labels to unlabeled nodes with high prediction confidence. Zhou et al. [159] proposed a dynamic self-training framework, which assigns a soft label confidence on the pseudo label loss to control their contribution to gradient update.

Despite offering promising results, the existing pseudo-labeling approaches on GNNs have not fully explored the power of self-training, due to two major limitations. First, these methods impose strict constraints that only unlabeled nodes with high prediction probabilities are selected for pseudo labeling. However, these selected nodes often convey similar information with given labels, causing *information redundancy* in the expanded label set. On the contrary, if unlabeled nodes with lower prediction probabilities are allowed to enlarge the label set, more pseudo label noise would be incurred to significantly degrade the classification accuracy. This creates a dilemma for pseudo-labeling strategies to achieve desirable performance improvements. Second, the existing methods all treat pseudo labels and genuine labels equally important. They are incorporated into the same loss function, such as the standard cross entropy loss, for node classification, neglecting their distinct contributions to the classification task. In the presence of unreliable or noisy pseudo labels, model performance might deteriorate during re-training.

Motivated by the above observations, we propose a novel informative pseudo-labeling framework called **InfoGNN** for semi-supervised node classification with few labels. Our aim is to fully harness the power of self-training by incorporating more pseudo labels, but at the same time, alleviate possible negative impact caused by noisy (i.e. incorrect) pseudo labels. To address **information redundancy**, we define node representativeness via neural estimation of mutual information between a node and its local context subgraph in the embedding space. This method offers two advantages: 1) It provides an informativeness measure to select unlabeled nodes for pseudo labeling, such that the added pseudo labels can bring in more information gain. 2) It implicitly encourages each node to approximate its own local neighborhood and depart away from other neighborhoods. The intuition behind is that an unlabeled node is considered informative when it can maximally reflect its local neighborhood. By integrating this informativeness measure with model prediction probabilities, our approach enables to selectively pseudo label nodes with maximum performance gains. To mitigate negative impact of **noisy pseudo labels**, we also propose a generalized cross entropy loss on pseudo labels to improve model robustness against noise. This loss allows us to maximize the pseudo-labeling capacity while minimizing the model collapsing risk. In addition, to cope with the potential **class-imbalance problem** caused by pseudo labeling under extremely few-label settings, we propose a class-balanced regularization that regularizes the number of pseudo labels to keep relative equilibrium in each class.

5.2 Problem Statement

Given an undirected graph $\mathscr{G} = \{\mathscr{V}, \mathscr{E}, X\}$ with the graph adjacent matrix $A \in \mathbb{R}^{n \times n}$. We assume that only a small fraction of nodes are labeled in the node set, where $\mathscr{L} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{|L|}$ denotes the set of labeled nodes, and $\mathscr{U}$ denotes the set of unlabeled nodes. We consider the semi-supervised node classification problem [63, 130] under a pseudo-labeling paradigm, which is formally defined as follows:

Problem 1 *Given an undirected graph $\mathscr{G} = \{\mathscr{V}, \mathscr{E}, X\}$ together with a small subset of labeled nodes $\mathscr{L} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{|L|}$, we aim to design a strategy* $1(\cdot)$ *for expanding the label set* *from unlabeled nodes, a method $\mathbb{Y}(\cdot)$ for generating reliable pseudo labels, and an exclusive loss function $\ell_U(\cdot)$ for pseudo labels, such that $1(\cdot)$, $\mathbb{Y}(\cdot)$ and $\ell_U(\cdot)$ can be combined together with the task-specific loss $\ell_L(\cdot)$ to maximize the classification performance of graph neural network $f_\theta(\cdot)$. This problem can be formally formulated as Eq.(5.1).*

$$\min_{\theta} \mathscr{J} = \sum_{\mathbf{x}_i \in \mathscr{L}} \ell_L(\mathbf{y}_i, f_\theta(\mathbf{x}_i)) + \sum_{\mathbf{x}_i \in \mathscr{U}} \ell_U(\mathbb{Y}(\mathbf{x}_i), f_\theta(\mathbf{x}_i)) \cdot 1(\mathbf{x}_i). \tag{5.1}$$

5.3 Methodology

5.3.1 Framework Overview

The primary aim of our work is to develop a robust pseudo-labeling framework for GNN training with few labels. As shown in Fig. 5.1, our proposed InfoGNN framework comprises of three key modules: 1) a GNN encoder; 2) informativeness estimator; 3) pseudo label selector. Taking a graph as input, the GNN encoder is first utilized to learn node embeddings as well as estimate class predictions and confidence scores. Then, the informativeness estimator closely follows to produce node informativeness scores for unlabeled nodes. Finally, according to informativeness and confidence scores, informative nodes are selected for pseudo labeling and model retraining. During GNN retraining phase, besides the standard cross entropy (SCE) loss applied on given labels, a generalized cross entropy loss (GCE) loss is applied on pseudo labels to improve model robustness against potentially noisy pseudo labels. A class-balanced regularization (CBR) is used to mitigate the potential class-imbalanced problem arising during pseudo labeling.

5.3.2 The GNN Encoder

The GNN encoder is the backbone for our framework. It mainly serves for generating node embeddings and giving class prediction probabilities that reflect model confidence in terms of predictions. Any GNN that focuses on node classification can be utilized here for embedding learning and classification. A GNN encoder generally learns node embeddings by recursively

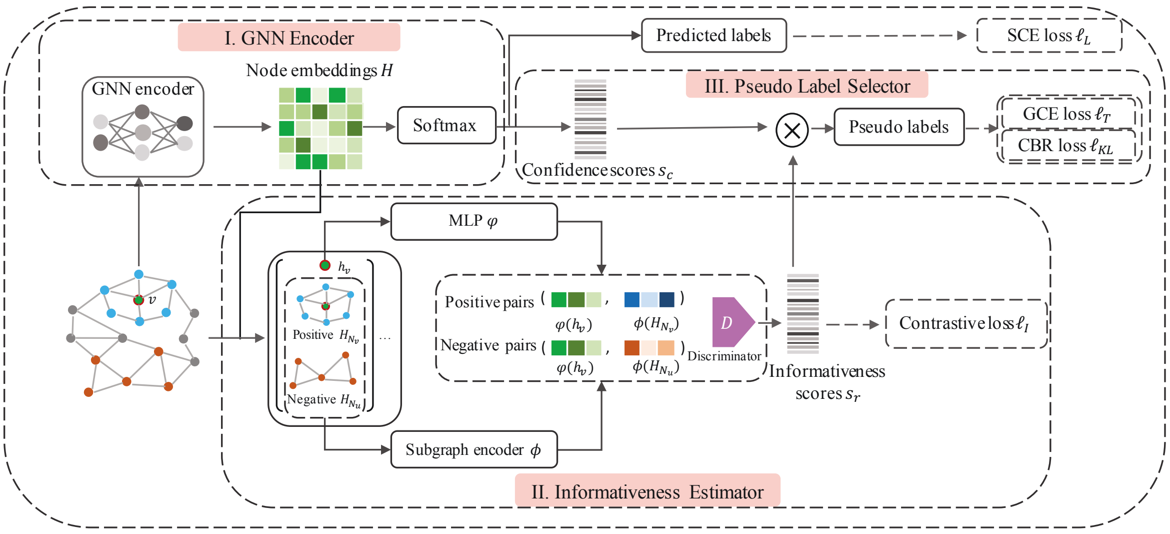


Fig. 5.1 Overview of the proposed InfoGNN framework, comprising of three main modules: GNN encoder, informativeness estimator and pseudo label selector. The GNN encoder is responsible for generating node embeddings and estimating confidence scores. Then, the informativeness estimator closely follows, in charge of measuring node informativeness and producing quantitative scores. Finally, according to both confidence and informativeness scores, informative nodes are selected for pseudo labeling and model retraining.

aggregating and transforming node features from topological neighborhoods. In our work, we utilize GCN [63] as our GNN encoder $f_\theta(\cdot)$. For $v \in \mathcal{V}$, the node embedding at *k-th* layer's prorogation can be obtained by:

$$h_v^k = \sigma(\sum_{v' \in \mathcal{N}_v} (\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2})_{v,v'} W^{k-1} h_{v'}^{k-1}), \tag{5.2}$$

$\sigma(\cdot)$ is the activation function, $\tilde{A} = A + I$ is the adjacency matrix of $\mathcal{G}$ with added self-connections. $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, and W^k is a layer-specific trainable weight matrix. We use the SCE loss to optimize GCN for node classification:

$$\ell_L(\mathbf{y}, f_\theta(\mathbf{x})) = -\sum_{i \in \mathcal{L}} \mathbf{y}_i log(f_\theta(\mathbf{x}_i)). \tag{5.3}$$

Finally, according to the class prediction probabilities, we can obtain the *confidence score* for each node v:

$$s_c(v) = \max_j f_\theta(\mathbf{x}_v)_j, \tag{5.4}$$

The confidence score $s_c(v)$ is utilized for node selection in combination with the representativeness score, which is detailed below.

5.3.3 Candidate Selection for Pseudo Labelling

Most existing pseudo-labeling methods typically select unlabeled nodes accounting for only model confidence or uncertainty [71, 92, 159]. These methods are in favor of selecting the nodes with high prediction probabilities, expecting to bring in less noisy pseudo labels for model re-training. However, these high-confidence nodes tend to carry redundant information with given labels, resulting in limited capacity to improve model performance. Therefore, besides model confidence, we propose to take node informativeness into account for node selection so as to maximally boost model performance. To this end, the key problem lies in how to measure node informativeness.

Informativeness Measure by MI Maximization We define the node informativeness as the representativeness of a node in relation to its contextual neighborhood. The intuition behind is that a node is informative when it could maximally represent its surrounding neighborhood while minimally reflect other arbitrary neighborhoods. Hence, the representativeness of a node can be measured by the mutual information between the node and its neighborhood with positive correlation. On account of this, we employ *MI maximization* techniques [8] to estimate the MI by measuring how much one node can represent its surrounding neighborhood and discriminate an arbitrary neighborhood. This thus provides a score to quantify the representativeness for each node. We achieve this by formulating it to be a *subgraph-based contrastive learning* task, which contrasts each node with its positive and negative context subgraphs.

Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, X\}$ with learned node embeddings H, for each node $v \in \mathcal{V}$, we define its local r-hop subgraph $\mathcal{N}_v$ as the *positive* sample, and an arbitrary r-hop subgraph $\mathcal{N}_u$ from node u as the *negative* sample. The mutual information $I^{\mathcal{G}}(v)$ between node v and its neighborhood can then be measured by a GAN-like divergence [94]:

$$I^{\mathcal{G}} \geqslant \hat{I}^{\mathcal{G}} = \max_{\omega} \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} [MI_{\omega}(h_v, H_{\mathcal{N}_v}) + MI_{\omega}(h_v, H_{\mathcal{N}_u})] \tag{5.5}$$

where h_v is node v's embedding generated from the GNN encoder, $H_{\mathcal{N}_v}$ and $H_{\mathcal{N}_u}$ are the embedding sets of subgraphs centered at node v and u. MI_{ω} is the trainable neural network parameterized by ω. $MI_{\omega}(h_v, H_{\mathcal{N}_v})$ indicates the affinity between positive pairs, while $MI_{\omega}(h_v, H_{\mathcal{N}_u})$ indicates the discrepancy between negative pairs. Our objective is to estimate $I^{\mathcal{G}}$ by maximizing $MI_{\omega}(h_v, H_{\mathcal{N}_v})$ while minimizing $MI_{\omega}(h_v, H_{\mathcal{N}_u})$, which is in essence a contrastive objective.

This contrastive objective is further achieved by employing a discriminator as illustrated in Figure 5.1. At each iteration, after obtaining the learned node embeddings H, both positive and negative subgraphs for each node are firstly sampled and paired. Then those nodes and their corresponding paired subgraphs are delivered to a discriminator network $MI_{\omega}(h_v, H_{\mathcal{N}_v})$, which is in charge of producing a representativeness score for each node by contrasting the node with its paired subgraphs. Formally, we specify $MI_{\omega}(h_v, H_{\mathcal{N}_v}) = \mathcal{D}(\varphi(h_v), \phi(H_{\mathcal{N}_v}))$. Here, $\phi(\cdot)$ is a *subgraph encoder* that aggregates embeddings of all nodes in the subgraph to generate a unified subgraph embedding. Here, I apply a *one-layer GCN* as the subgraph encoder, which can also be replaced by other alternative GNN models, such as GAT [130] or GraphSage [46]. The propagation principle of the GCN encoder on a r-hop subgraph is as follows:

$$\begin{aligned} \phi(H_{\mathcal{N}_v}) &= \sigma\Big(\sum_{v' \in \mathcal{N}_v} (D_r^{-1/2} A_r D_r^{-1/2})_{v,v'} h_{v'} W\Big), \\ A_r &= Bin(A * A_{r-1} + A_{r-1}) \end{aligned} \tag{5.6}$$

where W is the learnable parameters, and σ is the activation function. A is the original graph adjacent matrix, and $Bin(\cdot)$ is a binary function that guarantees $A_r(i,j) \in \{0,1\}$. D_r is the corresponding degree matrix of A_r. What we need to notice is that we use a r-hop adjacent matrix A_r, instead of the original adjacent matrix A for feature aggregation, and the aggregated embedding of the centre node in the subgraph will be the subgraph embedding.

