UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Graph Representation Learning for Graph-level Classification and Anomaly Detection

by Rongrong Ma

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Ling Chen

University of Technology Sydney
Faculty of Engineering and Information Technology

February 2025

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, *Rongrong Ma*, declare that this thesis is submitted in fulfilment of the requirements for the award of *the degree 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.

Production Note:
SIGNATURE: Signature removed prior to publication.

DATE: 13th February, 2025

Dedication

To my beloved parents, Yunhui Ma and Shuxia Cao.
To my husband Xiangyu Zhang.

Acknowledgments

I would like to thank my supervisor Professor Ling Chen for her insightful and valuable comments on my research works. In addition to help in research, Ling is also a great life mentor of me. She was always friendly and understanding and gave me large encouragement and many useful suggestions when I was down.

I greatly acknowledge Dr. Guansong Pang for guiding me during my candidature. Over the past four years, Guansong spent enormous time and effort on teaching me how to be an eligible researcher, including how to think independently and critically, do solid research and write academic papers. He gave me large freedom to independently explore interested research directions and think about possible solutions that we would discuss the feasibility and significance in each meeting. His comments on my ideas are always fruitful, critical and constructive. This mode enables me to think independently and discover mistake and shortage to avoid in my later research career. I will never forget his substantial suggestions on my works, which are significantly helpful for enhancing my research designs and paper writings. I cannot thank him more for all his helps.

I thank all my friends in UTS for their support and help in my life. I would also like to thank the UTS Graduate Research School for their quality administration service and financial support.

Last but not least, I thank my beloved mother and father for their eternal love, encouragement and support throughout my life. I would also like to thank my dear husband Xiangyu Zhang for being patient and extremely supportive during this tough period. He is always my spiritual backing and an enduring source of happiness. All pressure and unhappiness will fade away when he appears.

List of Publications

Papers that have been published:

1. **Rongrong Ma**, Guansong Pang, Ling Chen, and Anton van den Hengel. "Deep Graph-level Anomaly Detection by Glocal Knowledge Distillation". In: *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM-22)*, pp. 704-714.

2. **Rongrong Ma**, Guansong Pang, Ling Chen. "Imbalanced Graph Classification with Multi-scale Oversampling Graph Neural Networks". In: *2024 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-8.

3. **Rongrong Ma**, Guansong Pang, Ling Chen. "Harnessing Collective Structure Knowledge in Data Augmentation for Graph Neural Networks". In: *Neural Networks*, Volume 180, 2024, 106651, ISSN 0893-6080.

Abstract

Graph, due to its specific relation reservation ability, has become one of the most popular data storage modes in numerous applications, *e.g.*, bioinformatics, chemical and social networks. Graph representation learning is a crucial topic in graph data mining. Increasing attention is attracted by the graph neural networks (GNNs) as a result of their remarkable success in learning informative graph representations. Improving the expressiveness of GNN can help learn information-richer graph representations and thus is always a significant research goal. Besides, since the data collection is a costly process, how to learn expressive graph representations with limited supervision information is also important.

This thesis leverages three popular and practical graph-level classification tasks to formulate the circumstances with different amounts of supervision information and proposes three models to learn more expressive graph representations to solve them. In detail, our thesis makes the following contributions:

- We introduce a collective node and graph-level structural information harnessed model to improve the expressiveness of GNN. The proposed method significantly outperforms competing methods in graph classification task and is more generalized to out-of-distribution graphs, enabling it to be applied to real-world application in industry for higher accuracy and coping with unknown certainty in context. Besides, the proposed method is resource- and time-efficient, enabling the method to be applied to more platforms in industry.

- We propose a novel deep multi-scale oversampling framework and its instantiation to address the imbalanced graph classification problem. This is the *first work* that takes account of both within and between graph information to learn graph representations for imbalanced graph classification. The proposed method significantly outperforms its competing methods and offers a generic framework, in which different advanced imbalanced learning loss functions and GNN backbones can be easily plugged in and obtain significantly improved classification performance. This research improves the performance for data-insufficiency tasks and can save large amount of human resource in data collection.

- We formulate the graph-level anomaly detection problem as the task of detecting locally- or globally-anomalous graphs and empirically verify the presence of these two types of graph anomalies in real-world datasets. Then we introduce the *first*

approach and its instantiation specifically designed to effectively detect both types of anomalous graphs. The proposed method performs significantly better and can be trained much more sample-efficiently and with more robustness when compared with its advanced counterparts. This research would be of great importance to varying applications in industry, *e.g.*, omitting numerous experiments to identify toxic molecules from a set of chemical compounds, reducing human trials in recognizing drugs with severe side-effects or preventing the loss of millions of dollars by detecting fraud communities.

CONTENTS

Declaration i

Dedication iii

Acknowledgments v

List of Publications vii

Abstract ix

List of Tables xv

List of Figures xix

I Research Background and Foundation 1

1 Introduction 3

1.1 Research Motivation 3

1.2 Contributions 5

1.3 Organization 6

2 Preliminaries 7

2.1 Notations 7

2.2 Definitions and Problems on Graphs 9

2.3 Dataset and Performance Evaluation 10

2.3.1 Dataset 10

2.3.2 Performance Evaluation 11

3 Literature Review 15

3.1 Graph Representation Learning Methods 15

3.1.1 Traditional Methods 16
3.1.2 Graph Neural Networks 17
3.2 Graph Classification Methods 24
3.2.1 Balanced Graph Classification Methods 24
3.2.2 Imbalanced Graph Classification Methods 26
3.3 Graph Anomaly Detection 28
3.4 Summary 31

II Graph-level Problems and Solutions 33

4 Balanced Graph-level Classification with Collective Structure Knowledge Augmentation 35
4.1 Introduction 35
4.2 The Proposed CoS-GNN Model 37
4.2.1 Framework 37
4.2.2 Harnessing Collective Structure Knowledge for GNN 38
4.3 Theoretical Analysis 44
4.3.1 Expressive Power of CoS-GNN 44
4.3.2 Time Complexity Analysis 46
4.4 Experiments and Results 46
4.4.1 Competing Methods 46
4.4.2 Parameter Settings 47
4.4.3 Comparison to SOTA Models 47
4.4.4 Employ CoS-GNN as GNN Backbone 51
4.4.5 Enabling Out-of-distribution Generalization Tasks 54
4.4.6 Robustness w.r.t. Structure Contamination 54
4.4.7 Convergence Analysis 56
4.4.8 Ablation Study 56
4.5 Summary 61

5 Imbalanced Graph-level Classification with Multi-scale Oversampling 63
5.1 Introduction 63
5.2 The Proposed MOSGNN Model 66
5.2.1 Framework 66
5.2.2 Multi-scale Oversampling GNN 67

5.3 Theoretical Analysis . . . 73
5.4 Experiments and Results . . . 73
5.4.1 Competing Methods . . . 73
5.4.2 Parameter Settings . . . 74
5.4.3 Comparison to SOTA Models . . . 74
5.4.4 Enabling Different Loss Functions . . . 76
5.4.5 Enabling Different GNN Backbones . . . 77
5.4.6 Sample Efficiency . . . 78
5.4.7 Robustness w.r.t. Label Noise . . . 79
5.4.8 Ablation Study . . . 80
5.5 Summary . . . 82

6 Graph-level Anomaly Detection with Global and Local Knowledge Distillation 85
6.1 Introduction . . . 85
6.2 The Proposed GLocalKD Method . . . 88
6.2.1 Framework . . . 88
6.2.2 Joint Random Distillation of Graph and Node Representations . . . 90
6.3 Theoretical Analysis . . . 93
6.4 Experiments and Results . . . 94
6.4.1 Competing Methods . . . 94
6.4.2 Parameter Settings . . . 95
6.4.3 Comparison to SOTA Methods . . . 95
6.4.4 Sample Efficiency . . . 97
6.4.5 Robustness w.r.t. Anomaly Contamination . . . 97
6.4.6 Sensitivity Test . . . 98
6.4.7 Ablation Study . . . 99
6.5 Summary . . . 100

III Conclusions and Future Directions 103

7 Conclusion and Future Work 105
7.1 Conclusion . . . 105
7.2 Future Work . . . 106

A Appendix 109

A.1 The Influence of Subsample Size on LESINN 109

Bibliography **111**

List of Tables

Table **Page**

2.1 Commonly used notations and their descriptions. 8

2.2 The detailed information of 12 public datasets for balanced graph-level classification. The 'binary' in the 'Class' column denotes the dataset is for binary classification while 'multi' implies multi-class classification. The '#Graphs' is the total number of graphs in the dataset and the '#Nodes' means the average number of nodes in the dataset. The '✓' in the 'Attribute' column indicates the data contains attributed graphs, and otherwise they contain only plain graphs. 11

2.3 The imbalanced information of 16 public datasets for imbalanced graph-level classification. '#pos' and '#Graphs' denote the number of graph samples in the minority class and in the full dataset, respectively. 'Ratio' represents the ratio of the majority class size to the minority class. 12

2.4 The detailed information of 16 public datasets for graph-level anomaly detection. The '#Graphs' is the total number of graphs in the dataset and the '#Nodes' means the average number of nodes in the dataset. The '✓' in the 'Attribute' column indicates the data contains attributed graphs, and otherwise they contain only plain graphs. 13

3.1 A summary of several types of graph representation learning methods. 24

4.1 Accuracy (mean±std) of CoS-GNN and SOTA competing methods for graph classification on 12 real-world datasets. The best and second performance per dataset is boldfaced and underlined respectively. The following acronyms, PROTEINS_full (PROTS_full), IMDB-BINARY (I-BINARY), IMDB-MULTI (I-MULTI), REDDIT-BINARY (R-BINARY) and REDDIT-MULTI-5K (R-MULTI), are used. 'Rank' indicates the average performance ranking of a model across all datasets: a smaller rank value indicates a better overall performance. . . 48

4.2 AUPRC (mean±std) of CoS-GNN and SOTA competing methods for graph classification on 9 real-world binary classification datasets. The best and second performance per dataset is boldfaced and underlined respectively. The following acronyms, PROTEINS_full (PROTS_full), IMDB-BINARY (I-BINARY), IMDB-MULTI (I-MULTI), REDDIT-BINARY (R-BINARY) and REDDIT-MULTI-5K (R-MULTI), are used. 'Rank' indicates the average performance ranking of a model across all datasets: a smaller rank value indicates a better overall performance. 50

4.3 Results (mean±std) of CoS-GNN and corresponding vanilla GNN on OGB datasets – ogbg-molhiv and ogbg-molpcba. The best performance per dataset is boldfaced. 50

4.4 Training and inference time of augmentation methods in graph classification task. The following acronyms, PROTEINS_full (P_full), IMDB-BINARY (I-B), IMDB-MULTI (I-M), REDDIT-BINARY (R-B) and REDDIT-MULTI-5K (R-M), are used. All methods are with GIN as backbone. Each result is the time on the whole dataset. 51

4.5 Accuracy (mean±std) results of $\mathscr{G}$-mixup and Dummy using CoS-GCN as the GNN module, with $\mathscr{G}$-mixup and Dummy with GCN as baselines in graph classification. 'Different' denotes accuracy improvement (↑) or decrease (↓) brought by the replacement of CoS-GCN. Both of two methods suffer out of memory on REDDIT-BINARY and REDDIT-MULTI. 52

4.6 Accuracy (mean±std) results of GCN and CoS-GCN with MVPool as the readout operation. 'Different' denotes accuracy improvement (↑) or decrease (↓) brought by CoS-GCN compared to GCN. 53

4.7 Results of CoS-GNN with two baselines on three OOD datasets. G-X is short for the dataset name GOOD-X. 55

4.8 Results of the ablation study of the graph augmentation and the message passing mechanism in CoS-GCN in the graph classification task. 58

4.9 Efficiency of the augmented node features in graph classification. 59

4.10 Efficiency of the augmented graph features in graph classification. 60

5.1 F1 score (mean±std) of MOSGNN and five SOTA competing methods. #*pos* and #Graph denote the number of graph samples in the minority class and in the full dataset, respectively. 'Ratio' denotes the ratio of the majority class size to the minority class. 'Rank' indicates the average performance ranking of a model across all datasets: a smaller rank value indicates a better overall performance. The best performance per dataset is boldfaced. 75

5.2 Macro-F1 score (mean±std) of MOSGNN and five SOTA competing methods on 16 real-world imbalanced graph datasets. 'Rank' indicates the average performance ranking of a model across all datasets: a smaller rank value indicates a better overall performance. 75

5.3 AUPRC results (mean±std) of MOSGNN and five SOTA competing methods on 16 real-world imbalanced graph datasets. 'Rank' indicates the average performance ranking of a model across all datasets: a smaller rank value indicates a better overall performance. 76

5.4 F1 score (mean) using MOSGNN-enabled FocalLoss and LALoss, with the original FocalLoss and LALoss as baselines. 'Diff.' denotes the F1 score improvement (↑) or decrease (↓) of 'MOS-X' compared to the original 'X' loss. . . 77

5.5 Macro-F1 score (mean) using MOSGNN-enabled FocalLoss and LALoss, with the original FocalLoss and LALoss as baselines. 'Diff.' denotes the F1 score improvement (↑) or decrease (↓) of 'MOS-X' compared to the original 'X' loss. . 77

5.6 F1 score results (mean±std) of using GIN/GAT as backbones. The best performance is boldfaced. 'Baseline' denotes 'Oversampling'. 78

5.7 AUPRC results (mean±std) of using GIN/GAT as backbones. The best performance is boldfaced. 'Baseline' denotes 'Oversampling'. 79

5.8 F1 score (mean±std) of MOSGNN and its ablated variants. The best performance per dataset is boldfaced. 'Rank' indicates the average performance ranking of a model across all datasets: a smaller rank value indicates a better overall performance. 82

6.1 AUC results (mean±std) on 16 real-world graph datasets. The following acronyms, PROTEINS_full (PROTS_full), IMDB-BINARY (I-BINARY) and REDDIT-BINARY (R-BINARY), are used. The best performance is boldfaced. 96

6.2 Training time and test time on 3 datasets: REDDIT-BINARY, p53 and COLLAB (seconds on each epoch). 97

6.3 Detection of locally/globally-anomalous graphs. 101

A.1 The influence of subsample size on LESINN. ‘PROTEINS_full’ are shortened by ‘PROTS_full’. 110

List of Figures

Figure **Page**

4.1 1- and 2-WL tests fail to distinguish the two graphs as they obtain the same rooted subtree (node coloring). 36

4.2 The red and blue points represent the visualized graph representations of two classes in REDDIT-BINARY yielded by (b) CoS-GCN with augmented node-level structural features and (c) CoS-GCN with augmented structural features at both node and graph levels are more class-separable than those produced by (a) the original GCN. 37

4.3 A schematic depiction of our CoS-GNN. Our CoS-GNN first calculates the specific node- and graph-level structural features. Then a new message passing mechanism is devised to utilize the original node attributes and the augmented node structural features to compute the graph representation, which is further combined with the graph structural augmentations for down-stream tasks. 39

4.4 The strongly regular Rook's 4×4 graph (left) and Shrikhande graph (right) [14; 4]. The 3-WL/2-FWL test is not able to deem them as non-isomorphic. Rook's 4×4 graph possesses 4-cliques while the Shrikhande graph features 5-rings, which does not present in Rook's. 46

4.5 Accuracy performance of CoS-GCN and GCN w.r.t. different structural contamination rates. 56

4.6 Loss variation tendency of CoS-GCN on the training and validation dataset of REDDIT-BINARY. 57

5.1 Motivation of pairwise- and subgraph-scale oversampling. (a) Pairwise-scale classification. A minority graph wrongly classified based on graph-scale information can be correctly classified based on graph interactions; (b) Subgraph-scale classification. In some graphs, only their subgraphs are relevant to the classification; the other parts are noisy. Oversampling the whole graphs may lead to the inclusion of more noise. 64
5.2 Overview of the proposed framework. It augments and trains a GNN model with oversampled graph data at the subgraph, graph, and pairwise inter-graph levels, to capture diversified intra- and inter-graph information for the classification of minority graphs. To achieve this goal, two auxiliary objectives, *i.e.*, pairwise graph relation prediction and subgraph-based MIL, are combined with the standard graph classification objective to jointly optimize the GNN. 68
5.3 F1 score (y-axis) on nine NCI datasets with decreasing training data. 80
5.4 F1 score (y-axis) on nine NCI datasets with different levels of label noise. . . 81
5.5 F1 scores (y-axis) of MOSGNN with different hyperparameter (λ and β) settings on nine NCI datasets. 82

6.1 A set of graphs with two anomalous graphs indicated. The squares above/below the nodes represent node features. G_5 is a locally-anomalous graph due to the unusual local properties (*e.g.*, structure) of the orange node, while G_6 is a globally-anomalous graph because it does not conform to G_1 to G_4 in holistic graph properties. 86
6.2 Demonstration of GLocalKD working on a popular dataset – AIDS. (a) Representations of training graphs output by the random target network. (b) Representations of training graphs learned by the predictor network. (c) Prediction errors (anomaly scores) of GLocalKD on test graphs. Visualization in (a) and (b) is based on t-SNE. 87
6.3 The proposed framework. 89
6.4 AUC performance of GLocalKD and OCGCN using different amount of training data. 98
6.5 AUC performance of GLocalKD and OCGCN w.r.t. different anomaly contamination rates. 99
6.6 AUC results w.r.t. representation dimensionality. 99
6.7 AUC of GLocalKD with different GCN depths. 100

Part I

Research Background and Foundation

CHAPTER 1

INTRODUCTION

The motivation and the contributions of this research are introduced in this chapter, followed by the organization of this thesis.

1.1 Research Motivation

Graph has become one of the most powerful data storage modes due to its specific structure characteristics to model objects and pairwise relations between them. There is an increasing number of real-world high-impact tasks whose data are represented by graphs, including social science, e-commerce networks, biology, traffic networks, chemistry and physics. For example, in chemistry, chemical compounds can be denoted as graphs when regarding atoms as nodes and chemical bonds between them as edges; in a citation network, papers are regarded as nodes and the reference relationships between them are edge attributes.

In the extensive graph learning fields, graph representation learning plays a crucial role. A more informative graph representation can improve the results of the downstream tasks. Therefore, how to adequately utilize the known graph information to obtain more meaty graph representations is an important research topic. One of the key issues is to enhance the expressive power of the graph representation learning model, which means to learn more informative and discriminative graph representations. Numerous graph neural networks (GNNs) have demonstrated their excellent performance in graph representation learning when compared with tradition methods in recent years. How to

design GNN with *high expressiveness* is a challenging yet meaningful work. In addition, the data collection process in real world is an expensive work and the obtained sample set might be insufficient, *e.g.*, some class of data might be extremely less than others or even totally lacking. How to exploit expressive and discriminative graph representations based on *limited amount of supervision information* is another challenge.

To examine the performance of the graph representation learning models, applying them to down-stream tasks is a natural choice. The down-stream tasks can be various, *e.g.*, node-level, edge-level or graph-level, according to the aiming problems in real application. In this thesis, we turn the challenges above into learning graph representations for three graph-level classification problems due to their meaningful real-world applications while lack of focus, *i.e.*, balanced graph-level classification, imbalanced graph-level classification and one-class graph-level classification (also regarded as graph-level anomaly detection (GLAD)). One application example of these three problems in chemistry domain is as follows:

Example 1 (Example in Chemistry)**.** *In a graph dataset for chemical compounds, samples can be toxic or nontoxic. Discovering the toxicity of the remaining compounds with the toxicity of known samples can be regarded as the problem of balanced graph-level classification. When collected samples are insufficient and the distributions of known toxic and nontoxic samples are imbalanced, the problem turns to imbalanced graph-level classification. Under the circumstance that only nontoxic compounds are gathered and toxic items are required to discover, graph-level anomaly detection method is a good choice.*

Therefore, this thesis is centered on establishing mechanisms to learning informative and expressive graph representations to solve three challenging graph-level problems. The aims of these problems are different, leading to varying emphasis of graph representations for each problem: (i) for classification with sufficient data, the goal of graph representation learning is to learn expressive representations with more informative and discriminative knowledge of graphs; (ii) alleviating the influence of imbalanced data distribution with the limited minority samples is significant for imbalanced graph classification to learn representative graph embeddings; (iii) for graph-level anomaly detection, capturing characteristic graph information only with one type of abnormal samples is the focus.

1.2 Contributions

This thesis explores on expressive graph representation learning. Due to the fact that the data collection is an expensive and tough work, the gathered data in real-world application might be insufficient. How to utilize finite known samples to learn graph representation with high expressive power is an important research problem. We use three significant down-stream tasks in graph domain that have varying amount of supervision information to measure the expressiveness of the graph representation learning methods. Our contributions in this thesis are summarized as follows:

- We explore more expressive graph representations based on balanced sample distribution for the balanced graph-level classification problem in Chapter 4. Some popular message passing GNNs have been proved to have limited expressiveness [232]. We harness a diverse set of structure features that are learned with the original features simultaneously by a designed specific message passing mechanism to *enrich the learned graph representations*. This work has been published in *Neural Networks* [142].

- When the known samples are distributed off-balance, the graph embeddings for minority classes learned by the ordinary graph representation learning techniques might be influenced by the samples from majority classes. To learn expressive representations of the graphs in minority classes for imbalanced graph classification, we introduce a novel multi-scale oversampling scheme to learn graph representations based on intra- and inter-graph semantics in Chapter 5, resulting in *representations possessing discriminative information* embedded within and between the minority graphs. This work has been published in IJCNN-24 [143].

- In Chapter 6, we formulate the graph-level anomaly detection problem as the task of detecting locally- or globally-anomalous graphs, and empirically verify the presence of these two types of graph anomalies in real-world datasets. We introduce the *first approach* specifically designed to effectively *detect both types of anomalous graphs* by joint random distillation of graph and node representations. The work has been published in WSDM-22 [144].

Our research also has large potential practical values. Our proposed classification algorithms offer higher accuracy than some existing classification methods, which help alleviate error and would be of great significance in real-world tasks, *e.g.*, reducing inaccurate advertisement putting to social communities and saving a large amount

of money spent on human resources for molecule and chemical compound property determination. Our research on graph-level anomaly detection, as the first end-to-end deep method specifically designed to discover two types of abnormal graphs, is also important to varying applications, *e.g.*, omitting numerous experiments to identify toxic molecules from a set of chemical compounds, reducing human trials in recognizing drugs with severe side-effects or preventing the loss of millions of dollars by detecting fraud communities.

1.3 Organization

We next introduce the organization of the rest of this thesis. Chapter 2 presents some common and fundamental concepts used through the thesis, including the common notations, basic definitions on graphs, introduction of datasets and the measurements employed in the following chapters. In Chapter 3, we first review literature of graph representation learning methods, followed by works of balanced and imbalanced classification and anomaly detection in graph domain.

Chapters 4–6 contain main research of this thesis. Chapter 4 focuses on how to improve the expressive power of a message passing neural network to learn more informative graph representation for graph classification. Chapter 5 explores improving the graph representation learning of samples in minority class for imbalanced graph-level classification. Chapter 6 aims to detect abnormal graphs with local/global anomalies. Chapter 7 summarizes the thesis and discusses the possible future research directions.

CHAPTER 2

PRELIMINARIES

In this chapter, some most common symbols and notations that are important for the understanding of the methods and algorithms reviewed and proposed in this thesis are described, followed by the main definitions of our research problems. Later, we provide a detailed introduction of the datasets and the performance measurements employed in the thesis.

2.1 Notations

In this section, we first introduce some mathematical notations that will be used in the following review. Generally, a graph can be illustrated as $G = (\mathcal{V}_G, \mathcal{E}_G)$, where $\mathcal{V}_G$ is the set of nodes and $\mathcal{E}_G$ is the set of edges. The node set $\mathcal{V}_G = \{v_1, v_2, ..., v_{N_G}\}$ contains $N_G = |\mathcal{V}_G|$ nodes. The set of edges $\mathcal{E}_G = \{e_{ij} = (v_i, v_j)\}_{ij}$ consists of $M_G = |\mathcal{E}_G|$ edges which connect two nodes in $\mathcal{V}_G$. A graph data set with P graphs is denoted as $\mathcal{G} = \{G_1, ..., G_P\}$.

In a graph $G = (\mathcal{V}_G, \mathcal{E}_G)$, the adjacency matrix is defined as $A \in \{0,1\}^{N_G \times N_G}$. The element in i_{th} row, j_{th} column, $A_{i,j}$, represents the connectivity of two vertices v_i and v_j. In detail, if v_i is adjacent to v_j, $A_{i,j} = 1$, otherwise, $A_{i,j} = 0$. $D \in \mathbb{R}^{N_G \times N_G}$ is a diagonal degree matrix, *i.e.*, $D = diag(\sum_{j=1}^{N_G} A_{1,j}, ..., \sum_{j=1}^{N_G} A_{N_G,j})$. The neighborhood of node v_i is defined as $\mathcal{N}(v_i) = \{v_j \in \mathcal{V}_G | e_{ij} \in \mathcal{E}_G\}$. Some graphs are equipped with node attribute matrix X, in which row $\mathbf{x}_i$ is the attribute vector of node v_i. For other variables, we use uppercase letters for matrices and lowercase letters for vectors, for example, a matrix F and a vector $\mathbf{f}$.

We also introduce some common notations for deep learning models. In this thesis, superscripts are used to represent layers. The trainable parameters in each deep learning model is represented by W. During training, $H^{(l)}$ is the learned hidden representation matrix of nodes in l_{th} layer and its i_{th} row, *i.e.*, $\mathbf{h}_i^{(l)}$, is the hidden representation vector of node v_i. The rectified linear unit (ReLU) $\text{ReLU}(x) = \max(0, x)$ is a commonly used activation functions in deep learning models. $\rho(\cdot)$ represents a general nonlinear activation function. MLP$(\cdot)$ denotes a multi-layer perceptron.

Table 2.1 summarizes these common symbols and their descriptions. Additional symbols will be defined in the following chapters if necessary.

Notations	**Descriptions**
$\mathcal{G}$	A graph dataset.
G	A graph.
$\mathcal{V}_G$	The set of nodes in graph G.
$\mathcal{E}_G$	The set of edges in graph G.
v_i	Node $v_i \in \mathcal{V}_G$.
e_{ij}	Edge $e_{ij} \in \mathcal{E}_G$.
$\lvert\cdot\rvert$	The length of a set.
N_G	The number of nodes in graph G.
M_G	The number of edges in graph G.
X	The matrix of node attributes.
$\mathbf{x}_i$	The attribute vector of node v_i.
A	The graph adjacency matrix.
D	The diagonal degree matrix.
I	The identity matrix.
$\mathcal{N}(v_i)$	The neighborhood of node v_i.
$H^{(l)}$	The hidden representation matrix of nodes in l_{th} layer.
$\mathbf{h}_i^{(l)}$	The hidden representation vector of node v_i in l_{th} layer.
l	The layer index.
L	The layer number.
$W^{(l)}$	Trainable model parameters in l_{th} layer.
ReLU$(\cdot)$	The ReLU activation function.
$\rho(\cdot)$	A nonlinear activation function.
MLP$(\cdot)$	A multi-layer perceptron.

Table 2.1: Commonly used notations and their descriptions.

2.2 Definitions and Problems on Graphs

This section introduces some basic definitions of graph and problems on graphs that this thesis focuses on.

The nodes and edges in different graphs can have varying characteristics, resulting in different types of graphs. We first present definitions of several graphs in following:

Definition 1 (Directed/Undirected Graph)**.** *An edge is a directed edge if one of its endpoints is designated as the head and the other endpoint is designated as the tail, otherwise it is an undirected edge. A directed / undirected graph is a graph in which each edge is directed / undirected.*

Definition 2 (Weighted/Unweighted Graph)**.** *A weighted graph is a graph in which each edge is assigned with a weight, called its edge weight. If edges of a graph have no weights, it is an unweighted graph.*

Definition 3 (Attributed/Plain Graph)**.** *An attributed graph is a graph in which each node is assigned with an attribute vector. Otherwise, it is a plain graph.*

This thesis focuses on *undirected and unweighted graphs* due to their wide applications in real world.

Next, we introduce the three graph-level tasks researched in this thesis. Graph-level classification aims to predict the class label for an entire graph. In *Supervised Balanced Graph-level Classification* task, given a set of graphs $\mathcal{G} = \{G_i\}_i$, each graph is assigned with a class label and the class distribution is balanced, the aim is to learn an function $f : \mathcal{G} \rightarrow \mathbb{R}$, parameterized by W, such that $f(G;W)$ outputs the label of graph G. For *Supervised Imbalanced Graph-level Classification* task, the class distribution in training dataset is imbalanced.

Anomalous graphs in a graph set can be classified into two categories, *i.e.*, locally-anomalous graphs and globally-anomalous graphs, which are respectively defined as follows.

Definition 4 (Locally-anomalous Graph)**.** *Given a graph data set $\mathcal{G} = \{G_i\}_i$, with each graph $G \in \mathcal{G}$ denoted by $G = (\mathcal{V}_G, \mathcal{E}_G)$, graph $\hat{G}$ is a locally-anomalous graph if $\hat{G}$ does not conform to the graphs in $\mathcal{G}$ due to the presence of some anomalous nodes v, $\forall v \in \mathcal{V}_{\hat{G}}$, that significantly deviate from similar nodes in the graphs in $\mathcal{G}$.*

Definition 5 (Globally-anomalous Graph)**.** *Given a graph data set $\mathcal{G} = \{G_i\}_i$, graph $\hat{G}$ is a globally-anomalous graph if the holistic graph properties of $\hat{G}$ do not conform to that of the graphs in $\mathcal{G}$.*

The *Unsupervised Graph-level Anomaly Detection* focuses on identifying these two types of abnormal graphs. Specifically, given a set of normal graphs $\mathcal{G} = \{G_i\}_i$, graph-level anomaly detection problem aims at learning an anomaly scoring function $f : \mathcal{G} \to \mathbb{R}$, parameterized by W, such that $f(\hat{G}_i; W) > f(\hat{G}_j; W)$ if $\hat{G}_j$ conforms to $\mathcal{G}$ better than $\hat{G}_i$.

2.3 Dataset and Performance Evaluation

The section displays crucial datasets and the measurements employed in the experiments to evaluate the performance of models.

2.3.1 Dataset

The datasets used in this thesis are all public. Since the goals of this thesis concentrate on three different tasks, the datasets and the processing methods used for different tasks also should be varying. We will introduce these datasets respectively.

For balanced graph-level classification, 12 datasets from TUDataset graph classification benchmark [154] are employed. These datasets are from various areas, with different number of classes and various sparsity. For datasets with plain graphs, node labels will be used as node attributes. The detailed introduction of these datasets are presented in Table 2.2.

For imbalanced graph-level classification, 7 datasets from the TUDataset graph classification benchmark [154] are used. Besides, 9 NCI chemical compound graph datasets for anticancer activity prediction from Pubchem Library[1] are employed to enlarge the imbalanced-ratio diversity of datasets. They are all binary datasets. NCI1 and NCI109 used here are the whole datasets while the two datasets used in above balanced task are balanced version by [154]. We use NCI1* and NCI109* here to differentiate them. Table 2.3 displays the imbalanced ratios of these datasets. The imbalanced ratios of BZR and COX2 are about 3.5: 1, which is relatively small and usual classification method can obtain a satisfactory result. Aromatase, ATAD5, ER and p53 are with medium imbalanced ratios, while the NCI datasets are generally much more imbalanced.

[1] http://pubchem.ncbi.nlm.nih.gov

Table 2.2: The detailed information of 12 public datasets for balanced graph-level classification. The 'binary' in the 'Class' column denotes the dataset is for binary classification while 'multi' implies multi-class classification. The '#Graphs' is the total number of graphs in the dataset and the '#Nodes' means the average number of nodes in the dataset. The '✓' in the 'Attribute' column indicates the data contains attributed graphs, and otherwise they contain only plain graphs.

Dataset	Area	Class	#Graphs	#Nodes	Attribute
BZR	molecule	binary	405	35.75	✓
COX2	molecule	binary	467	41.22	✓
DD	bioinformatics	binary	1178	284.32	-
IMDB−BINARY	social	binary	1000	19.77	-
IMDB−MULTI	social	multi	1500	13.00	-
MUTAG	molecule	binary	188	17.93	-
NCI1	molecule	binary	4110	29.87	-
NCI109	molecule	binary	4127	29.68	-
PROTEINS_full	bioinformatics	binary	1113	39.06	✓
REDDIT−BINARY	social	binary	2000	429.63	-
REDDIT−MULTI	social	multi	4999	508.52	-
ENZYMES	bioinformatics	multi	600	32.63	✓

The use of datasets with different imbalanced ratios helps justify the applicability of our method in dealing with various imbalanced data.

15 datasets from the TUDataset graph classification benchmark [154] and one dataset from Toxicity Prediction Task[2], which are illustrated in Table 2.4, are used in unsupervised graph-level anomaly detection. Although these data sets are often used for classification tasks, using the class with less instances as the abnormal class to execute anomaly detection also makes sense. In the training dataset, the data of abnormal class will be discarded directly and not be used in the training phase.

Datasets that are used to examine other abilities of the models will be introduced in the corresponding chapter.

2.3.2 Performance Evaluation

Accuracy, the proportion of correct predictions among the total number of cases examined, is the most commonly used measurement for classification. We employ accuracy as the main measurement to examine the performance of models in balanced graph-level classification task. To increase the credibility of our results, we also utilize the area under the Precision-Recall curve (AUPRC) [15] as another evaluation metric, which

[2]https://tdcommons.ai/single_pred_tasks/tox/#herg−blockers

Table 2.3: The imbalanced information of 16 public datasets for imbalanced graph-level classification. '#*pos*' and '#Graphs' denote the number of graph samples in the minority class and in the full dataset, respectively. 'Ratio' represents the ratio of the majority class size to the minority class.

Dataset	#*pos*	**#Graphs**	**Ratio**
NCI1*	1793	37349	19.8 : 1
NCI33	1467	37022	24.2 : 1
NCI41	1350	25336	17.8 : 1
NCI47	1735	37298	20.5 : 1
NCI81	2081	37549	17.0 : 1
NCI83	1959	25550	12.0 : 1
NCI109*	1773	37518	20.2 : 1
NCI123	2715	36903	12.6 : 1
NCI145	1641	37043	21.6 : 1
BZR	86	405	3.7 : 1
COX2	102	467	3.6 : 1
P388	2298	41472	17.1 : 1
Aromatase	360	7226	19.1 : 1
ATAD5	338	9091	25.9 : 1
ER	937	7697	7.2 : 1
p53	537	8634	15.1 : 1

provides a single value that summarizes the overall performance of a model. AUPRC focuses on the performance of model on a specific class (defined as positive class) by considering the Precision and Recall simultaneously, which are the proportion of true positive predictions out of all positive predictions made by the model and the proportion of true positive predictions made by the model from all actual positive samples in the dataset, respectively. The AUPRC is the integration of the area under the Precision-Recall curve which plots the Precision against the Recall at different threshold settings. A perfect model would have a AUPRC of 1, indicating that higher AUPRC means better performance. We use AUPRC to examine the performance of the models on one class and be a supplementary to further testify our results.

F1-score [185], which is a weighted harmonic mean of Precision and Recall, is used as the measurement for imbalanced graph-level classification. F1-score balances the trade-off between Precision and Recall. If a model obtains high Precision but low Recall, it indicates that the model predicts fewer false positives but misses a lot of true positives. In contrast, a model with high Recall but low Precision makes more false positives but identifies more true positives. In these cases, the F1-score can evaluate the overall performance of the models. F1-score ranges from 0 to 1 and higher F1-score indicates

Table 2.4: The detailed information of 16 public datasets for graph-level anomaly detection. The '#Graphs' is the total number of graphs in the dataset and the '#Nodes' means the average number of nodes in the dataset. The '✓' in the 'Attribute' column indicates the data contains attributed graphs, and otherwise they contain only plain graphs.

Dataset	**Area**	**# Graphs**	**# Nodes**	**Attribute**
PROTEINS_full	bioinformatics	1113	39.06	✓
ENZYMES	bioinformatics	600	32.63	✓
AIDS	molecule	2000	15.69	✓
DHFR	molecule	467	42.43	✓
BZR	molecule	405	35.75	✓
COX2	molecule	467	41.22	✓
DD	bioinformatics	1178	284.32	–
NCI1	molecule	4110	29.87	–
IMDB-BINARY	social	1000	19.77	–
REDDIT-BINARY	social	2000	429.63	–
HSE	molecule	8417	16.89	–
MMP	molecule	7558	17.62	–
p53	molecule	8903	17.92	–
PPAR-gamma	molecule	8451	17.38	–
COLLAB	social	5000	74.49	–
hERG	molecule	655	26.48	–

better performance.

For graph-level anomaly detection, the detection rate of anomalous samples is the main indicator of the model performance. We employ the area under the ROC curve (AUC) [72] as the measurement in this task. The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) under different classification thresholds. TPR equals to Recall, while FPR is the proportion of false positive predictions to the total number of negative instances. The AUC is the area under the ROC curve, indicating the probability that a randomly chosen positive sample will be ranked higher by the model than a randomly chosen negative sample. Its consideration in the class-imbalance nature of anomaly detection problems enables AUC to be an ideal measurement in outlier detection tasks. A perfect model would obtain an AUC of 1, while a random model would achieve an AUC of 0.5, namely that higher AUC means better performance. Although AUPRC is also an eligible measurement for imbalanced datasets, it emphasizes the performance of the models on one class instead of all classes. Therefore, we utilize AUC as the measurement in the graph-level anomaly detection task.

The Wilcoxon signed rank test [37] is also leveraged in three tasks to measure the performance significance of our proposed methods against its competitors.

CHAPTER 3

LITERATURE REVIEW

Graph representation learning is a crucial step to solve various graph tasks since one can apply other machine learning tools to these representations conveniently for any down-stream tasks. From traditional methods to deep methods, graph representation learning is always a popular research topic in the past few decades. Especially in recent years, numerous deep methods have demonstrated their excellent performance in learning graph representation within rich graph knowledge. To solve various graph tasks, it is significant and important to have a comprehensive understanding of these methods. Therefore, this chapter first review the literature that is related to various graph representation learning models, including traditional methods and deep methods. Literature based on these methods to solve graph classification and anomaly detection problems is reviewed later. At the end of this chapter, a summary of this review is offered.

3.1 Graph Representation Learning Methods

Traditional methods constitute this research direction in the early stage and later graph kernel methods are proposed to improve the performance. With the popularity of deep learning, graph neural networks become dominant solutions at present. These methods are reviewed below in detail and summarized in Table 3.1.

3.1.1 Traditional Methods

Graph representation learning methods at the early stage are non-deep. Among these methods, methods based on matrix factorization are one of the pioneers, followed by random walk based methods and graph kernel based methods. We review these methods as follows.

Matrix Factorization Based Methods

Matrix factorization based methods are one of the pioneers in graph representation learning works. These methods aim to reduce the high-dimensional graph matrix into a low-dimensional space via matrix factorization. One type of these methods focuses on decomposing node proximity matrix into low-dimensional matrix directly, which can be divided into two phases, including the proximity-based matrix construction and the matrix dimension reduction [237]. Many well-known matrix decomposition methods are employed, *e.g.*, singular value decomposition (SVD) [100], principal component analysis (PCA) [89]. The elements of proximity-based matrix measure proximity of node pairs in the graph, whose construction is significant to the final performance of the method. Various structural characteristics are utilized to construct the information-rich proximity matrix, including adjacency matrix [1], different orders of graph adjacency matrices [20; 117; 237], katz index [162], rooted page rank [162], the number of common neighbors [162] and personalized page rank [240; 259]. Another type of matrix factorization based methods considers the minimum eigenvalue of the graph Laplacian matrix to obtain the node embeddings [1; 64]. However, the matrix factorization in these methods suffers from expensive time and memory complexity for large-scale graphs and is with limited generalization [75].

Random Walk Based Methods

Since structure is a significant and special characteristic of graph, researchers also consider using graph structure to construct graph representation. The main idea is to capture the graph structural information by random walks of each node and the nodes occurring in same random walk should have similar node representations. DeepWalk [175] and Node2Vec [67] are two representative methods. They first generate a set of random walks and then train a SkipGram model to obtain the node embeddings. The difference between DeepWalk and Node2Vec is mainly the way which the random walks are generated by. Many variations are later proposed to improve their perfor-

mance [272; 182; 141; 86; 229; 21; 83]. However, this type of methods has less robustness for noisy structure and less transductive learning ability [75].

Graph Kernel Based Methods

Graph kernel focuses on comparing the similarity between graphs or substructures in graph. As another vanguard of graph representation learning, it calculates node embeddings via mapping pairs of nodes to latent space by specific similarity measures. In [189], a graphlet kernel is introduced, which establishes graph feature by counting the number of different graphlets in graphs. With the inspiration from 1-dimensional Weisfeiler-Lehman (WL) test [103], authors in [188] propose Weisfeiler-Lehman kernels which represent graph as WL sequences with various height, including Weisfeiler-Lehman subtree kernel, the Weisfeiler-Lehman edge kernel and the Weisfeiler-Lehman shortest path kernel. Due to the limited expressive ability of 1-dimensional WL test, the WL subtree kernel based on k-dimensional WL algorithm is presented in [153]. Other WL algorithm based graph kernels are illustrated in [199; 157]. Since substructure is a significant feature in graph, there are also works constructing graph kernel via substructures, *e.g.*, random walks [93; 92; 156] and shortest paths [13; 56; 238]. For example, propagation kernels (PK) [156] capture structural information in graph by the early stage distributions from propagation schemes such as random walks. Although many works try to reduce the computational complexity of graph kernels, the calculation of graph kernel is still costly on large-scale graphs.