$\varphi(\cdot)$ is an MLP encoder, which is tasked to transform the node embedding to the same space with the subgraph embedding $\phi(H_{\mathcal{N}_v})$. With regard to the discriminator $\mathscr{D}(\cdot)$, we implement it using a bilinear layer:

$$\mathscr{D}(\varphi(h_v), \phi(H_{\mathcal{N}_v})) = \sigma(\varphi(h_v)B\phi(H_{\mathcal{N}_v})^T) \tag{5.7}$$

where B is the learnable parameter. To enable the discriminator $\mathscr{D}(\varphi(h_v), \phi(H_{\mathcal{N}_v}))$ to measure the affinity between node v and its corresponding local subgraph $\mathcal{N}_v$, we minimize the binary cross entropy loss between positive and negative pairs, which is formulated as the *contrastive loss*:

$$\ell_I = -\frac{1}{|\mathscr{V}|}\sum_{v\in\mathscr{V}}[log\mathscr{D}(\varphi(h_v), \phi(H_{\mathcal{N}_v})) + log(1-\mathscr{D}(\varphi(h_v), \phi(H_{\mathcal{N}_u})))], \tag{5.8}$$

By minimizing ℓ_I, the discriminator could maximally distinguish a node from any arbitrary subgraphs that it does not belong to in the embedding space. This process is equivalent to maximizing their MI in the sense of Eq.(5.5).

Pseudo-Labeling. The discriminator $\mathscr{D}(\cdot)$ measures the affinity between each node and its local subgraph. We utilize this affinity to define the *informativeness score* for each node:

$$s_r(v) = \mathscr{D}(\varphi(h_v), \phi(H_{\mathcal{N}_v})), \tag{5.9}$$

where $s_r(v)$ indicates to what extent a node could reflect its neighborhood, and a higher score means that the node is more informativeness. Therefore, by considering both the informativeness score and model prediction confidence, we derive the selection criterion to construct the pseudo-label set $\mathscr{U}_p$:

$$\mathscr{U}_p = \{v \in \mathscr{U} \,|\, (s_r(v)+s_c(v))/2 > k, s.t. s_c(v) > k\}. \tag{5.10}$$

where $s_c(v)$ is the confidence score as in Eq.(5.4), and k is a hyperparameter whose value can be empirically determined (See Fig. 5.4(b) in Section 5.4.6). We then produce the pseudo labels for $\mathscr{U}_p$ utilizing the GNN encoder $f_\theta(\cdot)$:

$$\hat{\mathbf{y}}_v = \operatorname*{argmax}_j f_\theta(\mathbf{x}_v)_j; v \in \mathscr{U}_p \tag{5.11}$$

Where the pseudo label $\hat{\mathbf{y}}_v$ is actually the predicted label by the GNN encoder.

5.3.4 Mitigating Noisy Pseudo Labels

During the re-training phase, existing pseudo-labeling methods regard given labels and pseudo labels equally important, so an identical loss function, e.g., the SCE loss, is applied. However, with more nodes added in, it is inevitable to introduce unreliable or noisy (i.e., incorrect) pseudo labels. If the same SCE loss is still applied on unreliable pseudo labels, it would degrade model performance. This is because, the SCE loss implicitly weighs more on the difficult nodes whose predictions deviate away from the supervised labels during gradient update [128, 154]. This is beneficial for training with clean labels and ensures faster convergence. However, when there exist noisy pseudo labels in the label set, more emphasis would be put on noisy pseudo labels as they are harder to fit than correct ones. This would ultimately cause the model to overfit incorrect labels, thereby degrading model performance.

To address this issue, we propose to apply the negative Box-Cox transformation [15] to the loss function $\ell_U(\cdot)$ on pseudo label set $\mathscr{U}_p$, inspired by [154]. The transformed loss function is given as follows:

$$\begin{gathered}\ell_U(\hat{\mathbf{y}}_i, f_\theta(\mathbf{x}_i)) = \frac{1 - f_\theta(\mathbf{x}_i)_j^q}{q}, \mathbf{x}_i \in \mathscr{U}_p, \\ \hat{\mathbf{y}}_i = \operatorname*{argmax}_j f_\theta(\mathbf{x}_i)_j\end{gathered} \tag{5.12}$$

where $q \in (0, 1]$, $\hat{\mathbf{y}}_i$ is the pseudo label. To further elaborate how this loss impacts parameter update, we have its gradient as follows:

$$\frac{\partial \ell_U(\hat{\mathbf{y}}_i, f_\theta(\mathbf{x}_i))}{\partial \theta} = f_\theta(\mathbf{x}_i)_j^q\left(-\frac{1}{f_\theta(\mathbf{x}_i)_j}\nabla_\theta f_\theta(\mathbf{x}_i)_j\right), \tag{5.13}$$

where $f_\theta(\mathbf{x}_i)_j \in (0,1]$ for $\forall i$. Compared with the SCE loss, it actually weighs each gradient by an additional $f_\theta(\mathbf{x}_i)_j^q$, which reduces the gradient descending on those unreliable pseudo labels with lower prediction probabilities. Actually, $\ell_U(\hat{\mathbf{y}}_i, f_\theta(\mathbf{x}_i))$ can be regarded as the *generalization of the SCE loss and the unhinged loss*. It is equivalent to SCE when q approaches zero, and becomes the unhinged loss when q is equal to *one*. Thus, this loss allows the network to collect more additional information from a larger amount of pseudo labels while alleviating their potential negative effect.

In practice, we apply a truncated version of $\ell_U(\cdot)$ to filter out potential impact from unlabeled nodes with low prediction probabilities, given by:

$$\ell_T(\hat{\mathbf{y}}_i, f_\theta(\mathbf{x}_i)) = \begin{cases} \ell_U(k), & f_\theta(\mathbf{x}_i)_j \leq k \\ \ell_U(\hat{\mathbf{y}}_i, f_\theta(\mathbf{x}_i)), & f_\theta(\mathbf{x}_i)_j > k \end{cases} \tag{5.14}$$

where $k \in (0,1)$, and $\ell_U(k) = (1-k^q)/q$. Formally, the truncated loss version is derived as:

$$\ell_T(\hat{\mathbf{y}}, f_\theta(\mathbf{x})) = \sum_{i \in \mathcal{U}} \lambda_i \ell_U(\hat{\mathbf{y}}_i, f_\theta(\mathbf{x}_i)) + (1-\lambda_i)\ell_U(k), \tag{5.15}$$

where $\lambda_i = 1$ if $i \in \mathcal{U}_p$, otherwise $\lambda_i = 0$. Intuitively, when the prediction probability of one node is lower than k, the corresponding truncated loss would be a constant. As the gradient of a constant loss is zero, this node would have no contribution to gradient update, thus eliminating negative effect of pseudo labels with low confidence.

5.3.5 Class-balanced Regularization

Under extreme cases where only very few labels are available for training, severe class-imbalance problem would occur during pseudo labeling. That means, one or two particular classes might dominate the whole pseudo label set, thus conversely impacting model retraining. To mitigate this, we propose to apply a Kullback–Leibler (KL) divergence between the pseudo label distribution and a default label distribution for class-balanced regularization:

$$\ell_{KL} = \sum_{j=1}^{c} p_j log \frac{p_j}{\overline{f(X)}_j}, \tag{5.16}$$

where p_j is the default probability of class j. Since the real label distribution is unknown, we apply the uniform distribution for this regularization. That is, we set the probability of each class as $p_j = 1/c$ in our situation. $\overline{f(X)}_j$ is the mean value of class prediction probability distribution over pseudo labels, which is calculated as follows:

$$\overline{f(X)}_j = \frac{1}{|\mathscr{U}_p|} \sum_{\mathbf{x}_i \in \mathscr{U}_p} f(\mathbf{x}_i)_j \tag{5.17}$$

It is worth noting that, under the uniform distribution assumption, we do not attempt to approximate the real label distribution, which is unknown a priori during training. Instead, we expect to regularize the class distribution in the pseudo label set to be more uniformly distributed, preventing only one or two classes dominating the selected pseudo labels. Accordingly, hyperparamter β is employed to control the impact of ℓ_{KL} as in Eq.(5.19). More empirical analysis on the impact of class-balanced regularization will be provided in Section 5.4.6

5.3.6 Model Training and Computational Complexity

Our proposed InfoGNN framework is given by Algorithm 2, which consists of one pre-training phase and one formal training phase. The pre-training phase (Step 2-4) is used to train a parameterized GNN with given labels. Accordingly, network parameters are updated by:

$$\ell_{pre} = \ell_L + \alpha \ell_I \tag{5.18}$$

At the beginning of the formal training phase, the pre-trained GNN is applied to generate prediction probabilities and informativeness score for each node, which are then used to produce pseudo labels (Step 6-8). Finally, both given labels and pseudo labels are used to re-train the GNN by minimizing the following loss function (Step 9):

$$\ell = \ell_L + \ell_T + \alpha \ell_I + \beta \ell_{KL} \tag{5.19}$$

Algorithm 2 Training InfoGNN with few labels

```
Input: Graph $G = \{\mathscr{V}, \mathscr{E}, \mathbf{X}\}$, $\alpha$, $\beta$, $r, q$ and $k$
Output: Label predictions
 1: Initialize network parameters
 2: for $t = 0; t < epoches; t = t + 1$ do
 3:     if $t <$ start_epoch then
 4:         pre-train the network according to Eq.(5.18);
 5:     else
 6:         Generate node prediction probabilities $f_\theta(\mathbf{x}_i)$
 7:         Generate informativeness scores based on Eq.(5.9);
 8:         Construct pseudo label set based on Eq.(5.10);
 9:         Update network parameters based on Eq.(5.19);
10:     end if
11: end for
12: return Label predictions
```

In terms of computational complexity, by comparison with GNN models based on the SCE loss, InfoGNN incurs slightly extra computational overhead in its attempt to mitigate label noise. The is mainly due to the calculation of the contrastive loss ℓ_I with subgraph encoder. Since we utilize a one-layer GCN as subgraph encoder on a r-hop subgraph, its computational complexity is linear with the number of edges $\mathscr{O}(|E_r|)$, where E_r is the number of edges in the r-hop subgraph, i.e. $|E_r| = sum(A_r)$. This is reasonably acceptable.

5.4 Experiments

To validate the effectiveness of the proposed pseudo-labeling framework, we carry out extensive experiments on six real-world graph datasets to compare against state-of-the-art baselines. We also conduct ablation study and sensitivity analysis to better understand key ingredients of our approach.

5.4.1 Datasets

Our experiments use six benchmark graph datasets in three different domains: 1) **Citation networks:** Cora, Citeseer[63] and Dblp[14]; 2) **Webpage networks:** Wikics[89]; 3) **Coauther networks**: Coauther-CS and Coauther-Phy[113]. Detailed dataset statistics are listed in Table 5.1 below.

Table 5.1 Details of Five Benchmark Datasets

Dataset	Nodes	Edges	Classes	Features
Citeseer	3327	4732	6	3703
Cora	2708	5429	7	1433
Dblp	17716	105734	4	1639
Wikics	11701	216123	10	300
Coauthor_CS	18333	81894	15	6805
Coauthor_Phy	34493	247962	5	8415

5.4.2 Baselines

For comparison, we use 12 representative methods as our baselines. Since all methods are based on the original GCN, GCN [63] is selected as the benchmark. A total of 11 recently proposed methods on graphs are used as strong competitors, which can be categorized into two groups:

- **Pseudo-labeling methods**: M3S [121], Self-training [71], Co-training [71], Union [71], Intersection [71], and DSGCN [159];
- **Self-supervised methods**: Super-GCN [61], GMI [98], SSGCN-clu [144], SSGCN-comp [144], SSGCN-par [144].

We run all experiments for 10 times with different random seeds, and report the mean Micro-F1 scores over 10 times. Due to algorithmic design, the number of selected pseudo labels might vary among different methods. Thus, we report the best performance of each baseline method with its optimized hyperparameter.

Table 5.2 Details of hyperparameters

Given labels (per class)	α	β	k
$\{1,3,5\}$	1.0	1.0	0.55
$\{10,15,20,30,40,50\}$	0.2	0.2	0.55

5.4.3 Experimental setup

Model Specification. For fair comparison, all baselines are adapted to use a two-layer GCN with 16 units of hidden layer. The hyper-parameters are the same with the GCN in [63], with L2 regularization of $5*10^{-4}$, learning rate of 0.01, dropout rate of 0.5. As for subgraph encoder $\phi(\cdot)$, we utilize a one-layer GCN with c outputs, where c is the number of classes. Both positive and negative subgraphs share the same subgraph encoder. $\varphi(\cdot)$ is also a one-layer MLP with 16-dimension output. The discriminator $\mathscr{D}(\cdot)$ is a one-layer bilinear network with sigmoid activation.

Following the setup of self-training methods [159], we split each dataset into training and test sets. To be specific, we randomly choose $\{1,3,5,10,15,20,30,40,50\}$ nodes per class for training as different settings, and the remaining nodes are used for testing. The performance of different methods is assessed on the test set for comparison.