3.1.2 Graph Neural Networks

The ability of traditional methods to deal with complex graph structures is limited. Considering the excellent performance of deep methods on image and text data, researchers begin to pay their attention to constructing deep models to learn graph information. Various deep graph neural network (GNN) models explode rapidly in recent years and have demonstrated their remarkable performance [263; 226; 90]. Graph Recurrent Neural Networks (Graph RNNs) are mostly pioneer works of GNNs, which exchange information between nodes and their neighbors until an equilibrium is reaching [65; 187; 119; 207; 250; 68]. However, the recurrent layers in these models are mostly with the same weights during the weight update phase, resulting in limited expressiveness in the relationship constraints between nodes and their neighbors [75]. Instead of iterating node knowledge with contractive constraints, Graph Convolutional Neural

Networks (GCNs) utilize a fixed number of layers with different weights to represent the mutual dependencies between nodes and their neighbors. Because of its effectiveness and convenience compared to other graph neural network methods, Graph Convolutional Neural Networks (GCNs) are undoubtedly the hottest topic in the area of deep learning on graph in recent years. Like CNNs, GCNs employ designed convolution and pooling function to learn the local and global structural patterns and features of graphs. In the following subsection, we will discuss these two functions respectively.

In CNNs, convolution operation is the most primary and crucial part. However, due to the lack of a grid structure in graphs, standard convolution operations for images or text cannot be directly employed by graphs [191]. The design of convolutional functions has significant impact on the performance of GCNs. In existing GCNs models, the convolution operations can be divided into two categories, including spectral convolutions and spatial convolutions. The spectral convolutions design the filters by the spectral graph theory while the spatial convolutions focus on the information propagation of node neighborhoods.

Spectral Based Graph Convolutional Neural Networks

Authors in [17] propose Spectral Convolutional Neural Network (Spectral CNN), which uses eigendecomposition of the graph Laplacian matrix $\mathbf{L}$ to introduce graph convolution from the spectral perspective. One disadvantage of it is the high requirement of computational complexity. Its another limitation is that learned filters cannot be utilized by graphs with different structures and sizes. To reduce the computational complexity and improve the efficiency, ChebNet [36] and CayleyNet [104] use some polynomial filters to construct graph convolution operators.

Graph Convolutional Network (GCN) [96], which is a simple yet efficient GNN backbone, is the first-order approximation of ChebNet and has been one of the most popular GNN backbone. The hidden representation of node v_i in the l_{th} layer, $\mathbf{h}_i^{(l)}$, is formulated as

$$\mathbf{h}_i^{(l)} = \rho\left(\sum_{v_j \in \tilde{\mathcal{N}}(v_i)} \frac{1}{\tilde{D}(i,i)\tilde{D}(j,j)} \mathbf{h}_j^{(l-1)} W^{(l)}\right), \tag{3.1}$$

where $\tilde{D} = D + I$ and $\tilde{\mathcal{N}}(v_i) = \mathcal{N}(v_i) \bigcup \{v_i\}$. Its matrix form can be written as

$$H^{(l)} = \rho(\tilde{D}^{\frac{1}{2}} \tilde{A} \tilde{D}^{\frac{1}{2}} H^{(l-1)} W^{(l)}), \tag{3.2}$$

where $\tilde{A} = A + I$. Some improvements are made on GCN in several recent works. In [112], Adaptive Graph Convolutional Network (AGCN) is proposed to learn hidden

structural relations that are not specified by the graph adjacency matrix. The Dual Graph Convolutional Network (DGCN) [276] constructs a dual graph convolutional architecture through two convolutions: one is the common convolution in GCN and another uses the positive pointwise mutual information (PPMI) matrix of the transition probability to replace the adjacency matrix. LanczosNets [120] use Lanczos algorithm to construct a low-rank approximation of graph Laplacian. In [230], the Graph Wavelet Neural Network (GWNN) utilizes sparse graph wavelet transform instead of matrix eigendecomposition to reduce computational cost. Authors in [275] employ a modified Markov Diffusion Kernel to establish the Simple Spectral Graph Convolution (S^2GC), which captures the global and local contexts of each node with lower computation and storage expense. GNNML3 [9] utilizes custom non-linear functions of eigenvalues and a masked convolution support with desired length of receptive field and is experimentally proven to be as powerful as 3-WL test.

Spatial Based Graph Convolutional Neural Networks

Another type of convolution operators is spatial convolutions, which transform and aggregate neighbor features to calculate the representation of corresponding node. The mostly pioneer work of spatial-based graph convolutional neural networks is Neural Network for Graph (NN4G) [150]. This work sums up the neighborhood information of a node to obtain graph convolutions. Diffusion Convolutional Neural Network (DCNN) [5] uses a diffusion transition probability between nodes to determine the neighborhoods of a node. GCN [96] also has its spatial interpretation, in which it applies weights to features of node and its corresponding neighbors. Message Passing Neural Network (MPNN) [63] provides a general framework of spatial-based graph convolution operations, which could cover many previous methods including GCN [96] and [17]. In MPNN, the graph convolution operation is regarded as a message passing process:

$$m_i^{(l)} = \sum_{v_j \in \mathcal{N}(v_i)} \mathscr{F}^{(l)}(\mathbf{h}_i^{(l-1)}, \mathbf{h}_j^{(l-1)}, \mathbf{x}_{i,j}^E), \ \mathbf{h}_i^{(l)} = \mathscr{U}^{(l)}(\mathbf{h}_i^{(l-1)}, m_i^{(l)}), \tag{3.3}$$

where $\mathscr{F}^{(l)}$ is the message passing function, $\mathscr{U}^{(l)}$ is the node update function, $\mathbf{x}_{i,j}^E$ is the attribute of edge e_{ij}, and $m_i^{(l)}$ is the message passed between vertices. Mixture Model Network (MoNet) [152] introduces the relative weight between nodes, which is defined by a mapping function. GCN [96] and DCNN [5] also can be generalized as special instances of MoNet.

However, most previous models are inherently transductive, meaning that they cannot address unseen nodes [70]. GraphSAGE [70] constructs an inductive GNN structure,

which samples and aggregates features from local neighborhood of a node hierarchically:

$$m_i^{(l)} = \text{AGGATE}^{(l)}(\{\mathbf{h}_j^{(l-1)}, \forall v_j \in \mathcal{N}(v_i)\}), \quad \mathbf{h}_i^{(l)} = \rho(W^{(l)} \left[\mathbf{h}_i^{(l-1)}, m_i^{(l)}\right]), \tag{3.4}$$

where $\text{AGGATE}^{(l)}(\cdot)$ is an aggregating function and $[\cdot,\cdot]$ is the concatenation operation.

Considering the remarkable performance of attention mechanism in image tasks, Graph Attention Network (GAT) [203] introduces attention mechanism into GCNs:

$$\mathbf{h}_i^{(l)} = \rho\left(\sum_{v_j \in \mathcal{N}(v_i)\cup\{v_i\}} \alpha_{i,j}^{(l)} \mathbf{h}_j^{(l-1)} W^{(l)}\right), \tag{3.5}$$

where $\alpha_{i,j}^{(l)}$ is the attention of v_i to v_j in the l_{th} layer and defined as

$$\alpha_{i,j}^{(l)} = \frac{\exp(\text{LeakyReLU}(\mathscr{A}(\mathbf{h}_i^{(l-1)}W^{(l)}, \mathbf{h}_j^{(l-1)}W^{(l)})))}{\sum_{v_k \in \mathcal{N}(v_i)\cup\{v_i\}} \exp(\text{LeakyReLU}(\mathscr{A}(\mathbf{h}_i^{(l-1)}W^{(l)}, \mathbf{h}_k^{(l-1)}W^{(l)})))}, \tag{3.6}$$

where $\mathscr{A}$ is an attention function to be learned. Authors in [203] also suggest that multi-head attention could improve the expressive capability of the model. Later, Gated Attention Network (GAAN) [253] proposes to learn extra attention score for different heads. A Heterogeneous Graph Attention Network (HAN) is proposed in [212] to study heterogeneous graph neural network by two attentions, i.e. the node-level and semantic-level attentions. GATv2 [16] uses dynamic attention to enhance the expressive power of GAT [203].

Previous GNNs are at most as powerful as the WL test in distinguishing graph structures and some graph structures cannot be distinguished by some popular GNN variants, such as GCN and GraphSAGE [232]. To solve such problem, [232] proposes Graph Isomorphism Network (GIN), which is as powerful as the WL test by adding a learnable parameter $\epsilon^{(l)}$ to adjust the weight of the central node:

$$\mathbf{h}_i^{(l)} = \text{MLP}\left((1+\epsilon^{(l)})\mathbf{h}_i^{(l-1)} + \sum_{v_j \in \mathcal{N}(v_i)} \mathbf{h}_j^{(l-1)}\right). \tag{3.7}$$

There are other GNN models focusing on varying effect of different neighbors during the aggregating process. Inspired by [150], Contextual Graph Markov Model (CGMM) constructs a deep architecture composed of layers of probabilistic models, which introduces probabilistic explainability while maintaining spatial locality [6]. In [118], Diffusion Graph Convolution (DGC) improves DCNN by summing up output of each diffusion step instead of concatenating them. However, the distant neighbors will contribute less under the transition probability matrix. PGC-DGCNN [200] establishes a shortest-path adjacency matrix to increase the influence of distant neighbors. Partition graph

convolution (PGC) [236] defines a criteria by which the neighbors of a node will be divided into several groups. Some other methods rank neighbors of each node via some certain criteria and select several neighbors with top ranks to share information [158; 59].

Some methods pay efforts to the message aggregation mechanism. Authors in [62] derive personalized propagation of neural predictions (PPNP) and its fast approximation from an improved propagation mechanism based on personalized PageRank. Principal Neighborhood Aggregation (PNA) [33] combines multiple aggregators with degree-scalers. GraphAIR [79] explicitly models neighborhood interaction as well as neighborhood aggregation to capture the complicated non-linear features in graph. Efficient Graph Convolution (EGC) [196] uses spatially-varying adaptive filters to improve the performance while reduce the memory complexity.

Combining Graph Kernels with Graph Neural Networks

There are some works combining graph kernels and neural networks, which can utilize the strengths of both two mechanism. K-dimensional Graph Neural Networks (k-GNN) [155] inaugurates this field, which combines the WL-subtree kernel with GNN. Graph Neural Tangent Kernels (GNTKs) [46] are a new category of graph kernels, which equals to infinitely wide multi-layer GNNs trained by gradient descent, and a general recipe which translates a GNN architecture to its corresponding GNTK is presented in [46]. Graph Convolutional Kernel Networks (GCKN) [23] present a family of multilayer graph kernels and combine GCN and kernel method together. Heterogeneous Graph Kernel based Graph Neural Network (HGK-GNN) [137] utilizes Mahalanobis Distance to build Heterogeneous Graph Kernel (HGK) and further combines it with GNN. Graph Structural Kernel Network (GSKN) [136] proposes an anonymous walk graph kernel (AWGK) and derives its GNN structure to calculate its kernel mapping, which is then compared with random walk kernel. Kernel Graph Neural Networks (KerGNNs) [54] construct graph filters by trainable hidden graphs and update node representations by graph kernels of them and subgraphs. These methods leverage the advantages of graph kernels and graph neural netoworks, but they are also with weak efficiency on large-scale graphs due to the limitation of graph kernels.

Obstacles and Improved GCNs

One of the main obstacles of using GCNs is the over-smoothing problem, which means that the output representations might not be isolated when the depth of network

is large. Many researches usually construct GCNs with 2 or 3 layers. To model deep GCNs, inspired by the idea of ResNet, residual connections are added to GCNs in some models [96; 176; 97; 233; 28]. Another problem that might hinder the applications of GCNs is that, the number of neighbors when training GCNs might be extremely large when a graph is large-scale or dense. Some sampling methods have been proposed to address this issue. PinSage [241] proposes to use random walks on the graph to sample neighbors. Stochastic Training of Graph Convolutional Network (StochasticGCN) [26] utilizes the historical activation of the last batch as a control variable to reduce the sampling variance, thereby theoretically guaranteeing an arbitrarily small sample size. In Fast Learning with Graph Convolutional Network (FastGCN) [25], nodes are interpreted as i.i.d. samples and the graph convolutions are deemed as integral transforms under probability measures. Adapt [82] samples nodes in the lower layer that are conditioned on the top layer. ClusterGCN [30] employs a graph clustering technique to sample a subgraph and the convolutions are implemented on the sampled subgraph.

Pooling Module

To obtain the graph-level representations for final down-stream tasks, another crucial module after acquiring node embeddings via a GNN is the readout operation. The basic and effective readout operations are simple averaging, summation and max pooling [51; 5] which are also popular in classical CNNs. However, such statistics might be not representative enough to assist the model to distinguish different graphs. A fully connected layer is commonly used as a final layer in the model to aggregate the representation of nodes [17]. In [94], fuzzy histograms are constructed for each dimension of the feature vectors and concatenated to obtain the graph-level representation. Works in [116; 63; 200] apply attention mechanisms to improve the averaging and summation pooling. Graph Multiset Transformer (GMT) [7] generates graph representation based on a multi-head attention. In DCGNN [256], a SortPooling method is proposed, which sorts the features in a consistent order and then feeds these sorted descriptors into a 1-D convolutional layer followed by a dense layer. Structural Semantic Readout (SSRead) [101] summarizes node representations by considering their position information. Distribution Knowledge Embedding (DKEPool) [27] views graphs as distributions and treats the pooling operation as summarizing the entire distribution information by simple predefined pooling operations. MPool [85] first utilizes motifs to model the relation between nodes and later develops two motif-based graph pooling models to learn graph representations.

There is one type of methods that captures rich graph structural information by

constructing coarsened graphs in sequence and then applies simple readout operation to the coarsened graphs to gain the final graph representation. For such methods, the construction of coarsened graph is vital. DiffPool [242] constructs differentiable soft cluster assignment matrix by graph convolution. GPool [57] coarsens graph according to the projection values of nodes calculated by a trainable projection vector. The self-attention graph pooling (SAGPool) [102] obtains self-attention scores by graph convolution. EigenPooling [146] coarsens graph based on graph Fourier transform. STRUCTPOOL [246] employs conditional random fields to compute the node cluster assignment matrix. Hierarchical Graph Pooling with Structure Learning (HGP-SL) [262] adaptively forms an induced subgraph for the subsequent layers according to the defined node information score. HaarPooling [220] coarsens graph based on the compressive Haar transform of the graph.

More complex criteria are proposed later for better model performance. A score generated from an importance score and a representativeness score is used to select significant nodes which are globally important and can represent more substructures during graph coarsening in RepPool [109]. Moreover, in RepPool [109], both selected and un-selected nodes are used when the coarsened graph is generated. Vertex infomax pooling (VIPool) [109] coarsens graph based on the neural estimation of mutual information between node features and neighborhood features. MinCutPool [12] hierarchically coarsens the graph with the cluster assignment matrix and can be optimized by the minCUT objective. Adaptive Structure Aware Pooling (ASAP) [180] applies a self-attention scheme to learn the cluster assignment matrix. Graph self-adaptive pooling (GSAPool) [254] utilizes the local structure and the feature information of the nodes to measure the importance of nodes. Topology-aware pooling (TAP) [58] considers local score for each node by attending each node to its neighboring nodes and importance score of each node globally in the entire graph together to choose nodes. MVPool [261] evaluates the importance of nodes in different views via a set of measurements and applies a structure learning mechanism with sparse attention to learn a refined graph structure for the coarsened graph. An interpretable neighborhood information gain criterion is defined to guide the node selection in iPool [61]. SEP [223] utilizes the concept of structural entropy to obtain the hierarchical cluster assignment matrix. HoscPool [50] constructs the cluster assignment matrix by minimizing relaxed formulations of motif spectral clustering. DMoN [201] utilizes spectral modularity maximization to calculate the cluster assignment matrix. Multi-channel Graph Pooling (MuchPool) [45] constructs two coarsened graphs based on the local structure and node features as well as graph

clustering and combines these two coarsened graphs to obtain the final pooled graph. Coarsened Graph Infomax Pooling (CGIPool) [166] constructs positive and negative coarsened graphs and makes the positive one with maximal mutual information with the input graph while the negative one with minimal mutual information with the input graph.

Some works propose to learn the representation of graph directly. In [187], an extra node is added into the graph to represent the entire graph. The added node is connected to all nodes and the network directly learns its features during the training process. GNs [11] learns the representation of graph directly by information passing from all nodes and edges.

Type	Method	Key idea	Drawbacks
Traditional	Matrix factorization	Reduce the high-dimensional graph matrix into a low-dimensional space via matrix factorization.	Limited generalization; High time and memory complexity.
	Random walk	Capture the graph structural information by random walks of each node; the nodes occurring in same random walk should have similar node representations.	Less robustness; Less transductive learning ability.
	Graph kernel	Map node pairs to latent space with particular similarity measures.	High time and memory complexity.
Deep	Graph convolutional neural network	Utilizes a fixed number of layers with different weights to represent the mutual dependencies between nodes and their neighbors.	Over-smoothing problem; Limited performance on disassortative graphs.

Table 3.1: A summary of several types of graph representation learning methods.

3.2 Graph Classification Methods

Classification is always a popular and significant task no matter in image, text or graph. Since the data collection is difficult, the data that we can leverage might be limited. My research focuses on classification with sufficient (balanced) and insufficient (imbalanced) data. GNNs have demonstrated their excellent achievement in various graph tasks when compared with traditional methods. Therefore, my research and this section mainly focus on GNN models.

3.2.1 Balanced Graph Classification Methods

Classification in graphs can be node-, subgraph- and graph-level. The graph representation learning methods mentioned in Section 3.1 can acquire node representations, which

can be leveraged directly by arbitrary classifier. To capture more node information and improve the classification performance, some tricks are introduced to applied to GNN backbones, including augmentation [183; 98; 216; 217; 269], sampling [25; 241; 277; 222; 278], k-hop [205; 31; 235]. Different from node classification, the above structures that perform excellently do not achieve well results on subgraph classification [214]. Until now, the research on subgraph classification is extremely limited [3; 214; 87; 178]. However, since graph-level classification focuses on learning discriminative information among graphs instead of inside graph that the above node or subgraph classification methods aim to, the above classification methods cannot be used directly for graph-level classification.

For graph-level classification, some works try to generalize convolutional neural networks (CNNs) to graphs [172; 202; 221], whose challenge is to construct the fixed-size graph receptive fields required by CNNs due to the specific structure of graphs. The convolutional filters designed in GNNs avoid this issue. The graph representation learning methods and the pooling methods that are reviewed in Section 3.1 can be combined together to learn informative graph-level representations, which are classified by an arbitrary classifier. The discriminability of the learned graph representations is crucial for the performance of the whole classification model. There are also other algorithms that are proposed based on those GNN backbones to enhance the representation expressiveness of GNNs and further improve the classification performance. One direction is to capture high-order knowledge. Some pooling mechanisms mentioned in Section 3.1.2 learn hierarchical information by coarsening graphs sequently to further improve the informativeness of the learned graph embeddings, *e.g.*, DiffPool [242], SAGPool [102], HGP-SL [262], VIPool [109], etc. Some works aggregate knowledge not only from its neighbors, but also from its k-hop neighborhood [159; 55]. A virtual node/edge is added in [84; 105; 63] to enable each node to obtain knowledge from nodes outside its one-hop neighborhood. Instead of learning from distant neighborhood, global topological information is calculated via extended persistence in [258].

Data augmentation also becomes an efficient scheme in graph classification. Some methods realize augmentation via enriching node features. For example, distance encoding is proposed in [110] to generate and add extra node features. RGIN [186] adds random node features, while GSN [14] and fast ID-GNN [243] use the count of various motifs to extend the node features. In [127], neighborhood features are augmented via a generative model conditioned on local structures and node features. Except to node feature extension, augmenting the graph from the structure perspective is another popular approach. For example, NestedGNN [257] and ID-GNN [243] sample a subgraph for each

node and use the subgraphs to compute node embeddings and add them to complement the original graph. The structural information used in these methods is local, while many useful graph-level structural information is ignored. To alleviate this issue, in [128], a dummy node that connects to all existing nodes is added without affecting original node and edge properties for better graph representation learning. Mixup [217], $\mathscr{G}$-mixup [71] and graph transplant [168] mixup graphs to obtain more graph data for data-hungry tasks. DropGNN [167] instead performs augmentation through iteratively removing nodes randomly and executing multiple different runs on these node-dropout graphs. TOGL [77] leverages persistent homology to incorporate global topological knowledge.

3.2.2 Imbalanced Graph Classification Methods

For data with imbalanced distribution, the classification methods reviewed above tend to favor the majority class and under-represent samples from minority classes, leading to sub-optimal performance. Most current methods for imbalanced classification in the graph domain focus on node-level tasks. Some existing imbalanced learning methods, *e.g.*, resampling, re-weighting [35; 123], re-margining [19; 42], ensemble methods, can be directly applied to node classification models. However, due to the topological properties between nodes in graphs, such operations might cannot obtain the optimized result. Some works try to improve them for the tasks in graph field. HSCL [34] adopts the hybrid sampling method in contrastive learning to obtain discriminative representations. ReNode [24] and TAM [194] design loss function incorporating graph structure knowledge. Authors in [125] design a loss function FD-Loss for imbalanced node classification to enable the model to focus on instances that are helpful for the task. Boosting-GNN [190] adjusts training samples that are wrongly classified by setting higher weights on them. In [211], the label difference index (LDI) is defined to establish the relationship between class imbalance and misclassification, and one new loss function and four new methods are proposed based on LDI, namely, improved focal loss (iFL), Graph Re-sampling (GRS), Graph Re-weighting (GRW), Graph Metric Learning (GML), and Graph Bilateral-branch Network (GBBN). Inspired by the ensemble-based methods, GraphDIVE [78] learns multi-view graph representations to capture intrinsic diverse graph topological structure characteristics and combines multi-view experts to make a more accurate prediction.

Besides, researchers also try to augment training samples to alleviate the deviation to the majority class. For example, GraphSMOTE [270], GATSMOTE [131], GNN-CL [114], Mixup [217], GraphMixup [225], ImGAGN [179], GraphENS [169], GraphTU [60], SemiMixup [113] and Graph-DAO [227] generate synthetic training samples. KINC-

GCN [8] introduces a kernel propagation method to augment the node features and a self-optimizing cluster analysis and a graph reconstruction module are utilized to help the classification. GraphSR [273] augments the minority classes with most similar unlabelled nodes. The specific aggregation process of GNN can be improved to reduce bias. Effective-aggregation Graph Convolutional Network (EGCN) [208] modifies the aggregation operation by limiting the aggregation of inter-class edges from a local perspective while focusing more on the minority class from a global perspective based on the imbalance ratio. Minority-weighted graph neural network (mGNN) [209] calculates node membership values as weights during aggregation process. Balanced Topological Augmentation (BAT) [134] dynamically locates and rectifies nodes crucially influenced during message passing to reduce the errors and biases. Considering the special structural property of graph, [210] inserts buffer nodes into the graph to modulate the impact of majority classes to enhance minor class representation. Distance-wise Prototypical Graph Neural Network (DPGNN) [215] transfers knowledge from majority instances to minority samples by the learned class prototypes and metric learning. Long-Tail Experts for Graphs (LTE4G) [248] introduces a class prototype-based inference method to adjust predictions. However, these methods balance classes inside a graph and cannot be utilized directly to balance graphs in a graph set for graph-level classification.

To realize graph-level imbalanced classification, the existing imbalanced learning methods can be used together with the graph representation learning methods. However, they are not designed to tackle the class imbalance problem in graph classification, resulting in sub-optimal performance. There is one method dedicated to develop transferable patterns on the structure-abundant head graphs in the cause of enriching the structure-scarce tail graphs for more expressive graph representations [133], but it is size oriented. Graph-of-Graph Neural Networks (G^2GNN) [219] are designed and trained using augmented graphs to learn graph representations for imbalanced graph classification. Specifically, it utilizes kernel similarity to construct a graph of graphs (GoG) and derives extra supervision for minority nodes from their neighborhoods by implementing GoG propagation. In addition, the stochastic topological augmentation is used to improve the model generalibility. However, the construction of GoG relies on kernel similarity among graphs, which is computationally costly for large-scale graph datasets. Retrieval Augmented Hybrid Network (RAHNet) [147] enriches the tail classes by using a graph retrieval module to search for relevant graphs and introduces a category-centered supervised contrastive loss to obtain discriminative representations. CoMe [239] optimizes the representation learning and classifier learning jointly via tailored balanced con-

trastive learning along with individual-expert classifier training, and then fuses and distills the multiple expert networks from both global and local views for more excellent collaboration ability. Authors in [234] find that feature-oriented augmentation and structure-oriented augmentation have different influences on different types of graphs and propose a degree-oriented optional augmentation to improve sample diversity. To enhance the generalibility of the model, a size-oriented GoGs is constructed. TopoImb [268] focuses on problem whose imbalance exists in topological motifs and designs a topology extractor to explicitly model and dynamically update the detection of structure groups, following by a training modulator that assigns importance weights to under-represented training samples to automatically and adaptively regulate the training process.

3.3 Graph Anomaly Detection

Anomalies in graph can be node, subgraph, edge (relation) inside a graph or whole graph in a set of graphs. Node/graph embeddings learned by graph representation methods can be input into traditional anomaly detection methods to identify outliers, but their results are sub-optimal. Researchers try to design specific methods for graph anomaly detection tasks, of which works that explore node-level anomaly detection (NLAD) in a graph make up a significant proportion. Before the popularity of GNNs, works focus on shallow models [80; 224; 255; 173; 106; 126]. For example, ANOMALOUS [173] employs CUR decomposition to select representative samples and uses residual analysis to calculate the normality of each sample. Due to the impressive performance of GNNs in other tasks, more attention is paid to GNN-based NLAD methods. One type of solutions is to use reconstruction errors of graph autoencoder (GAE) as anomaly scores. DOMINANT [39] is the pioneered work, which constructs a GAE structure with a GCN encoder to compress the input to low-dimensional representations and a decoder to reconstruct both the topological structure and nodal attributes and uses the reconstruction errors of nodes to spot anomalous nodes. Based on DOMINANT, several improved works are proposed [139; 10; 52; 115; 174; 184]. For example, in [10], two GAEs are applied to the attributes and adjacency matrix respectively for outlier-aware node embeddings learning and further an adversarial learning method is proposed; SpecAE [115] leverages GAE to extract low-dimensional embeddings and carries out detection via density estimation; GAD-NR [184] focuses on neighborhood reconstruction. Apart from the reconstruction models, self-supervised model is also a popular choice [218; 130; 271; 38; 88; 228; 49]. For instance, CoLA [130] constructs contrastive learning model on contrastive instance pair

which pairs node and subgraph and the agreement within the pair is used to evaluate the abnormality of nodes; AEGIS [38] employs a graph neural layer to learn anomaly-aware node representations and a generative adversarial network (Ano-GAN) to detect anomalies among new data inductively. In addition to the above two types of methods, there are also other notable models [213; 99; 44; 267; 41; 170; 274; 111; 132; 177]. For example, OCGNN [213] and [99] use representations of nodes to train a hypersphere to discover abnormal nodes. GDN [41] leverages a deviation loss to enlarge the deviations of the anomaly scores of abnormal and normal nodes. ResGCN [170] ranks anomalies based on residual information. AAGNN [274] utilizes subtractive aggregation to represent each node as the deviation from its neighbors. Normal nodes with high confidence are employed as labels to learn a tailored hypersphere as the criterion of anomalies. GraphConsis [132] improves the aggregation process of GNNs.

For abnormal edge detection, a probability distribution of nodes is established conditioned on a specific node and its neighbors in [163] and the edge with low existence probability has a high possibility to be regarded as an anomaly. Deep models with GNNs for this task mainly focus on GAE. Based on GAE, AANE [47] designs a new loss, which is composed of anomaly aware loss and adjusted fitting loss, to guide the choice of significant abnormal edges during model training. Duan et al. [48] extend AANE by denoting anomaly weights of edges with continuous parameters in a data-driven way. RGSE [135] employs common-neighbor-based local structure features aligning with GAE for higher robustness.

There are also some other approaches whose goal are subgraph anomaly detection. [151] and [138] construct model based on residual matrix of graph while [193] applies SPCA to the modularity matrix of a graph. Zhao et al. [265] formulate the problem as maximizing a non-parametric scan statistic and then approximate it to a submodular maximization problem. SADE [171] applies traditional anomaly detection methods to the subgraph embeddings learned by specific method. Methods with GNNs are also mainly GAE-based or contrastive learning-based. For example, AS-GAE [264] introduces a supermodular graph scoring function module to assign reasonable anomaly scores to the subgraphs in the anomalous areas identified by a GAE. [81] proposes an improved unsupervised contrastive learning method based on CoLA, which comprehensively compares both the internal and external aspects of subgraphs and leverages a trained teacher model as prior knowledge to modify the sampling probabilities for selectively aggregating neighbor nodes.

However, the above methods can only identify local anomalies inside a graph. Dis-

criminating abnormal graphs among graph set directly instead of anomalous components inside a graph is also meaningful in numerous domains. One simple method is to combine existing graph representation learning methods with anomaly detection methods directly [266]. For better performance, some specifically designed methods are introduced in recent years. One research direction is to learn more discriminative graph representations. iGAD [251] uses an anomalous graph substructure-aware deep Random Walk Kernel module to embed the knowledge of abnormal substructures into graph representations. Yu et al. [244] combine GIN with Siamese network architecture for more comprehensive graph representations. CVTGAD [108] designs a simplified transformer module and a cross-view attention module for better graph representations. TUAF [245] learns triple representations from the triple-unit graph which is transformed from original graph and inputs it into an adaptive fusion readout to obtain a high-quality graph-level representation. HRGCN [107] models the interactions among all the nodes and considers both source-to-destination node categories and their edge categories for better graph representations. Authors in [43] propose a spectral GNN, namely RQGNN, by incorporating Rayleigh Quotient learning with Chebyshev Wavelet GNN to explore the spectral aspects of anomalous graphs. GLADformer [231] incorporates a spatial-domain Graph Transformer module and a spectral energy distribution deviations to enhance global perception and proposes a Spectral GNN to guide the extraction of local anomaly features.

GAE is also a powerful mechanism for GLAD. GLADC [140] constructs a GAE-based contrastive learning mechanism and detects anomalies by the reconstruction and input graph representations simultaneously instead of relying on the reconstruction solely. HimNet [160] establishes a hierarchical memory structure via a GAE to detect both locally and globally anomalous graphs.

Except these methods, there are also some other notable models. SIGNET [129] measures the abnormality of each graph based on cross-view mutual information. GmapAD [145] maps abnormal and normal graphs into a representation space with large distance by calculating the similarity between graphs and inter-graph candidate nodes. MssGAD [121] learns a separate multi-representations space to differentiate normal and abnormal graph representation space. GLADST [122] trains two student modules by normal and abnormal data respectively under the guide of a teacher module and calculates the anomaly score for a graph based on the representation error value of two student models.

3.4 Summary

Traditional non-deep methods for graph representation learning often require more memory and calculation resources and cannot deal with graphs with complex structure. GNNs can leverage complex graph structure better through the specific message passing mechanism to achieve better performance and have become the main tool for various down-stream graph tasks in recent years. Different message passing mechanisms and pooling schemes are proposed for more informative graph representations.

The focus of down-stream tasks in graph domain can be varying, *i.e.*, node-level, edge-level, subgraph-level and graph-level. Node-, edge- and subgraph-level tasks focus on components inside a graph, while graph-level tasks pay main attention to whole graphs in a graph set. Therefore, methods centered on local components cannot be applied to graph-level tasks directly. Specifically designed methods are required to solve graph-level tasks. Methods for different-level classification and anomaly detection tasks in graph domain are reviewed in detail for a more comprehensive understanding of these types of tasks.

Part II

Graph-level Problems and Solutions

CHAPTER 4

BALANCED GRAPH-LEVEL CLASSIFICATION WITH COLLECTIVE STRUCTURE KNOWLEDGE AUGMENTATION

4.1 Introduction

In the past few years, GNNs have been emerging as one of the most powerful and successful techniques for graph representation learning. Message passing neural networks constitute a prevalent category of GNN models, which learns node features and graph structure information through recursively aggregating current representations of node and its neighbors. Diverse aggregation strategies have been introduced, giving rise to various GNN backbones, such as GCN, GIN, and among others [226; 96; 70; 203; 232]. However, the expressive power of these message passing GNNs is upper bounded by 1-dimensional Weisfeiler-Leman (1-WL) tests [232; 155] that encode a node's color via recursively expanding the neighbors of the node to construct a rooted subtree for the node. As shown in Figure 4.1, such rooted subtrees are with limited expressiveness and might be the same for graphs with different structures, leading to failure in distinguishing these graphs. This presents a bottleneck for applying WL tests or message passing neural networks to many real-world graph application domains.

The failure of WL test is mainly due to the rooted subtree's limited capabilities in capturing different substructures that can appear in the graph. Since the message passing scheme of GNNs mimics the 1-WL algorithm, one intuition to enhance the expressive power of GNNs is to enrich the passing information, especially structural

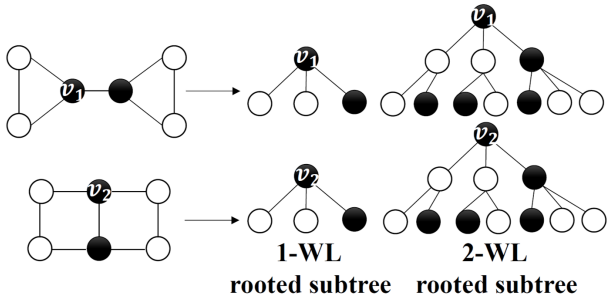


Figure 4.1: 1- and 2-WL tests fail to distinguish the two graphs as they obtain the same rooted subtree (node coloring).

knowledge, to help GNNs model diverse substructures. One popular approach to achieve this is data augmentation (DA) techniques [40]. One general framework in this line is to compute additional node features based on structural properties and attach them to original node features, such as DE [110], GSN [14], fast ID-GNN [243] and LAGNN [127]. Except extending node features, NestedGNN [257] and ID-GNN [243] compute and add node embeddings based on the local subgraph of each node. However, these methods only focus on local structure while many important global structure features are ignored. Also, GSN and fast ID-GNN often rely on a properly pre-defined substructure set to incorporate domain-specific inductive biases [257]. Further, these DA techniques are focused on augmenting the graph with some individual features, which are difficult to scale up to the incorporation of a diverse, large set of augmented features.

In this chapter, we propose a novel approach, namely *collective structure knowledge-augmented graph neural network (CoS-GNN)*, to leverage a variety of informative structural knowledge of graphs through DA for enhancing the expressiveness of existing GNNs. Instead of implicitly using structural information in other DA methods, we explicitly extract collective, domain-adaptive graph structural statistics at the graph and node levels as additional structure features. To fully leverage those augmented structural knowledge, we design a new message passing mechanism to respectively perform neighborhood aggregation on graph data using these augmented structure features and the original node attributes. Further, the new message passing can also model the interaction between the augmented features and the original node attributes. In doing so, our GNNs break down the upper bound of 1-WL tests and learn graph representations with significantly improved expressiveness (see the graph representations produced by CoS-GNN in Figure 4.2(b)(c) vs. those yielded by the original GCN).

In summary, our main contributions are as follows:

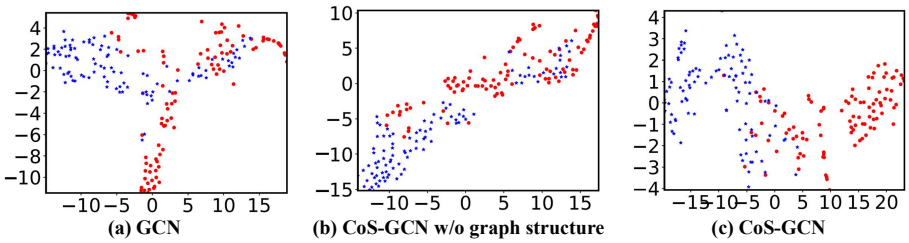


Figure 4.2: The red and blue points represent the visualized graph representations of two classes in REDDIT-BINARY yielded by (b) CoS-GCN with augmented node-level structural features and (c) CoS-GCN with augmented structural features at both node and graph levels are more class-separable than those produced by (a) the original GCN.

- We introduce a novel collective structure knowledge augmented GNN approach (CoS-GNN) that explicitly harnesses a diverse set of node and graph-level structural information for enhancing the expressiveness of GNN-based graph representations. The approach is generic and applicable to different GNN backbones.
- To effectively leverage the augmented structural features, a new message passing scheme is introduced in CoS-GNN, which simultaneously performs neighborhood aggregation on the augmented features and the original node attributes, enabling the learning of graph representations with significantly enriched structural knowledge.
- Comprehensive experiments on 12 graph datasets demonstrate that CoS-GNN (i) significantly outperforms competing methods in graph classification task; (ii) is more generalized to out-of-distribution graphs.

In the rest of this chapter, we introduce the proposed CoS-GNN and its two instantiations, namely CoS-GCN and CoS-GIN, in Section 4.2. Theoretical Analysis is presented in Section 4.3. Section 4.4 displays the experimental results. The summary of this chapter is provided in Section 4.5.

4.2 The Proposed CoS-GNN Model

4.2.1 Framework

The proposed CoS-GNN aggregates original and augmented structural features of single nodes and whole graph to learn expressive graph representations. The key intuition of CoS-GNN is to utilize various local (node) and global (graph) structural information

to enrich the original graph structural knowledge, through which we can learn a more informative and discriminative graph representation. The overall procedure of CoS-GNN is illustrated in Figure 4.3, which is composed of the following three major components:

- *Collective Graph Data Augmentation*. In this component, we generate a diverse set of specific structural features for each graph G (denoted by $\mathbf{x}_G^{gs}$) and each node v_i in G (denoted by $\mathbf{x}_i^{ns}$). These two types of features are added to augment each graph G from the structural knowledge perspective. It is a component that can be done offline.

- *Augmented Node-level Message Passing.* This component is designed to iteratively aggregate both the original and augmented node features, *i.e.*, $\mathbf{x}_i$ and $\mathbf{x}_i^{ns}$, to learn the node representation $\mathbf{h}_i$ with significantly enriched structural knowledge for each node v_i. To this end, a new message passing mechanism is introduced for this process. The node representations are then fed to a readout layer to gain the graph representation $\mathbf{h}_l$.

- *Graph-level Representation Fusion.* This component aims to synthesize the learned graph representation $\mathbf{h}_l$ and the pre-defined graph-level structural features $\mathbf{x}_G^{gs}$ via concatenation/fully-connected layers to obtain the final representation $\mathbf{h}_g$. $\mathbf{h}_g$ is then fed to a down-stream graph-level learning task.

4.2.2 Harnessing Collective Structure Knowledge for GNN

In this section, two instantiations of our CoS-GNN with the commonly-used GCN and GIN as the GNN backbone are introduced, namely CoS-GCN and CoS-GIN, respectively.

Collective Graph Data Augmentation

Graph is first augmented via computing some important node and graph statistics, which serve as additional node and graph features to complement the original node attributes. This component is shared by different model instantiations, and it can be performed before the model training.

Specifically, a number of widely-used and domain-adaptive node-level features are selected or generated, including the degree, triangle number, clique size, clique number, core number, cluster coefficient and square cluster coefficient, resulting seven new features in $\mathbf{x}_i^{ns}$ for each node v_i. The last two coefficient measures capture the tendency

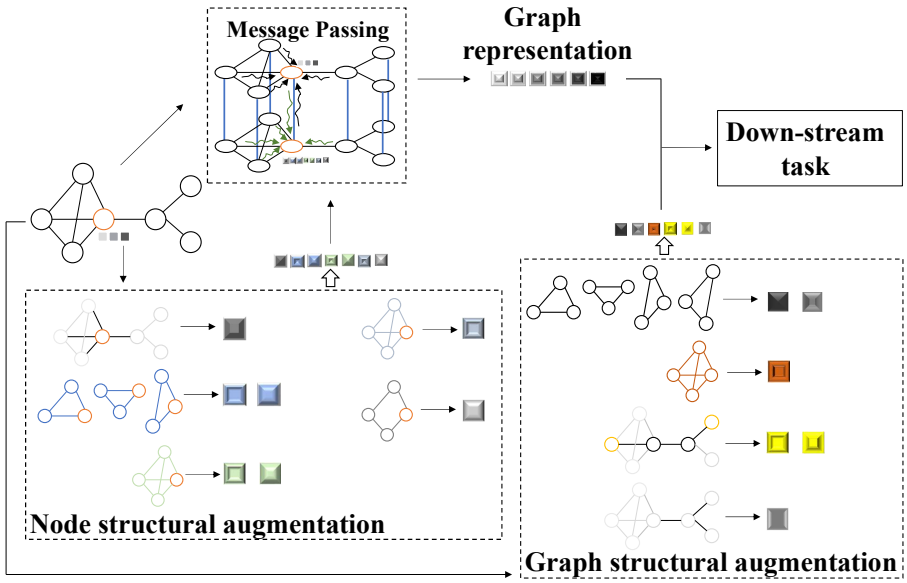


Figure 4.3: A schematic depiction of our CoS-GNN. Our CoS-GNN first calculates the specific node- and graph-level structural features. Then a new message passing mechanism is devised to utilize the original node attributes and the augmented node structural features to compute the graph representation, which is further combined with the graph structural augmentations for down-stream tasks.

of the node to form relatively dense communities, while other measures are to capture substructural information from varying scales. The detailed definitions of these features is as follows:

- **Degree.** The degree of a node/vertex is the number of edges that are incident to the node, which is an important and commonly-used node structure statistic.