Hyperparameter Specification. We specify hyperparameters conforming to the following rules: Generally, a larger α and β value would be beneficial to model training when the given labels are scarce, while smaller α and β values are more likely to achieve better performance as the number of given labels increases. For k, we fix its value to 0.55 for all settings. The specification of the three hyperparameters are summarized in Table 5.2. In terms of q, we empirically find that our model has relatively lower sensitivity to q with the regularization of loss ℓ_I, so its value is fixed under most of the settings. Specifically, we set $q = 1.0$ when one label per class is given, and $q = 0.1$ for all other label rates. The best r value for subgraph embedding in loss ℓ_I depends on the edge density of the input graph. Particularly, we apply $r = 3$ for edge-sparse graphs (Cora, Citeseer, Dblp, Coauther_cs), $r = 2$ for Wikics, and $r = 1$ for Coauther_phy.

Implementation Details. When training InfoGNN, we first pre-train the network to generate reliable predictions using Eq.(5.18) for 200 epoches, and then proceed with formal training using the full loss function Eq.(5.19) for another 200 epoches. During formal training, in order to get a steady model, we allow the model to update pseudo-label set every 5 epoches using Eq.(5.10). When updating the pseudo-label set, we use the mean score of unlabeled nodes in its last 10 training epoches, rather than the current prediction and informativeness score. Our framework is implemented using Pytorch. All experiments are run on a machine powered by Intel(R) Xeon(R) Gold 6126 @ 2.60GHz CPU and 2 Nvidia Tesla V100 32GB Memory Cards with Cuda version 10.2.

Table 5.3 The Micro-F1 performance comparison with various given labels on Cora and Citeseer.

Method	Cora						Citeseer					
	1	3	5	10	15	20	1	3	5	10	15	20
GCN	0.418	0.616	0.685	0.742	0.784	0.797	0.381	0.504	0.569	0.602	0.660	0.682
Super-GCN	0.522	0.673	0.720	0.760	0.788	0.799	0.499●	0.610	0.665●	0.700●	0.706●	0.712
GMI	0.502	0.672	0.715	0.757	0.783	0.797	0.497	0.568	0.621	0.632	0.670	0.683
SSGCN-clu	0.407	0.684	0.739	0.776	0.797●	0.810●	0.267	0.388	0.507	0.616	0.634	0.647
SSGCN-comp	0.451	0.609	0.676	0.741	0.772	0.794	0.433	0.547	0.638	0.682	0.692	0.709
SSGCN-par	0.444	0.649	0.692	0.734	0.757	0.770	0.457	0.578	0.643	0.693	0.705	0.716●
Cotraining	0.533	0.661	0.689	0.741	0.764	0.774	0.383	0.469	0.563	0.601	0.640	0.649
Selftraining	0.399	0.608	0.693	0.761	0.789	0.793	0.324	0.463	0.526	0.647	0.683	0.685
Union	0.505	0.663	0.713	0.764	0.792	0.797	0.366	0.491	0.560	0.631	0.663	0.667
Intersection	0.408	0.596	0.674	0.736	0.770	0.775	0.337	0.497	0.582	0.671	0.694	0.699
M3S	0.439	0.651	0.688	0.754	0.763	0.789	0.307	0.515	0.635	0.674	0.683	0.695
DSGCN	0.596●	0.712●	0.745●	0.777●	0.792	0.795	0.463	0.613●	0.652	0.674	0.681	0.684
InfoGNN	**0.601**	**0.735**	**0.776**	**0.792**	**0.813**	**0.828**	**0.540**	**0.652**	**0.717**	**0.721**	**0.725**	**0.733**

5.4.4 Comparison with State-of-the-art Baselines

Table 5.3-5.5 reports the mean Micro-F1 scores of our method and all baselines with respect to various label rates. The best performer is highlighted by **bold**, and the second best performer is highlighted by underline on each setting. We also perform *significance test* between the Micro-F1 scores achieved by InfoGNN and the best baseline methods, where we use ●(○) to indicate that InfoGNN is significantly better (worse) than the compared baseline methods at 95% significance level.

On the whole, our proposed InfoGNN algorithm outperforms other baseline methods by a large margin over almost all the settings. Compared with GCN, we averagely achieve

Table 5.4 The Micro-F1 performance comparison with various given labels on Dblp and Wikics.

Method	Dblp						Wikics					
	1	3	5	10	15	20	1	3	5	10	15	20
GCN	0.472	0.583	0.627	0.652	0.688	0.718	0.384	0.550	0.638	0.682	0.712	0.720
Super-GCN	0.472	0.583	0.685	0.708	0.729	0.738	0.399	0.552	0.599	0.683	0.712	0.721
GMI	0.544	0.597	0.656	0.728	0.739	0.754	0.325	0.484	0.546	0.654	0.683	0.700
SSGCN-clu	0.369	0.528	0.649	0.692	0.721	0.744	0.335	0.579	0.627	0.694	0.714	0.725
SSGCN-comp	0.458	0.525	0.598	0.634	0.674	0.707	0.224	0.261	0.358	0.381	0.343	0.356
SSGCN-par	0.418	0.545	0.639	0.683	0.708	0.733	0.332	0.593	0.659	0.706	0.732	0.740
Cotraining	0.545	0.646	0.634	0.674	0.703	0.701	0.367	0.584	0.645	0.692	0.724	0.737
Selftraining	0.437	0.580	0.634	0.707	0.738	0.759	0.350	0.602	**0.655**○	0.701	0.725	0.738
Union	0.485	0.618	0.652	0.712	0.737	0.746	0.351	0.584	0.646	0.694	0.723	0.740●
Intersection	0.458	0.581	0.566	0.665	0.715	0.734	0.359	0.599	0.654	0.706●	0.726●	0.740●
M3S	0.547	0.635	0.672	0.733	0.749●	0.752	0.401	0.593	0.621	0.685	0.711	0.734
DSGCN	0.587●	**0.671**○	0.720●	0.738●	0.744	0.764●	0.414●	0.607●	0.635	0.705	0.716	0.728
InfoGNN	**0.596**	0.669	**0.746**	**0.765**	**0.773**	**0.787**	**0.460**	**0.610**	0.650	**0.723**	**0.740**	**0.742**

Table 5.5 The Micro-F1 performance comparison with various given labels on Coauthor_cs and Coauthor_phy. OOM indicates Out-Of-Memory on a 32GB GPU

Method	Coauthor_cs						Coauthor_phy					
	1	3	5	10	15	20	1	3	5	10	15	20
GCN	0.640	0.799	0.847	0.893	0.901	0.909	0.700	0.849	0.868	0.901	0.912	0.918
Super-GCN	0.668	0.841	0.869	0.895	0.897	0.897	0.688	0.848	0.891	0.908	0.923	0.923
GMI	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
SSGCN-clu	**0.770**○	**0.886**○	0.890●	0.905●	0.908	0.911	**0.889**○	0.923●	0.930●	0.935●	0.936●	0.936●
SSGCN-comp	0.711	0.858	0.888	0.904	0.907	0.909	0.798	0.892	0.904	0.927	0.921	0.928
SSGCN-par	0.737	0.860	0.881	0.898	0.901	0.903	0.824	0.915	0.919	0.925	0.931	0.931
Cotraining	0.643	0.745	0.810	0.849	0.864	0.885	0.758	0.842	0.850	0.898	0.891	0.917
Selftraining	0.592	0.770	0.828	0.873	0.892	0.895	0.744	0.865	0.890	0.908	0.914	0.921
Union	0.621	0.772	0.812	0.856	0.864	0.885	0.750	0.855	0.870	0.908	0.902	0.910
Intersection	0.650	0.775	0.851	0.887	0.893	0.898	0.612	0.763	0.854	0.901	0.904	0.926
M3S	0.648	0.818	0.879	0.897	0.909●	0.912●	0.828	0.868	0.895	0.914	0.922	0.930
DSGCN	0.743○	0.829	0.863	0.879	0.883	0.892	0.781	0.812	0.862	0.896	0.908	0.916
InfoGNN	0.683	0.865	**0.892**	**0.906**	**0.913**	**0.918**	0.842	**0.924**	**0.934**	**0.938**	**0.942**	**0.942**

$12.1\%, 9.2\%, 8.0\%, 6.3\%, 4.1\%, 3.5\%$ of performance improvement on the six datasets when $1, 3, 5, 10, 15, 20$ nodes per class are labeled, respectively. In particular, InfoGNN achieves better classification results with lower label rates. With less than 10 nodes per class, InfoGNN succeeds in achieving similar Micro-F1 scores as GCN uses 20 nodes per class over all datasets. As for self-supervised baselines, their performance is inconsistent across different datasets. For example, SSGCN-clu obtains advantageous results on Coauthor-cs/phy, but achieves undesirable results on other four datasets. SSGCN-Comp performs poorly on Wikics. This is due to the fact that specific pretext tasks designed by SSGCN do not generalize well on graphs with different properties.

Table 5.6 further compares the performance of all methods w.r.t. higher label rates, with 30, 40, and 50 given labels per class. As can be seen, as the number of given labels per class increases beyond 20, Micro-F1 scores of all methods continue to increase but with a declining growth rate. It is expected that when the supervision information is relatively sufficient, a further increase in given labels can lead to only limited performance gains. Moreover, with more abundant labels provided for training, the advantages of pseudo labeling methods gradually diminish as compared to the original GCN. However, our InfoGNN still outperform other baselines in most cases, especially on Cora and Citeseer. For example, when 50 labels per class are given on Cora, our InfoGNN achieves an Micro-F1 score of 85.3%, markedly outperforming the second best performer (SSGCN-clu) and GCN by 1.6% and 2.4%, respectively. This proves that our InfoGNN is able to effectively alleviate the information redundancy problem when label information is relatively sufficient.

5.4.5 Ablation Study

To further analyze how different components of the proposed method take effect, we conduct a series of ablation experiments. Due to space limit, we only report experimental results on the settings where 3 and 10 nodes are labeled per class. The ablations are designed as follows:

- **InfoGNN-I**: only ℓ_I is applied based on GCN, which is used to evaluate the role of the contrastive loss;
- **InfoGNN-IT**: both ℓ_I and ℓ_T are applied, which is utilized to evaluate the impact of the GCE loss by comparing with InfoGNN-I. Note that only prediction score is applied here for ℓ_T, i.e. $\mathcal{U}_p = \{v \in \mathcal{U} | f(\mathbf{x}_v)_j > k\}$;
- **InfoGNN-ITS**: on the basis of InfoGNN-IT, the informativeness score, i.e., Eq.(5.10), is also applied for ℓ_T, which is to test the efficacy of the informativeness score by comparing with InfoGNN-IT. The impact of the ℓ_{KL} loss can be revealed by comparing with InfoGNN.

Table 5.6 The Micro-F1 performance comparison over six datasets with {30,40,50} given labels per class. OOM indicates Out-Of-Memory on a 32GB GPU

Method	Cora			Citeseer			Dblp		
	30	40	50	30	40	50	30	40	50
GCN	0.816	0.825	0.829	0.695	0.708	0.716	0.743	0.753	0.770
Super-GCN	0.812	0.828	0.836	0.720	0.728	0.737	0.760	0.767	0.775
GMI	0.806	0.815	0.820	0.692	0.695	0.701	$\underline{0.784}$●	**0.794**○	$\underline{0.794}$●
SSGCN-clu	$\underline{0.822}$●	$\underline{0.829}$●	$\underline{0.837}$●	0.682	0.683	0.680	0.756	0.766	0.775
SSGCN-comp	0.804	0.819	0.830	0.718	0.729	$\underline{0.739}$●	0.744	0.752	0.761
SSGCN-par	0.784	0.791	0.798	$\underline{0.724}$●	$\underline{0.732}$●	0.738	0.751	0.762	0.769
Cotraining	0.804	0.820	0.823	0.675	0.684	0.697	0.716	0.726	0.736
Selftraining	0.807	0.821	0.818	0.696	0.706	0.710	0.777	0.775	0.782
Union	0.807	0.819	0.827	0.688	0.691	0.694	0.757	0.764	0.757
Intersection	0.800	0.818	0.821	0.705	0.712	0.716	0.745	0.765	0.769
M3S	0.792	0.807	0.815	0.713	0.716	0.721	0.765	0.769	0.774
DSGCN	0.798	0.809	0.816	0.684	0.684	0.685	0.784	0.786	0.786
InfoGNN	**0.835**	**0.848**	**0.853**	**0.735**	**0.737**	**0.742**	**0.789**	$\underline{0.792}$	**0.795**

Method	Wikics			Coauthor_cs			Coauthor_phy		
	30	40	50	30	40	50	30	40	50
GCN	$\underline{0.752}$●	$\underline{0.761}$●	0.764	0.901	0.900	0.903	0.924	0.932	0.933
Super-GCN	0.742	0.752	0.763	0.908	0.909	0.909	0.929	0.930	0.933
GMI	0.713	0.730	0.746	OOM	OOM	OOM	OOM	OOM	OOM
SSGCN-clu	0.738	0.745	0.747	0.914	0.915	0.915	$\underline{0.938}$●	$\underline{0.939}$●	$\underline{0.940}$●
SSGCN-comp	0.361	0.375	0.412	0.909	0.918	$\underline{0.922}$●	0.928	0.933	0.937
SSGCN-par	0.741	0.750	0.755	0.906	0.908	0.908	0.933	0.933	0.934
Cotraining	0.750	0.756	0.765	0.889	0.895	0.898	0.926	0.924	0.927
Selftraining	0.743	0.760	**0.768**○	0.901	0.901	0.904	0.932	0.932	0.932
Union	$\underline{0.752}$●	0.761	0.765	0.893	0.901	0.898	0.921	0.931	0.925
Intersection	0.748	0.765	0.767	0.896	0.898	0.905	0.927	0.927	0.932
M3S	0.745	0.755	0.763	$\underline{0.916}$●	$\underline{0.920}$●	$\underline{0.922}$●	0.935	0.937	0.940
DSGCN	0.751	0.759	0.763	0.893	0.896	0.897	0.916	0.920	0.922
InfoGNN	**0.754**	**0.764**	$\underline{0.766}$	**0.919**	**0.922**	**0.923**	**0.943**	**0.944**	**0.945**

The ablation results are reported in Table 5.7. under two settings where the number of given labels per class is 3 and 10, respectively. The constrastive loss ℓ_I seems to make similar contributions with both label rates, achieving an average improvement of 3.7% and 3.9% over GCN on the six datasets. On top of ℓ_I, the use of GCE leads to further performance improvements. Taking Wikics as an example, GCE further boosts the accuracy by 3.8% and 3.0% on the basis of ℓ_I, with 3 and 10 given labels per class, respectively. By comparing the performance of InfoGNN-IT and InfoGNN-ITS, we can find that the informativeness scores make distinct contributions under the two settings, where, for example, it achieves an improvement of 3.3% with 3 given labels per class in contrast to 0.5% with 10 given labels per class on Citeseer. This is because an increasing number of training labels counteracts

Table 5.7 The Micro-F1 performance comparisons with various ablation studies

Method	Cora		Citeseer		Dblp		Wikics		Coauthor_cs		Coauthor_phy	
	3	10	3	10	3	10	3	10	3	10	3	10
GCN	0.616	0.742	0.504	0.602	0.583	0.652	0.550	0.682	0.799	0.893	0.849	0.901
InfoGNN-I	0.681	0.764	0.583	0.694	0.598	0.739	0.549	0.695	0.824	0.887	0.885	0.928
InfoGNN-IT	0.697	0.790	0.590	0.723	0.618	0.768	0.587	0.725	0.827	0.892	0.899	0.937
InfoGNN-ITS	0.720	0.792	0.623	0.728	0.646	0.766	0.593	0.723	0.827	0.886	0.906	0.937
InfoGNN	0.735	0.792	0.652	0.721	0.669	0.765	0.610	0.723	0.865	0.906	0.924	0.938

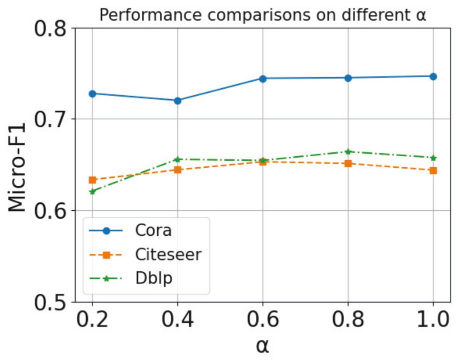
the effect of informativeness scoring. The similar phenomenon can also be observed on the contribution of ℓ_{KL}. It also plays a more significant role at a lower label rate, where imbalanced predictions are more likely to occur.

5.4.6 Sensitivity Analysis

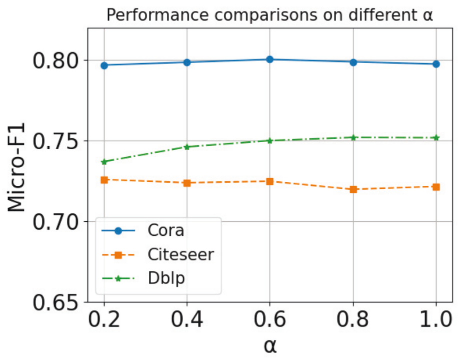
We also conduct experiments to test the impact of hyperparameters (α, β, q, k and r) on the performance of InfoGNN. We take turns to test the effect of each hyperparameter while fixing the values of the rest. Due to space limit, we only report the results when 3 and 10 labels per class are given for training.

Hyperparameter α controls the contribution of the contrastive loss ℓ_I to the total loss. Its impact on model performance is shown in Fig. 5.2. With 3 given labels per class provided, we find that a larger α could lead to better performance before $\alpha = 0.6$. After that, the performance retains at a good level with very slight changes. With 10 labels per class provided, except on Dblp, the changes of α do not largely impact model performance on Cora and Citeseer. This indicates that, when label information is very limited, our model requires stronger structural regularization to help generate discriminative node embeddings. On the other hand, when label information is relatively sufficient, network training is dominated by supervised loss from given labels. Thus, ℓ_I mainly takes effects when given labels are scarce.

Fig. 5.3 shows performance comparisons on different values of β. A similar trend with α can be observed on both settings. With only 3 labels per class provided, the class-imbalance problem is more likely to occur during pseudo labeling. Thus, our model favors a larger β to regularize numbers of each pseudo label class to be relatively equivalent, as shown in

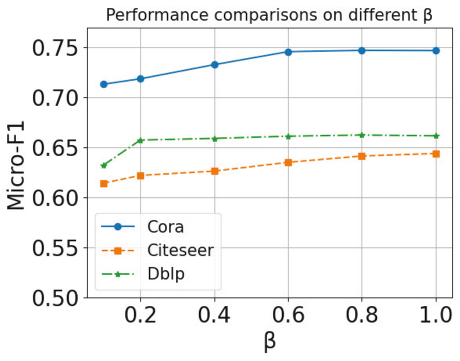


(a) α with 3 given labels per class

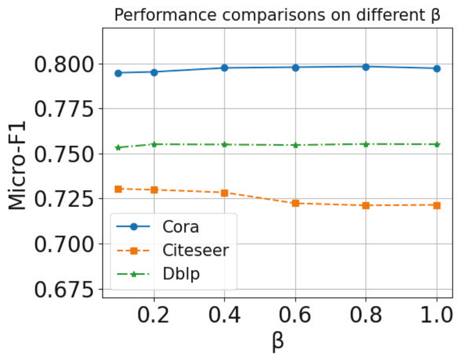


(b) α with 10 given labels per class

Fig. 5.2 Sensitivity analysis w.r.t. α on citation networks



(a) β with 3 given labels per class

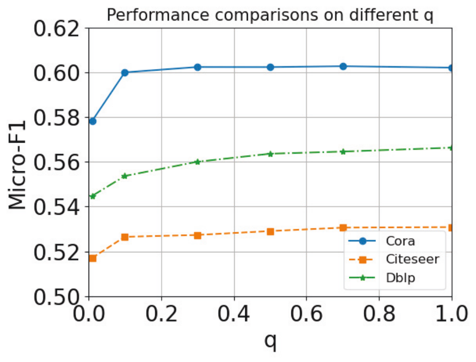


(b) β with 10 given labels per class

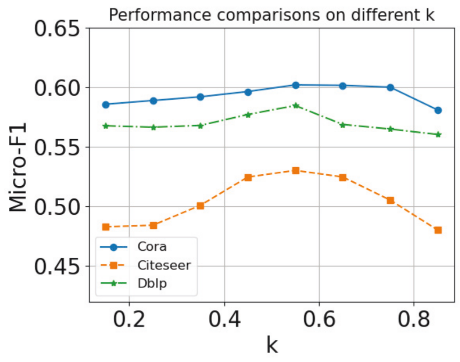
Fig. 5.3 Sensitivity analysis w.r.t. β on citation networks

Fig. 5.3(a). As β increases from 0.1 to 1.0, our model boosts its classification accuracy by around 3% on Citeseer and Cora. When 10 labels are given, as more label information can be exploited, the class-imbalance problem is less likely to arise. Hence, the change of β does not result in much impact on model performance.

Hyperparameter q is the generalization coefficient in ℓ_T. Fig. 5.4(a) illustrates model performance changes with an increase of q when one label per class is given. We can see that, as q rises, the performance of our method shows a gradual increase on the three datasets. This is because the severe lack of label information is more probable to incur noise in pseudo labels. A larger q is then able to decay the gradient update on unreliable samples that have lower prediction probabilities. This reduces the sensitiveness of our model towards incorrect



(a) q with 1 given labels per class



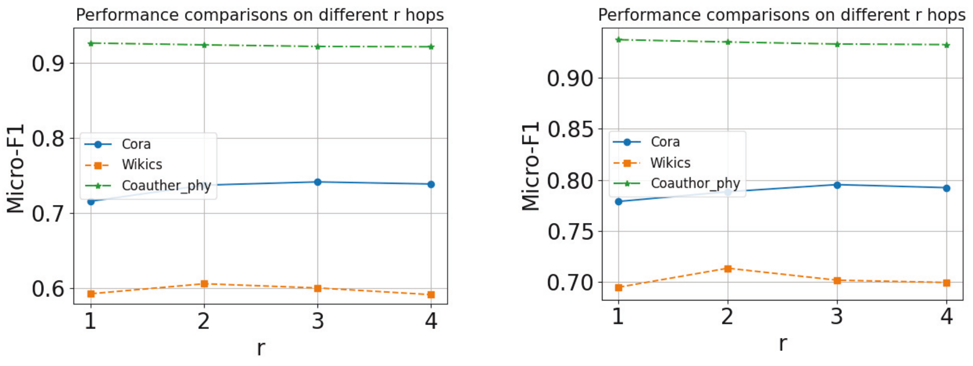
(b) k with 1 given labels per class

Fig. 5.4 Sensitivity analysis w.r.t. q & k on citation networks

pseudo labels, leading to better performance. On the other hand, *when descending q near zero, the GCE loss is approaching close to SCE,* and at the same time, the model has a significant performance degradation. This further proves the superiority of GCE over SCE loss when only few labels are given for training.

Hyperparameter k is the threshold for ℓ_T, which controls how many unlabeled nodes are selected for pseudo labeling. Fig. 5.4(b) depicts the performance changes by varying k with one given label per class. As we can see in this figure, a medium k achieves better accuracy, while either too small or too large k would undermine model performance.

Hyperparameter r indicates the number of hops for generating positive and negative subgraphs to calculate informativeness measures. The value of r controls the scale of sampled subgraphs. A larger r means the informativeness score is measured w.r.t. a larger-scale neighborhood. To investigate how the scale of sampled subgraphs affects model performance, we conduct experiments with varying values of r on three datasets (Cora, Wikics and Coauthor_phy) with diverse topology characteristics. As depicted in Fig.5.5, for edge-sparse graphs (*e.g.*, Cora), a larger r tends to result in better performance. For edge-dense graphs (*e.g.*, Coauthor_phy), a smaller r is more likely to exert better performance. For graphs with medium edge density (*e.g.*, Wikics), the best results are achieved with a medium r. When r exceeds 3 hops, model performance has a slight drop on all three datasets.



(a) r-hops with 3 given labels per class (b) r-hops with 10 given labels per class

Fig. 5.5 Sensitivity analysis w.r.t. number of hops r for sampled subgraphs

In summary, our method favors a smaller subgraph scale on dense graphs, but a larger scale on sparse graphs for sampling sufficient structural contexts in the local neighborhood.

5.5 Conclusion

In this chapter, I propose an informativeness augmented pseudo-labeling framework, called InfoGNN, to address semi-supervised node classification with few labels. I argue that all of the existing pseudo-labeling approaches on GNNs suffer from two major pitfalls: information redundancy and noisy pseudo labels. To address these issues, I propose to quantify node informativeness based on MI estimation maximization. Taking both informativeness and prediction confidence into account, more informative unlabeled nodes are selected for pseudo labeling. We then adapt a generalized cross entropy loss on pseudo labels to mitigate the adverse effect of unreliable pseudo labels. Furthermore, we apply a class-balanced regularization in response to the potential class-imbalance problem caused by pseudo labeling. Extensive experimental results and ablation studies verify the effectiveness of our proposed framework, and demonstrate its superior performance to state-of-the-art baseline models, especially under very few-label settings.

On the whole, this chapter offers an effective pseudo-labeling solution with information augmentation to cope with the label scarcity challenge in the semi-supervised node classification task. However, apart from the label sparsity problem, the GNN learning also suffers from the label noise problem, which can dramatically deteriorate the generalization ability of the model and result in poor predicting performance. Therefore, in the next chapter, I will explore how to learn a robust GNN to combat the label noise problem on graphs.

Chapter 6

Unified Robust Training for Graph Neural Networks against Label Noise[1]

In this chapter, I would investigate the issue of label-noise representation learning on graphs under the semi-supervised setting. To mitigate the label noise problem, a unified robust training framework has been proposed based on the label aggregation method. It can perform sample reweighting and label correction simultaneously, which can significantly reduce model sensitivity towards incorrect labels. Next, I will introduce details of the proposed framework.