- **Triangle.** Triangle is a simple and direct structure, and we count the number of triangles that use this node as a vertex.

- **Clique.** The clique is a substructure, in which every two distinct nodes are adjacent. We calculate the size of the maximal clique and the number of maximal cliques containing each given node.

- **K-core.** A k-core is defined as a maximal subgraph that is composed of nodes with degree k or more, the core number of a node is the largest value k of a k-core containing the given node. We collect the core number of each node as one of the augmented node-structural characteristics.

- **Quantized values.** Beyond the number, we also calculate the triangle/square clustering coefficient for each node, which is the fraction of possible triangles/squares through the given node that exist. This quantifies the tendency of nodes to form relatively dense network groups, *i.e.*, triangles or squares.

For graph-structural-level augmentation, a variety of important global statistics are utilized, including triangle number, clique size, the existence of bridge, average clustering coefficient, average global efficiency, and average local efficiency, to generate six graph-level structural features $\mathbf{x}_G^{gs}$ for each graph G. The three coefficients quantify the abundance of dense communities in the graph and the other statistics are the measurement of the node-to-node communication effectiveness within a graph. Detailed definition of each statistic is presented as follows:

- **Triangle.** We use the total number of triangles as one graph feature.
- **Clique.** We count the size of the largest clique in the graph as the second graph feature.
- **Bridge.** Another employed statistic is the existence of a bridge in the graph, which is an edge whose removal will cause the number of connected components of the graph to increase. The bridge is a specific characteristic of the graph.
- **Quantized values.** The average clustering coefficient for the graph is also included to measure the abundance of dense network groups in the graph. The efficiency of a pair of nodes is the multiplicative inverse of the shortest path distance between the nodes, and we calculate the average efficiency of all pairs of nodes in the graph, called average global efficiency, as one of the graph-structural statistics to measure the effectiveness of communication in the graph. The local efficiency of a node is defined as the average global efficiency of the subgraph induced by the neighbors of the node. We utilize the average local efficiency, which is the mean of local efficiencies of each node in the graph, as another statistic.

These collective statistics consider the configurations with different scales and complexities, which are normally adaptive to graphs from different domains.

Augmented Node-level Message Passing

Once the augmented node structural feature $\mathbf{x}^{ns}$ is obtained, we then aggregate the original feature $\mathbf{x}$ and the augmented features $\mathbf{x}^{ns}$ to learn the original node attributes

and their interaction with augmented structural knowledge of nodes. One straightforward solution that many previous methods do is to concatenate them directly and then apply GNN to perform the commonly-used neighborhood aggregation on nodes using the combined feature. This approach is easy-to-implement but fails to capture intricate interactions (e.g., higher-order and/or non-linear interactions) between the original node attributes and augmented features. To address this issue, a novel message passing mechanism for effectively capturing the diverse knowledge embedded in the two types of features and their interactions is proposed. The experiments also show that the proposed message passing mechanism outperforms the conventional message passing with the concatenated input (see results in Table 4.8).

To this end, a dual-graph structure that facilitates the modeling of the original node features, the modeling of the collective augmented node features, and the modeling of the interactions between these two types of features in each message passing step is constructed. In detail, given a graph G, we construct a new graph $\hat{G}$ with the same node and structure as the original graph but with the $\mathbf{x}^{ns}$ as its node attributes and link the corresponding nodes of G and $\hat{G}$. This results in an augmented graph with a dual-graph structure, $G^{'}$.

Message Passing in CoS-GCN

Next we perform message passing on the dual-graph structure $G^{'}$. When using GCN as the GNN backbone, the adjacent matrix $A^{'}$ of $G^{'}$ can be written as

$$A^{'} = \begin{pmatrix} A & I \\ I & A \end{pmatrix}, \tag{4.1}$$

and the degree matrix $D^{'}$ is

$$D^{'} = \begin{pmatrix} D+I & 0 \\ 0 & D+I \end{pmatrix}, \tag{4.2}$$

where A and D are the adjacent and degree matrices of G. We then convolute the node

features of $G^{'}$ by

$$
\begin{aligned}
H^{(l)} &= \rho\left(\tilde{D}^{'-\frac{1}{2}}\tilde{A}^{'}\tilde{D}^{'-\frac{1}{2}}\begin{pmatrix}H_n^{(l-1)}\\ H_{ns}^{(l-1)}\end{pmatrix}W^{(l)}\right)\\
&= \rho\left(\begin{pmatrix}\tilde{D}+I & 0\\ 0 & \tilde{D}+I\end{pmatrix}^{-\frac{1}{2}}\begin{pmatrix}\tilde{A} & I\\ I & \tilde{A}\end{pmatrix}\begin{pmatrix}\tilde{D}+I & 0\\ 0 & \tilde{D}+I\end{pmatrix}^{-\frac{1}{2}}\begin{pmatrix}H_n^{(l-1)}W^{(l)}\\ H_{ns}^{(l-1)}W^{(l)}\end{pmatrix}\right)\\
&= \rho\left(\begin{pmatrix}(\tilde{D}+I)^{-\frac{1}{2}}\tilde{A}(\tilde{D}+I)^{-\frac{1}{2}} & \tilde{D}+I\\ \tilde{D}+I & (\tilde{D}+I)^{-\frac{1}{2}}\tilde{A}(\tilde{D}+I)^{-\frac{1}{2}}\end{pmatrix}\begin{pmatrix}H_n^{(l-1)}W^{(l)}\\ H_{ns}^{(l-1)}W^{(l)}\end{pmatrix}\right)\\
&= \rho\begin{pmatrix}(\tilde{D}+I)^{-\frac{1}{2}}\tilde{A}(\tilde{D}+I)^{-\frac{1}{2}}H_n^{(l-1)}W^{(l)}+(\tilde{D}+I)H_{ns}^{(l-1)}W^{(l)}\\ (\tilde{D}+I)^{-\frac{1}{2}}\tilde{A}(\tilde{D}+I)^{-\frac{1}{2}}H_{ns}^{(l-1)}W^{(l)}+(\tilde{D}+I)H_n^{(l-1)}W^{(l)}\end{pmatrix},
\end{aligned}
\tag{4.3}
$$

where $\tilde{A}=A+I$, $\tilde{D}=D+I$ and $\tilde{D}^{'}=D^{'}+I$. $H_n^{(l-1)}$ and $H_{ns}^{(l-1)}$ is the node representation matrices of G and $\hat{G}$ after the $(l-1)_{th}$ convolutional layer. The feature input of the 0_{th} layer is node feature matrices X and X^{ns}, which stack $\mathbf{x}_i$ and $\mathbf{x}_i^{ns}$ ($v_i \in G$) across all graph nodes, respectively. $W^{(l)}$ is the parameter matrix of the l_{th} convolutional layer.

Since the original node features and augmented node structural features can be very different, two different convolutional filters (i.e., with different convolutional weights) is employed to learn their knowledge as follows:

$$
H^{(l)} = \begin{pmatrix}H_n^{(l)}\\ H_{ns}^{(l)}\end{pmatrix} \approx \rho\begin{pmatrix}(\tilde{D}+I)^{-\frac{1}{2}}\tilde{A}(\tilde{D}+I)^{-\frac{1}{2}}H_n^{(l-1)}W_n^{(l)}+(\tilde{D}+I)H_{ns}^{(l-1)}W_{ns}^{(l)}\\ (\tilde{D}+I)^{-\frac{1}{2}}\tilde{A}(\tilde{D}+I)^{-\frac{1}{2}}H_{ns}^{(l-1)}W_{ns}^{(l)}+(\tilde{D}+I)H_n^{(l-1)}W_n^{(l)}\end{pmatrix},
\tag{4.4}
$$

where $W_n^{(l)}$ and $W_{ns}^{(l)}$ are the parameter matrices of l_{th} layer for two types of features respectively, and $H_n^{(l)}$ and $H_{ns}^{(l)}$ are the node representation matrices of G and $\hat{G}$ after current l_{th} message passing layer.

After L message-passing layers, the node representations of two graphs G and $\hat{G}$ in each layer are aggregated to obtain the final node representation matrix as follows:

$$
H = \mathrm{AGGATE}_n(H_n^{(1)},\cdots,H_n^{(L)},H_{ns}^{(1)},\cdots,H_{ns}^{(L)}),
\tag{4.5}
$$

where $\mathrm{AGGATE}_n(\cdot)$ is an aggregate function, and concatenation is used in our experiments; H denotes the representation matrix that encapsulates the representation of all individual nodes. Then a readout function is applied to obtain the learned graph representation $\mathbf{h}_l$.

Message Passing in CoS-GIN

The framework can also be extended to other GNN backbones. Here we now present how the proposed message passing method can be adopted to the case using GIN as our

backbone. To this end, the GIN-based message passing is re-defined as follows:

$$\begin{aligned}\mathbf{h}_{v_i,n}^{(l)} &= \mathrm{MLP}_n^{(l)}\left((1+\epsilon^{(l)})\mathbf{h}_{v_i,n}^{(l-1)} + \sum_{v_j \in \mathcal{N}(v_i)} \mathbf{h}_{v_j,n}^{(l-1)} + \mathbf{h}_{v_i,ns}^{(l-1)}\right),\\ \mathbf{h}_{v_i,ns}^{(l)} &= \mathrm{MLP}_{ns}^{(l)}\left((1+\epsilon^{(l)})\mathbf{h}_{v_i,ns}^{(l-1)} + \sum_{v_j \in \mathcal{N}(v_i)} \mathbf{h}_{v_j,ns}^{(l-1)} + \mathbf{h}_{v_i,n}^{(l-1)}\right),\end{aligned} \tag{4.6}$$

Then the obtained representations are combined via summation. In detail,

$$\begin{aligned}\mathbf{h}_n &= \sum_l \mathrm{FC}_n^{(l)}(\mathrm{READOUT}(H_n^{(l)})),\\ \mathbf{h}_{ns} &= \sum_l \mathrm{FC}_{ns}^{(l)}(\mathrm{READOUT}(H_{ns}^{(l)})),\end{aligned} \tag{4.7}$$

where $\mathrm{FC}_n^{(l)}(\cdot)$ and $\mathrm{FC}_{ns}^{(l)}(\cdot)$ are fully-connected layers in the l_{th} layer. The learned graph representation $\mathbf{h}_l$ is gained through adding them together:

$$\mathbf{h}_l = \mathbf{h}_n + \mathbf{h}_{ns}. \tag{4.8}$$

The key insight of the message passing mechanism in CoS-GIN is analogous to that in CoS-GCN, but they are derived at different representation levels: matrix of node representations in CoS-GCN vs. vectorized node representations in CoS-GIN, which is mainly done for presentation brevity.

Graph-level Representation Fusion

After gaining the learned graph representation $\mathbf{h}_l$, MLPs are then employed to synthesize it, together with the augmented graph-structural feature $\mathbf{x}^{gs}$, to learn the final graph representations. In detail, $\mathbf{h}_l$ and $\mathbf{x}^{gs}$ are input into the two different MLPs as:

$$\mathbf{h}_l^{MLP} = \mathrm{MLP}^l(\mathbf{h}_l), \mathbf{h}_{gs}^{MLP} = \mathrm{MLP}^{gs}(\mathbf{x}^{gs}). \tag{4.9}$$

The information learned is then integrated to gain the final graph representation:

$$\mathbf{h}_g = \mathrm{AGGATE}_g(\mathbf{h}_l^{MLP}, \mathbf{h}_{gs}^{MLP}), \tag{4.10}$$

where $\mathrm{AGGATE}_g(\cdot)$ is the aggregation function and concatenation is used here. Then the graph representation can be used for any down-stream tasks. Algorithm 1 presents the procedure of CoS-GCN to calculate graph representations, which can be later input to any down-stream tasks.

Algorithm 1 Graph representation learning via CoS-GCN

Input: Graph set $\mathcal{G} = \{G_i\}_i$, two GNNs with parameter set $\{W_n^{(1)},...,W_n^{(L)}\}$ and $\{W_{ns}^{(1)},...,W_{ns}^{(L)}\}$, two MLP functions $\text{MLP}^l(\cdot)$ and $\text{MLP}^{gs}(\cdot)$

Output: Graph representation $\mathbf{h}_g$ for $G \in \mathcal{G}$

1: Augment node and graph structural knowledge to obtain X^{ns} and $\mathbf{x}^{gs}$ for each $G \in \mathcal{G}$
2: **for** G in $\mathcal{G}$ **do**
3: Compute $H^{(l)}, l \in \{1,\cdots,L\}$ with Eq. (4.4)
4: Aggregate $H^{(l)}, l \in \{1,\cdots,L\}$ with Eq. (4.5) to obtain H
5: Readout H to obtain $\mathbf{h}_l$
6: Input $\mathbf{h}_l$ and $\mathbf{x}^{gs}$ into MLP^l and MLP^{gs} respectively to gain $\mathbf{h}_l^{MLP}$ and $\mathbf{h}_{gs}^{MLP}$
7: Aggregate $\mathbf{h}_l^{MLP}$ and $\mathbf{h}_{gs}^{MLP}$ to obtain the final representation $\mathbf{h}_g$ for G
8: **end for**
9: **return** Graph representation $\mathbf{h}_g$ for $G \in \mathcal{G}$

4.3 Theoretical Analysis

4.3.1 Expressive Power of CoS-GNN

This section discusses the expressive power of CoS-GNN. When comparing the expressiveness of GNN models, we can define that:

Definition 6. *For any two GNN models: A and B, model A is said to be more expressive than model B, if and only if 1) model A can distinguish all samples that model B can distinguish, and 2) there exists samples which can be distinguished by model A but not by model B.*

To measure the expressive power of GNNs, the Weisfeiler-Lehman (WL) graph isomorphism test is commonly used, which is a family of algorithms (k-WL, k-FWL) used to test graph isomorphism [148; 66]. Two graphs G_1 and G_2 are called isomorphic if there exists an edge and color preserving bijection $\phi : \mathcal{V}_1 \to \mathcal{V}_2$. Next we show the strong expressive power of our model CoS-GNN from the WL-test perspective:

Theorem 1. *CoS-GNN is not less expressive than 1-WL and 2-WL tests.*

Proof. We first consider the comparison with 1-WL test. This equals to prove such statement: If CoS-GNN deems that two graphs are isomorphic, then 1-WL test will also deem them isomorphic. If after k iterations, the CoS-GNN regards two graphs G_1 and G_2 are isomorphic, we have $\mathbf{h}_{1,g}^{(k)} = \mathbf{h}_{2,g}^{(k)}$. Assuming that the AGGATE_g is injective, we can obtain that $\mathbf{h}_{1,l}^{MLP(k)} = \mathbf{h}_{2,l}^{MLP(k)}$ and $\mathbf{h}_{1,gs}^{MLP(k)} = \mathbf{h}_{2,gs}^{MLP(k)}$, followed by $\mathbf{h}_{1,l}^{(k)} = \mathbf{h}_{2,l}^{(k)}$ and

$\mathbf{x}_1^{gs} = \mathbf{x}_2^{gs}$. Thus we have $H_1^{(i)} = H_2^{(i)}$ and then $\mathbf{h}_{v,n}^{(i)} = \mathbf{h}_{u,n}^{(i)}$ and $\mathbf{h}_{v,ns}^{(i)} = \mathbf{h}_{u,ns}^{(i)}$ for $v \in \mathcal{V}_{G_1}$, $u \in \mathcal{V}_{G_2}$ and $i = 1, ..., k$ when the AGGATE$_n$ is injective.

What is needed to prove next is that the color extracted by 1-WL for node v and u is same, *i.e.*, $c_v^{(k)} = c_u^{(k)}$. We use the induction as [14] to demonstrate this. For $i = 0$, since the initial node features are the same for both CoS-GNN and 1-WL, we can get $c_v^{(0)} = c_u^{(0)}$ when $\mathbf{h}_{v,n}^{(0)} = \mathbf{h}_{u,n}^{(0)}$. Suppose $\mathbf{h}_{v,n}^{(j)} = \mathbf{h}_{u,n}^{(j)}, \mathbf{h}_{v,ns}^{(j)} = \mathbf{h}_{u,ns}^{(j)} \Rightarrow c_v^{(j)} = c_u^{(j)}$ holds for $j = 1, \cdots, k-1$, we later need to prove that it holds for $j = k$. Since each node representation, including $\mathbf{h}_{v,n}^{(j)}$ and $\mathbf{h}_{v,ns}^{(j)}$, is calculated by a COM function, if COM is injective, we have $\mathbf{h}_{v,n}^{(k-1)} = \mathbf{h}_{u,n}^{(k-1)}$, $\mathbf{h}_{v,ns}^{(k-1)} = \mathbf{h}_{u,ns}^{(k-1)}$, AGGATE$(\{\mathbf{h}_{q,n}^{(k-1)} | q \in \mathcal{N}_v\})$ = AGGATE$(\{\mathbf{h}_{p,n}^{(k-1)} | p \in \mathcal{N}_u\})$ and AGGATE$(\{\mathbf{h}_{q,ns}^{(k-1)} | q \in \mathcal{N}_v\})$ = AGGATE$(\{\mathbf{h}_{p,ns}^{(k-1)} | p \in \mathcal{N}_u\})$ when $\mathbf{h}_{v,n}^{(k)} = \mathbf{h}_{u,n}^{(k)}$ and $\mathbf{h}_{v,ns}^{(k)} = \mathbf{h}_{u,ns}^{(k)}$. According to Lemma 5 from [232], there exists an injective function. When AGGATE is injective, we have $\mathbf{h}_{q,n}^{(k-1)} = \mathbf{h}_{p,n}^{(k-1)}$ and $\mathbf{h}_{q,ns}^{(k-1)} = \mathbf{h}_{p,ns}^{(k-1)}$, which lead to $c_q^{(k-1)} = c_p^{(k-1)}$ for $q \in \mathcal{N}_v$ and $p \in \mathcal{N}_u$. Since we have $c_u^{(k-1)} = c_v^{(k-1)}$ according to the induction hypothesis, we can get $c_u^{(k)} = c_v^{(k)}$. Therefore, the 1-WL test regards two graphs isomorphic if the CoS-GNN regards them isomorphic.

Since 1-WL and 2-WL test have equivalent discrimination power [148; 257], CoS-GNN is also at least as expressive as 2-WL test. ■

The theorem states that CoS-GNN is at least as expressive as 1-WL and 2-WL tests. Some graphs that 1-WL and 2-WL tests cannot distinguish can be identified by the proposed CoS-GNN. For example, 1-WL and 2-WL fail to distinguish the two graphs in Figure 4.1, whereas CoS-GNN can easily differentiate them with the augmented features. Thus, our CoS-GNN can often learn more expressive representations than popular GNNs since they are mainly based on the 1-WL test, when handling complex graph datasets. For example, it has been shown in [29] that MPNNs cannot perform induced-subgraph-count of any connected pattern consisting of 3 or more nodes. For graphs with subgraphs that MPNNs cannot learn to count, there would be some pairs of graphs with different number of such uncounted subgraphs that are regarded as isomorphic by MPNNs. On the other hand, CoS-GNN can discriminate these graphs through including structural features that differentiate these subgraphs. As shown in Figure 4.1, the two graphs cannot be distinguished by MPNNs, but they can be differentiated by the triangle counting for both nodes and graphs, and the existence of bridge in the graphs as well.

When compared with higher-order WL tests, it can also be observed that our CoS-GNN can distinguish graphs that 2-FWL test (which is equivalent to 3-WL test [148]) fails to identify, meaning that 3-WL test is not more expressive than our CoS-GNN. For example, [4] and [14] have shown that the 2-FWL test fails to distinguish the well-known

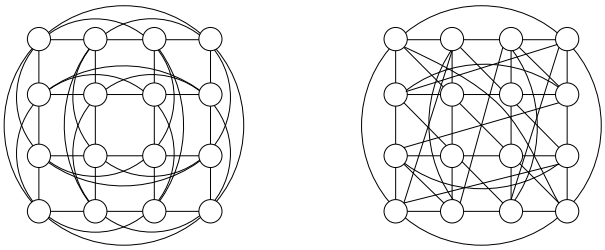


Figure 4.4: The strongly regular Rook's 4×4 graph (left) and Shrikhande graph (right) [14; 4]. The 3-WL/2-FWL test is not able to deem them as non-isomorphic. Rook's 4×4 graph possesses 4-cliques while the Shrikhande graph features 5-rings, which does not present in Rook's.

Rook's 4×4 and Shrikhande graphs, as illustrated in Figure 4.4. However, the clique features incorporated into our CoS-GNN model help effectively discriminate these two graphs.

4.3.2 Time Complexity Analysis

In this section, the time complexity of CoS-GNN is analyzed. The computation cost mostly concentrates on the feature extraction stage and the message passing stage. Let N and M be the number of nodes and edges in the graph respectively, in the feature learning phase, the degree and triangle counting cost are $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ time respectively. The complexity of clique and core finding are respectively bounded by $\mathcal{O}(N * 3^N)$ and $\mathcal{O}(N+M)$. The computation of triangle and square clustering coefficient is $\mathcal{O}(N^2)$. The bridge finding needs $\mathcal{O}(N+M)$ time. The average clustering coefficient, average global and local efficiency require $\mathcal{O}(N^2)$, $\mathcal{O}(N^3)$ and $\mathcal{O}(N^4)$ respectively. Therefore, the feature extraction stage requires $\mathcal{O}(N^4+N*3^N+M)$ time. As for the message passing stage, the time complexity of CoS-GNN equals to the corresponding vanilla GNN. Thus, the total time complexity of CoS-GNN is $\mathcal{O}(N^4+N*3^N+M)+\mathcal{O}_{GNN}$.

4.4 Experiments and Results

4.4.1 Competing Methods

The proposed CoS-GNN is compared with 13 state-of-the-art (SOTA) methods:

- **Graph kernels.** Two graph kernels, *i.e.*, **Weisfeiler-lehman subtree kernel (WL)** [188] and **Propagation graph kernels (PK)** [156] are used as baselines.

- **Basic graph neural networks.** Four popular networks, *i.e.*, **GCN** [96], **GraphSAGE (shortened by SAGE)** [70], **GAT** [203] and **GIN** [232], are considered as the network baselines.
- **GNN-based augmentation methods.** We also compare CoS-GNN with several augmentation models that are built based on GNNs, including $\mathscr{G}$**-mixup** [71], **Dummy** [128], **DropGNN** [167], **rGIN** [186], **NestedGNN** [257], **LAGNN** [127], and **GSN** [14].

The mean results and standard deviation based on 10-fold cross-validation is reported for all datasets.

4.4.2 Parameter Settings

The following parameters are set by default for CoS-GCN and its competing methods, including WL, PK, GAT, SAGE and GCN, on all 12 datasets: the learning rate is 0.001, the batch size is set to 512, the number of network layers is 3, the hidden layer dimension of network is 256, the classifier is a 3-layer MLP, pooling operation is max pooling, and the number of epochs is 1,000. The iteration number of WL is 3. For GIN and CoS-GIN, the learning rate is chosen from $\{0.01, 0.001, 0.0005, 0.0001\}$, the batch size is selected from $\{32, 64, 128, 256\}$, hidden layer dimension is ranged in $\{16, 64, 128, 256\}$ and the readout operation is either meanpooling or maxpooling. For other baselines, their public codes are run with their recommended settings.

4.4.3 Comparison to SOTA Models

The graph classification accuracy results of CoS-GNN models (including CoS-GCN and CoS-GIN) and 13 SOTA competing methods are reported in Table 4.1, where the GNN backbone used in $\mathscr{G}$-mixup, Dummy and DropGNN is all GIN due to its better performance; the results of $\mathscr{G}$-mixup on the IMDB and REDDIT datasets are taken from [71]; the result of Dummy on DD, NCI1 and NCI109 are from [128]; the results of NestedGCN and NestedGIN on DD, MUTAG and ENZYMES are from [257]; and '-' means the results are not reported in the original papers.

Table 4.1: Accuracy (mean±std) of CoS-GNN and SOTA competing methods for graph classification on 12 real-world datasets. The best and second performance per dataset is boldfaced and underlined respectively. The following acronyms, PROTEINS_full (PROTS_full), IMDB-BINARY (I-BINARY), IMDB-MULTI (I-MULTI), REDDIT-BINARY (R-BINARY) and REDDIT-MULTI-5K (R-MULTI), are used. 'Rank' indicates the average performance ranking of a model across all datasets: a smaller rank value indicates a better overall performance.

Method	BZR	COX2	DD	I-BINARY	I-MULTI	MUTAG	NCI1	NCI109	PROTS_full	R-BINARY	R-MULTI	ENZYMES	Rank	p-value
WL	0.790±0.045	0.760±0.043	0.747±0.023	0.727±0.035	0.503±0.021	0.797±0.072	0.643±0.025	0.647±0.021	0.737±0.028	0.611±0.039	0.382±0.019	0.338±0.074	13.5	0.0005
PK	0.818±0.026	0.790±0.015	0.742±0.026	0.729±0.050	0.488±0.038	0.697±0.054	0.634±0.017	0.620±0.036	0.734±0.026	0.635±0.049	0.411±0.024	0.392±0.036	14.3	0.0005
GCN	0.840±0.028	0.811±0.043	0.756±0.033	0.724±0.036	0.498±0.024	0.744±0.102	0.785±0.016	0.769±0.026	0.749±0.028	0.900±0.024	0.533±0.013	0.450±0.068	8.8	0.0005
GAT	0.837±0.036	0.790±0.065	0.773±0.027	0.704±0.044	0.496±0.035	0.738±0.105	0.772±0.015	0.765±0.027	0.748±0.035	0.851±0.036	0.502±0.025	0.422±0.069	10.8	0.0010
SAGE	0.842±0.055	0.820±0.055	0.771±0.031	0.733±0.046	0.495±0.024	0.766±0.059	0.795±0.018	0.780±0.027	0.748±0.034	0.878±0.029	0.508±0.017	0.395±0.079	8.3	0.0005
GIN	0.835±0.033	0.805±0.039	0.750±0.048	0.731±0.047	0.502±0.031	0.866±0.078	0.790±0.023	0.795±0.012	0.738±0.039	0.891±0.016	0.557±0.021	0.550±0.091	7.3	0.0005
$\mathcal{G}$**–mixup**	0.832±0.042	0.805±0.043	–	0.719±0.030	0.505±0.015	0.824±0.093	0.778±0.023	0.752±0.039	0.705±0.054	**0.929**±0.009	0.555±0.005	0.487±0.054	8.9	0.0049
Dummy	0.828±0.069	0.805±0.048	0.777±0.035	–	–	0.829±0.105	0.752±0.015	0.738±0.025	**0.770**±0.029	–	–	0.500±0.101	8.4	0.0034
DropGNN	0.845±0.030	0.807±0.055	–	0.741±0.027	0.502±0.023	0.834±0.095	0.809±0.014	0.810±0.013	0.747±0.032	–	–	0.588±0.053	5.7	0.0005
rGIN	0.827±0.046	0.805±0.051	0.698±0.020	0.746±0.022	0.512±0.026	0.851±0.052	0.808±0.019	0.801±0.016	0.757±0.024	0.897±0.020	0.549±0.023	0.663±0.073	6.0	0.0098
NestedGCN	0.808±0.033	0.782±0.009	0.763±0.038	0.733±0.052	0.499±0.037	0.829±0.011	0.720±0.080	0.708±0.076	0.730±0.017	–	–	0.312±0.067	12.8	0.0005
NestedGIN	0.832±0.082	0.790±0.066	**0.778**±0.039	0.745±0.064	0.510±0.023	0.879±0.082	0.777±0.017	0.779±0.026	0.738±0.035	–	–	0.290±0.080	7.8	0.0015
GSN-v	0.825±0.036	0.801±0.040	0.702±0.033	**0.760**±0.047	0.509±0.031	0.861±0.097	0.808±0.018	0.804±0.020	0.736±0.036	0.819±0.028	0.521±0.026	**0.687**±0.043	7.6	0.0190
LAGCN	0.837±0.039	0.807±0.064	0.752±0.033	0.727±0.041	0.499±0.022	0.761±0.087	0.752±0.027	0.713±0.031	0.755±0.035	0.884±0.016	0.513±0.026	0.590±0.047	9.3	0.0010
LAGIN	0.820±0.035	**0.831**±0.028	0.766±0.037	0.731±0.055	0.491±0.032	0.872±0.049	0.763±0.016	0.740±0.032	0.760±0.032	0.894±0.020	0.553±0.022	0.645±0.042	7.6	0.0156
CoS-GCN	0.832±0.036	0.824±0.055	**0.778**±0.032	0.754±0.052	0.503±0.019	0.878±0.059	0.816±0.013	0.801±0.021	0.757±0.028	0.917±0.022	0.554±0.028	0.533±0.064	3.8	–
CoS-GIN	**0.850**±0.043	0.822±0.094	0.773±0.038	0.753±0.033	**0.517**±0.023	**0.883**±0.066	**0.823**±0.026	**0.812**±0.013	0.751±0.019	0.906±0.018	**0.559**±0.021	0.653±0.036	**2.3**	–

It is clear that CoS-GIN and CoS-GCN achieve the best or second-best performance on most of the datasets and the two top-ranked methods among all methods. Specifically, CoS-GCN improves GCN by 0.8%, 2.2%, 3.0%, 3.1%, 3.2% and 8.3% for PROTEINS_full, DD, IMDB-BINARY, NCI1, NCI109 and ENZYMES respectively, while the improvements brought by CoS-GIN over GIN are 1.5%, 1.7%, 1.7%, 2.2%, 2.3%, 3.3% and 10.3% for BZR, COX2, IMDB-BINARY, DD, NCI1, NCI109 and ENZYMES respectively. These large performance advancement reveals that the structural information in these dataset is specific and the feature augmentation and message passing process in our CoS-GNN make full use of these structural information to improve its performance. When compared with other augmentation methods, our models can also perform better than the SOTA models on most datasets (*i.e.*, NCI109 (0.2%), REDDIT-MULTI (0.2%), MUTAG (0.4%), BZR (0.5%), IMDB-MULTI (0.5%) and NCI1 (1.4%)) and ranks top among all the competitors on overall performance. We also perform a paired Wilcoxon signed rank test to examine the significance of CoS-GNN against each of the competing methods across the 12 datasets. As shown by the p-values in Table 4.1, our CoS-GIN significantly outperforms GSN-v and LAGIN at the 95% confidence level and exceeds other competitors at the 99% confidence level. These results indicate that our collective node and graph structural knowledge augmented GNNs can learn more important graph structure information for graph classification. Besides, on individual datasets, CoS-GNN can gain 2%-11% accuracy improvement maximally on specific datasets when compared to the best-performing competing methods NestedGNN, GSN-v and LAGNN (for example, 5% enhancement of NestedGIN on NCI1, 9% improvement of GSN on REDDIT-BINARY). This means that the domain-adaptive graph structural knowledge in CoS-GNN can provide more generalized information to improve the model performance across different datasets while NestedGNN, GSN-v and LAGNN only consider the local structural information, which limits their performance. In summary, compared to each SOTA method, CoS-GNN may only have limited improvements on a few individual datasets, but the improvement on a set of datasets is substantial, and its improvement is significant across the 12 datasets used.

We report the AUPRC results of CoS-GNN and the competing methods on binary classification tasks in Table 4.2. Considering the limited performance of WL, PK and LAGCN, we omit their results. As can be seen in Table 4.2, although our CoS-GNN is not always the best model on every dataset, our CoS-GIN and CoS-GCN still achieve the top two performance on overall datasets, which is consistent to the results in Table 4.1 and further demonstrates the excellent ability of our CoS-GNN.

Table 4.2: AUPRC (mean±std) of CoS-GNN and SOTA competing methods for graph classification on 9 real-world binary classification datasets. The best and second performance per dataset is boldfaced and underlined respectively. The following acronyms, PROTEINS_full (PROTS_full), IMDB-BINARY (I-BINARY), IMDB-MULTI (I-MULTI), REDDIT-BINARY (R-BINARY) and REDDIT-MULTI-5K (R-MULTI), are used. 'Rank' indicates the average performance ranking of a model across all datasets: a smaller rank value indicates a better overall performance.

Method	BZR	COX2	DD	I-BINARY	MUTAG	NCI1	NCI109	PROTS_full	R-BINARY	Rank
GCN	0.674±0.124	0.541±0.152	0.760±0.055	0.831±0.038	0.783±0.113	0.842±0.023	0.824±0.0198	0.763±0.038	0.956±0.011	7.3
GAT	0.624±0.156	0.520±0.137	0.753±0.058	0.814±0.042	0.870±0.089	0.841±0.020	0.811±0.014	**0.772**±0.038	0.937±0.022	9.6
SAGE	0.661±0.170	0.565±0.144	**0.803**±0.038	0.813±0.042	0.899±0.0569	0.852±0.029	0.823±0.018	0.760±0.041	0.944±0.015	6.8
GIN	0.619±0.123	0.556±0.115	0.770±0.037	0.828±0.032	0.963±0.026	0.857±0.028	0.848±0.021	0.699±0.058	0.940±0.034	7.3
$\mathscr{G}-$**mixup**	0.691±0.130	0.439±0.160	–	0.829±0.033	0.945±0.040	0.833±0.021	0.796±0.043	0.688±0.078	–	10.6
Dummy	0.570±0.067	0.554±0.175	0.800±0.040	–	0.915±0.074	0.808±0.031	0.786±0.027	**0.772**±0.040	–	9.6
DropGNN	0.675±0.085	0.555±0.172	0.690±0.044	0.837±0.024	0.962±0.027	**0.879**±0.020	0.863±0.021	0.708±0.051	0.938±0.029	6.1
rGIN	0.576±0.096	0.573±0.147	0.704±0.026	0.821±0.022	0.968±0.025	0.868±0.020	0.862±0.010	0.706±0.068	0.928±0.039	7.7
NestedGIN	0.587±0.233	0.563±0.180	0.784±0.049	**0.840**±0.039	0.962±0.029	0.851±0.020	0.842±0.025	0.745±0.050	–	6.9
GSN-v	0.694±0.143	0.528±0.150	0.699±0.050	0.828±0.073	0.969±0.031	0.857±0.0215	0.847±0.021	0.670±0.132	0.876±0.037	8.2
LAGCN	0.652±0.077	0.560±0.113	0.757±0.045	0.826±0.038	0.912±0.034	0.800±0.020	0.755±0.026	0.747±0.040	0.961±0.025	8.8
LAGIN	0.552±0.089	0.539±0.130	0.785±0.066	0.833±0.025	**0.979**±0.012	0.838±0.024	0.811±0.044	0.723±0.067	0.955±0.020	7.7
CoS-GCN	0.594±0.102	**0.575**±0.133	0.795±0.052	0.837±0.046	0.974±0.020	**0.879**±0.014	0.861±0.027	0.760±0.039	0.960±0.022	**3.4**
CoS-GIN	**0.724**±0.147	0.562±0.160	0.777±0.046	0.829±0.034	0.977±0.019	0.872±0.015	**0.880**±0.021	0.725±0.035	**0.963**±0.011	3.7

We also compare our CoS-GNN with vanilla GNNs on Open Graph Benchmark (OGB) datasets – ogbg-molhiv and ogbg-molpcba in Table 4.3. Our CoS-GNN achieves better performance than corresponding vanilla GNN in most situations, indicating the positive contribution of the augmented features. The performance of CoS-GIN is a bit worse than that of GIN on ogbg-molpcba, which might be because that although our augmented features are useful, which is demonstrated by the improvement of CoS-GCN compared with GCN, GIN has also learned enough useful structural information and the augmentation operation in our CoS-GIN does not provide extra discriminative information.

Table 4.3: Results (mean±std) of CoS-GNN and corresponding vanilla GNN on OGB datasets – ogbg-molhiv and ogbg-molpcba. The best performance per dataset is boldfaced.

	ogbg-molhiv	ogbg-molpcba
Model	**AUC**	**AP**
GCN	0.7626±0.0098	0.1753±0.0023
CoS-GCN	**0.7662 ± 0.0165**	**0.2045 ± 0.0034**
GIN	0.7825±0.0077	**0.2288 ± 0.0027**
CoS-GIN	**0.7912 ± 0.0068**	0.2249±0.0034

We further calculate the training and inference time of our CoS-GNN and its competitors to demonstrate the efficiency of the CoS-GNN. We use the same GIN structure in all models. The results are reported in Table 4.4. We can see that our CoS-GIN is a little more costly than simple augmentation operation with conventional GIN module,

Table 4.4: Training and inference time of augmentation methods in graph classification task. The following acronyms, PROTEINS_full (P_full), IMDB-BINARY (I-B), IMDB-MULTI (I-M), REDDIT-BINARY (R-B) and REDDIT-MULTI-5K (R-M), are used. All methods are with GIN as backbone. Each result is the time on the whole dataset.

Stage	Method	BZR	COX2	DD	I-B	I-M	MUTAG	NCI1	NCI109	P_full	R-B	R-M	ENZYMES
Training	$\mathscr{G}$-mixup	0.028	0.033	-	0.055	0.068	0.011	0.255	0.203	0.068	-	-	0.037
	Dummy	0.019	0.022	0.236	-	-	0.010	0.156	0.153	0.047	-	-	0.023
	DropGNN	0.338	0.242	-	0.201	0.420	0.054	2.258	2.311	1.148	-	-	0.490
	rGIN	0.048	0.056	0.359	0.104	0.123	0.021	0.405	0.374	0.117	0.825	2.154	0.059
	NestedGIN	0.186	0.206	9.385	0.809	0.698	0.055	1.373	1.414	0.684	-	-	0.296
	GSN-v	1.738	1.970	6.111	4.632	3.312	1.196	1.385	1.387	1.394	39.601	50.874	1.327
	LAGIN	0.049	0.055	0.357	0.101	0.117	0.022	0.365	0.367	0.117	0.757	2.087	0.060
	CoS-GIN	0.087	0.099	0.704	0.172	0.198	0.036	0.585	0.586	0.199	1.537	4.389	0.102
Stage	**Method**	**BZR**	**COX2**	**DD**	**I-B**	**I-M**	**MUTAG**	**NCI1**	**NCI109**	**P_full**	**R-B**	**R-M**	**ENZYMES**
Inference	$\mathscr{G}$-mixup	0.003	0.003	-	0.004	0.005	0.002	0.012	0.013	0.005	-	-	0.003
	Dummy	0.003	0.003	0.019	-	-	0.002	0.012	0.012	0.005	-	-	0.003
	DropGNN	0.019	0.013	-	0.012	0.018	0.006	0.108	0.116	0.053	-	-	0.025
	rGIN	0.005	0.005	0.022	0.008	0.010	0.005	0.023	0.022	0.009	0.038	0.119	0.006
	NestedGIN	0.015	0.015	0.588	0.063	0.053	0.006	0.091	0.094	0.040	-	-	0.021
	GSN-v	0.009	0.010	0.031	0.016	0.015	0.008	0.025	0.025	0.010	0.112	0.370	0.008
	LAGIN	0.003	0.004	0.017	0.005	0.005	0.004	0.005	0.005	0.005	0.029	0.048	0.004
	CoS-GIN	0.009	0.009	0.041	0.012	0.013	0.008	0.032	0.032	0.013	0.081	0.233	0.010

which is caused by the feature augmentation operation, but it is more efficient than complex structural augmentation methods, including DropGNN and NestedGIN. Besides, although our CoS-GIN is a little time-costly on some large-scale datasets, it can still be successfully implemented on devices with limited computational ability, while $\mathscr{G}$-mixup, dummy, DropGNN and NestedGIN require more powerful devices on such instances, which restricts their application.

4.4.4 Employ CoS-GNN as GNN Backbone

Combined with GNN-based Methods

In this section, we examine the applicability of our CoS-GNN as GNN backbone in other GNN-based methods by replacing the GCN of $\mathscr{G}$-mixup and Dummy with CoS-GCN. We omit the results on REDDIT-BINARY and REDDIT-MULTI here because we can not run $\mathscr{G}$-mixup and Dummy on them by our device. The accuracy results of GCN-based and CoS-GCN-based $\mathscr{G}$-mixup and Dummy are reported in Table 4.5. The results show that the CoS-GCN-based $\mathscr{G}$-mixup outperforms GCN-based $\mathscr{G}$-mixup on all datasets and the largest improvement can be up to 32%. The performance of CoS-GCN-based Dummy

Table 4.5: Accuracy (mean±std) results of $\mathcal{G}$-mixup and Dummy using CoS-GCN as the GNN module, with $\mathcal{G}$-mixup and Dummy with GCN as baselines in graph classification. 'Different' denotes accuracy improvement (↑) or decrease (↓) brought by the replacement of CoS-GCN. Both of two methods suffer out of memory on REDDIT-BINARY and REDDIT-MULTI.

Dataset	**$\mathcal{G}$−mixup(GCN)**	**$\mathcal{G}$−mixup(CoS-GCN)**	**Difference**
BZR	0.8295 ± 0.0188	0.8666 ± 0.0471	$0.0371\uparrow$
COX2	0.7730 ± 0.0482	0.7967 ± 0.0416	$0.0237\uparrow$
DD	–	–	–
I−BINARY	0.7360 ± 0.0350	0.7410 ± 0.0243	$0.0050\uparrow$
I−MULTI	0.5013 ± 0.0283	0.5073 ± 0.0264	$0.0060\uparrow$
MUTAG	0.7173 ± 0.1130	0.8830 ± 0.0617	$0.1657\uparrow$
NCI1	0.5007 ± 0.0010	0.8212 ± 0.0146	$0.3205\uparrow$
NCI109	0.5038 ± 0.0007	0.7986 ± 0.0179	$0.2948\uparrow$
PROTS_full	0.7152 ± 0.0350	0.7512 ± 0.0246	$0.0360\uparrow$
ENZYMES	0.3456 ± 0.0435	0.4909 ± 0.0490	$0.1453\uparrow$
p-value	0.0039	-	-
Dataset	**Dummy(GCN)**	**Dummy(CoS-GCN)**	**Difference**
BZR	0.8296 ± 0.0203	0.8321 ± 0.0482	$0.0025\uparrow$
COX2	0.7899 ± 0.0509	0.8112 ± 0.0654	$0.0213\uparrow$
DD	0.7776 ± 0.0717	0.7699 ± 0.0433	$-0.0077\downarrow$
I−BINARY	–	–	–
I−MULTI	–	–	–
MUTAG	0.7813 ± 0.1292	0.8673 ± 0.0754	$0.0860\uparrow$
NCI1	0.6608 ± 0.1016	0.8092 ± 0.0146	$0.1484\uparrow$
NCI109	0.5527 ± 0.0851	0.8013 ± 0.0243	$0.2486\uparrow$
PROTS_full	0.7557 ± 0.0375	0.7556 ± 0.0286	$-0.0001\downarrow$
ENZYMES	0.4450 ± 0.1038	0.6067 ± 0.0602	$0.1617\uparrow$
p-value	0.0547	-	-

method is also better than the GCN-based Dummy on most datasets, achieving up to 25% improvement. The paired Wilcoxon signed rank test indicates that the improvement of CoS-GCN-based $\mathcal{G}$-mixup and Dummy across 10 datasets is significant at 99% and 90% confidence level, respectively. The decrease in accuracy of CoS-GCN-based Dummy on DD and PROTEINS_full might be because that the specific structural statistics augmented on the graphs are influenced and disturbed by the addition of the dummy node. When compared with the results of single CoS-GCN, there are also some improvement brought by CoS-GCN-based $\mathcal{G}$-mixup on BZR (3.5%), MUTAG (0.5%) and NCI1 (0.5%) and by CoS-GCN-based Dummy on ENZYMES (7.4%), which means that the combination with other GNN-based methods can also enhance the power of CoS-GNN. Overall, our CoS-GNN and other GNN-based methods are complementary and can be used as the basic GNN module in GNN-based methods to improve their performance successfully.

Combined with Other Pooling Methods

Table 4.6: Accuracy (mean±std) results of GCN and CoS-GCN with MVPool as the readout operation. 'Different' denotes accuracy improvement (↑) or decrease (↓) brought by CoS-GCN compared to GCN.

Dataset	**GCN-MVPool**	**CoS-GCN-MVPool**	**Difference**
BZR	0.8273 ± 0.0565	0.8345 ± 0.0301	$0.0072\uparrow$
COX2	0.7987 ± 0.0351	0.7986 ± 0.0430	$-0.0001\downarrow$
DD	0.7750 ± 0.0363	0.7962 ± 0.0390	$0.0212\uparrow$
I−BINARY	0.7280 ± 0.0268	0.7350 ± 0.0492	$0.0070\uparrow$
I−MULTI	0.5180 ± 0.0253	0.5080 ± 0.0332	$-0.0100\downarrow$
MUTAG	0.7178 ± 0.0858	0.8406 ± 0.1107	$0.1228\uparrow$
NCI1	0.7791 ± 0.0155	0.8015 ± 0.0094	$0.0224\uparrow$
NCI109	0.7754 ± 0.0233	0.7989 ± 0.0198	$0.0235\uparrow$
PROTS_full	0.7556 ± 0.0360	0.7664 ± 0.0298	$0.0108\uparrow$
R−BINARY	0.9050 ± 0.0219	0.9140 ± 0.0202	$0.0090\uparrow$
R−MULTI	0.5311 ± 0.0136	0.5515 ± 0.0255	$0.0204\uparrow$
ENZYMES	0.5833 ± 0.0516	0.5917 ± 0.0455	$0.0084\uparrow$
p-value	0.0093	-	-

There have been a type of pooling methods that hierarchically extract the graph information [242; 261]. We demonstrate that our CoS-GNN structure can be combined with these methods in this section. In each pooling layer, we use the real node features to calculate the pooling criterion and construct the coarsened graph.

We run our CoS-GCN with a hierarchical pooling – MVPool as the pooling operation to prove that our CoS-GCN can improve the performance of hierarchical pooling. The results of CoS-GCN with MVPool and GCN with MVPool as baseline are reported in Table 4.6, showing that CoS-GCN still can bring improvement on all datasets except COX2 and IMDB-MULTI. The average improvement is 2.02% and the maximal improvement can be up to about 12.28%. The paired Wilcoxon signed rank test indicates that the improvement across 12 datasets is significant at 99% confidence level. These results demonstrate that the augmented node and graph structural information also can provide extra useful information while the pooling operation is learning graph structural information. The accuracy decline of CoS-GCN with MVPool on COX2 is very marginal, only 0.01%. The 1% drop of CoS-GCN with MVPool on IMDB-MULTI might be because that the structural information in IMDB-MULTI might be limited and the hierarchical learning of MVPool can utilize most structural information in the graph. The feature augmentation in CoS-GCN provides some redundant information to the CoS-GCN-MVPool.

4.4.5 Enabling Out-of-distribution Generalization Tasks

We evaluate the generalization ability of CoS-GNN on out-of-distribution (OOD) data in this section. This experiment is designed to compare the performance of CoS-GNN with other two message passing neural networks (*i.e.*, GCN and GIN) in OOD generalization. These GNNs can be utilized as GNN backbone in various generalization methods to obtain better performance further. The datasets we utilize are from GOOD[1] [69]. GOOD-Motif is a synthetic dataset designed for structure shifts, GOOD-HIV is a molecular dataset, and GOOD-SST2 is a natural language sentiment analysis dataset. For each dataset, the GOOD benchmark selects one or two domain features (*e.g.*, base and size for GOOD-Motif, scaffold and size for GOOD-HIV, and length for GOOD-SST2) and then applies covariate and concept shift splits per domain to create diverse distribution shifts. Following [69], the metric we use for GOOD-HIV is AUC and classification accuracy is used for other datasets. We examine the generalization power of CoS-GNN with the baseline models taken from the GOOD benchmark [69]. The GNN backbone used in the baselines is GIN.

The results on the OOD and the in-distribution (ID) validation sets are reported in Table 4.7. It can be seen from the results that our model CoS-GCN outperforms the basic GCN on all settings except the one on GOOD-HIV; CoS-GIN gains better performance than GOOD on all settings except GOOD-HIV. This is mainly because that the node and graph structures augmented in our CoS-GNN are more generalizable w.r.t. different shifts of base, size, or length on the three GOOD datasets, while being less generalizable to the scaffold shift, a two-dimensional structural base of a molecule. The especially outstanding performance of CoS-GNN on GOOD-Motif also helps justify this. Each graph in GOOD-Motif is generated by connecting a base graph and a motif, and thus, the structure of base graphs and motifs is highly differentiated. Thus, the augmented structural information of each class enables the structure learning in CoS-GNN to obtain substantially improved OOD generalization performance, when compared with vanilla GNN.

4.4.6 Robustness w.r.t. Structure Contamination

Since the data collected in real applications may be with limited/noisy structural information, the performance of our CoS-GNN, which harnesses rich structural information, might be influenced by these contaminated information. In this section, we discuss the

[1] https://github.com/divelab/GOOD/

Table 4.7: Results of CoS-GNN with two baselines on three OOD datasets. G-X is short for the dataset name GOOD-X.

G-Motif	**Base**			
	Covariate		Concept	
Accuracy	OOD Validation	ID Validation	OOD Validation	ID Validation
GCN	0.321 ± 0.000	0.343 ± 0.025	0.395 ± 0.014	0.382 ± 0.017
GOOD	0.687 ± 0.034	0.700 ± 0.019	0.814 ± 0.006	0.809 ± 0.007
CoS-GCN	0.868 ± 0.004	0.865 ± 0.003	$\mathbf{0.934 \pm 0.000}$	$\mathbf{0.932 \pm 0.001}$
CoS-GIN	$\mathbf{0.888 \pm 0.020}$	$\mathbf{0.896 \pm 0.007}$	0.931 ± 0.001	0.923 ± 0.009
G-Motif	**Size**			
	Covariate		Concept	
Accuracy	OOD Validation	ID Validation	OOD Validation	ID Validation
GCN	0.346 ± 0.008	0.350 ± 0.003	0.391 ± 0.0172	0.385 ± 0.018
GOOD	0.517 ± 0.023	0.513 ± 0.019	0.708 ± 0.006	0.694 ± 0.009
CoS-GCN	$\mathbf{0.863 \pm 0.043}$	$\mathbf{0.816 \pm 0.077}$	$\mathbf{0.935 \pm 0.000}$	$\mathbf{0.933 \pm 0.002}$
CoS-GIN	0.598 ± 0.070	0.555 ± 0.096	0.918 ± 0.006	0.898 ± 0.014
G-HIV	**Scaffold**			
	Covariate		Concept	
AUC	OOD Validation	ID Validation	OOD Validation	ID Validation
GCN	0.669 ± 0.026	0.676 ± 0.016	0.700 ± 0.014	0.607 ± 0.016
GOOD	$\mathbf{0.696 \pm 0.020}$	0.689 ± 0.021	$\mathbf{0.723 \pm 0.010}$	$\mathbf{0.653 \pm 0.035}$
CoS-GCN	0.690 ± 0.017	$\mathbf{0.699 \pm 0.023}$	0.708 ± 0.009	0.605 ± 0.026
CoS-GIN	0.684 ± 0.021	0.663 ± 0.036	0.722 ± 0.011	0.636 ± 0.016
G-HIV	**Size**			
	Covariate		Concept	
AUC	OOD Validation	ID Validation	OOD Validation	ID Validation
GCN	0.591 ± 0.020	0.580 ± 0.012	0.638 ± 0.0110	0.533 ± 0.009
GOOD	0.600 ± 0.029	0.584 ± 0.025	0.633 ± 0.025	0.448 ± 0.029
CoS-GCN	$\mathbf{0.607 \pm 0.019}$	$\mathbf{0.619 \pm 0.005}$	0.654 ± 0.008	0.547 ± 0.007
CoS-GIN	0.585 ± 0.029	0.599 ± 0.028	$\mathbf{0.731 \pm 0.006}$	$\mathbf{0.622 \pm 0.016}$
G-SST2	**Length**			
	Covariate		Concept	
Accuracy	OOD Validation	ID Validation	OOD Validation	ID Validation
GCN	0.825 ± 0.008	0.805 ± 0.010	0.724 ± 0.012	0.677 ± 0.010
GOOD	0.813 ± 0.004	0.778 ± 0.011	0.724 ± 0.005	0.673 ± 0.001
CoS-GCN	$\mathbf{0.828 \pm 0.010}$	$\mathbf{0.814 \pm 0.014}$	0.730 ± 0.007	$\mathbf{0.685 \pm 0.023}$
CoS-GIN	0.822 ± 0.012	0.796 ± 0.021	$\mathbf{0.737 \pm 0.012}$	$\mathbf{0.685 \pm 0.013}$

impact of limited/noisy structural knowledge on our CoS-GNN. Specifically, we randomly remove $\{1\%, 5\%, 10\%, 15\%, 20\%\}$ edges of the data and compare their results with the results on original data. Our experiment is implemented on GCN backbone.

The results on NCI1 are displayed in Figure 4.5. It is obvious that both CoS-GCN and GCN suffer from performance decline due to the edge removal and the decline level of them is similar. Our CoS-GCN always have better performance than GCN under various structural contamination situation. This means that limited/noisy structure brings no more serious effects on the CoS-GCN. This might be because that the structures we augment are in different scales and parts of the extracted features will be infected while

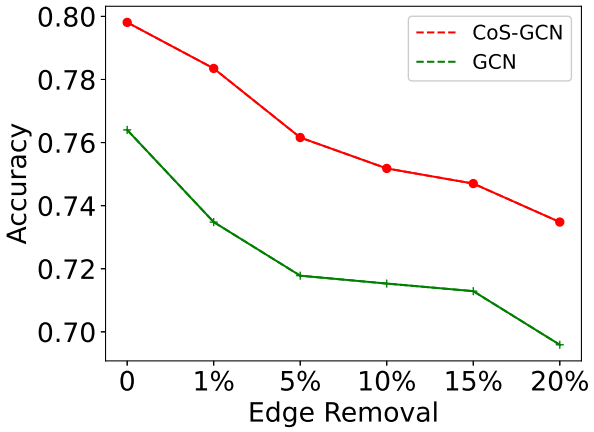


Figure 4.5: Accuracy performance of CoS-GCN and GCN w.r.t. different structural contamination rates.

others will still be exact. The unaffected structure features can correct the influence of the wrong information brought by the limited/noisy graph structure.

4.4.7 Convergence Analysis

In this section we run an experiment to illustrate the convergence ability of our CoS-GNN. In detail, we run the CoS-GCN on the REDDIT-BINARY dataset and record the loss tendency of training and validation dataset. The result is shown in Figure 4.6. It is obvious that both the training and validation loss will approach stability after a number of epochs. Besides, the early stopping used during training can ensure the model against overfitting and obtaining an excellent result.

4.4.8 Ablation Study

Ablation Study of the Graph Augmentation and the Specific Message Passing Scheme

This section examines the importance of the graph augmentation and the message passing scheme designed in CoS-GNN. All expeirments are based on CoS-GCN. We first evaluate the performance of GCN with original/augmented features as sole input ($\mathcal{V}_{nf}$ for real node feature, $\mathcal{V}_{ns}$ for augmented node structure features, and $\mathcal{V}_{gs}$ for augmented graph structure features), and then combine original and augmented node features by convolution after concatenation (cv_ct($\mathcal{V}_{nf}, \mathcal{V}_{ns}$)), concatenation after convolution (ct_cv($\mathcal{V}_{nf}, \mathcal{V}_{ns}$))), and convolution with our proposed message passing method

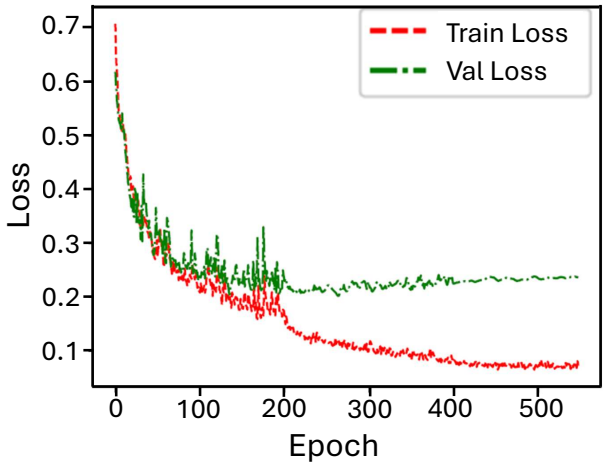


Figure 4.6: Loss variation tendency of CoS-GCN on the training and validation dataset of REDDIT-BINARY.

(cv($\mathcal{V}_{nf}, \mathcal{V}_{ns}$))). Incorporating the augmented graph features to cv($\mathcal{V}_{nf}, \mathcal{V}_{ns}$) leads to the full CoS-GCN.

The results of our ablation study using the graph classification task are displayed in Table 4.8. The paired Wilcoxon signed rank test shows when compared with other ablation parts except cv($\mathcal{V}_{nf}, \mathcal{V}_{ns}$), the improvement of CoS-GCN across 12 datasets is significant at 99% confidence level. The enhancement of CoS-GCN than cv($\mathcal{V}_{nf}, \mathcal{V}_{ns}$) across all datasets is significant at 85% confidence level. In detail, using node features ($\mathcal{V}_{nf}$) or augmented node/graph structural features ($\mathcal{V}_{ns}$/$\mathcal{V}_{gs}$) solely can achieve good performance, and using $\mathcal{V}_{nf}$ often outperforms $\mathcal{V}_{ns}$ and $\mathcal{V}_{gs}$ on most datasets. This indicates that both the original and augmented features are useful in graph representation learning but the augmented features is limitedly informative. The simple concatenation of $\mathcal{V}_{nf}$ and $\mathcal{V}_{ns}$, *i.e.*, **cv_ct**($\mathcal{V}_{\mathbf{nf}}, \mathcal{V}_{\mathbf{ns}}$) or **ct_cv**($\mathcal{V}_{\mathbf{nf}}, \mathcal{V}_{\mathbf{ns}}$)), helps improve the performance over the using of them solely, indicating the complementary information gained from the graph augmentation relative to the original node features. Our proposed message passing (convolution) method on top of the real and augmented node features, *i.e.*, **cv**($\mathcal{V}_{\mathbf{nf}}, \mathcal{V}_{\mathbf{ns}}$), further enhances the results substantially, which demonstrates the effectiveness of our proposed message passing in capturing intricate relations that cannot be captured in the vanilla GCN. Lastly, incorporating the augmented graph-level features would lead to the full model CoS-GNN that largely improves **cv**($\mathcal{V}_{\mathbf{nf}}, \mathcal{V}_{\mathbf{ns}}$), demonstrating that the generated global graph structural features are also important for the overall improvement.

Ablation Study of the Augmented Features

Table 4.8: Results of the ablation study of the graph augmentation and the message passing mechanism in CoS-GCN in the graph classification task.

Dataset	$\mathcal{V}_{\mathbf{nf}}$	$\mathcal{V}_{\mathbf{ns}}$	$\mathcal{V}_{\mathbf{gs}}$	**cv_ct**($\mathcal{V}_{\mathbf{nf}}, \mathcal{V}_{\mathbf{ns}}$)	**ct_cv**($\mathcal{V}_{\mathbf{nf}}, \mathcal{V}_{\mathbf{ns}}$)	**cv**($\mathcal{V}_{\mathbf{nf}}, \mathcal{V}_{\mathbf{ns}}$)	**CoS-GCN**
BZR	0.840 ± 0.028	$\mathbf{0.847 \pm 0.028}$	0.788 ± 0.010	0.808 ± 0.041	0.830 ± 0.025	0.845 ± 0.038	0.832 ± 0.036
COX2	0.811 ± 0.043	0.782 ± 0.008	0.782 ± 0.008	0.805 ± 0.043	0.805 ± 0.044	0.803 ± 0.058	$\mathbf{0.824 \pm 0.055}$
DD	0.756 ± 0.033	0.744 ± 0.034	0.756 ± 0.038	0.770 ± 0.043	0.774 ± 0.043	0.773 ± 0.032	$\mathbf{0.779 \pm 0.032}$
I−BINARY	0.724 ± 0.036	0.709 ± 0.056	0.706 ± 0.052	0.725 ± 0.069	0.741 ± 0.042	0.738 ± 0.033	$\mathbf{0.754 \pm 0.052}$
I−MULTI	0.498 ± 0.024	0.483 ± 0.032	0.479 ± 0.033	0.495 ± 0.028	0.491 ± 0.026	$\mathbf{0.509 \pm 0.024}$	0.503 ± 0.019
MUTAG	0.744 ± 0.102	0.867 ± 0.042	0.808 ± 0.091	0.823 ± 0.089	0.835 ± 0.064	0.872 ± 0.066	$\mathbf{0.878 \pm 0.059}$
NCI1	0.785 ± 0.016	0.690 ± 0.018	0.634 ± 0.015	0.790 ± 0.017	0.789 ± 0.019	0.808 ± 0.011	$\mathbf{0.816 \pm 0.013}$
NCI109	0.769 ± 0.026	0.699 ± 0.025	0.629 ± 0.025	0.774 ± 0.018	0.773 ± 0.029	0.799 ± 0.023	$\mathbf{0.801 \pm 0.021}$
PROTS_full	0.749 ± 0.028	0.728 ± 0.031	0.730 ± 0.023	0.758 ± 0.034	0.752 ± 0.029	$\mathbf{0.762 \pm 0.027}$	0.757 ± 0.028
R−BINARY	0.900 ± 0.024	0.912 ± 0.025	0.830 ± 0.020	0.910 ± 0.027	0.908 ± 0.027	0.911 ± 0.025	$\mathbf{0.917 \pm 0.022}$
R−MULTI	0.533 ± 0.013	0.544 ± 0.018	0.503 ± 0.027	0.543 ± 0.015	0.543 ± 0.021	0.554 ± 0.011	$\mathbf{0.554 \pm 0.028}$
ENZYMES	0.450 ± 0.068	0.278 ± 0.033	0.275 ± 0.065	0.432 ± 0.067	0.502 ± 0.063	0.462 ± 0.074	$\mathbf{0.533 \pm 0.064}$
Rank	4.7	4.9	6.6	3.9	3.8	2.4	**1.5**
p-value	0.0015	0.0034	0.0005	0.0010	0.0005	0.1289	–

In this section, we evaluate the effect of each augmented features on the final performance of CoS-GNN. We divided the augmented features into two categories, *i.e.*, one is the characteristics of some specific substructures, and another is some quantized values to measure structural properties of the node/graph. Then we remove each feature and their combinations in each category separately and compare the classification results with our CoS-GNN. The removal of node and graph features is implemented separately. The GNN backbone we use here is CoS-GCN.

Firstly, we delete degree, triangle, clique and k-core (denoted by w/o Dg, w/o Tri, w/o CK respectively) and then remove their combinations, *i.e.*, degree and triangle; degree, clique and k-core; triangle, clique and k-core; all the characteristics (shortened to w/o DT, w/o DCK, w/o TCK, w/o n_sub). We also remove quantized values – triangle clustering coefficient, square clustering coefficient and their combination (written as w/o TCo, w/o SCo and w/o n_quant respectively). The results are reported in Table 4.9. The performance of all operations is compared simultaneously. It is obvious that the removal of augmented features might cause better performance on some specific datasets but will result in decline on many other datasets, leading to a clear decline in the overall performance. Although our CoS-GCN still ranks first on the overall performance, the paired Wilcoxon signed rank test indicates that the performance drop of models with part of augmented features across 12 datasets is significant at 85% to 99% confidence level. Deletion of degree and clique and k-core characteristics respectively and their combinations often lead to worse performance, indicating their effect in the full CoS-GCN. Omitting all the node structural characteristics performs better than removing part

Table 4.9: Efficiency of the augmented node features in graph classification.

	Separated features					**Completed**
	Structural			**Quantized**		**Completed**
Dataset	**w/o Dg**	**w/o Tri**	**w/o CK**	**w/o TCo**	**w/o SCo**	**CoS-GCN**
BZR	0.847±0.048	0.832±0.043	0.857±0.039	0.835±0.052	0.857±0.048	0.832±0.036
COX2	0.818±0.049	0.829±0.043	0.814±0.043	0.827±0.045	0.805±0.072	0.824±0.055
DD	0.774±0.027	0.757±0.028	0.750±0.021	0.753±0.027	0.756±0.032	0.778±0.032
I-BINARY	0.749±0.040	0.738±0.041	0.726±0.042	0.747±0.026	0.734±0.036	0.754±0.052
I-MULTI	0.473±0.039	0.498±0.019	0.469±0.031	0.489±0.029	0.486±0.034	0.503±0.019
MUTAG	0.861±0.093	0.841±0.070	0.803±0.082	0.851±0.066	0.872±0.068	0.878±0.059
NCI1	0.780±0.019	0.782±0.023	0.782±0.021	0.793±0.016	0.791±0.017	0.816±0.013
NCI109	0.777±0.025	0.783±0.019	0.767±0.024	0.776±0.018	0.780±0.023	0.801±0.021
PROTS_full	0.750±0.039	0.749±0.034	0.753±0.039	0.758±0.038	0.757±0.039	0.757±0.028
R-BINARY	0.915±0.013	0.920±0.009	0.889±0.025	0.915±0.018	0.905±0.018	0.917±0.022
R-MULTI	0.541±0.015	0.554±0.014	0.523±0.018	0.558±0.021	0.546±0.019	0.554±0.028
ENZYMES	0.490±0.048	0.568±0.057	0.557±0.074	0.523±0.043	0.545±0.066	0.533±0.064
Rank	6.3	4.6	7.8	4.8	5.4	2.8
p-value	0.0093	0.1387	0.0093	0.0400	0.0986	–
	Combined features					**Completed**
	Structural			**Quantized**	**Structural**	**Completed**
Dataset	**w/o TCK**	**w/o DT**	**w/o DCK**	**w/o n_quant**	**w/o n_sub**	**CoS-GCN**
BZR	0.842±0.039	0.845±0.053	0.857±0.042	0.815±0.064	0.832±0.052	0.832±0.036
COX2	0.807±0.051	0.827±0.062	0.816±0.041	0.803±0.054	0.822±0.056	0.824±0.055
DD	0.750±0.035	0.743±0.036	0.750±0.024	0.777±0.030	0.770±0.030	0.778±0.032
I-BINARY	0.731±0.050	0.732±0.041	0.726±0.049	0.750±0.053	0.726±0.034	0.754±0.052
I-MULTI	0.495±0.024	0.492±0.022	0.458±0.025	0.493±0.033	0.503±0.024	0.503±0.019
MUTAG	0.808±0.077	0.851±0.057	0.781±0.103	0.862±0.060	0.835±0.077	0.878±0.059
NCI1	0.775±0.014	0.783±0.022	0.751±0.012	0.816±0.017	0.817±0.013	0.816±0.013
NCI109	0.760±0.029	0.772±0.025	0.722±0.030	0.796±0.024	0.795±0.022	0.801±0.021
PROTS_full	0.758±0.042	0.750±0.031	0.748±0.036	0.755±0.025	0.761±0.026	0.757±0.028
R-BINARY	0.890±0.022	0.918±0.017	0.902±0.025	0.906±0.018	0.895±0.033	0.917±0.022
R-MULTI	0.528±0.021	0.551±0.013	0.549±0.022	0.557±0.009	0.550±0.022	0.554±0.028
ENZYMES	0.510±0.051	0.528±0.050	0.503±0.045	0.520±0.059	0.535±0.072	0.533±0.064
Rank	7.9	5.9	8.8	5.0	4.9	2.8
p-value	0.0034	0.0220	0.0049	0.0049	0.0488	–

of them on some datasets and this might be because that the remaining structural characteristics increase the similarity among data.

Later, we delete augmented graph features sequentially (*i.e.*, w/o Tri, w/o Cli, w/o Bri, w/o ClCo and w/o Effi stand for removing triangle, clique, bridge numbers, average clustering coefficient and average local and global efficiency, respectively; w/o TBri, w/o ClBri and w/o TrCl denote deleting triangle and bridge numbers, clique and bridge numbers and triangle and clique numbers; w/o g_quant and w/o g_sub means removing the quantized values and graph substructural statistics, respectively). The results are shown in Table 4.10. The performance of all operations is compared simultaneously. The improvement of our CoS-GCN over the competing methods across the datasets is

significant at 80% to 99% confidence level. The removal of triangle, clique, bridge and their combination knowledge results in similar overall performance, which might be because that each feature contributes to the performance of CoS-GNN differently in different dataset. The deletion of average clustering coefficient or all quantized values has larger effect on the final performance, which indicates that the average clustering coefficient information is more discriminative. In summary, the graph-level substructural characteristics are also beneficial in our CoS-GNN since the removal of them leads to a clear decline of the overall performance of CoS-GNN.

Table 4.10: Efficiency of the augmented graph features in graph classification.

	Separated features					**Completed**
	Structural			**Quantized**		**Completed**
Dataset	**w/o Tri**	**w/o Cli**	**w/o Bri**	**w/o ClCo**	**w/o Effi**	**CoS-GCN**
BZR	0.850±0.056	0.852±0.032	0.857±0.043	0.840±0.053	0.840±0.050	0.832±0.036
COX2	0.812±0.056	0.820±0.052	0.827±0.070	0.812±0.061	0.824±0.048	0.824±0.055
DD	0.759±0.031	0.769±0.026	0.764±0.029	0.766±0.030	0.764±0.020	0.778±0.032
I-BINARY	0.744±0.040	0.730±0.035	0.738±0.035	0.733±0.026	0.744±0.042	0.754±0.052
I-MULTI	0.487±0.028	0.480±0.024	0.475±0.035	0.475±0.024	0.471±0.024	0.503±0.019
MUTAG	0.830±0.065	0.856±0.067	0.856±0.082	0.867±0.064	0.878±0.058	0.878±0.059
NCI1	0.796±0.022	0.794±0.013	0.795±0.025	0.753±0.081	0.798±0.017	0.816±0.013
NCI109	0.783±0.018	0.779±0.016	0.774±0.025	0.772±0.025	0.776±0.028	0.801±0.021
PROTS_full	0.766±0.046	0.741±0.044	0.750±0.039	0.757±0.042	0.769±0.041	0.757±0.028
R-BINARY	0.915±0.014	0.915±0.017	0.914±0.017	0.917±0.021	0.920±0.015	0.917±0.022
R-MULTI	0.551±0.021	0.563±0.024	0.547±0.017	0.543±0.024	0.551±0.019	0.554±0.028
ENZYMES	0.520±0.049	0.547±0.067	0.537±0.044	0.523±0.085	0.535±0.041	0.533±0.064
Rank	5.5	5.3	5.5	6.8	4.4	2.9
p-value	0.0278	0.0669	0.0542	0.0039	0.168	–
	Combined features					**Completed**
	Structural			**Quantized**	**Structural**	**Completed**
Dataset	**w/o TBri**	**w/o ClBri**	**w/o TrCl**	**w/o g_quant**	**w/o g_sub**	**CoS-GCN**
BZR	0.845±0.037	0.847±0.056	0.845±0.068	0.785±0.023	0.830±0.051	0.832±0.036
COX2	0.822±0.050	0.799±0.068	0.807±0.055	0.784±0.011	0.798±0.057	0.824±0.055
DD	0.768±0.034	0.773±0.034	0.761±0.029	0.745±0.044	0.784±0.040	0.778±0.032
I-BINARY	0.736±0.036	0.731±0.034	0.723±0.036	0.567±0.059	0.736±0.051	0.754±0.052
I-MULTI	0.477±0.025	0.475±0.028	0.499±0.025	0.366±0.033	0.501±0.026	0.503±0.019
MUTAG	0.825±0.062	0.856±0.078	0.846±0.060	0.856±0.076	0.861±0.083	0.878±0.059
NCI1	0.790±0.030	0.787±0.015	0.787±0.027	0.784±0.018	0.810±0.016	0.816±0.013
NCI109	0.782±0.015	0.781±0.017	0.782±0.024	0.772±0.024	0.799±0.016	0.801±0.021
PROTS_full	0.762±0.036	0.754±0.044	0.745±0.045	0.742±0.020	0.748±0.029	0.757±0.028
R-BINARY	0.909±0.029	0.917±0.022	0.917±0.022	0.757±0.045	0.911±0.024	0.917±0.022
R-MULTI	0.540±0.026	0.560±0.015	0.556±0.019	0.250±0.053	0.552±0.015	0.554±0.028
ENZYMES	0.552±0.074	0.535±0.068	0.563±0.077	0.173±0.039	0.525±0.040	0.533±0.064
Rank	6.0	5.4	5.9	10.3	5.5	2.9
p-value	0.0679	0.0420	0.0977	0.0005	0.0068	–

4.5 Summary

This chapter proposes a collective structure knowledge-augmented graph neural network (CoS-GNN) to enhance the expressive power of conventional message passing neural networks. The augmented node and graph features carry important and domain-adaptive structural knowledge, enabling the model to achieve excellent performance on various datasets. Then a message passing mechanism is proposed to integrate the original and augmented graph knowledge, resulting in graph representations with significantly improved expressiveness. This is justified by experiments in various down-stream tasks, including graph classification, and OOD generalization.

Although the calculation of CoS-GNN might be more costly than the computation of their corresponding vanilla GNN models, the performance improvement makes it negligible. Besides, CoS-GNN is a bit more costly than simple augmentation operation, which is caused by the feature augmentation operation, but it is more efficient than complex structural augmentation methods and requires less computational time and resources. Therefore, CoS-GNN provides a good trade-off between better performance and resource- and time-efficient.

Furthermore, when CoS-GNN is employed as GNN backbone and applied to other GNN-based methods, the performance of the model is successfully improved, indicating the potential of CoS-GNN to be used as basic GNN.

CHAPTER 5

IMBALANCED GRAPH-LEVEL CLASSIFICATION WITH MULTI-SCALE OVERSAMPLING

5.1 Introduction

In Chapter 4, we proposed an improved GNN for better performance in balanced graph-level classification. However, when the sample classes are distribution-imbalanced, such improvement can only have limited effect since it does not focus on the class imbalance issue. One main challenge in imbalanced graph classification is to learn expressive representations capturing discriminative local (*i.e.*, node/subgraph levels) and global (*i.e.*, graph level) knowledge of the graphs in under-represented (minority) classes. Solutions able to address this challenge could have advanced performance.

There are numerous conventional imbalanced learning methods [260], such as class re-sampling [73; 22; 91; 249], loss adjustment [123; 197; 19; 35; 181], logit adjustment [149; 95; 76] and information augmentation [252; 204; 247; 32]. Most of them are data-agnostic approaches, so they can often be combined with GNNs directly. For example, oversampling [73], which balances the dataset by randomly duplicating the samples from the minority class, is one of the most popular and easy-to-implement methods for imbalanced learning on graph data. Other approaches such as imbalanced learning loss functions (a.k.a. sample weighting methods) [123; 197; 19; 149; 95] address the imbalance problem by (adaptively) assigning larger weights to the minority samples during training. The resulting graph-level representations, however, ignore

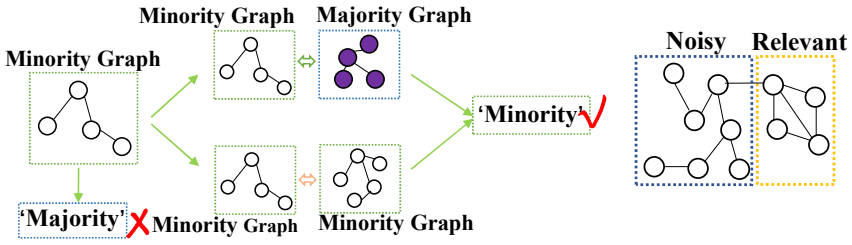


Figure 5.1: Motivation of pairwise- and subgraph-scale oversampling. (a) Pairwise-scale classification. A minority graph wrongly classified based on graph-scale information can be correctly classified based on graph interactions; (b) Subgraph-scale classification. In some graphs, only their subgraphs are relevant to the classification; the other parts are noisy. Oversampling the whole graphs may lead to the inclusion of more noise.

diverse discriminative information within the graphs and their interactions, especially for the minority graphs. As a result, they are often lack of discriminative power for the classification task.

There have been a number of deep methods designed specifically for imbalanced classification in the graph domain. Particularly, a few data augmentation-based methods, such as random edge/node elimination and mixup [217; 71], have been explored to capture intra-/inter-graph interactions and generate more graphs to enrich the GNN-based representations of minority graphs. However, it is difficult to perform meaningful intra-graph augmentation or interpolations between graphs due to their non-Euclidean nature and the lack of domain knowledge. Wang et al. [219] construct a graph of graphs (GoG) on the oversampled and augmented graphs and perform GoG propagation to aggregate neighboring graph representations for imbalanced graph classification. Nevertheless, as the GoG is constructed on kernel similarity, its time complexity increases drastically with dataset size (*i.e.*, polynomial w.r.t. the number of graphs and cubic w.r.t. the maximum number of nodes in a graph), and thus, it is prohibitively computationally costly for large datasets.

In this chapter, we propose a novel *multi-scale oversampling GNN, namely MOSGNN*, to learn various discriminative semantics within and between minority graphs for imbalanced graph classification. MOSGNN is specifically designed to leverage graph information at multiple scales, *i.e.*, subgraph, graph, and pairwise-graph scales, to generate a large number of minority-graph-oriented samples of diversified semantic for mitigating the bias toward the majority graphs at different granularities in GNN-

based models. To this end, in addition to the oversampling at the graph scale in general oversampling-based approaches, MOSGNN jointly optimizes two auxiliary objectives on pairwise graph relation prediction and multiple-instance-learning-based (MIL) subgraph classification.

The pairwise graph relation prediction is designed to extend the classification information of the minority graphs by exploiting the inter-graph interactions. Through pairing with other graphs, large-scale minority-oriented graph pairs can be generated easily, *i.e.*, majority-minority and minority-minority pairs vs. majority-majority pairs, from which MOSGNN can learn discriminative pairwise patterns of the minority graphs against that of majority graph pairs. This enables MOSGNN to correctly classify minority graphs that are otherwise wrongly classified by the graph-scale oversampling module (see Figure 5.1(a)). On the other hand, considering the specific structure characteristic of graphs and the fact that given a graph, only part of the graphs, *e.g.*, a few subgraphs, are relevant to the graph classification (Figure 5.1(b)), we design a subgraph-level oversampling to generate diverse subgraphs to represent graphs with different local semantics. Since one graph can have relevant and noisy subgraph patterns, we use MIL to allow MOSGNN to focus on only the relevant subgraphs of each graph. In doing so, MOSGNN learns expressive minority graph representations based on the rich semantics embedded within and between the graphs, enabling better classification performance.

In summary, we make three main contributions:

- We introduce a novel deep multi-scale oversampling framework to address the imbalanced graph classification problem. It learns discriminative representations of the minority graphs with the support of oversampling the minority graphs that have largely augmented semantics at the subgraph, graph, and pairwise inter-graph levels. This is the first work that takes account of both within and between graph information to learn graph representations for imbalanced graph classification.

- The framework is instantiated into a novel model, called MOSGNN. The model is specifically designed to learn the minority graphs at multiple scales using graph convolutional networks jointly optimized with subgraph-level MIL, graph-level classification, and pairwise graph relation prediction. The two auxiliary tasks offer much stronger inductive knowledge than the current graph augmentation operation-based methods.

- Comprehensive experiments on 16 imbalanced graph datasets with various imbalanced ratios show that MOSGNN (i) significantly outperforms five competing methods (Section 5.4.3), and (ii) offers a generic framework, in which different advanced imbalanced learning loss functions and GNN backbones can be easily plugged in and obtain significantly improved classification performance (Sections 5.4.4 and 5.4.5).

In the rest of this chapter, the proposed framework and its instantiation MOSGNN are introduced in Section 5.2. A theoretical analysis is presented in Section 5.3. Experimental results are provided in Section 5.4. This chapter is concluded in Section 5.5.

5.2 The Proposed MOSGNN Model

5.2.1 Framework

To solve the imbalanced graph classification problem, we propose a novel deep multi-scale oversampling framework that augments GNNs with oversampling at three different scales, *i.e.*, subgraph, graph, and pairwise graph relation, to learn discriminative representations of the minority graphs. The key idea here is to learn graph representations from multiple-scale oversampling to capture diverse discriminative information of the minority class graphs. To this end, we develop a framework with three classification objectives from the perspectives of subgraph, graph and pairwise inter-graph scales, respectively. The objective of each classification branch helps achieve an oversampling of a particular scale. The final classification model is based on the combined results of three branches that are jointly optimized. Particularly, the overall objective of our framework can be generally formulated as:

$$\mathscr{L} = L^g + \lambda L^p + \beta L^s, \tag{5.1}$$

where L^g, L^p and L^s are the loss functions for enforcing oversampling from the graphs, pairwise graphs, and subgraphs respectively and λ and β are hyperparameters that balance the importance of the three scales.

The overall procedure of the proposed framework is illustrated in Figure 5.2, which consists of the following three components:

- *Graph-level oversampling via standard graph classification*. In this component, a standard oversampling method is applied to the original dataset. The oversampled

dataset is used to train a GNN-based classifier with a loss function L^g, yielding a classification logit $\mathbf{r}_g$ for each graph.