6.1 Introduction

Recently, graph neural networks (GNNs) have been proposed to learn node embeddings and achieved state-of-the-art performance on the node classification task. The core of GNNs is to learn neural network primitives that generate node representations by passing, transforming, and aggregating node features from local neighborhoods [36]. As such, nearby nodes would have similar node representations [130]. By generalizing convolutional neural networks to graph data, graph convolutional networks (GCNs) [63] define the convolution operation via a

[1]Yayong Li, Jie Yin, Ling Chen. Unified Robust Training for Graph Neural Networks Against Label Noise. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 528-540. Springer, 2021. [74]

neighborhood aggregation function in the Fourier domain. The convolution of GCNs is a special form of Laplacian smoothing on graphs [71], which mixes the features of a node and its nearby neighbors. However, this smoothing operation can be disrupted when the training data is corrupted with label noise. As the training proceeds, GCNs would completely fit noisy labels, resulting in degraded performance and poor generalization. Hence, one key challenge is how to improve the robustness of GNNs against label noise, which is largely under explored yet.

Despite the lack of research attention on learning GNNs with noisy labels, it has been extensively studied on tasks with independent and identically distributed (IID) data in the domain of computer vision and natural language processing, such as the image classification [56, 60, 97, 127] and text classification task [5, 58, 85]. Label noise naturally stems from inter-observer variability, human annotator's error, and errors in crowdsourced annotations [60]. Existing methods attempt to correct the loss function by directly estimating a noise transition matrix [97, 127], or by adding extra layers to model the noise transition matrix [37, 58, 120]. However, it is difficult to accurately estimate the noise transition matrix particularly with a large number of classes. Alternative methods such as MentorNet [56] and Co-teaching [47] seek to separate clean samples from noisy samples, and use only the most likely clean samples to update model training. Other methods [4, 102] reweight each sample in the gradient update of the loss function, according to model's predicted probabilities. However, they require a large number of labeled samples or an extra clean set for training. Otherwise, reweighting would be unreliable and result in poor performance.

The aforementioned learning techniques, however, cannot be directly applied to tackle label noise on graphs. This is attributed to two significant challenges. (1) **Label sparsity**: graphs with inter-connected nodes are arguably harder to label than individual images. Very often, graphs are sparsely labeled, with only a small set of labeled nodes provided for training. Hence, we cannot simply drop "bad nodes" with corrupted labels like previous methods using "small-loss trick" [47, 56]. (2) **Label dependency**: graph nodes exhibit strong label dependency, so nodes with high structural proximity (directly or indirectly connected) tend to have a similar label. This presses a strong need to fully exploit graph topology and sparse node labels when training a robust model against label noise.

To tackle these challenges, we propose a novel approach for robustly learning GNN models against noisy labels under semi-supervised settings. Our approach provides a unified robust training framework for graph neural networks (UnionNET) that performs sample reweighting and label correction simulatenously. The core idea is twofold: (1) leverage random walks to perform label aggregation among nodes with structural proximity. (2) estimate node-level class distribution to guide sample reweighting and label correction. Intuitively, noisy labels could cause disordered predictions around context nodes, thus its derived node class distribution could in turn reflect the reliability of given labels. This provides an effective way to assess the reliability of given labels, guided by which sample reweighting and label correction are expected to weaken unreliable supervision and encourage label smoothing around context nodes. We verify the effectiveness of our proposed approach through experiments and ablation studies on real-world networks, demonstrating its superiority over competitive baselines.

6.2 Problem Statement

In this work, we consider the semi-supervised node classification setting. Given an undirected graph $G = \{\mathscr{V}, \mathscr{E}, \mathbf{X}\}$ with only a small fraction of nodes being labeled, let $\mathscr{L} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{|L|}$ denote the set of labeled nodes, and $\mathscr{U}$ denotes the unlabeled set. Under the GNN learning framework, the aim is to learn a representation $\mathbf{h}_{\mathbf{x}_i}$ for each node v_i such that its class label can be correctly predicted by $f(\mathbf{h}_{\mathbf{x}_i})$. For node classification, the standard cross entropy loss is used as the objective function:

$$\mathscr{J}(f(\mathbf{h}_{\mathbf{x}}), \mathbf{y}) = -\sum_{i \in \mathscr{L}} \sum_{j \in m} \mathbf{y}_{ij} \log(f(\mathbf{h}_{\mathbf{x}_i})_j). \tag{6.1}$$

where m denotes the number of the classes.

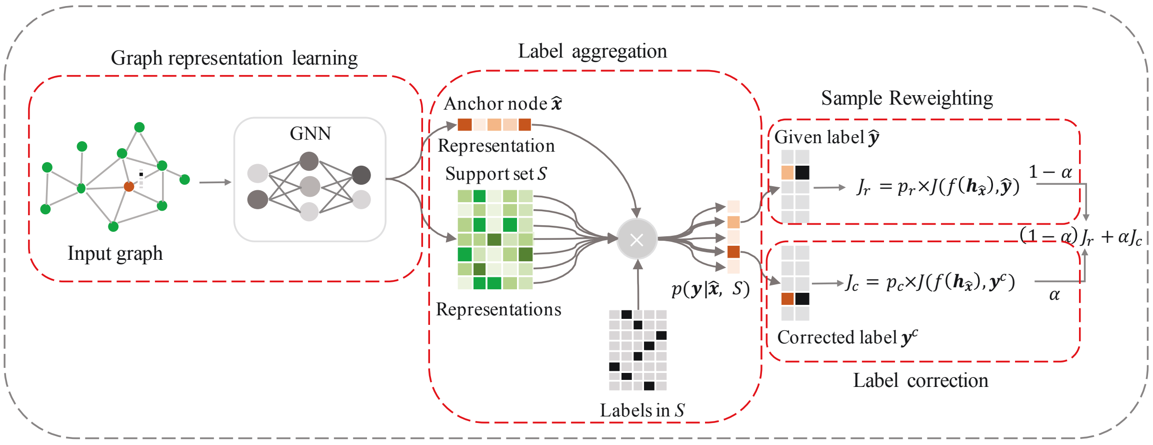

Graph representation learning
Input graph
GNN
Label aggregation
Anchor node $\hat{\boldsymbol{x}}$
Representation
Support set S
Representations
Labels in S
$p(\boldsymbol{y}|\hat{\boldsymbol{x}}, S)$
Sample Reweighting
Given label $\hat{\boldsymbol{y}}$
$J_r = p_r \times J(f(\boldsymbol{h}_{\hat{x}}), \hat{\boldsymbol{y}})$
$1-\alpha$
$(1-\alpha)J_r + \alpha J_c$
$J_c = p_c \times J(f(\boldsymbol{h}_{\hat{x}}), \boldsymbol{y}^c)$
α
Corrected label $\boldsymbol{y}^c$
Label correction

Fig. 6.1 Overview of the UnionNET Framework. The key idea is to infer the reliability of the given labels through estimating node-level class probability distributions via label aggregation. Based on this, the corresponding label weights and corrected labels are obtained to update model parameters during training.

However, when class labels in $\mathscr{L}$ are corrupted with label noise, the standard cross entropy would cause the GNN training to overfit incorrect labels, and in turn lead to degraded classification performance. Therefore, in our work, we aim to train a robust GNN model that is less sensitive to label noise.

Formally, given a small set of noisy labeled nodes $\mathscr{L}$, we aim to: (1) learn node representations $\mathbf{h}$ for all nodes $\mathscr{V}$, and (2) learn a model $f(\mathbf{h})$ to predict the labels of unlabeled nodes in $\mathscr{U}$ with maximum classification performance.

6.3 The UnionNET Learning Framework

To effectively tackle label noise on graphs, one desirable solution should consider the following key aspects. First, since only a small set of labeled nodes are available for training, we cannot simply drop “bad nodes” using “small-loss trick” [47, 56]. Second, graph nodes that share similar structural context exhibit label dependency. Thus, we propose a unified framework, UnionNET, for robustly training a GNN and performing label correction, as shown in Fig 6.1.

Taking a given graph as input, a GNN is first applied to learn node representations and generate the predicted label for each node. Then, label information is aggregated to estimate a class probability distribution per node. This aggregation is operated on a support set constructed by collecting context nodes with high structural proximity. According to node-level class probability distributions, our algorithm generates label weights and corrected labels for each labeled node. Those corrected labels generated from the support set could potentially provide extra “correct” supervision. Taken together, both given labels reweighted by label weights and corrected labels are used to update model parameters.

6.3.1 Label Aggregation

On graphs, it is well studied that nodes with high structural proximity tend to have the same labels [81, 160]. The supervision from noisy labels however disrupt such label smoothness around context nodes. Nevertheless, their smoothness degree could provide a reference to assess the reliability of given labels. Hence, we design a label aggregator that aggregates label information for each labeled node from its context nodes to estimate its class probability distribution. Specifically, for each labeled node $\hat{\mathbf{x}} \in \mathscr{L}$, called *anchor node*, we construct a *support set* of size k by performing random walks to collect context nodes with higher-order proximity. The generated support set is denoted as $S = \{(\mathbf{x}_i, \mathbf{y}_i)|\hat{\mathbf{x}}\}^k$, where $\mathbf{x}_i$ is the supportive node in S and $\mathbf{y}_i$ is one-hot encoding of $\mathbf{x}_i$'s class label. During the random walk, if node $\mathbf{x}_i \in \mathscr{L}$, the given label $\mathbf{y}_i$ is collected in S. Otherwise, the predicted label is collected.

Given anchor node $\hat{\mathbf{x}}$ and its support set S, we derive a node-level class probability distribution $P(\mathbf{y}|\hat{\mathbf{x}}, S)$ over m classes. It signifies the probabilities of the anchor node belonging to m classes in reference of its support set. Particularly, we specify a non-parametric attention mechanism given by,

$$P(\mathbf{y}|\hat{\mathbf{x}}, S) = \sum_{\mathbf{x}_i \in S} \mathscr{A}(\hat{\mathbf{x}}, \mathbf{x}_i)\mathbf{y}_i = \sum_{\mathbf{x}_i \in S} \frac{\exp(\mathbf{h}_{\mathbf{x}_i}^T \mathbf{h}_{\hat{\mathbf{x}}})}{\sum_{\mathbf{x}_j \in S} \exp(\mathbf{h}_{\mathbf{x}_j}^T \mathbf{h}_{\hat{\mathbf{x}}})} \mathbf{y}_i. \tag{6.2}$$

Here, the probability of the anchor node belonging to each class is calculated according to its proximity with nearby nodes in the support set. We define the proximity as the inner product in the embedding space, and apply softmax to measure the contribution made by each label in the support set to estimating the anchor node' class probability distribution. In the support set, if a node has a higher similarity with the anchor node (i.e., higher inner product), its label would contribute more to $P(\mathbf{y}|\hat{\mathbf{x}},S)$, and vice verse. This simple yet effective mechanism estimates a class probability distribution for each node, which is used to guide sample reweighting and label correction.

6.3.2 Sample Reweighting

For GNNs, the standard cross entropy loss implicitly puts more emphasis on the samples for which the predicted labels disagree with the provided labels during gradient update. This mechanism enables faster convergence and better fitting to the training data. However, if there exist corrupted labels in the training set, this implicit weighting scheme would conversely push the model to overfit noisy labels, leading to degraded performance [154]. To mitigate this, we devise a reweighting scheme for each node according to the reliability of its given label, so that the loss of reliable labels could contribute more during gradient update.

Specifically, we define the reweighting score of anchor node $\hat{\mathbf{x}}$ as:

$$p_r(\hat{\mathbf{y}}|\hat{\mathbf{x}},S) = \sum_{\mathbf{x}_i \in S, \mathbf{y}_i = \hat{\mathbf{y}}} \frac{\exp(\mathbf{h}_{\mathbf{x}_i}^T \mathbf{h}_{\hat{\mathbf{x}}})}{\sum_{\mathbf{x}_j \in S} \exp(\mathbf{h}_{\mathbf{x}_j}^T \mathbf{h}_{\hat{\mathbf{x}}})} \mathbf{y}_i. \tag{6.3}$$

The loss function for the labeled nodes is thus defined as:

$$\mathscr{J}_r = -\sum_{\hat{\mathbf{x}} \in \mathscr{L}} p_r(\hat{\mathbf{y}}|\hat{\mathbf{x}},S) \times \hat{\mathbf{y}} \log(f(\mathbf{h}_{\hat{\mathbf{x}}})), \tag{6.4}$$

where $p_r(\hat{\mathbf{y}}|\hat{\mathbf{x}},S)$ is the weight imposed on each labeled node $\hat{\mathbf{x}}$ according to the aggregated label information. If the given label $\hat{\mathbf{y}}$ is highly consistent with nearby labels, its gradient would be back-propagated as it is. Otherwise, it would be penalized by the weight during back-propagation.

6.3.3 Label Correction

The reweighting method reduces the sensitivity of the standard cross entropy to noisy labels, and boosts the robustness of the model. As labeled nodes are limited for training, we also augment the set of labeled nodes by correcting noisy labels. Accordingly, we define the label correction loss as

$$\mathscr{J}_c = -\sum_{\hat{\mathbf{x}} \in \mathscr{L}} p_c(\mathbf{y}^c|\hat{\mathbf{x}}, S) \times \mathbf{y}^c \log(f(\mathbf{h}_{\hat{\mathbf{x}}})), \tag{6.5}$$

$$p_c(\mathbf{y}^c|\hat{\mathbf{x}}, S) = \max_{\mathbf{y}_i} P(\mathbf{y}_i|\hat{\mathbf{x}}, S) = \max_{\mathbf{y}_i} \sum_{\mathbf{x}_i \in S} \frac{\exp(\mathbf{h}_{\mathbf{x}_i}^T \mathbf{h}_{\hat{\mathbf{x}}})}{\sum_{\mathbf{x}_j \in S} \exp(\mathbf{h}_{\mathbf{x}_j}^T \mathbf{h}_{\hat{\mathbf{x}}})} \mathbf{y}_i. \tag{6.6}$$

This provides additional supervision for $\hat{\mathbf{x}}$ with the corrected label $\mathbf{y}^c$, encouraging it to have the same label with the most consistent one in its support set. This approach aggregates labels from context nodes via a linear combination based on their similarity in the embedding space. It thus helps diminish the gradient update of corrupted labels, and boosts the supervision from consistent labels.