- *Pairwise-graph oversampling via pairwise relation prediction*. This module aims to utilize the relation between every pair of graphs to gain discriminative inter-graph-level information for the minority class. The graph pairs can be majority-majority, majority-minority, and minority-minority graph combinations. The relation prediction is to discriminate the majority-majority pairs against the two types of minority-graph-oriented pairs via a loss function L^p. The oversampling is applied at the pairwise graph level such that we have a balanced set of majority-majority graph pairs and the majority-minority or minority-minority graph pairs. The relation prediction will output a prediction logit $\mathbf{r}_p$ for each graph pair.

- *Subgraph-level oversampling via MIL*. This component is focused to apply the oversampling to the subgraph level. It assumes that, given a graph, only a few subgraphs within this graph are relevant to the graph classification. To accommodate this idea, we transform the graph classification into a weakly-supervised subgraph classification where a graph is represented by a bag of subgraphs and graph-level labels are given to perform the classification of the subgraphs. A loss function of top-k multiple instance learning (MIL), L^s, is used to achieve this goal. Oversampling is applied to the minority class samples before the generation of the subgraphs of each graph. Since L^s is enforced on the subgraph samples, the oversampling is equivalent to a subgraph oversampling. This component will combine the prediction logits of subgraph bags to yield a graph-level prediction score $\mathbf{r}_s$ for each graph.

The three loss functions L^g, L^p, and L^s are jointly minimized at the training stage. During inference, three prediction logits $\mathbf{r}_g$, $\mathbf{r}_p$ and $\mathbf{r}_s$ are lastly combined to define the prediction probability of a test graph $\hat{G}$ as

$$\mathbf{r}(\hat{G}) = \mathbf{r}_g + \lambda \mathbf{r}_p + \beta \mathbf{r}_s. \tag{5.2}$$

5.2.2 Multi-scale Oversampling GNN

In this section, an instantiation of the framework, called MOSGNN, is introduced. Various GNNs and pooling operations can be used as the graph representation learning backbone. In our model, the commonly used GCN with the recently proposed MVPool [261] and shared weight parameters is used by default in all three branches.

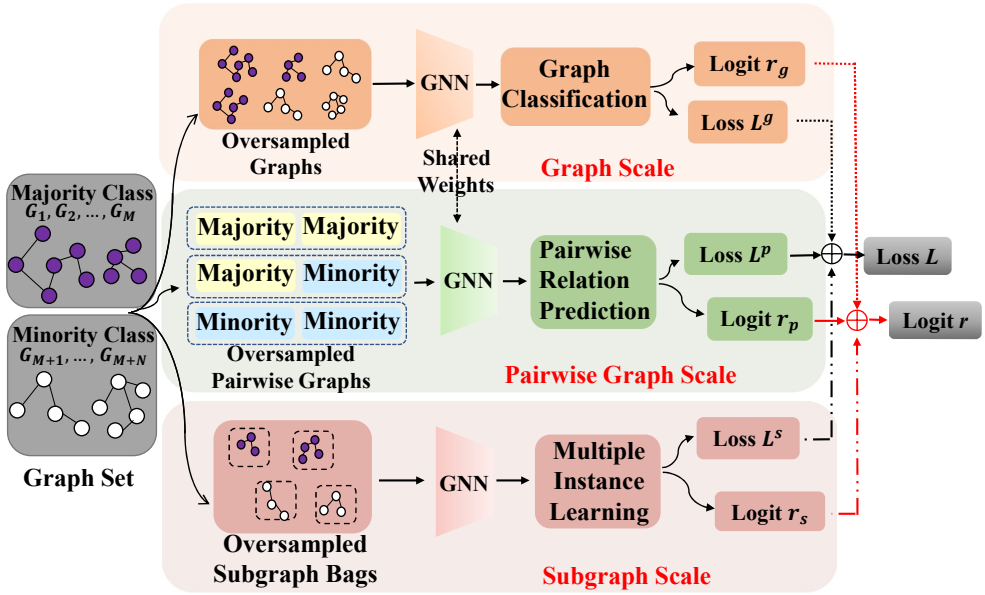


Figure 5.2: Overview of the proposed framework. It augments and trains a GNN model with oversampled graph data at the subgraph, graph, and pairwise inter-graph levels, to capture diversified intra- and inter-graph information for the classification of minority graphs. To achieve this goal, two auxiliary objectives, *i.e.*, pairwise graph relation prediction and subgraph-based MIL, are combined with the standard graph classification objective to jointly optimize the GNN.

Graph-level Oversampling

A general graph-level oversampling is used to address the class imbalance in the learning of the graph representations via GNNs. Specifically, this branch feeds the balanced graph embeddings obtained from a GCN backbone into a multi-layer perception (MLP) classifier to calculate the probability of each graph belonging to each class as follows:

$$\mathbf{s}_i^g = \mathrm{MLP}^g(\mathbf{h}_i; W_g), \tag{5.3}$$

where $\mathbf{h}_i = \mathrm{GCN}(G_i; W_{gcn})$ is the embedding of a graph G_i from the oversampled graph set. To optimize the GCN and MLP layers, a standard cross-entropy loss function is used:

$$L^g = \sum_i \mathrm{CE}(\mathbf{s}_i^g, y_i), \tag{5.4}$$

where $\{y_i\}_i$ is the label set and $\mathrm{CE}(\cdot)$ is the cross entropy loss function. This standard loss enforces the MOSGNN model to learn discriminative representations from the full graphs. The two auxiliary loss functions below complement this loss function to learn

more discriminative graph representations from the inter-graph relation and subgraph perspectives.

Pairwise-graph-level Oversampling

The pairwise graph oversampling is devised to extend the classification information of the minority graphs by exploiting the pairwise inter-graph interactions. To this end, a self-supervised learning task is designed to reformulate the primary classification problem as a pairwise relation prediction task. In particular, we first pair the majority and minority graphs into three types of graph pairs, including majority-majority, majority-minority, and minority-minority pairs, creating a new pairwise graph set $\mathscr{P} = \{(\mathbf{h}_i, \mathbf{h}_j), y_{ij} | \mathbf{h}_i, \mathbf{h}_j \in \mathscr{H}\}$, where $\mathscr{H}$ is the graph embedding set produced by a GCN backbone, $(\mathbf{h}_i, \mathbf{h}_j)$ is a graph pair randomly sampled from the oversampled dataset, and y_{ij} is a surrogate label of the pair $(\mathbf{h}_i, \mathbf{h}_j)$. Each pair $(\mathbf{h}_i, \mathbf{h}_j)$ has one of the three pairwise relations: $P_{maj,maj}$, $P_{maj,min}$ and $P_{min,min}$. To significantly extend the minority class data and balance the pairwise graph samples, the objective is further reformulated as a new binary classification task, aiming at discriminating majority graph pairs $P_{maj,maj}$ from minority-graph-related pairs $P_{maj,min}/P_{min,min}$ instead. In doing so, we leverage the interactions within the minority graphs and between the minority and majority graphs to generate a substantially large number of minority-graph-oriented pair samples. Learning to differentiate $P_{maj,maj}$ from $P_{maj,min}/P_{min,min}$ would then capture discriminative information of the minority graphs at the inter-graph level. More specifically, the pairwise graph learning is formally defined as:

$$L^p = \sum_{(i,j)\in\mathscr{P}} \mathrm{CE}(\mathbf{s}_{ij}^p, y_{ij}), \tag{5.5}$$

where $\mathbf{s}_{ij}^p = \mathscr{R}^p(\mathbf{h}_i, \mathbf{h}_j; W_p)$ and $\mathbf{h}_i = \mathrm{GCN}(G_i; W_{gcn})$ (the same GCN is used to obtain $\mathbf{h}_j$), $\mathscr{R}^p(\cdot,\cdot;W)$ is the relation learning function, and y_{ij} is the relation label of pair $(\mathbf{h}_i, \mathbf{h}_j)$. Since the intrinsic relationship between samples can be diverse across the datasets, it is difficult to explicitly define it. In MOSGNN, we employ a minority graph prediction network to learn it as:

$$\mathbf{s}_{ij}^p = \mathrm{MLP}^p(\mathbf{h}_i \odot \mathbf{h}_j; W_p), \tag{5.6}$$

where $\odot$ is the concatenate operation and $y_{ij} = 0$ if $(\mathbf{h}_i, \mathbf{h}_j)$ is a $P_{maj,maj}$ pair and $y_{ij} = 1$ otherwise.

Subgraph-level Oversampling

Graph has rich local information, *e.g.*, subgraph information can support the accurate classification of graphs, while using full graph may fail due to the presence of possible noisy nodes or subgraphs in the graphs. Considering this, we further devise a subgraph-level oversampling to improve the graph classification results. This equals to a weakly-supervised classification task where graph-level labels are given to perform the classification of subgraphs, with each graph represented by a bag of subgraphs. Thus, we employ a widely-used weakly-supervised learning technique MIL to implement this component.

Specifically, we first randomly remove some nodes and edges to generate a bag of q subgraphs $\{G_i^n\}_{n=1}^q$ for each graph G_i. A GCN with parameters W_{sgcn} is then applied to each subgraph bag to gain their embeddings $\mathcal{B}_{G_i} = \{\mathbf{h}_i^n\}_{n=1}^q$, where $\mathbf{h}_i^n$ is the embedding of subgraph G_i^n. Considering the small number of minority data, we also pair the embedding bags, *i.e.*, majority-majority, majority-minority and minority-minority subgraph pairs to produce more minority data. Similar to pair bags in pairwise graph oversampling, the majority-majority pair bags are regarded as one class and the other two types of pairs are treated as another class. That is, the subgraph bags are paired with each other, resulting in $\mathcal{B} = \{(\mathcal{B}_{G_i}, \mathcal{B}_{G_j}), y_{ij}^s\}$, where $y_{ij}^s = 0$ if $(\mathcal{B}_{G_i}, \mathcal{B}_{G_j})$ is a $P_{maj,maj}$ pair and $y_{ij}^s = 1$ otherwise. The pairwised subgraph bags focus on discriminative substructures while the graphs in pairwise level are centered on interaction between graphs. They can find similar patterns, but they also complement to each other.

In implementing the MIL task, we employ the recently proposed feature magnitude learning-based top-k MIL approach [198]. Specifically, given a pair of subgraph bags $(\mathcal{B}_{G_i}, \mathcal{B}_{G_j})$, the top-k MIL learning is implemented to map the pair to a prediction score by:

$$\mathbf{s}_{ij}^s = \text{MIL}((\mathcal{B}_{G_i}, \mathcal{B}_{G_j}); W_s) = \text{mean}(\mathbf{S}_{ij}^k; W_s), \tag{5.7}$$

where $\mathbf{S}_{ij}^k = topk(\{\mathbf{s}_i^n\}_{n=1}^q \cup \{\mathbf{s}_j^n\}_{n=1}^q)$ are the top-k prediction scores of the predictions from both subgraph bags $\mathcal{B}_{G_i}$ and $\mathcal{B}_{G_j}$, and $s_i^n = \text{MLP}^s(\mathbf{h}_i^n; W_s)$ is the prediction score of a single subgraph $\mathbf{h}_i^n \in \mathcal{B}_{G_i}$. Note that $\text{MLP}^s(\cdot)$ takes individual bags of subgraph embeddings, rather than the concatenated embeddings of subgraphs to avoid prohibitive computational cost, since there are a large number of possible subgraph pairs per two graphs. The pairwise subgraph interaction is captured instead at the output layer via the top-k prediction across the two subgraph bags. Then, the binary cross-entropy loss is applied to optimize the MIL model:

$$L_{mil} = \sum_{(i,j)\in\mathcal{B}} \text{CE}(\mathbf{s}_{ij}^s, y_{ij}^s). \tag{5.8}$$

In [198], it is shown that enforcing a feature magnitude-based gap between the classes at the feature representation layers can further improve the classification performance. We accordingly employ this regularization here as:

$$L_{reg} = \sum_{(i,j)\in\mathscr{B}} \max\Big(0, m - g_k(\mathscr{B}_{G_i}) + g_k(\mathscr{B}_{G_j})\Big), \tag{5.9}$$

where G_i is a minority class graph, G_j is a majority class graph, m is a pre-defined margin, and $g_k(\cdot)$ denotes the mean of the top-k L_2-norm values across the subgraph embeddings in each bag. Thus, the overall loss function of the subgraph-scale branch is as:

$$L^s = L_{mil} + \eta L_{reg}, \tag{5.10}$$

where η is a parameter that balances the two loss terms.

Training and Inference

At the training stage, we first use oversampling and GCNs (with MVPool-based graph pooling) to generate class-balanced batches of graph/subgraph embeddings in each branch, and then jointly optimize the following overall loss:

$$\mathscr{L} = \underbrace{L^g}_{graphs} + \underbrace{\lambda L^p}_{pairwise\,graphs} + \underbrace{\beta L^s}_{subgraphs}, \tag{5.11}$$

where L^g, L^p, and L^s are respectively specified as Eq. (5.4), (5.5), and (5.10), and λ and β are hyperparameters that balance the influence of the three components on the overall classification. It yields a MOSGNN model with optimal parameters $W^* = \{W^*_{gcn}, W^*_{sgcn}, W^*_g, W^*_p, W^*_s\}$.

Algorithm 2 presents the procedure of training MOSGNN. After the data oversampling in Step 1, MOSGNN performs stochastic gradient descent-based optimization to train GCNs and three MLPs in Steps 2-11. Particularly, Step 4 computes the graph-scale loss, Steps 5-6 calculate the pairwise-scale loss and Steps 7-8 obtain the subgraph-scale loss. Step 9 gathers the losses and Step 10 performs gradient descent steps on the loss to optimize the parameters in GCNs and MLPs. We finally gain the graph representation learning networks-GCN(W^*_{gcn}), GCN(W^*_{sgcn}), and MLPs in each scale-MLP^g(W^*_g), MLP^p(W^*_p), MLP^s(W^*_s).

During inference, given a test graph sample G, its graph embedding should be paired with other graph embeddings to extract relationships between them. We evaluated the pair of the test graph with itself and graphs randomly sampled from the training

Algorithm 2 Training MOSGNN

Input: Imbalanced training graph set $\mathscr{G} = \{G_i\}_i$

Output: Two graph representation learning networks-GCN(W^*_{gcn}), GCN(W^*_{sgcn}), MLP in each scale-MLP^g(W^*_g), MLP^p(W^*_p), MLP^s(W^*_s)

1: Randomly oversample the graph set $\mathscr{G}$ to obtain the balanced set
2: **for** $j = 1$ to $n\_batches$ **do**
3: Compute graph representations $\{\mathbf{h}_G\}_G$ by using GCN(W_{gcn})
4: Compute graph-scale prediction probability $\mathbf{s}^g_i$ with Eq. (5.3) and further loss function with Eq. (5.4)
5: Randomly pairwise $\{\mathbf{h}_G\}_G$ to obtain pairwise graph set $\mathscr{P}$
6: Compute pairwise-interaction prediction probability $\mathbf{s}^p_{i,j}$ with Eq. (5.6) and further loss function with Eq. (5.5)
7: Establish subgraph bags and obtain subgraph embedding bags by using GCN(W_{sgcn})
8: Compute subgraph-scale loss with Eqs. (5.7)-(5.10)
9: Gather the losses from three scales to calculate the final loss with Eq. (5.11)
10: Perform a gradient descent step on Eq. (5.11)
11: **end for**
12: **return** The graph representation learning networks-GCN(W^*_{gcn}), GCN(W^*_{sgcn}), MLP in each scale-MLP^g(W^*_g), MLP^p(W^*_p), MLP^s(W^*_s)

data and obtained similar results. Therefore, we choose to easily pair test graphs with themselves to obtain the prediction of the pairwise scale. Finally, the classification probability predicted by MOSGNN for G is defined as:

$$\mathbf{r}(G) = \text{MLP}^g(\mathbf{h}; W^*_g) + \lambda \text{MLP}^p(\mathbf{h} \odot \mathbf{h}; W^*_p) + \beta \text{MIL}(\mathscr{B}_G, W^*_s), \tag{5.12}$$

where $\mathbf{h} = \text{GCN}(G, W^*_{gcn})$. Class with maximum probability in $\mathbf{r}(G)$ is the class of G predicted by MOSGNN.

Enabling Other Imbalanced Learning Loss Functions

There have been many imbalanced learning loss functions adaptively assigning larger weights to the minority samples during training. We found empirically that the performance of these advanced loss functions can be further improved under our MOSGNN framework by simply plugging these losses to replace the cross entropy loss in MOSGNN (see Section 5.4.4).

5.3 Theoretical Analysis

The time complexity of the proposed MOSGNN is analysed. The graph-scale oversampling is the vanilla oversampling-based GNN whose computational requirements come from the oversampling operation, GNN and the MLP classifier. Let N be the number of majority graphs in the original data set. Our oversampling operation is simply duplicating data and its complexity is $O(N)$. Since the GNN encoder can be various and we use O_{GNN} to stand for its complexity. The MLP classifier whose layer number and dimension are constant requires $O(N)$ time. The extra computational requirements are due to the pairwise-graph-scale and subgraph-scale modules. As for the pairwise graph scale, the oversampling and GNN operations can use the results from the graph scale. Both the graph pairing and relation prediction steps require $O(N)$. Therefore, the pairwise graph scale module needs $O(N)$ additional time. As in [219], the complexity of subgraph augmentation is linearly proportional to the size of graph and imposes no additional time compared with GNN encoder. The GNN operation needs extra O_{GNN} time, while the MIL module takes an extra $O(N)$ time for score calculation and $O(kN\log q) \approx O(N)$ for the selection of top-k scores in each bag. Therefore, the total time complexity of MOSGNN is $O(N)+O_{GNN}$. For example, with GCN as GNN backbone, the time complexity will be $O(N(M+1))$, where M denotes the maximal number of edges in graphs.

5.4 Experiments and Results

5.4.1 Competing Methods

There are three exisiting GNN-based imbalanced graph classification methods, *i.e.*, mixup [217], G^{2}GNN [219] and SOLT-GNN[133]. The construction of GoG in G^{2}GNN relies on GraKeL [192], which supports only small datasets. Since most of our datasets have a large scale (NCI and Tox datasets have 30K and 8K samples) and they are not supported by GraKeL, we cannot apply G^{2}GNN to these datasets. Besides, the imbalance issue in SOLT-GNN is on the graph size, rather than the class imbalance, which is different from our setting. Therefore, our MOSGNN is compared to five state-of-the-art (SOTA) competing methods from four different directions: i) **Re-sampling:** Oversampling [73] and SMOTE [22]; ii) **Data Augmentation:** Mixup [217]; iii) **Re-weighting:** FocalLoss [123]; and iv) **Logit Adjustment:** LALoss [149].

5.4.2 Parameter Settings

The following parameters are set by default for MOSGNN and its competing methods on all 16 datasets: the batch size is set to 256, the number of GCN layers combined with MVPool is 3, the dimension of hidden layer in the MVPool-based graph pooling is 128, the pooling ratio of MVPool is 0.8, and the number of epochs is 200. The learning rate is 10^{-3} for all method except LALoss, which use a linear learning rate warming up in the first five epochs to reach the base learning rate 0.1 as recommended in [149]. All methods are trained with Adam except LALoss, which is trained with the SGD optimizer following [149]. All methods select the best decision threshold with the classification probability varying from 0.3 to 0.9 using a step size of 0.1. The Mixup hyperparameter used to combine two graphs into one is selected from $\{0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$. The parameters λ and β in MOSGNN are selected from $\{1.0, 0.5, 0.25, 0.0\}$. The parameter q in the subgraph branch of MOSGNN is set to 10, while the other three parameters m, k and η are set to 100, 3 and 0.0001 respectively, as suggested in [198]. Other parameters in the competitors are set as suggested in their original works.

5.4.3 Comparison to SOTA Models

The results of comparing MOSGNN to five competing methods are reported in Table 5.1. Our MOSGNN performs best on all datasets. The improvement of MOSGNN compared with the best competing method per dataset is large on most datasets, *e.g.*, NCI33 (2.9%), NCI109* (4.3%), NCI81 (4.4%), BZR (4.7%), NCI41 (5.1%), COX2 (5.5%), Aromatase (6%) and ATAD5 (7.8%). Oversampling, FocalLoss and LALoss are the three most effective competing methods, but they cannot well leverage the intra- and inter-graph structure information, learning less discriminative representations. MOSGNN achieves this by the multi-scale oversampling and largely outperforms them. The p-value results indicate that the improvement of MOSGNN over all the competitors is significant at the 99% confidence level, demonstrating the superiority of MOSGNN on graph datasets with diverse imbalanced ratios.

In addition, we report the macro-F1 score and AUPRC of the proposed MOSGNN and its competitors in Table 5.2 and Table 5.3. It is obvious that the proposed model achieves top-1 performance and the paired signed-rank test indicates the improvement across the 16 datasets is significant at the 99% confidence level, which further demonstrates the better ability of MOSGNN to discriminate the imbalanced samples.

Table 5.1: F1 score (mean±std) of MOSGNN and five SOTA competing methods. #*pos* and #Graph denote the number of graph samples in the minority class and in the full dataset, respectively. 'Ratio' denotes the ratio of the majority class size to the minority class. 'Rank' indicates the average performance ranking of a model across all datasets: a smaller rank value indicates a better overall performance. The best performance per dataset is boldfaced.

Dataset	#*pos*	#Graph	Ratio	SMOTE	Mixup	Oversampling	FocalLoss	LALoss	MOSGNN
NCI1*	1793	37349	19.8 : 1	0.253 ± 0.030	0.384 ± 0.038	0.447 ± 0.037	0.425 ± 0.033	0.371 ± 0.003	**0.477 ± 0.023**
NCI33	1467	37022	24.2 : 1	0.208 ± 0.014	0.336 ± 0.023	0.439 ± 0.020	0.415 ± 0.016	0.318 ± 0.040	**0.468 ± 0.011**
NCI41	1350	25336	17.8 : 1	0.252 ± 0.021	0.382 ± 0.040	0.432 ± 0.021	0.415 ± 0.010	0.302 ± 0.005	**0.482 ± 0.015**
NCI47	1735	37298	20.5 : 1	0.263 ± 0.002	0.316 ± 0.032	0.428 ± 0.009	0.383 ± 0.022	0.324 ± 0.003	**0.448 ± 0.019**
NCI81	2081	37549	17.0 : 1	0.274 ± 0.029	0.359 ± 0.033	0.437 ± 0.019	0.428 ± 0.026	0.380 ± 0.009	**0.481 ± 0.013**
NCI83	1959	25550	12.0 : 1	0.288 ± 0.009	0.340 ± 0.035	0.412 ± 0.022	0.428 ± 0.002	0.354 ± 0.015	**0.440 ± 0.014**
NCI109*	1773	37518	20.2 : 1	0.264 ± 0.023	0.347 ± 0.036	0.424 ± 0.006	0.389 ± 0.011	0.339 ± 0.009	**0.466 ± 0.020**
NCI123	2715	36903	12.6 : 1	0.270 ± 0.025	0.291 ± 0.082	0.389 ± 0.017	0.400 ± 0.011	0.352 ± 0.009	**0.426 ± 0.013**
NCI145	1641	37043	21.6 : 1	0.272 ± 0.022	0.320 ± 0.037	0.451 ± 0.001	0.409 ± 0.028	0.335 ± 0.015	**0.468 ± 0.018**
BZR	86	405	3.7 : 1	0.520 ± 0.062	0.485 ± 0.033	0.542 ± 0.030	0.459 ± 0.052	0.502 ± 0.032	**0.589 ± 0.025**
COX2	102	467	3.6 : 1	0.328 ± 0.048	0.283 ± 0.201	0.438 ± 0.057	0.343 ± 0.245	0.089 ± 0.126	**0.494 ± 0.055**
P388	2298	41472	17.1 : 1	0.323 ± 0.022	0.522 ± 0.004	0.547 ± 0.013	0.532 ± 0.016	0.465 ± 0.009	**0.561 ± 0.013**
Aromatase	360	7226	19.1 : 1	0.044 ± 0.063	0.089 ± 0.069	0.222 ± 0.086	0.101 ± 0.074	0.172 ± 0.058	**0.282 ± 0.091**
ATAD5	338	9091	25.9 : 1	0.061 ± 0.048	0.000 ± 0.000	0.120 ± 0.058	0.139 ± 0.055	0.146 ± 0.016	**0.223 ± 0.027**
ER	937	7697	7.2 : 1	0.049 ± 0.042	0.124 ± 0.049	0.195 ± 0.028	0.137 ± 0.048	0.088 ± 0.018	**0.201 ± 0.023**
p53	537	8634	15.1 : 1	0.230 ± 0.047	0.162 ± 0.115	0.211 ± 0.026	0.255 ± 0.029	0.220 ± 0.024	**0.258 ± 0.050**
			Rank	5.4	4.8	2.4	3.1	4.3	1
			p-value	0.0004	0.0004	0.0004	0.0004	0.0004	-

Table 5.2: Macro-F1 score (mean±std) of MOSGNN and five SOTA competing methods on 16 real-world imbalanced graph datasets. 'Rank' indicates the average performance ranking of a model across all datasets: a smaller rank value indicates a better overall performance.

Dataset	#*pos*	#Graph	Ratio	SMOTE	Mixup	Oversampling	FocalLoss	LALoss	MOSGNN
NCI1*	1793	37349	19.8 : 1	0.608 ± 0.015	0.679 ± 0.018	0.708 ± 0.023	0.699 ± 0.016	0.668 ± 0.001	**0.726 ± 0.012**
NCI33	1467	37022	24.2 : 1	0.583 ± 0.011	0.657 ± 0.011	0.709 ± 0.010	0.696 ± 0.008	0.646 ± 0.021	**0.724 ± 0.006**
NCI41	1350	25336	17.8 : 1	0.601 ± 0.010	0.675 ± 0.018	0.700 ± 0.010	0.693 ± 0.005	0.633 ± 0.005	**0.727 ± 0.006**
NCI47	1735	37298	20.5 : 1	0.610 ± 0.001	0.645 ± 0.015	0.699 ± 0.004	0.678 ± 0.012	0.645 ± 0.002	**0.710 ± 0.010**
NCI81	2081	37549	17.0 : 1	0.607 ± 0.017	0.663 ± 0.016	0.701 ± 0.008	0.697 ± 0.014	0.674 ± 0.004	**0.725 ± 0.008**
NCI83	1959	25550	12.0 : 1	0.604 ± 0.004	0.647 ± 0.016	0.684 ± 0.011	0.689 ± 0.003	0.654 ± 0.007	**0.693 ± 0.009**
NCI109*	1773	37518	20.2 : 1	0.612 ± 0.011	0.661 ± 0.018	0.698 ± 0.005	0.680 ± 0.006	0.653 ± 0.005	**0.721 ± 0.010**
NCI123	2715	36903	12.6 : 1	0.602 ± 0.012	0.625 ± 0.039	0.672 ± 0.011	0.679 ± 0.007	0.652 ± 0.005	**0.692 ± 0.006**
NCI145	1641	37043	21.6 : 1	0.617 ± 0.011	0.648 ± 0.018	0.713 ± 0.001	0.691 ± 0.014	0.651 ± 0.006	**0.722 ± 0.010**
BZR	86	405	3.7 : 1	0.691 ± 0.053	0.678 ± 0.015	0.709 ± 0.028	0.659 ± 0.025	0.677 ± 0.012	**0.725 ± 0.043**
COX2	102	467	3.6 : 1	0.532 ± 0.086	0.565 ± 0.090	0.635 ± 0.013	0.596 ± 0.113	0.465 ± 0.037	**0.648 ± 0.028**
P388	2298	41472	17.1 : 1	0.642 ± 0.011	0.749 ± 0.001	0.759 ± 0.008	0.754 ± 0.009	0.718 ± 0.004	**0.767 ± 0.009**
Aromatase	360	7226	19.1 : 1	0.489 ± 0.032	0.521 ± 0.034	0.567 ± 0.038	0.518 ± 0.032	0.552 ± 0.027	**0.601 ± 0.057**
ATAD5	338	9091	25.9 : 1	0.502 ± 0.024	0.476 ± 0.000	0.521 ± 0.021	0.535 ± 0.024	0.529 ± 0.008	**0.568 ± 0.020**
ER	937	7697	7.2 : 1	0.481 ± 0.010	0.525 ± 0.024	**0.558 ± 0.014**	0.523 ± 0.021	0.509 ± 0.006	0.557 ± 0.008
p53	537	8634	15.1 : 1	0.549 ± 0.031	0.552 ± 0.058	0.565 ± 0.015	0.580 ± 0.003	0.566 ± 0.012	**0.584 ± 0.051**
			Rank	5.7	4.4	2.3	3.1	4.4	1.1
			p-value	0.0004	0.0004	0.0005	0.0004	0.0004	-

Table 5.3: AUPRC results (mean±std) of MOSGNN and five SOTA competing methods on 16 real-world imbalanced graph datasets. 'Rank' indicates the average performance ranking of a model across all datasets: a smaller rank value indicates a better overall performance.

Dataset	$\#pos$	# Graph	Ratio	SMOTE	Mixup	Oversampling	FocalLoss	LALoss	MOSGNN
NCI1*	1793	37349	19.8 : 1	0.189 ± 0.016	0.348 ± 0.025	0.401 ± 0.033	0.379 ± 0.030	0.304 ± 0.013	**0.413 ± 0.031**
NCI33	1467	37022	24.2 : 1	0.142 ± 0.020	0.275 ± 0.028	0.383 ± 0.009	0.341 ± 0.025	0.267 ± 0.026	**0.407 ± 0.015**
NCI41	1350	25336	17.8 : 1	0.175 ± 0.010	0.336 ± 0.007	0.361 ± 0.024	0.363 ± 0.027	0.253 ± 0.007	**0.417 ± 0.015**
NCI47	1735	37298	20.5 : 1	0.173 ± 0.004	0.298 ± 0.015	0.356 ± 0.010	0.333 ± 0.020	0.266 ± 0.011	**0.401 ± 0.013**
NCI81	2081	37549	17.0 : 1	0.190 ± 0.026	0.317 ± 0.025	0.387 ± 0.013	0.389 ± 0.025	0.330 ± 0.022	**0.423 ± 0.027**
NCI83	1959	25550	12.0 : 1	0.208 ± 0.006	0.322 ± 0.023	0.375 ± 0.014	0.388 ± 0.008	0.331 ± 0.016	**0.403 ± 0.004**
NCI109*	1773	37518	20.2 : 1	0.170 ± 0.009	0.320 ± 0.025	0.383 ± 0.007	0.354 ± 0.020	0.270 ± 0.009	**0.411 ± 0.019**
NCI123	2715	36903	12.6 : 1	0.196 ± 0.016	0.315 ± 0.005	0.349 ± 0.027	0.379 ± 0.023	0.302 ± 0.011	**0.387 ± 0.012**
NCI145	1641	37043	21.6 : 1	0.177 ± 0.013	0.314 ± 0.008	0.386 ± 0.026	0.373 ± 0.033	0.263 ± 0.024	**0.404 ± 0.009**
BZR	86	405	3.7 : 1	0.521 ± 0.066	0.460 ± 0.040	0.550 ± 0.041	0.536 ± 0.050	0.476 ± 0.032	**0.629 ± 0.047**
COX2	102	467	3.6 : 1	0.366 ± 0.055	0.350 ± 0.066	0.457 ± 0.064	0.403 ± 0.126	0.286 ± 0.028	**0.461 ± 0.072**
P388	2298	41472	17.1 : 1	0.270 ± 0.022	0.508 ± 0.018	0.530 ± 0.019	0.521 ± 0.020	0.449 ± 0.003	**0.558 ± 0.008**
Aromatase	360	7226	19.1 : 1	0.101 ± 0.012	0.164 ± 0.031	0.158 ± 0.023	0.148 ± 0.010	0.157 ± 0.037	**0.217 ± 0.073**
ATAD5	338	9091	25.9 : 1	0.119 ± 0.020	0.121 ± 0.014	0.166 ± 0.018	0.148 ± 0.010	0.119 ± 0.013	**0.193 ± 0.060**
ER	937	7697	7.2 : 1	0.108 ± 0.009	0.142 ± 0.028	**0.199 ± 0.020**	0.162 ± 0.028	0.142 ± 0.006	0.147 ± 0.013
p53	537	8634	15.1 : 1	0.184 ± 0.026	0.260 ± 0.042	0.222 ± 0.021	**0.261 ± 0.008**	0.202 ± 0.010	0.205 ± 0.022
			Rank	5.7	4.1	2.3	2.7	4.9	1.3
			p-value	0.0004	0.0009	0.0072	0.0045	0.0004	-

5.4.4 Enabling Different Loss Functions

This section evaluates the applicability of our model to enable other imbalanced learning loss functions. In detail, we replace the cross-entropy loss in MOSGNN with FocalLoss and LALoss, named as MOS-Focal and MOS-LA respectively. The results are reported in Table 5.4, showing that MOS-Focal and MOS-LA substantially outperform FocalLoss and LALoss respectively on all datasets except ATAD5 and p53 for FocalLoss and p53 for LALoss. The average improvement is 4.6% and 7.6% and the maximal improvement can be up to about 13% and 33% w.r.t. FocalLoss and LALoss respectively. The paired signed-rank test indicates the improvement across the 16 datasets is significant at 99% confidence level. The performance drop of MOS-Focal w.r.t. FocalLoss on Aromatase are very marginal, having only a difference of 0.28%. The decreased performance on p53 is relatively large, since the subgraph-scale oversampling in MOSGNN works adversely on p53 (as shown in Table 5.8) and this effect is further enlarged likely due to the sample weighting schemes in two losses.

We also report the F1-macro score result of classification when using MOSGNN-enabled FocalLoss and LALoss in Table 5.5. We can see from Table 5.5 that the MOSGNN structure brings improvement on most datasets, which is consistent with the F1 score result above.

Table 5.4: F1 score (mean) using MOSGNN-enabled FocalLoss and LALoss, with the original FocalLoss and LALoss as baselines. 'Diff.' denotes the F1 score improvement (↑) or decrease (↓) of 'MOS-X' compared to the original 'X' loss.

Dataset	Focal	MOS-Focal	Diff.	LA	MOS-LA	Diff.
NCI1*	0.4250	0.4971	0.0721 ↑	0.3712	0.4179	0.0467 ↑
NCI33	0.4153	0.4558	0.0405 ↑	0.3179	0.3962	0.0783 ↑
NCI41	0.4154	0.4503	0.0349 ↑	0.3016	0.4211	0.1195 ↑
NCI47	0.3829	0.4342	0.0513 ↑	0.3240	0.3778	0.0538 ↑
NCI81	0.4282	0.4859	0.0577 ↑	0.3799	0.4193	0.0394 ↑
NCI83	0.4281	0.4453	0.0172 ↑	0.3543	0.4263	0.0720 ↑
NCI109*	0.3892	0.4788	0.0896 ↑	0.3388	0.4302	0.0914 ↑
NCI123	0.4004	0.4267	0.0263 ↑	0.3516	0.3888	0.0372 ↑
NCI145	0.4086	0.4604	0.0518 ↑	0.3346	0.4105	0.0759 ↑
BZR	0.4591	0.5296	0.0705 ↑	0.5024	0.5688	0.0664 ↑
COX2	0.3426	0.4725	0.1299 ↑	0.0889	0.4177	0.3288 ↑
P388	0.5319	0.5628	0.0309 ↑	0.4646	0.5379	0.0733 ↑
Aromatase	0.1005	0.1554	0.0549 ↑	0.1721	0.2230	0.0509 ↑
ATAD5	0.1393	0.1365	−0.0028 ↓	0.1455	0.1739	0.0284 ↑
ER	0.1373	0.2610	0.1237 ↑	0.0875	0.1848	0.0973 ↑
p53	0.2550	0.1536	−0.1014 ↓	0.2202	0.1736	−0.0466 ↓
p-value	0.0061	-		0.0009	-	

Table 5.5: Macro-F1 score (mean) using MOSGNN-enabled FocalLoss and LALoss, with the original FocalLoss and LALoss as baselines. 'Diff.' denotes the F1 score improvement (↑) or decrease (↓) of 'MOS-X' compared to the original 'X' loss.

Dataset	Focal	MOS-Focal	Diff.	LA	MOS-LA	Diff.
NCI1*	0.6991	0.7365	0.0374↑	0.6684	0.6875	0.0191↑
NCI33	0.6956	0.7169	0.0213↑	0.6464	0.6746	0.0282↑
NCI41	0.6925	0.7117	0.0192↑	0.6332	0.6956	0.0624↑
NCI47	0.6776	0.6998	0.0222↑	0.6452	0.6643	0.0191↑
NCI81	0.6970	0.7269	0.0299↑	0.6735	0.6906	0.0171↑
NCI83	0.6892	0.7015	0.0123↑	0.6535	0.6918	0.0383↑
NCI109*	0.6799	0.7267	0.0468↑	0.6529	0.7024	0.0495↑
NCI123	0.6790	0.6926	0.0136↑	0.6521	0.6709	0.0188↑
NCI145	0.6913	0.7192	0.0279↑	0.6505	0.6871	0.0366↑
BZR	0.6592	0.7042	0.0450↑	0.6773	0.7199	0.0426↑
COX2	0.5956	0.6385	0.0429↑	0.4649	0.6161	0.1512↑
P388	0.7535	0.7687	0.0152↑	0.7177	0.7553	0.0376↑
Aromatase	0.5184	0.4974	-0.0210↓	0.5516	0.5608	0.0092↑
ATAD5	0.5354	0.5323	-0.0031↓	0.5292	0.5448	0.0156↑
ER	0.5227	0.5827	0.0600↑	0.5092	0.5462	0.0370↑
p53	0.5799	0.5412	-0.0387↓	0.5663	0.5319	-0.0344↓

5.4.5 Enabling Different GNN Backbones

This section measures the applicability of MOSGNN framework to other GNN backbones. We replace the GCN in our model with GIN [232] and GAT [203] respectively and use sum/mean pooling as the readout module. Since the Oversampling performs best in the

competitors, we only compare Oversampling with our method in this experiment. The results are shown in Table 5.6. It is obvious that the usage of GIN brings improvement on most datasets for both MOSGNN and Oversampling and our MOSGNN has better results on all datasets. The enhancement brought by GAT is limited, which might be because of the usage of MVPool with GCN in original MOSGNN. However, our MOSGNN still has better performance than Oversampling on all datasets. We also report the AUPRC results of replacing GCN in MOSGNN with GIN and GAT in Table 5.7 to further demonstrate this.

Table 5.6: F1 score results (mean±std) of using GIN/GAT as backbones. The best performance is boldfaced. 'Baseline' denotes 'Oversampling'.

Dataset	**GIN**		**GAT**	
	Baseline	**MOSGNN**	**Baseline**	**MOSGNN**
NCI1*	0.541 ± 0.006	$\mathbf{0.542 \pm 0.033}$	0.440 ± 0.035	$\mathbf{0.472 \pm 0.016}$
NCI33	0.489 ± 0.008	$\mathbf{0.534 \pm 0.013}$	0.417 ± 0.022	$\mathbf{0.456 \pm 0.022}$
NCI41	0.509 ± 0.007	$\mathbf{0.537 \pm 0.018}$	0.433 ± 0.008	$\mathbf{0.458 \pm 0.004}$
NCI47	0.498 ± 0.026	$\mathbf{0.512 \pm 0.004}$	0.399 ± 0.015	$\mathbf{0.453 \pm 0.016}$
NCI81	0.534 ± 0.011	$\mathbf{0.540 \pm 0.017}$	0.459 ± 0.019	$\mathbf{0.474 \pm 0.021}$
NCI83	0.489 ± 0.015	$\mathbf{0.504 \pm 0.007}$	0.436 ± 0.018	$\mathbf{0.457 \pm 0.017}$
NCI109*	0.490 ± 0.023	$\mathbf{0.527 \pm 0.009}$	0.438 ± 0.023	$\mathbf{0.442 \pm 0.015}$
NCI123	0.459 ± 0.025	$\mathbf{0.498 \pm 0.016}$	0.417 ± 0.015	$\mathbf{0.431 \pm 0.016}$
NCI145	0.536 ± 0.009	$\mathbf{0.545 \pm 0.010}$	0.438 ± 0.012	$\mathbf{0.467 \pm 0.012}$
BZR	0.503 ± 0.044	$\mathbf{0.586 \pm 0.017}$	0.486 ± 0.078	$\mathbf{0.557 \pm 0.063}$
COX2	0.443 ± 0.055	$\mathbf{0.502 \pm 0.040}$	0.530 ± 0.046	$\mathbf{0.539 \pm 0.074}$
P388	0.614 ± 0.014	$\mathbf{0.622 \pm 0.007}$	0.547 ± 0.013	$\mathbf{0.566 \pm 0.008}$
Aromatase	0.238 ± 0.059	$\mathbf{0.285 \pm 0.009}$	0.236 ± 0.027	$\mathbf{0.307 \pm 0.030}$
ATAD5	0.241 ± 0.074	$\mathbf{0.281 \pm 0.088}$	0.187 ± 0.093	$\mathbf{0.231 \pm 0.047}$
ER	0.165 ± 0.026	$\mathbf{0.207 \pm 0.037}$	0.156 ± 0.040	$\mathbf{0.224 \pm 0.042}$
p53	0.220 ± 0.005	$\mathbf{0.225 \pm 0.027}$	0.205 ± 0.077	$\mathbf{0.229 \pm 0.030}$
p-value	0.0004	-	0.0004	-

5.4.6 Sample Efficiency

As the minority samples are typically difficult or costly to obtain, this section examines the performance of our model w.r.t. decreasing sample size of the minority class in training data, *i.e.*, sample efficiency. We perform the experiment using 1%, 5%, 10%, 25%, 50% and 100% of training minority samples, respectively. The experiment is focused on the nine NCI datasets, with $\lambda = 1$ and $\beta = 1$ by default in MOSGNN; the competing methods use the hyperparameter settings that are optimal on the original training data per dataset. The F1 score results are displayed in Figure 5.3. MOSGNN obtains consistently higher F1 scores than the five competing methods in almost all cases. This superiority is particularly clear when only 5% and 1% minority samples are used. This

Table 5.7: AUPRC results (mean±std) of using GIN/GAT as backbones. The best performance is boldfaced. 'Baseline' denotes 'Oversampling'.