However, in the presence of extreme label noise, this approach would produce biased correction that deviates far away from its original prior distribution over the training data. This bias could exacerbate the overfitting problem caused by noisy labels. To overcome this, we employ a KL-divergence loss between the prior and predicted distributions to push them as close as possible [125]. It is given by:

$$\mathscr{J}_p = \sum_{j=1}^{m} p_j \log \frac{p_j}{\overline{f(\mathbf{h}_\mathbf{X})}_j}, \tag{6.7}$$

Where p_j is the prior probability of class j in $\mathscr{L}$, and $\overline{f(\mathbf{h}_\mathbf{X})}_j = \frac{1}{|L|} \sum_{\mathbf{x} \in \mathscr{L}} f(\mathbf{h}_\mathbf{x})_j$ is the mean value of predicted probability distribution on the training set.

6.3.4 Model Training

The training of UnionNET is given in Algorithm 3, which consists of the pre-training phase (Step 1-4) and the training phase (Step 6-11). The pre-training is employed to obtain a parameterized GNN. The pre-trained GNN then generates node representations **h**, which are used to compute sample weights and corrected labels. After that, model parameters are updated according to the loss function:

$$\mathscr{J}_f = (1-\alpha)\mathscr{J}_r + \alpha\mathscr{J}_c + \beta\mathscr{J}_p. \tag{6.8}$$

Compared with GNNs with the standard cross entropy loss, the training of UnionNET incurs an extra computational complexity of $\mathscr{O}(|L|ml)$ to estimate node-level class distributions, where $|L|$ is number of labeled nodes, m is number of classes, and l is number of nodes including context nodes in the support set.

Algorithm 3 Robust training for GNNs against label noise

```
Input: Graph $G = \{\mathscr{V}, \mathscr{E}, \mathbf{X}\}$, node sets $\mathscr{L}, \mathscr{U}$, $\alpha$, $\beta$
Output: label predictions
 1: Initialize network parameters
 2: for $t = 0; t < epoches; t = t+1$ do
 3:     if $t <$ start_epoch then
 4:         pre-train the network according to Eq.(6.1);
 5:     else
 6:         Generate node representations $\mathbf{h_x}$
 7:         Construct support set S for each node $\hat{\mathbf{x}} \in \mathscr{L}$
 8:         Aggregate labels to produce node-level class distribution $P(\mathbf{y}|\hat{\mathbf{x}}, S)$
 9:         Compute weight $p_r(\hat{\mathbf{y}}|\hat{\mathbf{x}}, S)$ using Eq.(6.3)
10:         Generate corrected label $\mathbf{y}^c$ and its weight $p_c(\mathbf{y}^c|\hat{\mathbf{x}}, S)$ using Eq.(6.6)
11:         Update parameters by descending gradient of Eq.(6.8)
12:     end if
13: end for
14: return Label predictions
```

6.4 Experiments

6.4.1 Datasets and Baselines.

Three benchmark datasets are used in our experiments: Cora, Citeseer, and Pubmed. We use the same data split as in [63], with 500 nodes for validation, 1000 nodes for testing, and the remaining for training. Of these training sets, only a small fraction of nodes are labeled (3.6% on Citeseer, 5.2% on Cora, 0.3% on Pubmed) and the rest of nodes are unlabeled. Details about the datasets can be found in [63].

As far as we are concerned, there has not yet been any method exclusively proposed to deal with the label noise problem on GNNs for semi-supervised node classification. We select three strong competing methods from image classification, and adapt them to work with GCN [63] under our setting as baselines.

- **Co-teaching** [47] trains two peer networks simultaneously, and each network selects the samples that have small losses to update the other network.
- **Decoupling** [84] also trains two networks simultaneously, but it updates the model parameters using only the samples with which the two networks disagree.
- **GCE** [154] utilizes a negative Box-Cox transformation as the loss function.

As a general robust training framework, UnionNET can be applied to any semi-supervised GNNs for node classification. Hereby, we instantiate UnionNET with two state-of-the-art GNNs, GCN [63] and GAT [130], denoted as **UnionNET-GCN** and **UnionNET-GAT**, respectively.

6.4.2 Experimental Setup.

Due to the fact that there are not yet benchmark graph datasets corrupted with noisy labels, we manually generate noisy labels on public datasets to evaluate our algorithm. We follow commonly used label noise generation methods in the domain of images [47, 56]. Given a noise rate r, we generate noisy labels over all classes according to a noise transition matrix $Q^{m\times m}$, where $Q_{ij} = p(\tilde{y} = j|y = i)$ is the probability of clean label $\mathbf{y}$ being flipped to noisy label $\tilde{\mathbf{y}}$. We consider two types of noise: 1). **Symmetric noise**: label i is corrupted to other labels with a uniform random probability, s.t. $Q_{ij} = Q_{ji}$; 2). **Pairflip noise**: mislabeling only occurs between similar classes. For instance, given $r = 0.4$ and $m = 3$, the two types of noise transition matrices are given by

$$Q^{\text{symmetric}} = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.2 & 0.6 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{bmatrix}; \; Q^{\text{pairflip}} = \begin{bmatrix} 0.6 & 0.4 & 0. \\ 0. & 0.6 & 0.4 \\ 0.4 & 0. & 0.6 \end{bmatrix}$$

Our experiments follow a transductive setting, where the noise transition matrix is only applied to $\mathscr{L}$, while both validation and test sets are kept clean. For UnionNET-GCN, we apply a two-layer GCN, which has 16 units of hidden layer. The hyper-parameters are set as L2 regularization of $5 * 10^{-4}$, learning rate of 0.01, dropout rate of 0.5. For UnionNET-GAT, we apply a two-layer GAT, with the first layer consisting of 8 attention heads, each computing 8 features. The learning rate is 0.005, dropout rate is 0.6, L2 regularization is $5 * 10^{-4}$.

We set the random walk length as 10 on Cora and Citeseer, and 4 on Pubmed, and the random walk is repeated for 10 times for each node to create the support set. We first pre-train the network, during which only the standard cross entropy are used, i.e. $\mathscr{J}_{pre} = \mathscr{J}(f(\mathbf{h_x}), \mathbf{y})$. After that, it proceeds to the formal training, which uses $\mathscr{J}_f$ in Eq.(6.8) as the loss function. And α and β are set as 0.5 and 1.0.

With regard to the hyperparameters of the baseline methods, their backbone GCN architectures remain the same with UnionNET-GCN for fair comparison. And then the grid search strategy is applied to determine their optimal hyperparameters over the validation set such that they can reach their best performance for evaluation.

6.4.3 Comparison with State-of-the-art Methods

Table 6.1 compares the node classification performance of all methods w.r.t. both the symmetric and asymmetric noise types under various noise rates. The best performer is highlighted by **bold**, and the second best performer is highlighted by underline on each setting. For GCN-based baselines, UnionNET-GCN generally outperforms all baselines by a large margin. Compared with GCN in case of symmetric noise type, UnionNET-GCN achieves an accuracy improvement of 3.4%, 6.3%, 13.1% and 7.1% under the noise rate

Table 6.1 Performance comparison (Micro-F1 score) on node classification

Dataset	Methods	Symmetric label noise				Asymmetric label noise			
		noise rate (%)							
		10	20	40	60	10	20	30	40
Cora	GCN	0.778	0.732	0.576	0.420	0.768	$\underline{0.696}$●	0.636	$\underline{0.517}$●
	Co-teaching	0.775	0.665	0.486	0.249	$\underline{0.773}$●	0.630	0.542	0.393
	Decoupling	0.738	0.708	0.564	$\underline{0.436}$●	0.743	0.683	0.574	0.518
	GCE	$\underline{0.794}$●	$\underline{0.741}$●	$\underline{0.621}$●	0.402	$\underline{0.773}$●	0.714	$\underline{0.652}$●	0.509
	UnionNET-GCN	**0.812**	**0.795**	**0.707**	**0.491**	**0.801**	**0.771**	**0.710**	**0.584**
	GAT	0.755	0.709	0.566	0.389	0.764	0.683	0.616	0.534
	UnionNET-GAT	0.797	0.784	0.692	0.546	0.774	0.745	0.660	0.540
Citeseer	GCN	0.670	0.634	0.480	$\underline{0.360}$●	0.667	0.624	0.531	$\underline{0.501}$●
	Co-teaching	0.673	0.541	0.379	0.273	0.677	0.583	0.472	0.418
	Decoupling	0.588	0.584	0.402	0.348	0.615	0.548	0.537	0.468
	GCE	$\underline{0.690}$●	$\underline{0.649}$●	$\underline{0.542}$●	0.358	$\underline{0.701}$●	$\underline{0.633}$●	$\underline{0.552}$●	0.498
	UnionNET-GCN	**0.701**	**0.673**	**0.567**	**0.401**	**0.706**	**0.667**	**0.587**	**0.521**
	GAT	0.649	0.604	0.475	0.338	0.651	0.599	0.551	0.480
	UnionNET-GAT	0.695	0.667	0.585	0.424	0.697	0.654	0.604	0.512
Pubmed	GCN	0.748	0.672	0.508	0.367	0.739	0.686	$\underline{0.618}$●	0.528
	Co-teaching	**0.769●**	0.660	0.478	0.345	$\underline{0.761}$●	0.634	0.576	0.472
	Decoupling	0.650	0.625	0.422	0.334	0.641	0.592	0.428	0.396
	GCE	$\underline{0.750}$	$\underline{0.699}$●	$\underline{0.561}$●	$\underline{0.393}$●	0.753	$\underline{0.696}$●	0.609	**0.567**○
	UnionNET-GCN	**0.769**	**0.725**	**0.588**	**0.409**	**0.776**	**0.719**	**0.649**	$\underline{0.556}$
	GAT	0.736	0.670	0.525	0.381	0.737	0.657	0.594	0.536
	UnionNET-GAT	0.751	0.726	0.570	0.361	0.758	0.702	0.626	0.552

of 10%, 20%, 40% and 60% on Cora, respectively. Similar improvements can be seen on Citeseer and Pubmed, where the smallest improvement is 2.1% on Pubmed with a noise rate of 10%, and the largest improvement is 8.7% on Citeseer with a noise rate of 40%. In case of asymmetric noise type, UnionNET-GCN has the similar performance. Quantitatively, UnionNET-GCN outperforms GCN by an average of 3.6%, 5.0%, 5.4%, 3.8% on the four noise rates on three datasets.

In most cases, GCE is the second best performer, but its advantage comes at the cost of worse converging capability, leading to sub-optimal performance. Co-teaching and Decoupling do not exhibit robustness towards noisy labels as reported in fully supervised image classification. Their performance drops are expected, as labeled data is further reduced when they prune the training data. This exacerbates the label scarcity problem in our semi-supervised setting.

In order to further verify the significance of the improvements, we have also conducted the paired t-test between UnionNET-GCN and the best baseline methods. We utilize the $\bullet(\circ)$ to indicate UnionNET-GCN is significantly better (worse) than the best baseline performer at the 95% significance level. And the t-test results exhibit consistent performance with the Micro-F1 score, which further proves the effectiveness of the proposed method.

In addition, in terms of the GAT backbone, UnionNET-GAT also surpasses GAT on the three datasets w.r.t. most noise rates. Similar to UnionNET-GCN, UnionNET-GAT generally exhibits greater superiority on higher noise rates. For example, in case of symmetric noise type, UnionNET-GAT outperforms GAT by an average of 3.4%, 6.4%, 9.4% and 7.4% at the four noise rates. Such performance gains validate the generality of UnionNET on improving robustness of different GNN models against noisy labels.

6.4.4 Ablation Study

We conduct ablation studies to evaluate the effectiveness of various components in UnionNET. Our ablation study is based on GCN, with two ablation versions: 1) **UnionNET-R** with only sample reweighting; 2) **UnionNET-RC** with sample reweighting and label correction. The ablation results are summarized in Table 6.2. When only reweighting is applied, UnionNET-R consistently exhibits advantages over GCN, though the advantageous margins vary over different noise rates and noise types. When it comes to UnionNET-RC, both smaple reweighting and label correction are applied, but, surprisingly, the performance becomes worse than UnionNET-R in some extreme cases with higher noise rates. Therefore, label correction does not guarantee performance gains, whose utility is exerted only with the regularization of the prior distribution loss.