Dataset	GIN		GAT	
	Baseline	**MOSGNN**	**Baseline**	**MOSGNN**
NCI1*	$\mathbf{0.514 \pm 0.014}$	0.503 ± 0.020	0.372 ± 0.033	$\mathbf{0.420 \pm 0.007}$
NCI33	0.433 ± 0.012	$\mathbf{0.466 \pm 0.015}$	0.368 ± 0.014	$\mathbf{0.394 \pm 0.021}$
NCI41	0.462 ± 0.012	$\mathbf{0.490 \pm 0.026}$	0.356 ± 0.022	$\mathbf{0.393 \pm 0.011}$
NCI47	$\mathbf{0.471 \pm 0.005}$	0.461 ± 0.015	0.359 ± 0.015	$\mathbf{0.411 \pm 0.023}$
NCI81	$\mathbf{0.503 \pm 0.031}$	0.494 ± 0.039	0.412 ± 0.024	$\mathbf{0.421 \pm 0.015}$
NCI83	0.451 ± 0.031	$\mathbf{0.475 \pm 0.017}$	0.388 ± 0.005	$\mathbf{0.422 \pm 0.012}$
NCI109*	0.459 ± 0.014	$\mathbf{0.478 \pm 0.019}$	0.378 ± 0.011	$\mathbf{0.387 \pm 0.025}$
NCI123	0.404 ± 0.040	$\mathbf{0.463 \pm 0.005}$	0.371 ± 0.022	$\mathbf{0.401 \pm 0.015}$
NCI145	0.493 ± 0.011	$\mathbf{0.512 \pm 0.004}$	0.376 ± 0.010	$\mathbf{0.427 \pm 0.006}$
BZR	$\mathbf{0.606 \pm 0.035}$	0.590 ± 0.055	$\mathbf{0.573 \pm 0.018}$	0.546 ± 0.061
COX2	0.476 ± 0.067	$\mathbf{0.478 \pm 0.055}$	0.530 ± 0.063	$\mathbf{0.551 \pm 0.104}$
P388	0.594 ± 0.025	$\mathbf{0.609 \pm 0.014}$	0.539 ± 0.004	$\mathbf{0.547 \pm 0.004}$
Aromatase	$\mathbf{0.221 \pm 0.005}$	0.218 ± 0.031	0.168 ± 0.013	$\mathbf{0.207 \pm 0.039}$
ATAD5	0.217 ± 0.068	$\mathbf{0.258 \pm 0.063}$	0.160 ± 0.026	$\mathbf{0.214 \pm 0.042}$
ER	0.127 ± 0.011	$\mathbf{0.138 \pm 0.014}$	0.139 ± 0.014	$\mathbf{0.152 \pm 0.011}$
p53	0.156 ± 0.006	$\mathbf{0.178 \pm 0.036}$	0.168 ± 0.014	$\mathbf{0.194 \pm 0.034}$
p-value	0.0174	-	0.0019	-

might be because that MOSGNN is able to utilize the fine-grained subgraph information and the majority graph samples via MIL and pairwise relation prediction respectively, which helps alleviate the impacts of the decreasing minority data to some extent and maintain the consistent improvement.

5.4.7 Robustness w.r.t. Label Noise

Noisy class labels are one common challenge to supervised tasks. The challenge is much greater in imbalanced classification if the minority class samples are incorrectly labeled as the majority one. This section evaluates the capability of the models in handling this challenge. Specifically, we randomly flip a subset of the labels of the minority samples to the majority class label, with the subset size ranging from 0% up to 30% of the minority samples. Other experiment settings are as in Section 5.4.6.

The F1 scores of the six methods under different levels of label noise are reported in Figure 5.4. The results show that our model MOSGNN achieves the best robustness on all nine datasets, outperforming all the competing methods under almost all label noise levels across the nine datasets. It is impressive that, compared to the rapid decreasing performance of the other methods like Mixup, FocalLoss and LALoss, the performance of MOSGNN and Oversampling is barely affected by the increasing label noises on most of the datasets, indicating the advantage of the oversampling methods over the

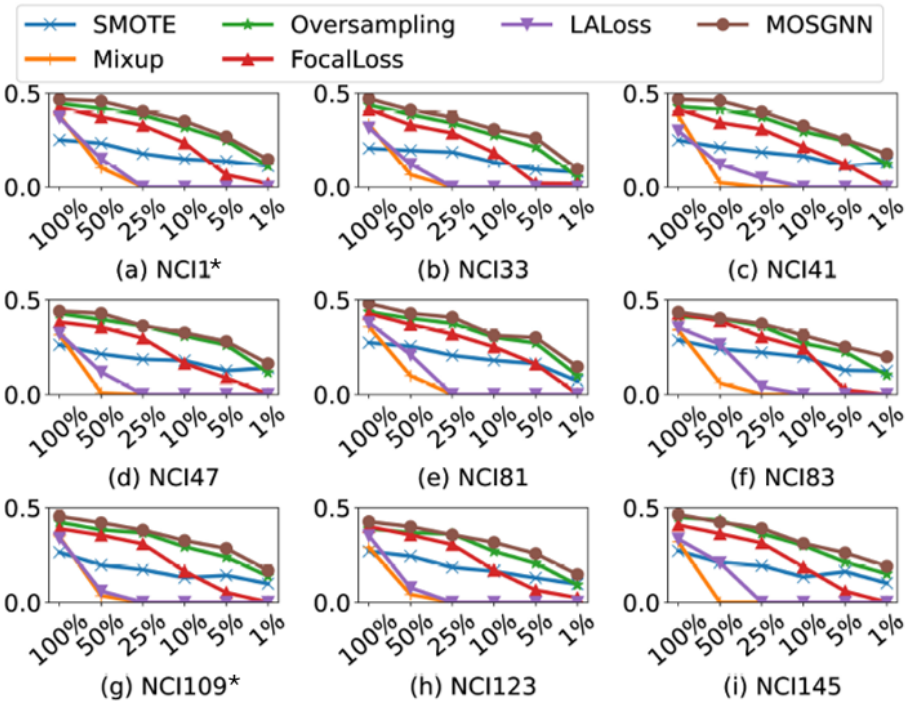


Figure 5.3: F1 score (y-axis) on nine NCI datasets with decreasing training data.

data generation and sample weighting (imbalanced loss) based methods in handling label noises. MOSGNN is substantially better than Oversampling due to its additional subgraph and inter-graph oversampling.

5.4.8 Ablation Study

This section examines the importance of three oversampling modules in our model. We first evaluate three oversampling branches separately (L^g, L^p, or L^s), and then incrementally add pairwise-graph-scale and subgraph-scale branches until we obtain the full model MOSGNN. The results are reported in Table 5.8. It is shown that using graph-scale (L^g) or pairwise-graph-scale oversampling (L^p) individually can achieve good performance. Jointly optimizing L^g and L^p further improves the individual performance, indicating the complementary information gained from two types of oversampling. Using L^s only cannot classify samples successfully, since the subgraph-based classification discards many nodes per graph, but it can still achieve fairly good F1 scores on some datasets, indicating that important information in some datasets are actually embedded in subgraphs. Therefore, combining L^s with L^g or L^p can obtain better performance than using single scale on those datasets. The worse performance obtained by combinations

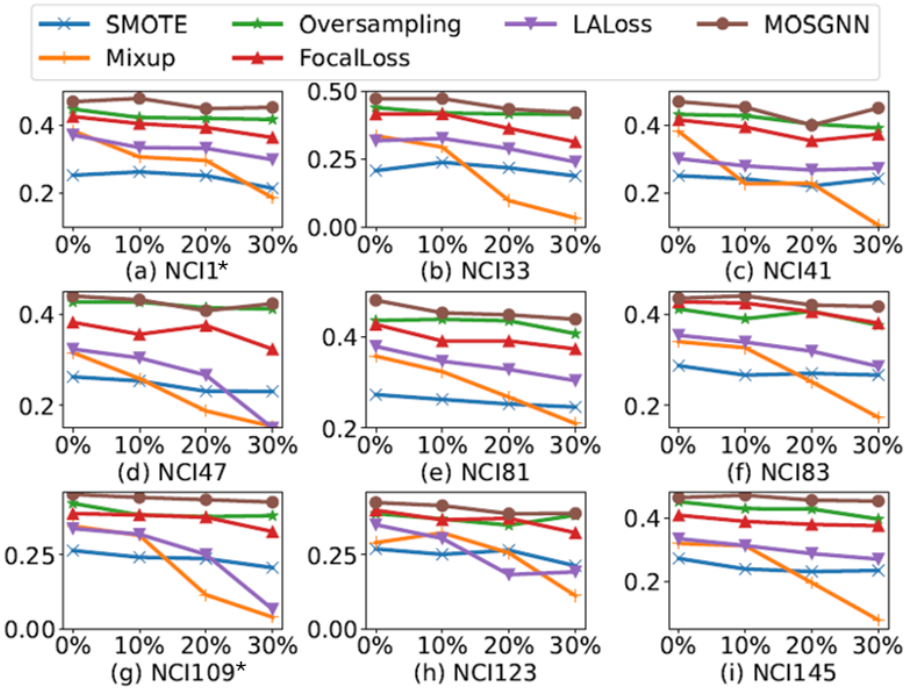


Figure 5.4: F1 score (y-axis) on nine NCI datasets with different levels of label noise.

of two scales in L^g, L^p and L^s might be because that pairs of the three scales do not capture the full graph semantic well and part of their results are not consistent. This problem can be alleviated when three scales are used together, which is exactly what we have in our MOSGNN. Overall, the three oversampling done by L^g, L^p, and L^s helps significantly enhance the representations of minority graphs over their individual/pairwise uses, indicating they all have important contributions to the superior performance of MOSGNN.

We also evaluate the effects of two hyperparameters (λ and β) to the performance of MOSGNN. In Figure 5.5, we show the results with $\beta = 1.0$ fixed and varying λ in $\{0, 0.25, 0.5, 1\}$, and vice versa. Clearly, larger λ can bring better results in most datasets, which indicates that the information gained from the pairwise-graph-scale oversampling is general useful for the performance of MOSGNN. On the other hand, more efforts are required to tune the parameter β for an effective use of the subgraph-scale oversampling. Nevertheless, both parameters can be well tuned on the validation dataset; the results here suggest that more careful tuning of β is desired in practice.

Table 5.8: F1 score (mean±std) of MOSGNN and its ablated variants. The best performance per dataset is boldfaced. 'Rank' indicates the average performance ranking of a model across all datasets: a smaller rank value indicates a better overall performance.

Dataset	L^g	L^p	L^s	L^g **&** L^p	L^g **&** L^s	L^p **&** L^s	MOSGNN
NCI1*	0.447 ± 0.037	0.455 ± 0.010	0.215 ± 0.152	0.438 ± 0.028	0.457 ± 0.032	0.468 ± 0.041	**0.477 ± 0.023**
NCI33	0.439 ± 0.020	0.444 ± 0.023	0.209 ± 0.148	0.429 ± 0.006	0.441 ± 0.008	0.438 ± 0.032	**0.468 ± 0.011**
NCI41	0.432 ± 0.021	0.437 ± 0.038	0.283 ± 0.019	0.452 ± 0.016	0.438 ± 0.006	0.461 ± 0.027	**0.482 ± 0.015**
NCI47	0.428 ± 0.009	0.402 ± 0.036	0.196 ± 0.139	0.426 ± 0.016	0.387 ± 0.043	0.433 ± 0.025	**0.448 ± 0.019**
NCI81	0.43pm0.019	0.450 ± 0.011	0.206 ± 0.146	0.456 ± 0.017	0.440 ± 0.020	0.439 ± 0.010	**0.481 ± 0.013**
NCI83	0.412 ± 0.022	0.421 ± 0.029	0.183 ± 0.134	**0.440 ± 0.014**	0.420 ± 0.011	0.414 ± 0.014	**0.440 ± 0.014**
NCI109*	0.424 ± 0.006	0.426 ± 0.008	0.296 ± 0.045	0.458 ± 0.014	0.440 ± 0.008	0.445 ± 0.020	**0.466 ± 0.020**
NCI123	0.389 ± 0.017	0.409 ± 0.008	0.296 ± 0.011	0.425 ± 0.015	0.391 ± 0.014	0.403 ± 0.004	**0.426 ± 0.013**
NCI145	0.451 ± 0.001	0.451 ± 0.015	0.201 ± 0.142	0.465 ± 0.011	0.447 ± 0.007	0.448 ± 0.013	**0.468 ± 0.018**
BZR	0.542 ± 0.030	0.576 ± 0.009	0.353 ± 0.048	**0.589 ± 0.025**	0.515 ± 0.025	0.583 ± 0.045	**0.589 ± 0.025**
COX2	0.438 ± 0.057	0.452 ± 0.058	0.353 ± 0.086	0.468 ± 0.077	0.405 ± 0.069	0.456 ± 0.047	**0.494 ± 0.055**
P388	0.547 ± 0.013	0.551 ± 0.012	0.275 ± 0.195	**0.561 ± 0.013**	0.546 ± 0.017	0.530 ± 0.017	**0.561 ± 0.013**
Aromatase	0.222 ± 0.086	0.259 ± 0.082	0.084 ± 0.060	0.243 ± 0.079	0.125 ± 0.033	0.206 ± 0.025	**0.282 ± 0.091**
ATAD5	0.120 ± 0.058	0.200 ± 0.064	0.171 ± 0.059	**0.223 ± 0.027**	0.186 ± 0.065	0.182 ± 0.091	**0.223 ± 0.027**
ER	0.195 ± 0.028	0.152 ± 0.050	0.095 ± 0.038	**0.201 ± 0.023**	0.173 ± 0.022	0.191 ± 0.053	**0.201 ± 0.023**
p53	0.211 ± 0.026	**0.263 ± 0.028**	0.137 ± 0.028	0.258 ± 0.050	0.255 ± 0.026	0.241 ± 0.026	0.258 ± 0.050
Rank	4.9	3.6	6.9	2.4	4.7	4.0	1.1
p-value	0.0002	0.0004	0.0002	0.0156	0.0002	0.0002	-

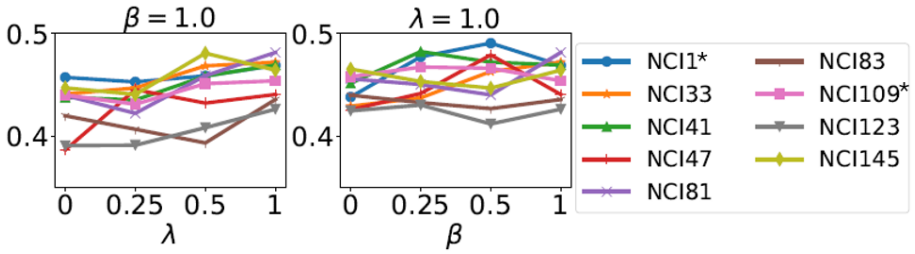


Figure 5.5: F1 scores (y-axis) of MOSGNN with different hyperparameter (λ and β) settings on nine NCI datasets.

5.5 Summary

This chapter introduces a novel deep multi-scale oversampling approach MOSGNN for imbalanced graph classification. It can significantly extend the graph samples of the minority class with rich intra- and inter-graph semantics. These semantic-augmented minority graphs enable more effective training of GNNs on imbalanced graphs. The experiments show that MOSGNN learns significantly improved imbalanced graph classifiers over SOTA competing models.

In addition, the sample efficiency experiment demonstrates that MOSGNN performs better than other competitors when the sample number of the minority class is insufficient. Moreover, the robustness experiment proves that MOSGNN achieves the best

robustness when suffering from different level of sample contamination. Thus, MOSGNN can have more stable performance when the samples are a little unreliable.

We further find that MOSGNN can be a generic framework, in which different advanced imbalanced learning loss functions and GNN backbones can be easily plugged in and obtain significantly improved classification performance.

However, the inter-graph relationship learning via pairwise operation is defined as binary classification, and thus the proposed MOSGNN only supports binary classification. In many real-world applications, the number of classes might exceed two. The proposed method cannot be employed to solve such tasks. How to define the inter-graph relationship between multi-class samples and establish efficient scheme to extract the relationships needs further research.

CHAPTER 6

GRAPH-LEVEL ANOMALY DETECTION WITH GLOBAL AND LOCAL KNOWLEDGE DISTILLATION

6.1 Introduction

Chapter 4 and Chapter 5 discuss the graph-level classification problems with sufficient and imbalanced samples. However, there is an extreme case, in which only one type of samples are known. This classification problem formulates one-class classification that aims to classify other samples from the known class of samples. We also call this problem graph-level anomaly detection (GLAD) when regard the known type as normal class and other types are abnormal.

Despite the prevalence of graph data and the importance of anomaly detection therein, GLAD has received little attention compared to anomaly detection in other types of data [2; 164]. One primary challenge in GLAD is to learn expressive graph representations that capture local and global normal patterns in the graph structure and attributes (*e.g.*, descriptive features of nodes). This is essential for the detection of both locally-anomalous graph – relating to individual nodes and their local neighborhood (G_5 in Figure 6.1) – and globally-anomalous graph – relating to holistic graph characteristics (G_6 in Figure 6.1).

In this chapter, inspired by Knowledge Distillation (KD) [74; 18; 206], we introduce a novel deep anomaly detection approach for GLAD that learns both global and local normal patterns by *joint random distillation* of graph and node representations – global

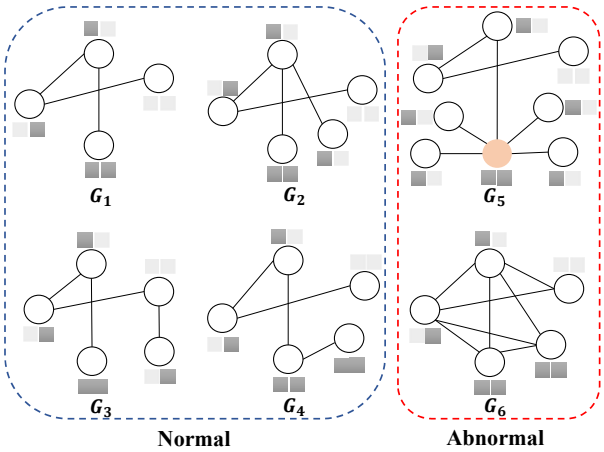
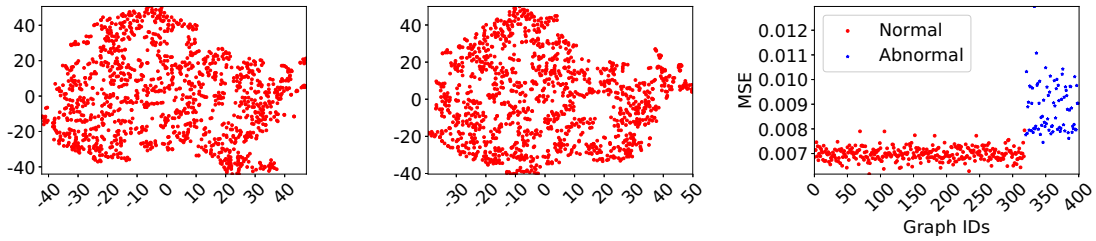


Figure 6.1: A set of graphs with two anomalous graphs indicated. The squares above/-below the nodes represent node features. G_5 is a locally-anomalous graph due to the unusual local properties (*e.g.*, structure) of the orange node, while G_6 is a globally-anomalous graph because it does not conform to G_1 to G_4 in holistic graph properties.

and local (*i.e.*, glocal) graph representation distillation. The random representation distillation is done by training one GNN to predict a *random GNN* that has its neural network weights fixed to random initialization, *i.e.*, the predictor network learns to produce the same representations as that in the random network, as shown in Figure 6.2(a) and (b). To accurately predict these fixed randomly-projected representations, the predictor network is enforced to learn all major patterns in the training data. By applying such a random distillation on both graph and node representations, our model learns glocal graph patterns across the given training graphs. When the training data consists of exclusively (or mostly) normal graphs, the learned patterns are a summarization of multi-scale graph regularity/normality information. As a result, given a graph that shows node/graph-level irregularity/abnormality w.r.t. these learned patterns, the model cannot accurately predict its representations, leading to a much larger prediction error than that of normal graphs, as shown in Figure 6.2(c). Thus, this prediction error can be defined as anomaly score to detect the aforementioned two types of graph anomalies.

Accordingly, this chapter makes the following major contributions:

- The GLAD problem is formulated as the task of detecting locally- or globally-anomalous graphs, and the presence of these two types of graph anomalies in real-world datasets is empirically verified (Section 6.4.7).
- A novel deep anomaly detection framework that models *glocal graph regularity* and learns graph anomaly scores in an end-to-end fashion is introduced. This results in



(a) Random features (b) Learned features (c) Prediction errors

Figure 6.2: Demonstration of GLocalKD working on a popular dataset – AIDS. (a) Representations of training graphs output by the random target network. (b) Representations of training graphs learned by the predictor network. (c) Prediction errors (anomaly scores) of GLocalKD on test graphs. Visualization in (a) and (b) is based on t-SNE.

the first approach specifically designed to effectively detect both types of anomalous graphs.

- A new GLAD model, namely Global and Local Knowledge Distillation (GLocalKD), is further instantiated from the framework. GLocalKD implements the joint random distillation of graph and node representations by minimizing the graph- and node-level prediction errors of approximating a random graph convolutional neural network. GLocalKD is easy-to-implement without requiring the challenging graph generation, and it can effectively learn diverse glocal normal patterns with small training data. It also shows remarkable robustness to anomaly contamination, indicating its applicability in both exclusively normal training data setting and anomaly-contaminated unlabeled training data setting.

- Extensive empirical results on 16 real-world datasets from chemistry, medicine, and social network domains show that GLocalKD (i) significantly outperforms seven state-of-the-art competing methods (Section 6.4.3); (ii) is substantially more sample-efficient than other deep detectors (Section 6.4.4), *e.g.*, it can use 95% less training samples to achieve the accuracy that still outperforms the competing methods by a large margin; and (iii) by using a single default GNN architecture, performs very stably w.r.t. different anomaly contamination rates (Section 6.4.5) and the dimensionality of the representations (Section 6.4.6).

The rest of this chapter is organized as follows. The proposed framework and its instantiation GLocalKD are detailed in Section 6.2, followed by a theoretical analysis

in Section 6.3. The evaluation results are provided in Section 6.4. This work is then summarized in Section 6.5.

6.2 The Proposed GLocalKD Method

6.2.1 Framework

To solve the GLAD problem, we propose an end-to-end scoring framework that synthesizes two graph neural networks and joint random knowledge distillation of graph and node representations to train a deep anomaly detector. The resulting model can effectively detect both types of anomalous graphs.

Overview of the Framework

Our framework jointly distills graph-level and node-level representations of each graph, to learn both global and local graph normality information. It consists of two graph neural networks – a fixed randomly initialized target network and a predictor network – with exactly the same architecture and two distillation losses. It learns the holistic (fine-grained) graph normality by training the predictor network to predict the graph (node) level representations produced by the random target network. Let $\mathbf{h}_G$ and $\hat{\mathbf{h}}_G$ respectively be the graph representation of G yielded by the predictor and target networks, and $\mathbf{h}_i$ and $\hat{\mathbf{h}}_i$ be the respective node representation for a node v_i in G produced by the two networks, the overall objective of our approach can be given as:

$$\mathscr{L} = L_{graph} + \lambda L_{node}, \tag{6.1}$$

where λ is a hyperparameter that balances the importance of the two loss functions, L_{graph} and L_{node} are respective graph-level and node-level distillation loss functions:

$$L_{graph} = \frac{1}{|\mathscr{G}|} \sum_{G \in \mathscr{G}} \mathrm{KD}\left(\mathbf{h}_G, \hat{\mathbf{h}}_G\right), \tag{6.2}$$

$$L_{node} = \frac{1}{|\mathscr{G}|} \sum_{G \in \mathscr{G}} \left(\frac{1}{|G|} \sum_{v_i \in \mathscr{V}_G} \mathrm{KD}\left(\mathbf{h}_i, \hat{\mathbf{h}}_i\right) \right), \tag{6.3}$$

where $\mathrm{KD}(\cdot,\cdot)$ is a distillation function that measures the difference between two feature representations and $|\mathscr{G}|$ is the number of graphs in $\mathscr{G}$.

The overall procedure of the training stage of our framework is shown in Figure 6.3, which works as follows:

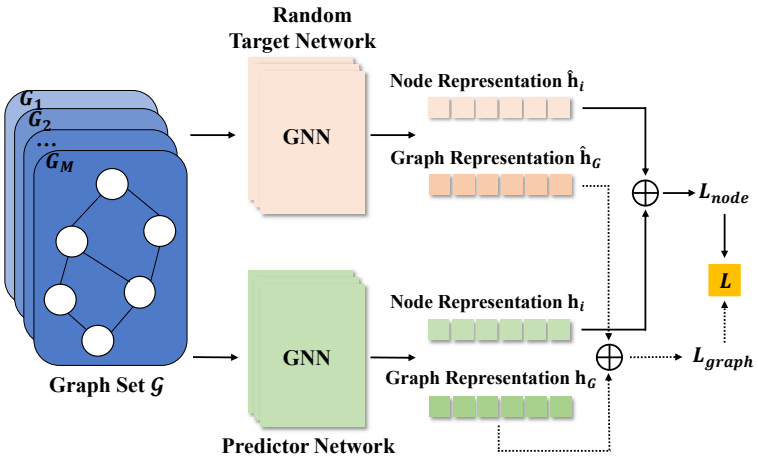


Figure 6.3: The proposed framework.

- We first randomly initialize a graph network $\hat{\phi}(\cdot;W)$ as the target network and fix its weight parameters $\hat{W}$. For every given graph G, it will yield a graph-level representation $\hat{\mathbf{h}}_G$ and node-level representation $\hat{\mathbf{h}}_i$ for each node v_i in G.
- A predictor network $\phi(\cdot;W)$, with the same architecture as $\hat{\phi}$, is parameterized by W and trained to predict the representation outputs of the target network $\hat{\phi}$. That is, for every given graph G, it produces the graph-level representation $\mathbf{h}_G$ and the node-level representation $\mathbf{h}_i$, $\forall v_i \in \mathcal{V}_G$.
- Lastly, for graph G, $\hat{\mathbf{h}}_G$, $\mathbf{h}_G$, $\hat{\mathbf{h}}_i$, and $\mathbf{h}_i$ are integrated into a loss function $\mathscr{L}$, which is minimized to train the predictor network $\phi(\cdot;W)$.

At the evaluation stage, the anomaly score for a given graph G is defined as

$$score(G;\hat{W},W^*) = \mathrm{KD}\left(\mathbf{h}_G,\hat{\mathbf{h}}_G\right) + \lambda\frac{1}{|G|}\sum_{v_i\in\mathcal{V}_G}\mathrm{KD}\left(\mathbf{h}_i,\hat{\mathbf{h}}_i\right), \tag{6.4}$$

where W^* are the learned parameters of the predictor network.

Key Intuition

The graph-level and node-level representations of graphs are learned by GNNs, whose powerful capabilities of capturing graph structure and semantic information have been proved in various learning tasks and applications. The joint random distillation in our framework forces both graph representations and node representations of the predictor network to be as close as possible to the corresponding outputs of the fixed random target

network on normal graph data. This resembles the extraction of different patterns (either frequently or infrequently) presented in the random representations of graphs and nodes, respectively. If a pattern frequently occurs in the random representation space, the pattern would be distilled better, *i.e.*, the prediction error in Eq. (6.2) or (6.3) is small due to a large sample size of the pattern; and the prediction error is large otherwise. As a result, our joint random distillation learns such regularity information from both graph and node representations. For a given test graph G, its anomaly score $score(G;\hat{W},W^*)$ would be large if it does not conform to the regularity information embedded in the training graph set $\mathcal{G}$ at either the graph or the node level, *e.g.*, G_5 and G_6 in Figure 6.1; and $score(G;\hat{W},W^*)$ would be small otherwise, *e.g.*, G_1 – G_4 in Figure 6.1.

6.2.2 Joint Random Distillation of Graph and Node Representations

An instantiation of the proposed framework called Global and Local Knowledge Distillation (GLocalKD) is presented in the following section, in which we use widely-used graph convolutional network (GCN) to learn node and graph representations and the joint distillation is driven by two mean square error-based loss functions.

Random Target Network

We first establish a target network with randomly initialized weights to obtain graph- and node-level representations in the random space. Different graph representation approaches may be used to generate the required representations as the prediction targets of the predictor network. Theoretically, various deep graph networks, such as GCN, GAT and GIN, can be employed as the graph representation learning module. In our work, a standard GCN is used, because GCN and its variants have proved their power to learn expressive features of graphs and good computational efficiency [263; 226].

Specifically, $\hat{\text{GCN}}(\cdot,\hat{W}): G=(\mathcal{V}_G,\mathcal{E}_G) \rightarrow \mathbb{R}^{N_G \times k}$ is a GCN with fixed randomly initialized weights $\hat{W}$ (*i.e.*, the GCN is frozen after random weight initialization), where N_G is the number of nodes in G and k is a predefined dimensionality size of node representations. For each graph $G=(\mathcal{V}_G,\mathcal{E}_G)$, $\hat{\text{GCN}}(\cdot)$ takes adjacency matrix A and feature matrix X as input, and maps each node $v_i \in \mathcal{V}_G$ to the representation space using $\hat{W}$. Let $\hat{\mathbf{h}}_i^{(l)}$ be the hidden representation of node v_i in the l_{th} layer, which is formally computed as

follows:

$$\hat{\mathbf{h}}_i^{(l)} = \text{ReLU}\left(\sum_{v_j \in \widetilde{\mathcal{N}}(v_i)} \frac{1}{\sqrt{\widetilde{D}(i,i)\widetilde{D}(j,j)}} \hat{\mathbf{h}}_j^{(l-1)} \hat{W}^{(l)}\right) \tag{6.5}$$

where $\hat{\mathbf{h}}_j^{(l-1)}$ represents the hidden representation of node v_j in the $(l-1)_{th}$ layer, $\mathcal{N}(v_i)$ denotes the 1_{st}-order neighbors of v_i and $\widetilde{\mathcal{N}}(v_i) = \mathcal{N}(v_i) \cup \{v_i\}$, and the input representation of v_i in the 0_{th} layer, $\hat{\mathbf{h}}_i^{(0)}$, is initialized by its feature vector in X, *i.e.*, $\hat{\mathbf{h}}_i^{(0)} = X(i,:)$. Thus, the output random node representation $\hat{\mathbf{h}}_i$ for node v_i can be written as:

$$\hat{\mathbf{h}}_i = \text{ReLU}\left(\sum_{v_j \in \widetilde{\mathcal{N}}(v_i)} \frac{1}{\sqrt{\widetilde{D}(i,i)\widetilde{D}(j,j)}} \hat{\mathbf{h}}_j^{(L-1)} \hat{W}^{(L)}\right) \tag{6.6}$$

where L is the number of layers of $\hat{\text{GCN}}(\cdot)$. The feature matrix X is composed of node attributes for attributed graphs. For plain graphs, following [256], we use the node degree as the node feature to construct a simple X, since the degree of nodes is one of the key information for the discriminability of nodes and graphs.

Next, a READOUT operation is applied to the node representations to obtain the graph-level representation for G. Considering that our goal is to detect anomalies, we need to aggregate extreme features across the node representations. Thus, the maxpooling is employed in the READOUT operation:

$$\hat{\mathbf{h}}_G = \max\{\hat{\mathbf{h}}_i, \forall v_i \in \mathcal{V}_G\}. \tag{6.7}$$

Predictor Network

The predictor network is a graph network used to predict the output representations of the target network, $\hat{\mathbf{h}}_i$ and $\hat{\mathbf{h}}_G$. We employ a GCN with the exactly same structure as the target network as the predictor network, which is denoted as $\text{GCN}(\cdot, W) : G = (\mathcal{V}_G, \mathcal{E}_G) \rightarrow \mathbb{R}^{N_G \times k}$ with the weight parameters W to be learned. Then, similar to $\hat{\text{GCN}}(\cdot, \hat{W})$, $\text{GCN}(\cdot, W)$ yields the node representation $\mathbf{h}_i$ for node v_i by the following formulation:

$$\mathbf{h}_i = \text{ReLU}\left(\sum_{v_j \in \widetilde{\mathcal{N}}(v_i)} \frac{1}{\sqrt{\widetilde{D}(i,i)\widetilde{D}(j,j)}} \mathbf{h}_j^{(L-1)} W^{(L)}\right) \tag{6.8}$$

After the same READOUT operation as in $\hat{\text{GCN}}(\cdot, \hat{W})$, the graph representation $\mathbf{h}_G$ is computed as follows:

$$\mathbf{h}_G = \max\{\mathbf{h}_i, \forall v_i \in \mathcal{V}_G\}. \tag{6.9}$$

Thus, the only difference between the random target network $\hat{\text{GCN}}(\cdot, \hat{W})$ and the predictor network $\text{GCN}(\cdot, W)$ is that $\hat{W}$ is fixed after random initialization while W needs to be learned through the following glocal knowledge distillation.

Glocal Regularity Distillation

We further perform glocal regularity distillation by minimizing the distance between the (graph- and node-level) representations produced by the predictor network and the target network. Specifically, the graph-level and node-level distillation loss are defined as:

$$L_{graph} = \frac{1}{|\mathcal{G}|} \sum_{G \in \mathcal{G}} \|\mathbf{h}_G - \hat{\mathbf{h}}_G\|^2, \tag{6.10}$$

$$L_{node} = \frac{1}{|\mathcal{G}|} \sum_{G \in \mathcal{G}} \left(\frac{1}{|G|} \sum_{v_i \in \mathcal{V}_G} \|\mathbf{h}_i - \hat{\mathbf{h}}_i\|^2 \right). \tag{6.11}$$

To learn the global and local graph regularity information simultaneously, our model is optimized by jointly minimizing the above two losses:

$$\mathcal{L} = L_{graph} + L_{node}. \tag{6.12}$$

That is, λ in Eq. (6.1) is set to one in Eq. (6.12) since it is believed that it is equivalently important to detect both of locally- and globally-anomalous graphs. We will discuss in Section 6.3 in more details about why our model can learn the global and local graph regularity.

Algorithm 3 presents the procedure of training GLocalKD. After random weight initialization of $\hat{W}$ and W in Step 1, GLocalKD performs stochastic gradient descent-based optimization to learn W of the predictor network in Steps 2-11, while the parameters in $\hat{W}$ are fixed. Particularly, Step 4 samples a mini-batch $\mathcal{B}$ with size $batch\_size$. We obtain node representations and graph representations from both of $\hat{\text{GCN}}(\cdot, \hat{W})$ and $\text{GCN}(\cdot, W)$ in Steps 6-7, respectively. Step 9 then performs gradient descent steps on our loss Eq. (6.12) w.r.t. the parameters in W. We finally obtain the predictor network $\text{GCN}(\cdot, W^*)$ with the learned W^* and the random target network $\hat{\text{GCN}}(\cdot, \hat{W})$.

Anomaly Detection of Using GLocalKD

By joint global and local random distillation, the learned representations in our predictor network capture the regularity information at both the graph and node levels.

Algorithm 3 Training GLocalKD

Input: Normal training graph set $\mathcal{G} = \{G_i\}_i$
Output: Target network – $\hat{\text{GCN}}(\cdot, \hat{W})$, predictor network – $\text{GCN}(\cdot, W^*)$

```
Randomly initialize Ŵ and W, with Ŵ fixed
for i = 1 to n_epochs do
  for j = 1 to n_batches do
    ℬ ←Randomly sample batch_size graphs from 𝒢
    for G in ℬ do
      Compute node representations ĥ_i and h_i, ∀v_i ∈ 𝒱_G
      Compute graph representations ĥ_G and h_G
    end for
    Perform a gradient descent step on Eq. (6.12) w.r.t. the parameters in W
  end for
end for
return ĜCN(·, Ŵ), GCN(·, W*)
```

Specifically, given a test graph sample G, its anomaly score is defined by the prediction errors in both graph and node-level representations:

$$score(G; \hat{W}, W^*) = \left\| \mathbf{h}_G - \hat{\mathbf{h}}_G \right\|^2 + \frac{1}{|G|} \sum_{v_i \in \mathcal{V}_G} \|\mathbf{h}_i - \hat{\mathbf{h}}_i\|^2. \tag{6.13}$$

This indicates that the locally- and globally-anomalous graph anomalies are treated equally important in our anomaly scoring, sharing the same spirit as the overall objective in Eq. (6.12).

6.3 Theoretical Analysis

We show below that GLocalKD can normally produce a larger anomaly score for an abnormal graph than that for a normal one. Specifically, consider a regression problem with data distribution $\hat{\mathcal{G}} = \{G_i, y_i\}_i$ (y_i is the regression target) and a Bayesian setting in which a prior $p(W^\star)$ over the parameters of a GCN, $\text{GCN}(\cdot, W^\star)$, is considered. The aim is to calculate the posterior after iteratively updating on the data. According to [18], our task can then be formulated as the optimization problem below:

$$\min_W \mathrm{E}_{(G_i, y_i) \sim \hat{\mathcal{G}}} \|\text{GCN}(G_i, W) + \text{GCN}(G_i, W^\star) - y_i\|^2 + \mathscr{R}(W), \tag{6.14}$$

where $\mathscr{R}(W)$ is a regularization term from the prior [161]. Let $\mathscr{F}$ be the distribution over functions $f_W = \text{GCN}(\cdot, W) + \text{GCN}(\cdot, W^\star)$, where W is the solution of Eq. (6.14) and $W^\star$ is drawn from $p(W^\star)$, then the ensemble $\mathscr{F}$ can bee seen as an approximation of the posterior [161].

When we select the graphs from the same distribution and set the label y_i to zero, the optimization problem

$$\arg\min_{W} \mathrm{E}_{(G_i,y_i)\sim\hat{\mathcal{G}}} \|\mathrm{GCN}(G_i,W) + \mathrm{GCN}(G_i,W^\star)\|^2 \tag{6.15}$$

is equivalent to distilling a randomly drawn function from the prior. From this perspective, each entry of the representation outputs of the target and the predictor networks would correspond to a part of an ensemble and the prediction error would be an estimate of the predictive variance of the ensemble when the ensemble is assumed to be unbiased, as discussed in [18]. If we consider $\mathrm{GCN}(\cdot,W^\star)$ as the target network with randomly initialized $W^\star$ and regard $\mathrm{GCN}(\cdot,W)$ as the predictor network, the prediction errors of the node representations as well as graph representations in the predictor network would be an estimate of the predictive variance of the results of two networks. In other words, our training process aims to train a predictor network so that the node representations and graph representations of the two networks on each training sample are as close as possible. Then, for the graph with patterns similar to many other training graphs, the prediction errors in Eqs. (6.10) and (6.11) are small, *i.e.*, small predictive variance in Eq. (6.14), because there are sufficient such samples to train the prediction model; the abnormal graphs, by contrast, are drawn from different distributions from the training graphs and dissimilar to most of the training data, leading to large predictive variance in Eq. (6.14). Thus, the prediction errors in our joint random distillation can distinguish both locally- and globally-anomalous graphs from normal graphs.

6.4 Experiments and Results

6.4.1 Competing Methods

Seven competing methods from two types of approaches are used.

- **Two-step Methods.** This approach first uses state-of-the-art graph representation-based methods to obtain vectorized graph representations, and then applies advanced off-the-shelf shallow anomaly detectors on top of the representations to calculate anomaly scores. InfoGraph [195], WL [188] and PK graph kernels [156] are used in our experiments. Anomaly detectors, including iForest [124] and kNN ensemble (LESINN) [165], are utilized. The combination of these embedding methods and detectors leads to six two-step methods.

- **End-to-end Methods**. We also compare GLocalKD with the one-class GCN-based method, namely OCGCN [266], which can be trained in an end-to-end manner as GLocalKD. OCGCN is optimized using a SVDD objective on top of GCN-based representation learning.

We report the mean AUC and standard deviation based on 5-fold cross-validation for all datasets, except HSE, MMP, p53 and PPAR-gamma that have widely-used training and test splits. For these four datasets, the results are based on five runs with different random seeds.

6.4.2 Parameter Settings

The target network and the predictor network in GLocalKD share the same network architecture – a network with three GCN layers. The dimension of the hidden layer is 512 and the output layer has 256 neural units. The learning rate is selected through the grid search, varying from 10^{-1} to 10^{-5}. The batch size is 300 for all data sets except the four largest datasets HSE, MMP, p53 and PPAR-gamma, for which the batch size is 2000. For the competing methods, the network architecture and the optimization of OCGCN is the same as our model. The other methods are taken from their authors. We probed a wide range of hyperparameter settings in both iForest and LESINN. We found that the performance of iForest does not change much with varying hyperparameter settings, while LESINN can obtain large improvement of using one subsampling size setting over the others (see Table A.1 in Appendix A.1). Due to these observations, iForest with subsampling size and the number of trees respectively set to 256 and 100 is used by default, while LESINN with the subsampling size setting that performs best on most of the datasets is used.

6.4.3 Comparison to SOTA Methods

The AUC results of GLocalKD and its seven competing methods are reported in Table 6.1. Our GLocalKD model is the best performer on 7 datasets, achieving improvement ranging from 1% to 12% on many of these datasets compared to the best contenders per dataset, *e.g.*, AIDS (3.7%), PROTEINS_full (6.7%), PPAR-gamma (10.3%), MMP (10.5%), p53 (11.9%); and its performance is very close to the best contenders on some other datasets, such as DD and COLLAB. The consistent superiority of GLocalKD is mainly due to its capability in learning both global and local graph regularity. Its performance may drop significantly, *e.g.*, decrease to performance equivalent to a random detector, if only

one of these patterns is captured (see Table 6.3). The seven competing methods fail to work in many datasets mainly because their graph representations capture only partial local/global pattern information.

We also perform a paired Wilcoxon signed rank test to examine the significance of GLocalKD against each of the competing methods across the 16 datasets. As shown by the p-values in Table 6.1, GLocalKD significantly outperforms the iForest-based methods and OCGCN at the 99% confidence level. The confidence level of the superiority of GLocalKD over LESINN-based methods ranges from 85% and 95%. However, note that LESINN heavily relies on its subsample size (see Table A.1 for the full results of InfoGraph-LESINN, WL-LESINN and PK-LESINN in Appendix A.1). GLocalKD works less effectively on COLLAB than some contenders, which may be due to the inseparability of anomalies from the normal graphs as the contenders also do not perform well on it.

Table 6.1: AUC results (mean±std) on 16 real-world graph datasets. The following acronyms, PROTEINS_full (PROTS_full), IMDB-BINARY (I-BINARY) and REDDIT-BINARY (R-BINARY), are used. The best performance is boldfaced.

Dataset	**InfoGraph**		**WL**		**PK**		**OCGCN**	**GLocalKD**
	iForest	**LESINN**	**iForest**	**LESINN**	**iForest**	**LESINN**		
PROTS_full	0.464±0.019	0.336±0.047	0.639±0.018	0.712±0.053	0.627±0.009	0.572±0.031	0.718±0.036	**0.785**±0.034
ENZYMES	0.483±0.027	0.528±0.046	0.498±0.029	0.624±0.050	0.493±0.013	0.608±0.033	0.613±0.087	**0.636**±0.061
AIDS	0.703±0.036	0.955±0.023	0.632±0.050	0.584±0.016	0.476±0.014	0.421±0.010	0.664±0.080	**0.992**±0.004
DHFR	0.489±0.015	**0.625**±0.028	0.466±0.013	0.596±0.056	0.467±0.013	0.568±0.054	0.495±0.080	0.558±0.030
BZR	0.528±0.060	0.731±0.071	0.533±0.032	0.720±0.032	0.525±0.052	**0.775**±0.063	0.658±0.071	0.679±0.065
COX2	0.580±0.052	0.670±0.079	0.532±0.027	0.590±0.056	0.515±0.036	**0.671**±0.039	0.628±0.072	0.589±0.045
DD	0.475±0.012	0.310±0.034	0.699±0.006	0.638±0.045	0.706±0.010	**0.833**±0.023	0.605±0.086	0.805±0.017
NCI1	0.494±0.009	0.598±0.035	0.545±0.008	**0.743**±0.015	0.532±0.006	0.670±0.012	0.627±0.015	0.683±0.015
I-BINARY	0.520±0.028	0.565±0.017	0.442±0.032	**0.612**±0.046	0.442±0.035	0.585±0.047	0.536±0.148	0.514±0.039
R-BINARY	0.457±0.003	0.262±0.027	0.450±0.013	0.239±0.028	0.450±0.012	0.487±0.013	0.759±0.056	**0.782**±0.016
HSE	0.484±0.026	**0.657**±0.051	0.477±0.000	0.528±0.000	0.489±0.003	0.469±0.016	0.388±0.041	0.591±0.001
MMP	0.539±0.022	0.571±0.037	0.475±0.000	0.307±0.000	0.488±0.002	0.322±0.008	0.457±0.038	**0.676**±0.001
p53	0.511±0.014	0.520±0.025	0.473±0.000	0.390±0.000	0.486±0.004	0.329±0.001	0.483±0.017	**0.639**±0.002
PPAR-gamma	0.521±0.023	0.541±0.036	0.510±0.000	0.461±0.000	0.499±0.017	0.388±0.015	0.431±0.043	**0.644**±0.001
COLLAB	0.453±0.003	0.319±0.033	0.506±0.020	0.536±0.014	0.529±0.023	**0.550**±0.043	0.401±0.183	0.525±0.014
hERG	0.607±0.033	0.701±0.048	0.665±0.042	**0.802**±0.047	0.679±0.034	0.798±0.052	0.569±0.049	0.704±0.049
p-value	0.0005	0.0262	0.0004	0.1089	0.0005	0.1337	0.0018	–

In terms of computational efficiency, we record the training and test time of each method on 3 datasets: REDDIT, p53 and COLLAB. REDDIT and COLLAB are the datasets with most average number of nodes and edges in all 16 datasets, respectively. P53 contains the most number of graphs. The training time of the two-stage methods is only the graph representations/embeddings learning time. The results are shown in Table 6.2. As shown in Table 6.2, GLocalKD and OCGCN have a similar time complexity and run much faster than the other methods in online detection, since iForest/LESINN methods require extra steps on top of the graph representations to compute the anomaly

scores. On the other hand, GLocalKD and OCGCN are generally more computationally costly than the WL and PK based methods because GLocalKD and OCGCN typically require multiple iterations to perform well.

Table 6.2: Training time and test time on 3 datasets: REDDIT-BINARY, p53 and COLLAB (seconds on each epoch).

	Dataset	InfoGraph		WL		PK		OCGCN	GLocalKD
		iForest	LESINN	iForest	LESINN	iForest	LESINN		
Training Time	REDDIT-BINARY	3536.36	3536.36	8.01	8.01	127.09	127.09	1397.40	1395
	p53	60.61	60.61	9.42	9.42	821.29	821.29	297.37	337.80
	COLLAB	2059.66	2059.66	63.24	63.24	416.95	416.95	2421.83	2510.52
Test Time	REDDIT-BINARY	5.79	15.19	3.72	29.37	84.46	112.20	4.65	4.97
	p53	19.67	24.54	225.44	207.95	301.89	250.79	0.66	0.97
	COLLAB	12.41	34.44	39.50	313.65	273.82	573.90	9.28	8.88

6.4.4 Sample Efficiency

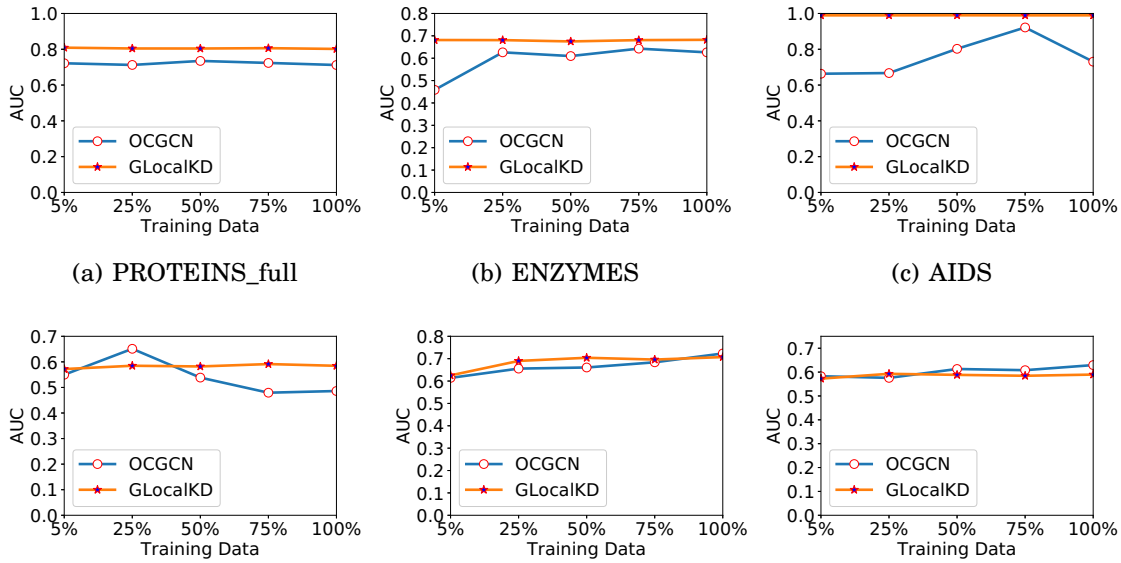
This section examines the performance of our model w.r.t. the amount of training data, *i.e.*, sample efficiency, using the deep competing method OCGCN as baseline. We use respective 5%, 25%, 50%, 75% and 100% of original training samples to train the models, and evaluate the performance on the same test data set. We report the results on the attributed graph datasets only. Similar results can be found on the other datasets.

The AUC results are shown in Figure 6.4. It is very impressive that even when 95% less training data are used, GLocalKD can retain similarly good performance across nearly all the six datasets. By contrast, the performance of OCGCN can drop significantly on some datasets, such as ENZYMES and AIDS, if the same amount of training data is reduced. As a result, GLocalKD can outperform OCGCN by large margins even it uses 95% less training data than OCGCN on such datasets.

6.4.5 Robustness w.r.t. Anomaly Contamination

Recall that we tackle the semi-supervised anomaly detection setting with exclusively normal training samples. However, the data collected in real applications may be contaminated by some anomalies or data noises. This section investigates the robustness of GLocalKD w.r.t. different anomaly contamination levels in the training data. We vary the contamination rates from 0% up to 16%. Again, we report the results on the six attributed graph datasets only due to page limitation; OCGCN is used as baseline.

AUC results of GLocalKD and OCGCN with different anomaly contamination rates are shown in Figure 6.5. GLocalKD is barely affected by the contamination and performs



(a) PROTEINS_full (b) ENZYMES (c) AIDS

(d) DHFR (e) BZR (f) COX2

Figure 6.4: AUC performance of GLocalKD and OCGCN using different amount of training data.

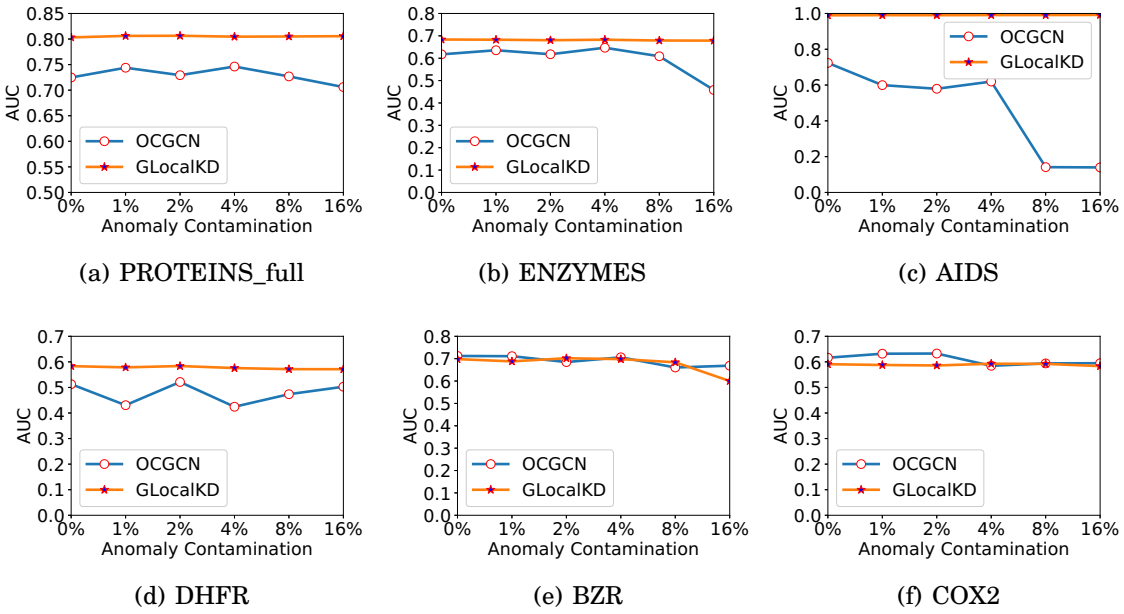
very stably on all the datasets, contrasting to OCGCN whose performance decreases largely with increasing contamination rate on ENZYMES and AIDS. This is mainly because GLocalKD essentially learns all types of patterns in the training data by the random distillation, by which it is able to detect the anomalies as long as those anomalous patterns are not as frequent as the normal patterns in the training data; whereas OCGCN is sensitive since its anomaly measure, SVDD, is sensitive to the anomaly contamination.

6.4.6 Sensitivity Test

This section tests the sensitivity of GLocalKD to the representation dimension and the GCN depth. For the first test, we vary the output dimension of GCN in $\{32, 64, 128, 256, 512\}$; for the GCN depth, we evaluate the performance of GLocalKD using L GCN layers, with $L \in \{1, 2, 3, 5\}$. The results are illustrated in Figures 6.6 and 6.7.

As can be seen from the results, GLocalKD performs stably using different representation dimensionality sizes on most datasets. The dimensionality size – 256 – is generally recommended as this setting enables GLocalKD to perform well on diverse datasets.

Besides, GLocalKD achieves better performance with increasing depth on nearly all the datasets, but the performance is flatten when increasing the depth from three to five. A network depth of three is generally recommended, since deeper GCN does not help



(a) PROTEINS_full (b) ENZYMES (c) AIDS

(d) DHFR (e) BZR (f) COX2

Figure 6.5: AUC performance of GLocalKD and OCGCN w.r.t. different anomaly contamination rates.

achieve better performance but is more computationally costly.

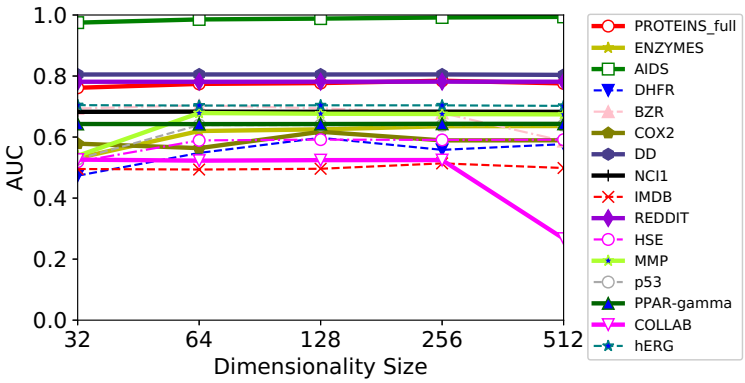


Figure 6.6: AUC results w.r.t. representation dimensionality.

6.4.7 Ablation Study

In this section, we examine the importance of the two components, L_{graph} and L_{node}, in our model. To do that, we derive two variants of GLocalKD, including GLocalKD w/o L_{node} that denotes the use of random distillation on the graph representations only,

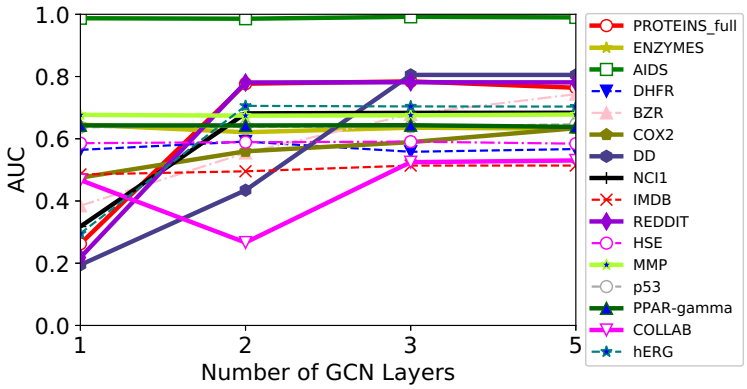


Figure 6.7: AUC of GLocalKD with different GCN depths.

and GLocalKD w/o L_{graph} that represents the use of random distillation on the node representations only.

The results of GLocalKD and its two variants are shown in Table 6.3. It is clear that using L_{graph} (or L_{node}) only can obtain better performance on some datasets, while it may perform worse on the other datasets, compared with GLocalKD. Joint random distillation by using both of L_{graph} and L_{node} can achieve a good trade-off and perform generally good across all the datasets.

It is interesting that GLocalKD w/o L_{graph} significantly outperforms GLocalKD w/o L_{node} in a number of datasets, *e.g.*, AIDS, DHFR, DD, MMP, p53, PPAR-gamma and hERG, indicating the dominant presence of locally-anomalous graphs in those data; on the other hand, the inverse cases occur on ENZYMES, IMDB and HSE, indicating the dominance of globally-anomalous graphs in these three datasets. These results show that modeling fine-grained graph regularity is as important as, if not more important than, the holistic graph regularity for the GLAD task, since both types of graph anomalies can present in the graph datasets.

6.5 Summary

This chapter proposes a novel framework and its instantiation GLocalKD to detect abnormal graphs in a set of graphs. As shown in the experimental results, graph datasets can contain different types of anomalies – locally- and globally-anomalous graphs. To the best of our knowledge, GLocalKD is the first model designed to detect both types of graph anomalies. Extensive experiments demonstrate that GLocalKD performs significantly better in AUC.

Table 6.3: Detection of locally/globally-anomalous graphs.

Dataset	**GLocalKD**	**w/o L_{node}**	**w/o L_{graph}**
PROTEINS_full	**0.785**±0.034	0.686±0.045	0.757±0.040
ENZYMES	0.636±0.061	**0.642**±0.096	0.505±0.036
AIDS	0.992±0.004	0.963±0.014	**0.997**±0.006
DHFR	0.558±0.030	0.459±0.036	**0.596**±0.030
BZR	**0.679**±0.065	0.623±0.079	0.671±0.049
COX2	**0.589**±0.045	0.585±0.051	0.557±0.055
DD	0.805±0.017	0.528±0.093	**0.805**±0.017
NCI1	**0.683**±0.015	0.458±0.058	0.682±0.015
IMDB	0.514±0.039	**0.610**±0.103	0.490±0.044
REDDIT	**0.782**±0.016	0.574±0.085	0.781±0.016
HSE	0.591±0.001	**0.655**±0.007	0.589±0.000
MMP	0.676±0.001	0.543±0.016	**0.680**±0.000
p53	0.639±0.002	0.495±0.016	**0.641**±0.000
PPAR-gamma	0.644±0.001	0.600±0.044	**0.644**±0.000
COLLAB	0.525±0.014	0.501±0.055	**0.526**±0.012
hERG	**0.704**±0.049	0.566±0.043	0.703±0.057

In addition, GLocalKD can be trained much more sample-efficiently when compared with its advanced counterparts. It is also shown that GLocalKD achieves promising AUC performance even when there is large anomaly contamination in the training data, indicating that GLocalKD can be applied to not only settings with exclusively normal training data but also settings with anomaly-contaminated unlabeled training data.

However, the proposed GLocalKD lacks of interpretability. That is, it cannot indicate the type and location of the anomalies. Anomaly explaining and locating can be very useful for human to discover and correct the problem in time. Thus, further research needs to be explored to construct a model which can detect abnormal graphs with various types of anomalies and offer the explanation and location of the outliers.

Part III

Conclusions and Future Directions

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

The specific structure of graph enables it to be a powerful data storage mode in various fields, including social network, biology, chemistry and physics. Numerous real-world applications in these domains can be formulated by problems in graph. Graph representation learning is a significant module for solving these problems efficiently.

To obtain better results for the down-stream tasks, improving the expressiveness of the obtained graph representations is crucial. Besides, how to deal with the limited amonut of supervision information resulting from the expensive data collection process is another issue. In this thesis, we explore graph representation learning for three graph-level classification problems with different ratios of known samples, *i.e.*, balanced graph-level classification, imbalanced graph-level classification and one-class graph-level classification (graph-level anomaly detection). Our explorations result in three expressive graph representation learning modules in Chapters 4-6. A detailed summary is offered as follows:

- We proposed CoS-GNN framework and its two instances, CoS-GCN and CoS-GIN, that utilized collective augmented node and graph structure knowledge and a specific designed message passing mechanism to learn graph representations for *balanced graph-level classification*. Our experiments on data sets from various domains demonstrated remarkable improvement over several state-of-the-art mod-

els, indicating the expressiveness of the learned representations brought by the augmented knowledge.

- To address the data insufficiency issue, we introduced a multi-scale oversampling framework and its instance MOSGNN that learned expressive minority graph representations based on intra- and inter-graph semantics resulting from oversampled graphs at multiple scales - subgraph, graph, and pairwise graphs for *imbalanced graph-level classification*. Excellent experimental results over five state-of-the-art models demonstrated the effect of semantic-augmented minority graphs in MOSGNN.

- We empirically demonstrated that anomalies in graph could be local, *i.e.*, individual abnormal nodes and their first-order neighborhood or global, *i.e.*, abnormal subgraph/full graph characteristics. We introduced a deep framework and its instance GLocalKD that learned rich local and global normal pattern information by joint random distillation of graph and node representations to identify graphs with one or both types of anomalies simultaneously for *graph-level anomaly detection*. Experiments on graph data sets from diverse fields proved the ability of our GLocalKD to capture informative patterns from normal graph representations.

The improvement of the proposed methods indicates that these methods address the varying amount of supervision information well. This provides possible research directions for the academic community to better leverage the limited amount of known information. To the industry, the improvement brought by the proposed methods may mean millions of dollars saving by reducing mistake operations or preventing fraud.

7.2 Future Work

In the previous chapters, we proposed methods to learn graph representations for three graph problems with different amount of supervision information. To build more easy-to-use and more practical methods for more extensive application scenarios, there are several more interesting yet challenging directions remaining exploration.

Large Language Models (LLMs), especially the ChatGPT, impress everyone and start a trend of Artificial Intelligence. Their revolutionary success in natural language processing and computer vision tasks attracts attention to apply LLMs to the graph domain. It is believed to be a potential research direction for solving various graph tasks. Some pioneers have tried to apply LLMs to graphs and obtained improved performance [53].

However, due to the specific characteristics of graph data, there are still some problems remaining to be solved.

Exploring Improving the Ability of GNNs

One worthy research direction is to use LLMs to improve the performance of GNNs. There have been some existing works combining LLMs with GNNs that gain large performance improvement [53]. In these methods, LLMs are often used as enhancers, predictors or alignment components. Such applications may not meet the full potential of LLMs. Deeper fusion of LLMs and GNNs may obtain better performance and remains to be researched.

Another study direction is to establish a large graph model separately. Such research is extremely challenging with the following two problems but significantly meaningful for graph domain.

High-quality Data Collection

The success of LLMs requires rich data for training. Graphs can have different types, *e.g.*, static or dynamic, directed or undirected, weighted or unweighted, attributed or plain. In addition, statistics of graphs, including size, density and degree, also can be various. Collecting samples with varying types and characteristics is helpful for empowering the large graph model address diverse down-stream graphs. However, gathering such large amount of graph data is difficult and expensive. How to accelerate this process and save human resources and cost is a meaningful question required to solve.

Exploring Generalization to Various Applications

The previous graph models only focus on one specific task and lack of generalization ability to other applications. In fact, problems in graph domain formulated from various real-world applications are distinct. For example, the problem can be node-level, edge-level or graph-level; the aim of the problem can be classification, prediction or graph generation. A unified large graph model that can solve all these problems will be a revolutionary finding and be significant to numerous real-world graph applications.

APPENDIX

A.1 The Influence of Subsample Size on LESINN

Table A.1 shows the effect of subsample size on the performance of LESINN. We fix the ensemble size and vary the sumsample size with $\{2, 4, 8, 16, 32, 64, 128, 256\}$ to record the result.

Table A.1: The influence of subsample size on LESINN. 'PROTEINS_full' are shortened by 'PROTS_full'.

Dataset	Method	size=2	size=4	size=8	size=16	size=32	size=64	size=128	size=256
PROTS_full	Info-LESINN	0.399±0.048	0.398±0.049	0.390±0.049	0.380±0.049	0.367±0.047	0.357±0.047	0.345±0.048	0.336±0.047
	WL-LESINN	0.769±0.017	0.779±0.014	0.780±0.014	0.779±0.016	0.777±0.021	0.769±0.029	0.742±0.047	0.712±0.053
	PK-LESINN	0.759±0.023	0.766±0.024	0.769±0.020	0.769±0.018	0.765±0.019	0.702±0.068	0.633±0.057	0.572±0.031
ENZYMES	Info-LESINN	0.465±0.078	0.465±0.072	0.462±0.059	0.465±0.051	0.466±0.042	0.477±0.041	0.496±0.044	0.528±0.046
	WL-LESINN	0.519±0.066	0.488±0.069	0.485±0.065	0.498±0.050	0.518±0.040	0.538±0.037	0.577±0.044	0.624±0.050
	PK-LESINN	0.562±0.053	0.553±0.041	0.564±0.030	0.579±0.025	0.590±0.034	0.596±0.043	0.594±0.038	0.608±0.033
AIDS	Info-LESINN	0.883±0.041	0.889±0.039	0.900±0.038	0.912±0.035	0.924±0.033	0.935±0.030	0.944±0.027	0.955±0.023
	WL-LESINN	0.651±0.016	0.526±0.021	0.437±0.019	0.424±0.012	0.447±0.016	0.483±0.013	0.528±0.014	0.584±0.016
	PK-LESINN	0.578±0.026	0.468±0.037	0.385±0.021	0.352±0.017	0.358±0.010	0.374±0.009	0.393±0.010	0.421±0.010
DHFR	Info-LESINN	0.460±0.042	0.473±0.046	0.486±0.040	0.509±0.042	0.541±0.039	0.575±0.035	0.608±0.034	0.625±0.028
	WL-LESINN	0.365±0.038	0.401±0.047	0.457±0.049	0.480±0.048	0.509±0.052	0.538±0.055	0.573±0.059	0.596±0.056
	PK-LESINN	0.368±0.032	0.400±0.027	0.431±0.021	0.453±0.027	0.474±0.040	0.503±0.051	0.541±0.056	0.568±0.054
BZR	Info-LESINN	0.557±0.043	0.568±0.037	0.600±0.039	0.632±0.039	0.658±0.040	0.690±0.050	0.721±0.068	0.737±0.071
	WL-LESINN	0.540±0.054	0.549±0.050	0.576±0.061	0.620±0.055	0.679±0.053	0.700±0.050	0.717±0.043	0.720±0.032
	PK-LESINN	0.528±0.070	0.542±0.067	0.578±0.075	0.631±0.073	0.693±0.072	0.739±0.068	0.764±0.066	0.775±0.063
COX2	Info-LESINN	0.588±0.064	0.611±0.050	0.616±0.052	0.628±0.058	0.639±0.066	0.661±0.069	0.673±0.066	0.670±0.079
	WL-LESINN	0.444±0.101	0.487±0.089	0.512±0.074	0.557±0.075	0.583±0.073	0.599±0.074	0.605±0.067	0.590±0.056
	PK-LESINN	0.443±0.093	0.465±0.085	0.472±0.080	0.523±0.073	0.568±0.067	0.608±0.061	0.648±0.046	0.671±0.039
DD	Info-LESINN	0.320±0.038	0.318±0.033	0.315±0.032	0.313±0.031	0.310±0.032	0.308±0.032	0.307±0.032	0.310±0.034
	WL-LESINN	0.543±0.052	0.540±0.048	0.535±0.050	0.547±0.055	0.560±0.054	0.578±0.055	0.605±0.051	0.638±0.045
	PK-LESINN	0.800±0.023	0.811±0.028	0.816±0.028	0.819±0.027	0.822±0.026	0.827±0.027	0.831±0.025	0.833±0.023
NCI1	Info-LESINN	0.479±0.016	0.482±0.018	0.487±0.019	0.495±0.023	0.508±0.027	0.532±0.031	0.561±0.034	0.598±0.035
	WL-LESINN	0.533±0.029	0.566±0.029	0.590±0.024	0.621±0.019	0.650±0.015	0.676±0.014	0.710±0.014	0.743±0.015
	PK-LESINN	0.525±0.021	0.542±0.024	0.558±0.019	0.586±0.019	0.607±0.017	0.624±0.015	0.646±0.013	0.670±0.012
IMDB	Info-LESINN	0.431±0.033	0.438±0.033	0.441±0.029	0.467±0.045	0.482±0.043	0.505±0.037	0.541±0.023	0.565±0.017
	WL-LESINN	0.398±0.040	0.397±0.028	0.404±0.028	0.437±0.027	0.504±0.055	0.586±0.058	0.605±0.057	0.612±0.046
	PK-LESINN	0.392±0.045	0.384±0.037	0.385±0.033	0.406±0.023	0.462±0.045	0.552±0.057	0.582±0.050	0.585±0.047
REDDIT	Info-LESINN	0.449±0.023	0.461±0.030	0.418±0.038	0.346±0.048	0.290±0.032	0.276±0.028	0.268±0.027	0.262±0.027
	WL-LESINN	0.231±0.026	0.234±0.026	0.237±0.027	0.239±0.027	0.239±0.027	0.239±0.028	0.239±0.028	0.239±0.028
	PK-LESINN	0.224±0.024	0.295±0.035	0.422±0.017	0.440±0.013	0.448±0.013	0.457±0.010	0.471±0.010	0.487±0.013
HSE	Info-LESINN	0.586±0.116	0.589±0.107	0.596±0.100	0.606±0.092	0.617±0.083	0.629±0.071	0.644±0.060	0.657±0.051
	WL-LESINN	0.341±0.000	0.421±0.000	0.468±0.000	0.482±0.000	0.495±0.000	0.507±0.000	0.518±0.000	0.528±0.000
	PK-LESINN	0.361±0.005	0.393±0.011	0.407±0.011	0.419±0.006	0.435±0.004	0.446±0.008	0.462±0.013	0.469±0.016
MMP	Info-LESINN	0.626±0.051	0.612±0.051	0.600±0.048	0.587±0.043	0.579±0.039	0.574±0.038	0.571±0.038	0.571±0.037
	WL-LESINN	0.422±0.000	0.363±0.000	0.344±0.000	0.333±0.000	0.330±0.000	0.320±0.000	0.313±0.000	0.307±0.000
	PK-LESINN	0.400±0.010	0.362±0.002	0.354±0.004	0.348±0.004	0.341±0.006	0.332±0.005	0.326±0.007	0.322±0.008
p53	Info-LESINN	0.573±0.046	0.567±0.045	0.551±0.041	0.537±0.037	0.532±0.033	0.525±0.030	0.520±0.028	0.520±0.025
	WL-LESINN	0.435±0.000	0.429±0.000	0.413±0.000	0.413±0.000	0.409±0.000	0.403±0.000	0.396±0.000	0.390±0.000
	PK-LESINN	0.341±0.007	0.342±0.005	0.340±0.004	0.339±0.003	0.339±0.004	0.336±0.003	0.332±0.001	0.329±0.001
PPAR-gamma	Info-LESINN	0.629±0.026	0.625±0.038	0.616±0.042	0.605±0.044	0.594±0.049	0.574±0.049	0.553±0.043	0.541±0.036
	WL-LESINN	0.379±0.000	0.409±0.000	0.428±0.000	0.444±0.000	0.460±0.000	0.461±0.000	0.458±0.000	0.461±0.000
	PK-LESINN	0.408±0.005	0.405±0.006	0.404±0.007	0.400±0.012	0.400±0.014	0.397±0.016	0.388±0.015	0.388±0.015
COLLAB	Info-LESINN	0.286±0.048	0.272±0.038	0.264±0.032	0.260±0.026	0.255±0.023	0.255±0.024	0.275±0.029	0.319±0.033
	WL-LESINN	0.603±0.029	0.587±0.026	0.535±0.029	0.450±0.030	0.373±0.025	0.365±0.019	0.445±0.018	0.536±0.014
	PK-LESINN	0.621±0.037	0.606±0.031	0.558±0.046	0.474±0.057	0.394±0.054	0.382±0.051	0.472±0.051	0.550±0.043
hERG	Info-LESINN	0.574±0.044	0.601±0.046	0.610±0.050	0.628±0.053	0.641±0.052	0.659±0.051	0.685±0.049	0.701±0.048
	WL-LESINN	0.742±0.035	0.753±0.026	0.764±0.027	0.772±0.031	0.782±0.035	0.795±0.043	0.802±0.048	0.802±0.047
	PK-LESINN	0.762±0.036	0.769±0.037	0.775±0.039	0.779±0.042	0.791±0.043	0.798±0.049	0.800±0.054	0.798±0.052

BIBLIOGRAPHY

[1] A. AHMED, N. SHERVASHIDZE, S. NARAYANAMURTHY, V. JOSIFOVSKI, AND A. J. SMOLA, *Distributed large-scale natural graph factorization*, in Proceedings of the 22nd international conference on World Wide Web, 2013, pp. 37–48.

[2] L. AKOGLU, H. TONG, AND D. KOUTRA, *Graph based anomaly detection and description: a survey*, Data Mining and Knowledge Discovery, 29 (2015), pp. 626–688.

[3] E. ALSENTZER, S. FINLAYSON, M. LI, AND M. ZITNIK, *Subgraph neural networks*, Advances in Neural Information Processing Systems, 33 (2020), pp. 8017–8029.

[4] V. ARVIND, F. FUHLBRÜCK, J. KÖBLER, AND O. VERBITSKY, *On weisfeiler-leman invariance: Subgraph counts and related graph properties*, Journal of Computer and System Sciences, 113 (2020), pp. 42–59.

[5] J. ATWOOD AND D. TOWSLEY, *Diffusion-convolutional neural networks*, in Advances in neural information processing systems, 2016, pp. 1993–2001.

[6] D. BACCIU, F. ERRICA, AND A. MICHELI, *Contextual graph markov model: A deep and generative approach to graph processing*, arXiv preprint arXiv:1805.10636, (2018).

[7] J. BAEK, M. KANG, AND S. J. HWANG, *Accurate learning of graph representations with graph multiset pooling*, arXiv preprint arXiv:2102.11533, (2021).

[8] X.-E. BAI, J. AN, Z.-B. YU, H.-Q. BAO, AND K.-F. WANG, *A kernel propagation-based graph convolutional network imbalanced node classification model on graph data*, in 2022 IEEE International Conference on Networking, Sensing and Control (ICNSC), IEEE, 2022, pp. 1–6.

[9] M. BALCILAR, P. HÉROUX, B. GAUZERE, P. VASSEUR, S. ADAM, AND P. HONEINE, *Breaking the limits of message passing graph neural networks*, in International Conference on Machine Learning, PMLR, 2021, pp. 599–608.

[10] S. BANDYOPADHYAY, L. N, S. V. VIVEK, AND M. N. MURTY, *Outlier resistant unsupervised deep architectures for attributed network embedding*, in Proceedings of the 13th international conference on web search and data mining, 2020, pp. 25–33.

[11] P. W. BATTAGLIA, J. B. HAMRICK, V. BAPST, A. SANCHEZ-GONZALEZ, V. ZAMBALDI, M. MALINOWSKI, A. TACCHETTI, D. RAPOSO, A. SANTORO, R. FAULKNER, ET AL., *Relational inductive biases, deep learning, and graph networks*, arXiv preprint arXiv:1806.01261, (2018).

[12] F. M. BIANCHI, D. GRATTAROLA, AND C. ALIPPI, *Spectral clustering with graph neural networks for graph pooling*, in International conference on machine learning, PMLR, 2020, pp. 874–883.

[13] K. M. BORGWARDT AND H.-P. KRIEGEL, *Shortest-path kernels on graphs*, in Fifth IEEE international conference on data mining (ICDM'05), IEEE, 2005, pp. 8–pp.

[14] G. BOURITSAS, F. FRASCA, S. ZAFEIRIOU, AND M. M. BRONSTEIN, *Improving graph neural network expressivity via subgraph isomorphism counting*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 45 (2022), pp. 657–668.

[15] K. BOYD, K. H. ENG, AND C. D. PAGE, *Area under the precision-recall curve: point estimates and confidence intervals*, in Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13, Springer, 2013, pp. 451–466.

[16] S. BRODY, U. ALON, AND E. YAHAV, *How attentive are graph attention networks?*, arXiv preprint arXiv:2105.14491, (2021).

[17] J. BRUNA, W. ZAREMBA, A. SZLAM, AND Y. LECUN, *Spectral networks and locally connected networks on graphs*, arXiv preprint arXiv:1312.6203, (2013).

[18] Y. BURDA, H. EDWARDS, A. STORKEY, AND O. KLIMOV, *Exploration by random network distillation*, arXiv preprint arXiv:1810.12894, (2018).

[19] K. CAO, C. WEI, A. GAIDON, N. ARECHIGA, AND T. MA, *Learning imbalanced datasets with label-distribution-aware margin loss*, Advances in neural information processing systems, 32 (2019).

[20] S. CAO, W. LU, AND Q. XU, *Grarep: Learning graph representations with global structural information*, in Proceedings of the 24th ACM international on conference on information and knowledge management, 2015, pp. 891–900.

[21] S. CHANPURIYA AND C. MUSCO, *Infinitewalk: Deep network embeddings as laplacian embeddings with a nonlinearity*, in Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1325–1333.

[22] N. V. CHAWLA, K. W. BOWYER, L. O. HALL, AND W. P. KEGELMEYER, *Smote: synthetic minority over-sampling technique*, Journal of artificial intelligence research, 16 (2002), pp. 321–357.

[23] D. CHEN, L. JACOB, AND J. MAIRAL, *Convolutional kernel networks for graph-structured data*, in International Conference on Machine Learning, PMLR, 2020, pp. 1576–1586.

[24] D. CHEN, Y. LIN, G. ZHAO, X. REN, P. LI, J. ZHOU, AND X. SUN, *Topology-imbalance learning for semi-supervised node classification*, Advances in Neural Information Processing Systems, 34 (2021), pp. 29885–29897.

[25] J. CHEN, T. MA, AND C. XIAO, *Fastgcn: Fast learning with graph convolutional networks via importance sampling*, in International Conference on Learning Representations, 2018.

[26] J. CHEN, J. ZHU, AND L. SONG, *Stochastic training of graph convolutional networks with variance reduction*, in International Conference on Machine Learning, 2018, pp. 942–950.

[27] K. CHEN, J. SONG, S. LIU, N. YU, Z. FENG, G. HAN, AND M. SONG, *Distribution knowledge embedding for graph pooling*, IEEE Transactions on Knowledge and Data Engineering, (2022).

[28] M. CHEN, Z. WEI, Z. HUANG, B. DING, AND Y. LI, *Simple and deep graph convolutional networks*, in International conference on machine learning, PMLR, 2020, pp. 1725–1735.

[29] Z. CHEN, L. CHEN, S. VILLAR, AND J. BRUNA, *Can graph neural networks count substructures?*, Advances in neural information processing systems, 33 (2020), pp. 10383–10395.

[30] W.-L. CHIANG, X. LIU, S. SI, Y. LI, S. BENGIO, AND C.-J. HSIEH, *Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks*, in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 257–266.

[31] E. CHIEN, J. PENG, P. LI, AND O. MILENKOVIC, *Adaptive universal generalized pagerank graph neural network*, arXiv preprint arXiv:2006.07988, (2020).

[32] H.-P. CHOU, S.-C. CHANG, J.-Y. PAN, W. WEI, AND D.-C. JUAN, *Remix: rebalanced mixup*, in Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16, Springer, 2020, pp. 95–110.

[33] G. CORSO, L. CAVALLERI, D. BEAINI, P. LIÒ, AND P. VELIČKOVIĆ, *Principal neighbourhood aggregation for graph nets*, Advances in Neural Information Processing Systems, 33 (2020), pp. 13260–13271.

[34] C. CUI, J. WANG, W. WEI, AND J. LIANG, *Hybrid sampling-based contrastive learning for imbalanced node classification*, International Journal of Machine Learning and Cybernetics, 14 (2023), pp. 989–1001.

[35] Y. CUI, M. JIA, T.-Y. LIN, Y. SONG, AND S. BELONGIE, *Class-balanced loss based on effective number of samples*, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 9268–9277.

[36] M. DEFFERRARD, X. BRESSON, AND P. VANDERGHEYNST, *Convolutional neural networks on graphs with fast localized spectral filtering*, in Advances in neural information processing systems, 2016, pp. 3844–3852.