6.4.5 Hyper-parameter Sensitivity

We further test the sensitivity of UnionNET-GCN w.r.t. the hyper-parameters (α, β) in Eq.(6.8) and the random walk length for the support set construction. We report the results

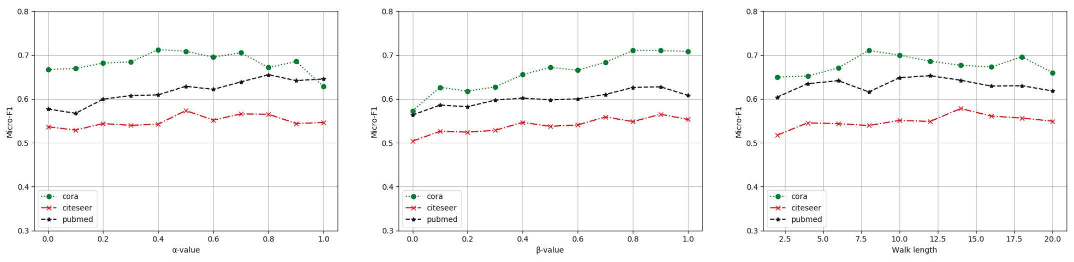
Table 6.2 Performance comparison of ablation experiments based on GCN

Dataset	Methods	Symmetric label noise				Asymmetric label noise			
		noise rate (%)							
		10	20	40	60	10	20	30	40
Cora	GCN	0.778	0.732	0.576	0.420	0.768	0.696	0.636	0.517
	UnionNET-R	0.785	0.770	0.659	0.480	0.796	0.709	0.646	0.521
	UnionNET-RC	0.788	0.759	0.626	0.339	0.783	0.703	0.601	0.516
	UnionNET-GCN	**0.812**	**0.795**	**0.707**	**0.491**	**0.801**	**0.771**	**0.710**	**0.584**
Citeseer	GCN	0.670	0.634	0.480	0.360	0.667	0.624	0.531	0.501
	UnionNET-R	0.692	0.643	0.507	0.363	0.699	0.627	0.547	0.484
	UnionNET-RC	0.657	0.645	0.495	0.330	0.660	0.642	0.511	0.431
	UnionNET-GCN	**0.701**	**0.673**	**0.567**	**0.401**	**0.706**	**0.667**	**0.587**	**0.521**
Pubmed	GCN	0.748	0.672	0.508	0.367	0.739	0.686	0.618	0.528
	UnionNET-R	0.766	0.710	0.573	**0.417**	0.759	0.705	0.624	0.560
	UnionNET-RC	**0.770**	0.695	0.573	0.362	0.757	0.650	0.608	0.497
	UnionNET-GCN	0.769	**0.725**	**0.588**	0.409	**0.776**	**0.719**	**0.649**	**0.556**

on the three datasets at 40% symmetric noise rate in Fig. 6.2. α controls the trade-off between sample reweighting and label correction. When α is zero, our method is only a reweighting method. When α reaches 1, our method evolves as a self-learning based label correction method, where given labels are replaced with predicted labels after the initial epoches. On Cora and Citeseer, our method achieves the best results at a medium α value. But on Pubmed, its performance improves as α increases, and reaches its best when $\alpha = 1.0$. This is possibly because Pubmed has stronger clustering property with only three classes, enabling the predicted labels to be more reliable for correction. The performance changes w.r.t. β exhibits similar trends on the three datasets, where our method gradually improves its performance as β increases. The random walk length determines the order of proximity the support set could cover. Either too small or too large of the random walk length would impair the reliability of the supportive nodes, and thus undermine performance improvements. Empirically, our method achieves its best at a medium range of random walk lengths.

6.5 Conclusion

In this chapter, a novel semi-supervised framework, UnionNET, is built up for learning with noisy labels on graphs. I argued that, existing methods on image classification fail to work on



(a) α (b) β (c) Walk Length

Fig. 6.2 Hyper-parameter sensitivity analysis on α, β, and the random walk length

graphs, as they often take a fully supervised approach, and requires extra clean supervision or explicit estimation of the noise transition matrix. Our approach provides a unified solution to robustly training a GNN model and performing label correction simultaneously. UnionNET is a general framework that can be instantiated with any state-of-the-art semi-supervised GNNs to improve model robustness, and it can be trained in an end-to-end manner. Experiments on three real-world datasets demonstrated that our method is effective in improving model robustness w.r.t. different label noise types and rates, and outperform competitive baselines.

In summary, this chapter provides an effective solution to combat the label noise problem by proposing a unified robust training framework for GNNs. In the next chapter, I will make a conclusion of this thesis and outline a few potential directions for my future research.

Chapter 7

Conclusion and Future Work

In this chapter, I would first make a summarization of the whole thesis, and then offers some directions for my further research.

7.1 Conclusion

In this thesis, I focus on the research of handling label sparse and noise problems on attributed graphs with the expectation of relieving GNNs' high dependence on large amounts of high-quality labels. The comprehensive literature review allows me to gain a deep insight into the limitations and challenges with regard to the label-associated problems on GNNs. Based on these knowledge, I develop my research from three perspectives, i.e. active learning, pseudo-labeling and label-noise representation learning on attributed graphs. Active learning has been proposed to tackle the label scarcity problem by constructing the train set with the most useful nodes. It is tasked to select the most informative nodes and query their labels for training within a given labeling budget, such that the selected nodes can maximize the classification performance of the model. To address this problem, a semi-supervised adversarial active learning (SEAL) framework is proposed on graphs in Chapter 4. It selects the nodes that share the least information with the existing labeled nodes. Inspired by the generative adversarial network, a GAN-like architecture is established with two adversarial components, i.e. a GNN encoder and a semi-supervised discriminator network, to measure the informativeness of the unlabeled candidates via the adversarial min-max game. The nodes having the largest divergence scores with the existing label set will be selected for querying labels. Experimental results show that the nodes selected by SEAL are able to well refine the decision boundary and obtain superior classification performance compared with baseline methods.

Then, I investigate a strict case of semi-supervised node classification where only very few labels are available for learning GNN classifiers. I argue that the existing methods suffer from a pair of contradictory problems, i.e. information redundancy and pseudo-label noise. In response to this challenge, an informativeness augmented pseudo-labeling method (InfoGNN) is proposed in Chapter 5, expecting to bring in more informative candidates while avoiding negative impacts from noisy pseudo-labels. This method consists of two main components, where an informativeness estimator is designed to measure node informativeness via the mutual information maximization technique, and a generalized cross entropy loss is applied to mitigate the problem of pseudo-label noise. Besides, a class-balance regularization is also applied to relieve the potential class imbalance problem during pseudo-labeling. Combining them together, InfoGNN can take the aspects of informativeness, reliability and class-imbalance into consideration at the same time. Extensive experiments over six real-world datasets demonstrate that the proposed method can effectively improve model performance by a large margin.

Finally, to combat the label noise problem, a robust training framework (UnionNET) is proposed for learning GNNs with noisy labels under the semi-supervised setting in Chapter 6. This approach provides a unified solution for performing sample reweighting and label correction simultaneously. This solution is based on the intuition that since noisy labels usually cause disordered predictions around neighboring nodes, the derived node class distribution by neighboring nodes can in turn reflect the reliability of given labels. Therefore, to estimate the label reliability, a label aggregation method is proposed to estimate node-level class probability distributions, which further provide guidance for the operation of sample reweighting and label correction. The extensive experimental results show that the proposed UnionNET achieves state-of-the-art performance over multiple datasets in the context of different noise types and rates. To the best of our knowledge, this is the first work that studies the label noise problem on node classification tasks under the semi-supervised setting.

7.2 Future Work

In the future, I will continue to focus on the label-associated problems on attributed graphs, and I am going to extend my research to more related topics in this area. Here, I outline several directions for my future research:

- **Class-imbalanced problem:** Current GNNs are primarily learned based on balanced-splitting label sets, ignoring the fact that the class distribution of given labels is usually inherently skewed in real-world datasets. Under the class-imbalanced setting, few classes (i.e. majority classes) usually dominate the whole labeled set in terms of their numbers, while the rest of classes (i.e. minority classes) only occupy a small portion of labeled nodes. This would induce GNNs to produce coarse representations in minority classes and compromise the classification performance [117]. Besides inheriting challenges existing in the IID data, the complex topological relationships among nodes in minority and majority classes would further aggravate this problem. However, how to address the class imbalanced problem on graphs is still under-explored.

  Confronting this challenge, the following two directions may provide me with possible solutions for handling the class-imbalanced problem: 1) the adversarial oversampling method; Instead of increasing the number of minority samples by random data augmentation technique, adversarial oversampling method can generate minority nodes that reside close to the decision boundary between different classes, thereby better refining the classifier; 2) the edge drop method; Since excessive links between minority and majority classes would weaken the characteristic of the minority nodes during feature aggregation process, selectively dropping redundant links connecting different classes would benefit to eliminating the negative effect of imbalance bias.

- **Non-homophily graph:** Most message passing mechanisms of GNNs are designed based on the homophily assumption that connected nodes are more likely to share the same labels. Thus the connected nodes with high structural proximity are usually pushed together during training, so as to learn similar representations for them and serve for the downstream node classification tasks. However, this learning mechanism is conflicting with non-homophily (heterophily) graphs, where nodes with different labels tend to be connected together. Under this situation, the feature aggregation operation among different classes would conversely make the learned representations more indistinguishable, and thus deteriorating the model performance.

  To address this challenge, I would make efforts from the two perspectives: 1) since the nodes with high structural and semantic similarities might be far away from each other in a heterophily graph, I would increase the reception field of GNNs, such that they can have the capability of capturing long-distance dependencies among nodes; 2) Instead of the uniform aggregation mechanism, I would try to design the discriminative aggregation mechanism, which can assign specific weights to local neighbors to demonstrate their distinct contributions to learning the class-differentiable representations for center nodes.

- **Graph incremental learning:** Currently, most of our research efforts have been on investigating static graphs, however, how to deal with dynamic graphs is still under explored. In real-world applications, graphs usually rapidly evolve with novel class nodes being incrementally added in different time periods. Under this situation, the GNN models are desired to be able to undertake a sequence of incremental learning sessions, where each session involves fulfilling the classification task on a set of novel (unseen) classes. Graph incremental learning is tasked to learn such a GNN model that can acquire novel knowledge from the new sessions while preserving the capability of recognizing all encountered classes in the previous sessions. Nevertheless, during the incremental learning session, the up-to-date GNN models that are trained on a set of new classes often suffer from the catastrophic forgetting problem, and have a poor performance on previously encountered classes.

In order to address this catastrophic forgetting problem, I would resort to knowledge distillation techniques, where the distillation loss allows the model to retain previously acquired knowledge by minimizing the divergence of representations between old and new models, thereby assisting the new model to maintain the ability of distinguishing old class. In addition, it would be also a promising solution to mitigate the forgetting problem by selectively storing and replaying a small number of samples that can well represent previous tasks, which can avoid the model to be completely overwritten by the novel classes, and thus preserving the previous knowledge.

References

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pages 21–29. PMLR, 2019.

[2] Charu C Aggarwal. *Data classification: algorithms and applications*. CRC press, 2014.

[3] Paul Almasan, José Suárez-Varela, Krzysztof Rusek, Pere Barlet-Ros, and Albert Cabellos-Aparicio. Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. *Computer Communications*, 196:184–194, 2022.

[4] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. *arXiv:1904.11238*, 2019.

[5] Ehsan Mohammady Ardehaly and Aron Culotta. Learning from noisy label proportions for classifying online social data. *Social Network Analysis and Mining*, 8(1):1–18, 2018.

[6] Martin Arjovsky and Leon Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*, 2017.

[7] James Atwood and Don Towsley. Diffusion-convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

[8] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International Conference on Machine Learning*, pages 531–540. PMLR, 2018.

[9] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(11), 2006.

[10] Dimitris Berberidis and Georgios B Giannakis. Active sampling for graph-aware classification. In *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 648–652. IEEE, 2017.

[11] Dimitris Berberidis and Georgios B Giannakis. Data-adaptive active sampling for efficient graph-cognizant classification. *IEEE Transactions on Signal Processing*, 66 (19):5167–5179, 2018.

[12] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. Node classification in social networks. In *Social network data analytics*, pages 115–148. Springer, 2011.

[13] Mustafa Bilgic, Lilyana Mihalkova, and Lise Getoor. Active learning for networked data. In *International Conference on Machine Learning*, pages 79–86, 2010.

[14] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018.

[15] George EP Box and David R Cox. An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(2):211–243, 1964.

[16] Joan Bruna, Wojciech Zaremba, Arthur D. Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2014.

[17] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. Active learning for graph embedding. *arXiv preprint arXiv:1705.05085*, 2017.

[18] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems*, 32, 2019.

[19] Ming-Wei Chang, Lev-Arie Ratinov, Nicholas Rizzolo, and Dan Roth. Learning and inference with constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1513–1518, 2008.

[20] Jie Chen, Tengfei Ma, and Cao Xiao. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*, 2018.

[21] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pages 1725–1735. PMLR, 2020.

[22] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271, 2020.

[23] Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan R Salakhutdinov. Good semi-supervised learning that requires a bad GAN. In *Advances in Neural Information Processing Systems*, pages 6510–6520, 2017.

[24] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[25] Yue Deng, KaWai Chen, Yilin Shen, and Hongxia Jin. Adversarial active learning for sequences labeling and generation. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 4012–4018, 2018.

[26] Kaize Ding, Jianling Wang, Jundong Li, Kai Shu, Chenghao Liu, and Huan Liu. Graph prototypical networks for few-shot learning on attributed networks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 295–304, 2020.

[27] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.

[28] Meng Fang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Active class discovery and learning for networked data. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 315–323, 2013.

[29] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33:22092–22103, 2020.

[30] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. *Advances in neural information processing systems*, 30, 2017.

[31] Yifan Fu, Xingquan Zhu, and Bin Li. A survey on instance selection for active learning. *Knowledge and Information Systems*, 35(2):249–283, 2013.

[32] Atsushi Fujii, Takenobu Tokunaga, Kentaro Inui, and Hozumi Tanaka. Selective sampling for example-based word sense disambiguation. *Computational Linguistics*, 24(4):573–597, 1998.

[33] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *International Conference on Machine Learning*, pages 2083–2092. PMLR, 2019.

[34] Li Gao, Hong Yang, Chuan Zhou, Jia Wu, Shirui Pan, and Yue Hu. Active discriminative network representation learning. In *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

[35] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

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

[37] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *International Conference on Learning Representations*, 2017.

[38] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[39] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[40] Quanquan Gu and Jiawei Han. Towards active learning on graphs: An error bound minimization approach. In *2012 IEEE International Conference on Data Mining*, pages 882–887. IEEE, 2012.

[41] Andrew Guillory and Jeff A Bilmes. Label selection on graphs. In *Advances in Neural Information Processing Systems*, pages 691–699, 2009.

[42] Andrew Guillory and Jeff A. Bilmes. Active semi-supervised learning using submodular functions. In *The 27th Conference on Uncertainty in Artificial Intelligence*, 2011.

[43] Sheng Guo, Weilin Huang, Haozhi Zhang, Chenfan Zhuang, Dengke Dong, Matthew R Scott, and Dinglong Huang. Curriculumnet: Weakly supervised learning from large-scale web images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 135–150, 2018.

[44] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 922–929, 2019.

[45] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. Variational graph recurrent neural networks. *Advances in neural information processing systems*, 32, 2019.

[46] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[47] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems*, pages 8527–8537, 2018.

[48] Jiangfan Han, Ping Luo, and Xiaogang Wang. Deep self-learning from noisy labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5138–5147, 2019.

[49] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pages 4116–4126. PMLR, 2020.

[50] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *ArXiv*, abs/1506.05163, 2015.

[51] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

[52] Richang Hong, Yuan He, Le Wu, Yong Ge, and Xindong Wu. Deep attributed network embedding by preserving structure and attribute information. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.

[53] Kexin Huang and Marinka Zitnik. Graph meta learning via local subgraphs. *Advances in Neural Information Processing Systems*, 33:5862–5874, 2020.

[54] Wei Huang, Yayong Li, weitao Du, Richard Xu, Jie Yin, Ling Chen, and Miao Zhang. Towards deepening graph neural networks: A GNTK-based optimization perspective. In *International Conference on Learning Representations*, 2022.

[55] Ming Ji and Jiawei Han. A variance minimization criterion to active learning on graphs. In *Artificial Intelligence and Statistics*, pages 556–564, 2012.

[56] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, pages 2304–2313. PMLR, 2018.

[57] Yu Jin and Joseph F JaJa. Learning graph-level representations with recurrent neural networks. *arXiv preprint arXiv:1805.07683*, 2018.

[58] Ishan Jindal, Daniel Pressel, Brian Lester, and Matthew Nokleby. An effective label noise model for dnn text classification. *Annual Conference of the North American*, page 3246–3256, 2019.

[59] Ajay J Joshi, Fatih Porikli, and Nikolaos P Papanikolopoulos. Scalable active learning for multiclass image classification. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2259–2273, 2012.

[60] Davood Karimi, Haoran Dou, S. Warfield, and Ali Gholipour. Deep learning with noisy labels: exploring techniques and remedies in medical image analysis. *Medical image analysis*, 65:101759, 2020.

[61] Dongkwan Kim and Alice H. Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations*, 2021.

[62] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[63] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[64] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2019.

[65] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, International Conference on Machine Learning*, volume 3, 2013.

[66] John Boaz Lee, Ryan Rossi, and Xiangnan Kong. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1666–1674, 2018.

[67] David D Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pages 148–156. Elsevier, 1994.

[68] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer, 1994.

[69] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9267–9276, 2019.

[70] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020.

[71] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[72] Wenyuan Li, Zichen Wang, Jiayun Li, Jennifer Polson, William Speier, and Corey Arnold. Semi-supervised learning based on generative adversarial network: a comparison between good GAN and bad GAN approach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshop*, 2019.

[73] Yayong Li, Jie Yin, and Ling Chen. Seal: Semisupervised adversarial active learning on attributed graphs. *IEEE Transactions on Neural Networks and Learning Systems*, 32(7):3136–3147, 2021.

[74] Yayong Li, Jie Yin, and Ling Chen. Unified robust training for graph neural networks against label noise. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 528–540. Springer, 2021.

[75] Yayong Li, Jie Yin, and Ling Chen. Informative pseudo-labeling for graph neural networks with few labels. *arXiv preprint arXiv:2201.07951*, 2022.

[76] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 556–559, 2003.

[77] Ralph Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

[78] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and Philip Yu. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2022. doi: 10.1109/TKDE.2022.3172903.

[79] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. Geniepath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4424–4431, 2019.

[80] Jun Long, Jianping Yin, Wentao Zhao, and En Zhu. Graph-based active learning based on label propagation. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 179–190. Springer, 2008.

[81] Qing Lu and Lise Getoor. Link-based classification. In *International Conference on Machine Learning*, pages 496–503, 2003.

[82] Yifei Ma, Roman Garnett, and Jeff Schneider. σ-optimality for active learning on gaussian random fields. In *Advances in Neural Information Processing Systems*, pages 2751–2759, 2013.

[83] David JC MacKay. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992.

[84] Eran Malach and Shai Shalev-Shwartz. Decoupling" when to update" from" how to update". In *Advances in Neural Information Processing Systems*, pages 960–970, 2017.

[85] Hassan H Malik and Vikas S Bhardwaj. Automatic training data cleaning for text classification. In *2011 IEEE 11th international conference on data mining workshops*, pages 442–449. IEEE, 2011.

[86] Andrew Kachites McCallumzy and Kamal Nigamy. Employing em and pool-based active learning for text classification. In *International Conference on Machine Learning*, pages 359–367. Citeseer, 1998.

[87] Luke McDowell and David Aha. Semi-supervised collective classification via hybrid label regularization. In *International Conference on Machine Learning*, 2012.

[88] Luke K McDowell and David W Aha. Labels or attributes? rethinking the neighbors for collective classification in sparsely-labeled networks. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 847–852, 2013.

[89] Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

[90] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[91] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017.

[92] Subhabrata Mukherjee and Ahmed Hassan Awadallah. Uncertainty-aware self-training for text classification with few labels. *arXiv preprint arXiv:2006.15315*, 2020.

[93] Jennifer Neville and David Jensen. Iterative classification in relational data. In *Proc. AAAI-2000 workshop on learning statistical models from relational data*, pages 13–20, 2000.

[94] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

[95] Hoang NT, Choong Jin, and Tsuyoshi Murata. Learning graph neural networks with noisy labels. In *The 2nd Learning from Limited Labeled Data Workshop*, 2019.

[96] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. In *International Joint Conference on Artificial Intelligence*, pages 2609–2615, 2018.

[97] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1944–1952, 2017.

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

[99] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 701–710. ACM, 2014.

[100] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1150–1160, 2020.

[101] Meng Qu, Yoshua Bengio, and Jian Tang. Gmnn: Graph markov neural networks. In *International Conference on Machine Learning*, pages 5241–5250. PMLR, 2019.

[102] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pages 4334–4343. PMLR, 2018.

[103] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

[104] Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05) - Volume 1*, 1:29–36, 2005.

[105] Nicholas Roy and Andrew McCallum. Toward optimal active learning through monte carlo estimation of error reduction. In *International Conference on Machine Learning*, pages 441–448, 2001.

[106] Luana Ruiz, Fernando Gama, and Alejandro Ribeiro. Gated graph recurrent neural networks. *IEEE Transactions on Signal Processing*, 68:6303–6318, 2020.

[107] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[108] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[109] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[110] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the 2008 conference on empirical methods in natural language processing*, pages 1070–1079. ACM, 2008.

[111] Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. In *Advances in Neural Information Processing Systems*, pages 1289–1296, 2008.

[112] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM, 1992.

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

[114] Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. M-walk: Learning to walk over graphs using monte carlo tree search. *Advances in Neural Information Processing Systems*, 31, 2018.

[115] Lixin Shi, Yuhang Zhao, and Jie Tang. Combining link and content for collective active learning. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1829–1832. ACM, 2010.

[116] Lixin Shi, Yuhang Zhao, and Jie Tang. Batch model active learning for networked data. *ACM Transactions on Intelligent Systems and Technology*, 3(2):1–25, 2012.

[117] Min Shi, Yufei Tang, Xingquan Zhu, David Wilson, and Jianxun Liu. Multi-class imbalanced graph convolutional network learning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 2020.

[118] Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5972–5981, 2019.

[119] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[120] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014.

[121] Ke Sun, Zhanxing Zhu, and Zhouchen Lin. Multi-stage self-supervised learning for graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[122] Shiliang Sun and David R Hardoon. Active learning with extremely sparse labeled examples. *Neurocomputing*, 73(16-18):2980–2988, 2010.

[123] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. Adversarial graph augmentation to improve graph contrastive learning. *Advances in Neural Information Processing Systems*, 34:15920–15933, 2021.

[124] Partha Pratim Talukdar and Koby Crammer. New regularized algorithms for transductive learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 442–457. Springer, 2009.

[125] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5552–5560, 2018.

[126] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.

[127] Arash Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *Advances in Neural Information Processing Systems*, pages 5596–5605, 2017.

[128] Brendan Van Rooyen, Aditya Menon, and Robert C Williamson. Learning with symmetric label noise: The importance of being unhinged. In *Advances in Neural Information Processing Systems*, pages 10–18, 2015.

[129] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations (Poster)*, 2019.

[130] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[131] Vikas Verma, Meng Qu, Alex Lamb, Yoshua Bengio, Juho Kannala, and Jian Tang. Graphmix: Regularized training of graph neural networks for semi-supervised learning. *arXiv preprint arXiv:1909.11715*, 2019.

[132] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

[133] Hongwei Wang and Jure Leskovec. Unifying graph convolutional neural networks and label propagation. *arXiv preprint arXiv:2002.06755*, 2020.

[134] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53 (3):1–34, 2020.

[135] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[136] Jun Wu, Jingrui He, and Jiejun Xu. Net: Degree-specific graph neural networks for node and graph classification. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 406–415, 2019.

[137] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2691–2699, 2015.

[138] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. Graph contextualized self-attention network for session-based recommendation. In *International Joint Conference on Artificial Intelligence*, volume 19, pages 3940–3946, 2019.

[139] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pages 5453–5462. PMLR, 2018.

[140] Nuo Xu, Pinghui Wang, Long Chen, Jing Tao, and Junzhou Zhao. Mr-gnn: Multi-resolution and dual graph neural network for predicting structured entity interactions. In *International Joint Conference on Artificial Intelligence*, pages 3968–3974, 2019.

[141] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7017–7025, 2019.

[142] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

[143] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33, 2020.

[144] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. When does self-supervision help graph convolutional networks? In *International Conference on Machine Learning*, pages 10871–10880. PMLR, 2020.

[145] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *International Joint Conference on Artificial Intelligence*, pages 3634–3640, 2018.

[146] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Wai-Hung Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In *International Conference on Machine Learning*, 2019.

[147] Victoria Zayats and Mari Ostendorf. Conversation modeling on reddit using a graph-structured lstm. *Transactions of the Association for Computational Linguistics*, 6: 121–132, 2018.

[148] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020.

[149] Kun Zhan and Chaoxi Niu. Mutual teaching for graph convolutional networks. *Future Generation Computer Systems*, 115:837–843, 2021.

[150] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Attributed network embedding via subspace discovery. *Data Mining and Knowledge Discovery*, 33(6): 1953–1980, 2019.

[151] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *The International Conference on Learning Representations (ICLR)*, 2018.

[152] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *The 34th Conference on Uncertainty in Artificial Intelligence*, 2018.

[153] Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Ustebay. Bayesian graph convolutional neural networks for semi-supervised classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5829–5836, 2019.

[154] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems*, pages 8778–8788, 2018.

[155] Wentao Zhao, Jun Long, En Zhu, and Yun Liu. A scalable algorithm for graph-based active learning. In *International Workshop on Frontiers in Algorithmics*, pages 311–322. Springer, 2008.

[156] Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16, 2003.

[157] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[158] Kaixiong Zhou, Xiao Huang, Yuening Li, Daochen Zha, Rui Chen, and Xia Hu. Towards deeper graph neural networks with differentiable group normalization. *Advances in neural information processing systems*, 33:4917–4928, 2020.

[159] Ziang Zhou, Shenzhong Zhang, and Zengfeng Huang. Dynamic self-training framework for graph convolutional networks. *arXiv preprint arXiv:1910.02684*, 2019.

[160] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning*, pages 912–919, 2003.

[161] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning workshop*, volume 3, 2003.

[162] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020.

[163] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, pages 2069–2080, 2021.