[37] J. DEMŠAR, *Statistical comparisons of classifiers over multiple data sets*, The Journal of Machine learning research, 7 (2006), pp. 1–30.

[38] K. DING, J. LI, N. AGARWAL, AND H. LIU, *Inductive anomaly detection on attributed networks*, in Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence, 2021, pp. 1288–1294.

[39] K. DING, J. LI, R. BHANUSHALI, AND H. LIU, *Deep anomaly detection on attributed networks*, in Proceedings of the 2019 SIAM International Conference on Data Mining, SIAM, 2019, pp. 594–602.

[40] K. DING, Z. XU, H. TONG, AND H. LIU, *Data augmentation for deep graph learning: A survey*, ACM SIGKDD Explorations Newsletter, 24 (2022), pp. 61–77.

[41] K. DING, Q. ZHOU, H. TONG, AND H. LIU, *Few-shot network anomaly detection via cross-network meta-learning*, in Proceedings of the Web Conference 2021, 2021, pp. 2448–2456.

[42] Q. DONG, S. GONG, AND X. ZHU, *Imbalanced deep learning by minority class incremental rectification*, IEEE transactions on pattern analysis and machine intelligence, 41 (2018), pp. 1367–1381.

[43] X. DONG, X. ZHANG, AND S. WANG, *Rayleigh quotient graph neural networks for graph-level anomaly detection*, in The Twelfth International Conference on Learning Representations, 2023.

[44] Y. DOU, Z. LIU, L. SUN, Y. DENG, H. PENG, AND P. S. YU, *Enhancing graph neural network-based fraud detectors against camouflaged fraudsters*, in Proceedings of the 29th ACM international conference on information & knowledge management, 2020, pp. 315–324.

[45] J. DU, S. WANG, H. MIAO, AND J. ZHANG, *Multi-channel pooling graph neural networks.*, in IJCAI, 2021, pp. 1442–1448.

[46] S. S. DU, K. HOU, R. R. SALAKHUTDINOV, B. POCZOS, R. WANG, AND K. XU, *Graph neural tangent kernel: Fusing graph neural networks with graph kernels*, Advances in neural information processing systems, 32 (2019).

[47] D. DUAN, L. TONG, Y. LI, J. LU, L. SHI, AND C. ZHANG, *Aane: Anomaly aware network embedding for anomalous link detection*, in 2020 IEEE International Conference on Data Mining (ICDM), IEEE, 2020, pp. 1002–1007.

[48] D. DUAN, C. ZHANG, L. TONG, J. LU, C. LV, W. HOU, Y. LI, AND X. ZHAO, *An anomaly aware network embedding framework for unsupervised anomalous link detection*, Data Mining and Knowledge Discovery, 38 (2024), pp. 501–534.

[49] J. DUAN, P. ZHANG, S. WANG, J. HU, H. JIN, J. ZHANG, H. ZHOU, AND X. LIU, *Normality learning-based graph anomaly detection via multi-scale contrastive learning*, in Proceedings of the 31st ACM International Conference on Multimedia, 2023, pp. 7502–7511.

[50] A. DUVAL AND F. MALLIAROS, *Higher-order clustering and pooling for graph neural networks*, in Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 426–435.

[51] D. K. DUVENAUD, D. MACLAURIN, J. IPARRAGUIRRE, R. BOMBARELL, T. HIRZEL, A. ASPURU-GUZIK, AND R. P. ADAMS, *Convolutional networks on graphs for learning molecular fingerprints*, in Advances in neural information processing systems, 2015, pp. 2224–2232.

[52] H. FAN, F. ZHANG, AND Z. LI, *Anomalydae: Dual autoencoder for anomaly detection on attributed networks*, in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020, pp. 5685–5689.

[53] W. FAN, S. WANG, J. HUANG, Z. CHEN, Y. SONG, W. TANG, H. MAO, H. LIU, X. LIU, D. YIN, ET AL., *Graph machine learning in the era of large language models (llms)*, arXiv preprint arXiv:2404.14928, (2024).

[54] A. FENG, C. YOU, S. WANG, AND L. TASSIULAS, *Kergnns: Interpretable graph neural networks with graph kernels*, in Proceedings of the AAAI conference on artificial intelligence, vol. 36, 2022, pp. 6614–6622.

[55] J. FENG, Y. CHEN, F. LI, A. SARKAR, AND M. ZHANG, *How powerful are k-hop message passing graph neural networks*, Advances in Neural Information Processing Systems, 35 (2022), pp. 4776–4790.

[56] A. FERAGEN, N. KASENBURG, J. PETERSEN, M. DE BRUIJNE, AND K. BORGWARDT, *Scalable kernels for graphs with continuous attributes*, Advances in neural information processing systems, 26 (2013).

[57] H. GAO AND S. JI, *Graph u-nets*, in Proceedings of the 36th International Conference on Machine Learning, 2019.

[58] H. GAO, Y. LIU, AND S. JI, *Topology-aware graph pooling networks*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 43 (2021), pp. 4512–4518.

[59] H. GAO, Z. WANG, AND S. JI, *Large-scale learnable graph convolutional networks*, in Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, 2018, pp. 1416–1424.

[60] J. GAO, J. LI, K. ZHANG, AND Y. KONG, *Topology uncertainty modeling for imbalanced node classification on graphs*, in ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2023, pp. 1–5.

[61] X. GAO, W. DAI, C. LI, H. XIONG, AND P. FROSSARD, *ipool‚Äîinformation-based pooling in hierarchical graph neural networks*, IEEE Transactions on Neural Networks and Learning Systems, 33 (2021), pp. 5032–5044.

[62] J. GASTEIGER, A. BOJCHEVSKI, AND S. GÜNNEMANN, *Predict then propagate: Graph neural networks meet personalized pagerank*, arXiv preprint arXiv:1810.05997, (2018).

[63] J. GILMER, S. S. SCHOENHOLZ, P. F. RILEY, O. VINYALS, AND G. E. DAHL, *Neural message passing for quantum chemistry*, in ICML, 2017.

[64] C. GONG, D. TAO, J. YANG, AND K. FU, *Signed laplacian embedding for supervised dimension reduction*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 28, 2014.

[65] M. GORI, G. MONFARDINI, AND F. SCARSELLI, *A new model for learning in graph domains*, in Proceedings. 2005 IEEE international joint conference on neural networks, 2005., vol. 2, IEEE, 2005, pp. 729–734.

[66] M. GROHE, *Descriptive complexity, canonisation, and definable graph structure theory*, vol. 47, Cambridge University Press, 2017.

[67] A. GROVER AND J. LESKOVEC, *node2vec: Scalable feature learning for networks*, in Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.

[68] F. GU, H. CHANG, W. ZHU, S. SOJOUDI, AND L. EL GHAOUI, *Implicit graph neural networks*, Advances in Neural Information Processing Systems, 33 (2020), pp. 11984–11995.

[69] S. GUI, X. LI, L. WANG, AND S. JI, *Good: A graph out-of-distribution benchmark*, Advances in Neural Information Processing Systems, 35 (2022), pp. 2059–2073.

[70] W. L. HAMILTON, R. YING, AND J. LESKOVEC, *Inductive representation learning on large graphs*, arXiv preprint arXiv:1706.02216, (2017).

[71] X. HAN, Z. JIANG, N. LIU, AND X. HU, *G-mixup: Graph data augmentation for graph classification*, in International Conference on Machine Learning, PMLR, 2022, pp. 8230–8248.

[72] D. J. HAND AND R. J. TILL, *A simple generalisation of the area under the roc curve for multiple class classification problems*, Machine learning, 45 (2001), pp. 171–186.

[73] H. HE AND E. A. GARCIA, *Learning from imbalanced data*, IEEE Transactions on knowledge and data engineering, 21 (2009), pp. 1263–1284.

[74] G. HINTON, O. VINYALS, AND J. DEAN, *Distilling the knowledge in a neural network*, arXiv preprint arXiv:1503.02531, (2015).

[75] V. T. HOANG, H.-J. JEON, E.-S. YOU, Y. YOON, S. JUNG, AND O.-J. LEE, *Graph representation learning and its applications: a survey*, Sensors, 23 (2023), p. 4168.

[76] Y. HONG, S. HAN, K. CHOI, S. SEO, B. KIM, AND B. CHANG, *Disentangling label distribution for long-tailed visual recognition*, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 6626–6636.

[77] M. HORN, E. DE BROUWER, M. MOOR, Y. MOREAU, B. RIECK, AND K. BORGWARDT, *Topological graph neural networks*, arXiv preprint arXiv:2102.07835, (2021).

[78] F. HU, L. WANG, Q. LIU, S. WU, L. WANG, AND T. TAN, *Graphdive: Graph classification by mixture of diverse experts.*, in IJCAI, 2022, pp. 2080–2086.

[79] F. HU, Y. ZHU, S. WU, W. HUANG, L. WANG, AND T. TAN, *Graphair: Graph representation learning with neighborhood aggregation and interaction*, Pattern Recognition, 112 (2021), p. 107745.

[80] R. HU, C. C. AGGARWAL, S. MA, AND J. HUAI, *An embedding approach to anomaly detection*, in 2016 IEEE 32nd International Conference on Data Engineering (ICDE), IEEE, 2016, pp. 385–396.

[81] S. HU AND M. SHAO, *Dual perspective contrastive learning based subgraph anomaly detection on attributed networks*, in International Conference on Artificial Neural Networks, Springer, 2022, pp. 481–493.

[82] W. HUANG, T. ZHANG, Y. RONG, AND J. HUANG, *Adaptive sampling towards fast graph representation learning*, in Advances in neural information processing systems, 2018, pp. 4558–4567.

[83] Z. HUANG, A. SILVA, AND A. SINGH, *A broader picture of random-walk based graph embedding*, in Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, 2021, pp. 685–695.

[84] K. ISHIGURO, S.-I. MAEDA, AND M. KOYAMA, *Graph warp module: an auxiliary module for boosting the power of graph neural networks in molecular graph analysis*, arXiv preprint arXiv:1902.01020, (2019).

[85] M. I. K. ISLAM, M. KHANOV, AND E. AKBAS, *Mpool: motif-based graph pooling*, in Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2023, pp. 105–117.

[86] S. IVANOV AND E. BURNAEV, *Anonymous walk embeddings*, in International conference on machine learning, PMLR, 2018, pp. 2186–2195.

[87] S. A. JACOB, P. LOUIS, AND A. SALEHI-ABARI, *Stochastic subgraph neighborhood pooling for subgraph classification*, in Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, 2023, pp. 3963–3967.

[88] M. JIN, Y. LIU, Y. ZHENG, L. CHI, Y.-F. LI, AND S. PAN, *Anemone: Graph anomaly detection with multi-scale contrastive learning*, in Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 3122–3126.

[89] I. T. JOLLIFFE AND J. CADIMA, *Principal component analysis: a review and recent developments*, Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences, 374 (2016), p. 20150202.

[90] W. JU, Z. FANG, Y. GU, Z. LIU, Q. LONG, Z. QIAO, Y. QIN, J. SHEN, F. SUN, Z. XIAO, ET AL., *A comprehensive survey on deep graph representation learning*, Neural Networks, (2024), p. 106207.

[91] B. KANG, S. XIE, M. ROHRBACH, Z. YAN, A. GORDO, J. FENG, AND Y. KALANTIDIS, *Decoupling representation and classifier for long-tailed recognition*, arXiv preprint arXiv:1910.09217, (2019).

[92] U. KANG, H. TONG, AND J. SUN, *Fast random walk graph kernel*, in Proceedings of the 2012 SIAM international conference on data mining, SIAM, 2012, pp. 828–838.

[93] H. KASHIMA, K. TSUDA, AND A. INOKUCHI, *Marginalized kernels between labeled graphs*, in Proceedings of the 20th international conference on machine learning (ICML-03), 2003, pp. 321–328.

[94] S. KEARNES, K. MCCLOSKEY, M. BERNDL, V. PANDE, AND P. RILEY, *Molecular graph convolutions: moving beyond fingerprints*, Journal of computer-aided molecular design, 30 (2016), pp. 595–608.

[95] G. R. KINI, O. PARASKEVAS, S. OYMAK, AND C. THRAMPOULIDIS, *Label-imbalanced and group-sensitive classification under overparameterization*, Advances in Neural Information Processing Systems, 34 (2021), pp. 18970–18983.

[96] T. N. KIPF AND M. WELLING, *Semi-supervised classification with graph convolutional networks*, arXiv preprint arXiv:1609.02907, (2016).

[97] J. KLICPERA, A. BOJCHEVSKI, AND S. GÜNNEMANN, *Predict then propagate: Graph neural networks meet personalized pagerank*, in International Conference on Learning Representations, 2018.

[98] K. KONG, G. LI, M. DING, Z. WU, C. ZHU, B. GHANEM, G. TAYLOR, AND T. GOLDSTEIN, *Flag: Adversarial data augmentation for graph neural networks*, (2020).

[99] A. KUMAGAI, T. IWATA, AND Y. FUJIWARA, *Semi-supervised anomaly detection on attributed graphs*, in 2021 International Joint Conference on Neural Networks (IJCNN), IEEE, 2021, pp. 1–8.

[100] K. LANGE AND K. LANGE, *Singular value decomposition*, Numerical analysis for statisticians, (2010), pp. 129–142.

[101] D. LEE, S. KIM, S. LEE, C. PARK, AND H. YU, *Learnable structural semantic readout for graph classification*, in 2021 IEEE International Conference on Data Mining (ICDM), IEEE, 2021, pp. 1180–1185.

[102] J. LEE, I. LEE, AND J. KANG, *Self-attention graph pooling*, in 36th International Conference on Machine Learning, ICML 2019, International Machine Learning Society (IMLS), 2019, pp. 6661–6670.

[103] A. LEMAN AND B. WEISFEILER, *A reduction of a graph to a canonical form and an algebra arising during this reduction*, Nauchno-Technicheskaya Informatsiya, 2 (1968), pp. 12–16.

[104] R. LEVIE, F. MONTI, X. BRESSON, AND M. M. BRONSTEIN, *Cayleynets: Graph convolutional neural networks with complex rational spectral filters*, IEEE Transactions on Signal Processing, 67 (2018), pp. 97–109.

[105] J. LI, D. CAI, AND X. HE, *Learning graph-level representation for drug discovery*, arXiv preprint arXiv:1709.03741, (2017).

[106] J. LI, H. DANI, X. HU, AND H. LIU, *Radar: Residual analysis for anomaly detection in attributed networks.*, in IJCAI, vol. 17, 2017, pp. 2152–2158.

[107] J. LI, G. PANG, L. CHEN, AND M.-R. NAMAZI-RAD, *Hrgcn: Heterogeneous graph-level anomaly detection with hierarchical relation-augmented graph neural networks*, in 2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA), IEEE, 2023, pp. 1–10.

[108] J. LI, Q. XING, Q. WANG, AND Y. CHANG, *Cvtgad: Simplified transformer with cross-view attention for unsupervised graph-level anomaly detection*, in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2023, pp. 185–200.

[109] M. LI, S. CHEN, Y. ZHANG, AND I. TSANG, *Graph cross networks with vertex infomax pooling*, Advances in Neural Information Processing Systems, 33 (2020), pp. 14093–14105.

[110] P. LI, Y. WANG, H. WANG, AND J. LESKOVEC, *Distance encoding: Design provably more powerful neural networks for graph representation learning*, Advances in Neural Information Processing Systems, 33 (2020), pp. 4465–4478.

[111] Q. LI, Y. HE, C. XU, F. WU, J. GAO, AND Z. LI, *Dual-augment graph neural network for fraud detection*, in Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 4188–4192.

[112] R. LI, S. WANG, F. ZHU, AND J. HUANG, *Adaptive graph convolutional neural networks*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, 2018.

[113] W.-Z. LI, C.-D. WANG, H. XIONG, AND J.-H. LAI, *Graphsha: Synthesizing harder samples for class-imbalanced node classification*, in Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023, pp. 1328–1340.

[114] X. LI, Z. FAN, F. HUANG, X. HU, Y. DENG, L. WANG, AND X. ZHAO, *Graph neural network with curriculum learning for imbalanced node classification*, Neurocomputing, 574 (2024), p. 127229.

[115] Y. LI, X. HUANG, J. LI, M. DU, AND N. ZOU, *Specae: Spectral autoencoder for anomaly detection in attributed networks*, in Proceedings of the 28th ACM international conference on information and knowledge management, 2019, pp. 2233–2236.

[116] Y. LI, D. TARLOW, M. BROCKSCHMIDT, AND R. ZEMEL, *Gated graph sequence neural networks*, arXiv preprint arXiv:1511.05493, (2015).

[117] Y. LI, Y. WANG, T. ZHANG, J. ZHANG, AND Y. CHANG, *Learning network embedding with community structural information*, in Proceedings of the 28th international joint conference on artificial intelligence, 2019.

[118] Y. LI, R. YU, C. SHAHABI, AND Y. LIU, *Diffusion convolutional recurrent neural network: Data-driven traffic forecasting*, in International Conference on Learning Representations, 2018.

[119] Y. LI, R. ZEMEL, M. BROCKSCHMIDT, AND D. TARLOW, *Gated graph sequence neural networks*, in Proceedings of ICLR'16, 2016.

[120] R. LIAO, Z. ZHAO, R. URTASUN, AND R. S. ZEMEL, *Lanczosnet: Multi-scale deep graph convolutional networks*, arXiv preprint arXiv:1901.01484, (2019).

[121] F. LIN, H. GONG, M. LI, Z. WANG, Y. ZHANG, AND X. LUO, *Multi-representations space separation based graph-level anomaly-aware detection*, in Proceedings of the 35th International Conference on Scientific and Statistical Database Management, 2023, pp. 1–11.

[122] F. LIN, X. LUO, J. WU, J. YANG, S. XUE, Z. WANG, AND H. GONG, *Discriminative graph-level anomaly detection via dual-students-teacher model*, in International Conference on Advanced Data Mining and Applications, Springer, 2023, pp. 261–276.

[123] T.-Y. LIN, P. GOYAL, R. GIRSHICK, K. HE, AND P. DOLLÁR, *Focal loss for dense object detection*, in Proceedings of the IEEE international conference on computer vision, 2017, pp. 2980–2988.

[124] F. T. LIU, K. M. TING, AND Z.-H. ZHOU, *Isolation forest*, in 2008 eighth ieee international conference on data mining, IEEE, 2008, pp. 413–422.

[125] M. LIU, S. JIN, L. JIN, S. WANG, Y. FANG, AND Y. SHI, *Imbalanced nodes classification for graph neural networks based on valuable sample mining*, in Proceedings of the 2022 6th International Conference on Electronic Information Technology and Computer Engineering, 2022, pp. 1957–1962.

[126] N. LIU, X. HUANG, AND X. HU, *Accelerated local anomaly detection via resolving attributed networks.*, in IJCAI, 2017, pp. 2337–2343.

[127] S. LIU, R. YING, H. DONG, L. LI, T. XU, Y. RONG, P. ZHAO, J. HUANG, AND D. WU, *Local augmentation for graph neural networks*, in International conference on machine learning, PMLR, 2022, pp. 14054–14072.

[128] X. LIU, J. CHENG, Y. SONG, AND X. JIANG, *Boosting graph structure learning with dummy nodes*, in International Conference on Machine Learning, PMLR, 2022, pp. 13704–13716.

[129] Y. LIU, K. DING, Q. LU, F. LI, L. Y. ZHANG, AND S. PAN, *Towards self-interpretable graph-level anomaly detection*, Advances in Neural Information Processing Systems, 36 (2024).

[130] Y. LIU, Z. LI, S. PAN, C. GONG, C. ZHOU, AND G. KARYPIS, *Anomaly detection on attributed networks via contrastive self-supervised learning*, IEEE transactions on neural networks and learning systems, 33 (2021), pp. 2378–2392.

[131] Y. LIU, Z. ZHANG, Y. LIU, AND Y. ZHU, *Gatsmote: Improving imbalanced node classification on graphs via attention and homophily*, Mathematics, 10 (2022), p. 1799.

[132] Z. LIU, Y. DOU, P. S. YU, Y. DENG, AND H. PENG, *Alleviating the inconsistency problem of applying graph neural network to fraud detection*, in Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, 2020, pp. 1569–1572.

[133] Z. LIU, Q. MAO, C. LIU, Y. FANG, AND J. SUN, *On size-oriented long-tailed graph classification of graph neural networks*, in Proceedings of the ACM Web Conference 2022, 2022, pp. 1506–1516.

[134] Z. LIU, Z. ZENG, R. QIU, H. YOO, D. ZHOU, Z. XU, Y. ZHU, K. WELDEMARIAM, J. HE, AND H. TONG, *Class-imbalanced graph learning without class rebalancing*, (2023).

[135] Z. LIU, W. ZUO, D. ZHANG, AND X. FENG, *Rgse: Robust graph structure embedding for anomalous link detection*, IEEE Transactions on Big Data, (2023).

[136] Q. LONG, Y. JIN, Y. WU, AND G. SONG, *Theoretically improving graph neural networks via anonymous walk graph kernels*, in Proceedings of the Web Conference 2021, 2021, pp. 1204–1214.

[137] Q. LONG, L. XU, Z. FANG, AND G. SONG, *Hgk-gnn: heterogeneous graph kernel based graph neural networks*, in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 1129–1138.

[138] M. LUAN, B. WANG, Y. ZHAO, AND F. HU, *Anomalous subgraph detection in given expected degree networks with deep learning*, IEEE Access, 9 (2021), pp. 60052–60062.

[139] X. LUO, J. WU, A. BEHESHTI, J. YANG, X. ZHANG, Y. WANG, AND S. XUE, *Comga: Community-aware attributed graph anomaly detection*, in Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, 2022, pp. 657–665.

[140] X. LUO, J. WU, J. YANG, S. XUE, H. PENG, C. ZHOU, H. CHEN, Z. LI, AND Q. Z. SHENG, *Deep graph level anomaly detection with contrastive learning*, Scientific Reports, 12 (2022), p. 19867.

[141] T. LYU, Y. ZHANG, AND Y. ZHANG, *Enhancing the network embedding quality with structural similarity*, in Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 147–156.

[142] R. MA, G. PANG, AND L. CHEN, *Harnessing collective structure knowledge in data augmentation for graph neural networks*, Neural Networks, 180 (2024), p. 106651.

[143] R. MA, G. PANG, AND L. CHEN, *Imbalanced graph classification with multi-scale oversampling graph neural networks*, in 2024 International Joint Conference on Neural Networks (IJCNN), 2024, pp. 1–8.

[144] R. MA, G. PANG, L. CHEN, AND A. VAN DEN HENGEL, *Deep graph-level anomaly detection by glocal knowledge distillation*, in Proceedings of the fifteenth ACM international conference on web search and data mining, 2022, pp. 704–714.

[145] X. MA, J. WU, J. YANG, AND Q. Z. SHENG, *Towards graph-level anomaly detection via deep evolutionary mapping*, in Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023, pp. 1631–1642.

[146] Y. MA, S. WANG, C. C. AGGARWAL, AND J. TANG, *Graph convolutional networks with eigenpooling*, in Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 723–731.

[147] Z. MAO, W. JU, Y. QIN, X. LUO, AND M. ZHANG, *Rahnet: Retrieval augmented hybrid network for long-tailed graph classification*, in Proceedings of the 31st ACM International Conference on Multimedia, 2023, pp. 3817–3826.

[148] H. MARON, H. BEN-HAMU, H. SERVIANSKY, AND Y. LIPMAN, *Provably powerful graph networks*, Advances in neural information processing systems, 32 (2019).

[149] A. K. MENON, S. JAYASUMANA, A. S. RAWAT, H. JAIN, A. VEIT, AND S. KUMAR, *Long-tail learning via logit adjustment*, arXiv preprint arXiv:2007.07314, (2020).

[150] A. MICHELI, *Neural network for graphs: A contextual constructive approach*, IEEE Transactions on Neural Networks, 20 (2009), pp. 498–511.

[151] B. A. MILLER, M. S. BEARD, P. J. WOLFE, AND N. T. BLISS, *A spectral framework for anomalous subgraph detection*, IEEE Transactions on Signal Processing, 63 (2015), pp. 4191–4206.

[152] F. MONTI, D. BOSCAINI, J. MASCI, E. RODOLA, J. SVOBODA, AND M. 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, 2017, pp. 5115–5124.

[153] C. MORRIS, K. KERSTING, AND P. MUTZEL, *Glocalized weisfeiler-lehman graph kernels: Global-local feature maps of graphs*, in 2017 IEEE International Conference on Data Mining (ICDM), IEEE, 2017, pp. 327–336.

[154] C. MORRIS, N. M. KRIEGE, F. BAUSE, K. KERSTING, P. MUTZEL, AND M. NEUMANN, *Tudataset: A collection of benchmark datasets for learning with graphs*, in ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020), 2020.

[155] C. MORRIS, M. RITZERT, M. FEY, W. L. HAMILTON, J. E. LENSSEN, G. RATTAN, AND M. GROHE, *Weisfeiler and leman go neural: Higher-order graph neural networks*, in Proceedings of the AAAI conference on artificial intelligence, vol. 33, 2019, pp. 4602–4609.

[156] M. NEUMANN, R. GARNETT, C. BAUCKHAGE, AND K. KERSTING, *Propagation kernels: efficient graph kernels from propagated information*, Machine learning, 102 (2016), pp. 209–245.

[157] D. H. NGUYEN, C. H. NGUYEN, AND H. MAMITSUKA, *Learning subtree pattern importance for weisfeiler-lehman based graph kernels*, Machine Learning, 110 (2021), pp. 1585–1607.

[158] M. NIEPERT, M. AHMED, AND K. KUTZKOV, *Learning convolutional neural networks for graphs*, in International conference on machine learning, PMLR, 2016, pp. 2014–2023.

[159] G. NIKOLENTZOS, G. DASOULAS, AND M. VAZIRGIANNIS, *K-hop graph neural networks*, Neural Networks, 130 (2020), pp. 195–205.

[160] C. NIU, G. PANG, AND L. CHEN, *Graph-level anomaly detection via hierarchical memory networks*, in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2023, pp. 201–218.

[161] I. OSBAND, J. ASLANIDES, AND A. CASSIRER, *Randomized prior functions for deep reinforcement learning*, arXiv preprint arXiv:1806.03335, (2018).

[162] M. OU, P. CUI, J. PEI, Z. ZHANG, AND W. ZHU, *Asymmetric transitivity preserving graph embedding*, in Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 1105–1114.

[163] L. OUYANG, Y. ZHANG, AND Y. WANG, *Unified graph embedding-based anomalous edge detection*, in 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–8.

[164] G. PANG, C. SHEN, L. CAO, AND A. V. D. HENGEL, *Deep learning for anomaly detection: a review*, ACM Computing Surveys (CSUR), 54 (2021), pp. 1–38.

[165] G. PANG, K. M. TING, AND D. ALBRECHT, *Lesinn: Detecting anomalies by identifying least similar nearest neighbours*, in 2015 IEEE international conference on data mining workshop (ICDMW), IEEE, 2015, pp. 623–630.

[166] Y. PANG, Y. ZHAO, AND D. LI, *Graph pooling via coarsened graph infomax*, in Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 2177–2181.

[167] P. A. PAPP, K. MARTINKUS, L. FABER, AND R. WATTENHOFER, *Dropgnn: Random dropouts increase the expressiveness of graph neural networks*, Advances in Neural Information Processing Systems, 34 (2021), pp. 21997–22009.

[168] J. PARK, H. SHIM, AND E. YANG, *Graph transplant: Node saliency-guided graph mixup with local structure preservation*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, 2022, pp. 7966–7974.

[169] J. PARK, J. SONG, AND E. YANG, *Graphens: Neighbor-aware ego network synthesis for class-imbalanced node classification*, in International conference on learning representations, 2021.

[170] Y. PEI, T. HUANG, W. VAN IPENBURG, AND M. PECHENIZKIY, *Resgcn: Attention-based deep residual modeling for anomaly detection on attributed networks*, Machine Learning, 111 (2022), pp. 519–541.

[171] Y. PEI, F. LYU, W. VAN IPENBURG, AND M. PECHENIZKIY, *Subgraph anomaly detection in financial transaction networks*, in Proceedings of the First ACM International Conference on AI in Finance, 2020, pp. 1–8.

[172] H. PENG, J. LI, Q. GONG, Y. NING, S. WANG, AND L. HE, *Motif-matching based subgraph-level attentional convolutional network for graph classification*, in Proceedings of the AAAI conference on artificial intelligence, vol. 34, 2020, pp. 5387–5394.

[173] Z. PENG, M. LUO, J. LI, H. LIU, Q. ZHENG, ET AL., *Anomalous: A joint modeling approach for anomaly detection on attributed networks.*, in IJCAI, vol. 18, 2018, pp. 3513–3519.

[174] Z. PENG, M. LUO, J. LI, L. XUE, AND Q. ZHENG, *A deep multi-view framework for anomaly detection on attributed networks*, IEEE Transactions on Knowledge and Data Engineering, 34 (2020), pp. 2539–2552.

[175] B. PEROZZI, R. AL-RFOU, AND S. SKIENA, *Deepwalk: Online learning of social representations*, in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701–710.

[176] T. PHAM, T. TRAN, D. PHUNG, AND S. VENKATESH, *Column networks for collective classification*, in AAAI-17: Proceedings of the 31st Artificial Intelligence AAAI Conference, AAAI Press, 2017, pp. 2485–2491.

[177] H. QIAO AND G. PANG, *Truncated affinity maximization: One-class homophily modeling for graph anomaly detection*, Advances in Neural Information Processing Systems, 36 (2024).

[178] D. QIN, X. TANG, AND J. LU, *Subgraph representation learning with self-attention and free adversarial training*, Applied Intelligence, (2024), pp. 1–18.

[179] L. QU, H. ZHU, R. ZHENG, Y. SHI, AND H. YIN, *Imgagn: Imbalanced network embedding via generative adversarial graph networks*, in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 1390–1398.

[180] E. RANJAN, S. SANYAL, AND P. TALUKDAR, *Asap: Adaptive structure aware pooling for learning hierarchical graph representations*, in Proceedings of the AAAI conference on artificial intelligence, vol. 34, 2020, pp. 5470–5477.

[181] J. REN, C. YU, X. MA, H. ZHAO, S. YI, ET AL., *Balanced meta-softmax for long-tailed visual recognition*, Advances in neural information processing systems, 33 (2020), pp. 4175–4186.

[182] L. F. RIBEIRO, P. H. SAVERESE, AND D. R. FIGUEIREDO, *struc2vec: Learning node representations from structural identity*, in Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, 2017, pp. 385–394.

[183] Y. RONG, W. HUANG, T. XU, AND J. HUANG, *Dropedge: Towards deep graph convolutional networks on node classification*, arXiv preprint arXiv:1907.10903, (2019).

[184] A. ROY, J. SHU, J. LI, C. YANG, O. ELSHOCHT, J. SMEETS, AND P. LI, *Gad-nr: Graph anomaly detection via neighborhood reconstruction*, in Proceedings of the 17th ACM International Conference on Web Search and Data Mining, 2024, pp. 576–585.

[185] Y. SASAKI ET AL., *The truth of the f-measure*, Teach tutor mater, 1 (2007), pp. 1–5.

[186] R. SATO, M. YAMADA, AND H. KASHIMA, *Random features strengthen graph neural networks*, in Proceedings of the 2021 SIAM international conference on data mining (SDM), SIAM, 2021, pp. 333–341.

[187] F. SCARSELLI, M. GORI, A. C. TSOI, M. HAGENBUCHNER, AND G. MONFARDINI, *The graph neural network model*, IEEE transactions on neural networks, 20 (2008), pp. 61–80.

[188] N. SHERVASHIDZE, P. SCHWEITZER, E. J. VAN LEEUWEN, K. MEHLHORN, AND K. M. BORGWARDT, *Weisfeiler-lehman graph kernels.*, Journal of Machine Learning Research, 12 (2011).

[189] N. SHERVASHIDZE, S. VISHWANATHAN, T. PETRI, K. MEHLHORN, AND K. BORGWARDT, *Efficient graphlet kernels for large graph comparison*, in Artificial intelligence and statistics, PMLR, 2009, pp. 488–495.

[190] S. SHI, K. QIAO, S. YANG, L. WANG, J. CHEN, AND B. YAN, *Boosting-gnn: boosting algorithm for graph networks on imbalanced node classification*, Frontiers in neurorobotics, 15 (2021), p. 775688.

[191] D. I. SHUMAN, S. K. NARANG, P. FROSSARD, A. ORTEGA, AND P. VANDERGHEYNST, *The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains*, IEEE signal processing magazine, 30 (2013), pp. 83–98.

[192] G. SIGLIDIS, G. NIKOLENTZOS, S. LIMNIOS, C. GIATSIDIS, K. SKIANIS, AND M. VAZIRGIANNIS, *Grakel: A graph kernel library in python*, Journal of Machine Learning Research, 21 (2020), pp. 1–5.

[193] N. SINGH, B. A. MILLER, N. T. BLISS, AND P. J. WOLFE, *Anomalous subgraph detection via sparse principal component analysis*, in 2011 IEEE statistical signal processing workshop (SSP), IEEE, 2011, pp. 485–488.

[194] J. SONG, J. PARK, AND E. YANG, *Tam: topology-aware margin loss for class-imbalanced node classification*, in International Conference on Machine Learning, PMLR, 2022, pp. 20369–20383.

[195] F.-Y. SUN, J. HOFFMANN, V. VERMA, AND J. TANG, *Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization*, arXiv preprint arXiv:1908.01000, (2019).

[196] S. A. TAILOR, F. L. OPOLKA, P. LIO, AND N. D. LANE, *Do we need anisotropic graph neural networks?*, arXiv preprint arXiv:2104.01481, (2021).

[197] J. TAN, C. WANG, B. LI, Q. LI, W. OUYANG, C. YIN, AND J. YAN, *Equalization loss for long-tailed object recognition*, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 11662–11671.

[198] Y. TIAN, G. PANG, Y. CHEN, R. SINGH, J. W. VERJANS, AND G. CARNEIRO, *Weakly-supervised video anomaly detection with robust temporal feature magnitude learning*, in Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 4975–4986.

[199] M. TOGNINALLI, E. GHISU, F. LLINARES-LÓPEZ, B. RIECK, AND K. BORGWARDT, *Wasserstein weisfeiler-lehman graph kernels*, Advances in neural information processing systems, 32 (2019).

[200] D. V. TRAN, N. NAVARIN, AND A. SPERDUTI, *On filter size in graph convolutional networks*, in 2018 ieee symposium series on computational intelligence (ssci), IEEE, 2018, pp. 1534–1541.

[201] A. TSITSULIN, J. PALOWITCH, B. PEROZZI, AND E. MÜLLER, *Graph clustering with graph neural networks*, Journal of Machine Learning Research, 24 (2023), pp. 1–21.

[202] R.-C. TZENG AND S.-H. WU, *Distributed, egocentric representations of graphs for detecting critical structures*, in International Conference on Machine Learning, PMLR, 2019, pp. 6354–6362.

[203] P. VELIČKOVIĆ, G. CUCURULL, A. CASANOVA, A. ROMERO, P. LIO, AND Y. BENGIO, *Graph attention networks*, arXiv preprint arXiv:1710.10903, (2017).

[204] V. VERMA, A. LAMB, C. BECKHAM, A. NAJAFI, I. MITLIAGKAS, D. LOPEZ-PAZ, AND Y. BENGIO, *Manifold mixup: Better representations by interpolating hidden states*, in International conference on machine learning, PMLR, 2019, pp. 6438–6447.

[205] G. WANG, R. YING, J. HUANG, AND J. LESKOVEC, *Multi-hop attention graph neural network*, arXiv preprint arXiv:2009.14332, (2020).

[206] H. WANG, G. PANG, C. SHEN, AND C. MA, *Unsupervised representation learning by predicting random distances*, in Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, 2021, pp. 2950–2956.

[207] J. WANG, V. W. ZHENG, Z. LIU, AND K. C.-C. CHANG, *Topological recurrent neural network for diffusion prediction*, in 2017 IEEE international conference on data mining (ICDM), IEEE, 2017, pp. 475–484.

[208] K. WANG, J. AN, AND Q. KANG, *Effective-aggregation graph convolutional network for imbalanced classification*, in 2022 IEEE International Conference on Networking, Sensing and Control (ICNSC), IEEE, 2022, pp. 1–5.

[209] K. WANG, J. AN, M. ZHOU, Z. SHI, X. SHI, AND Q. KANG, *Minority-weighted graph neural network for imbalanced node classification in social networks of internet of people*, IEEE Internet of Things Journal, 10 (2023), pp. 330–340.

[210] Q. WANG, Z. LIU, Z. ZHANG, AND B. HE, *Buffgraph: Enhancing class-imbalanced node classification via buffer nodes*, arXiv preprint arXiv:2402.13114, (2024).

[211] R. WANG, W. XIONG, Q. HOU, AND O. WU, *Tackling the imbalance for gnns*, in 2022 International Joint Conference on Neural Networks (IJCNN), IEEE, 2022, pp. 1–8.

[212] X. WANG, H. JI, C. SHI, B. WANG, Y. YE, P. CUI, AND P. S. YU, *Heterogeneous graph attention network*, in The World Wide Web Conference, 2019, pp. 2022–2032.

[213] X. WANG, B. JIN, Y. DU, P. CUI, Y. TAN, AND Y. YANG, *One-class graph neural networks for anomaly detection in attributed networks*, Neural computing and applications, 33 (2021), pp. 12073–12085.

[214] X. WANG AND M. ZHANG, *Glass: Gnn with labeling tricks for subgraph representation learning*, in International Conference on Learning Representations, 2021.

[215] Y. WANG, C. AGGARWAL, AND T. DERR, *Distance-wise prototypical graph neural network for imbalanced node classification*, 2022.

[216] Y. WANG, J. JIN, W. ZHANG, Y. YU, Z. ZHANG, AND D. WIPF, *Bag of tricks for node classification with graph neural networks*, arXiv preprint arXiv:2103.13355, (2021).

[217] Y. WANG, W. WANG, Y. LIANG, Y. CAI, AND B. HOOI, *Mixup for node and graph classification*, in Proceedings of the Web Conference 2021, 2021, pp. 3663–3674.

[218] Y. WANG, J. ZHANG, S. GUO, H. YIN, C. LI, AND H. CHEN, *Decoupling representation learning and classification for gnn-based anomaly detection*, in Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval, 2021, pp. 1239–1248.

[219] Y. WANG, Y. ZHAO, N. SHAH, AND T. DERR, *Imbalanced graph classification via graph-of-graph neural networks*, in Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 2067–2076.

[220] Y. G. WANG, M. LI, Z. MA, G. MONTUFAR, X. ZHUANG, AND Y. FAN, *Haar graph pooling*, in International conference on machine learning, PMLR, 2020, pp. 9952–9962.

[221] Z. WANG, Q. CAO, H. SHEN, B. XU, K. CEN, AND X. CHENG, *Location-aware convolutional neural networks for graph classification*, Neural Networks, 155 (2022), pp. 74–83.

[222] X. WEI, X. GONG, Y. ZHAN, B. DU, Y. LUO, AND W. HU, *Clnode: Curriculum learning for node classification*, in Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, 2023, pp. 670–678.

[223] J. WU, X. CHEN, K. XU, AND S. LI, *Structural entropy guided graph hierarchical pooling*, in International conference on machine learning, PMLR, 2022, pp. 24017–24030.

[224] L. WU, X. HU, F. MORSTATTER, AND H. LIU, *Adaptive spammer detection with sparse group modeling*, in Proceedings of the international AAAI conference on web and social media, vol. 11, 2017, pp. 319–326.

[225] L. WU, J. XIA, Z. GAO, H. LIN, C. TAN, AND S. Z. LI, *Graphmixup: Improving class-imbalanced node classification by reinforcement mixup and self-supervised context prediction*, in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2022, pp. 519–535.

[226] Z. WU, S. PAN, F. CHEN, G. LONG, C. ZHANG, AND S. Y. PHILIP, *A comprehensive survey on graph neural networks*, IEEE transactions on neural networks and learning systems, 32 (2020), pp. 4–24.

[227] R. XIA, C. ZHANG, Y. ZHANG, X. LIU, AND B. YANG, *A novel graph oversampling framework for node classification in class-imbalanced graphs*, Science China Information Sciences, 67 (2024), pp. 1–16.

[228] C. XIAO, X. XU, Y. LEI, K. ZHANG, S. LIU, AND F. ZHOU, *Counterfactual graph learning for anomaly detection on attributed networks*, IEEE Transactions on Knowledge and Data Engineering, (2023).

[229] W. XIAO, H. ZHAO, V. W. ZHENG, AND Y. SONG, *Vertex-reinforced random walk for network embedding*, in Proceedings of the 2020 SIAM International Conference on Data Mining, SIAM, 2020, pp. 595–603.

[230] B. XU, H. SHEN, Q. CAO, Y. QIU, AND X. CHENG, *Graph wavelet neural network*, arXiv preprint arXiv:1904.07785, (2019).

[231] F. XU, N. WANG, H. WU, X. WEN, D. ZHANG, S. LU, B. LI, W. GONG, H. WAN, AND X. ZHAO, *Gladformer: A mixed perspective for graph-level anomaly detection*, arXiv preprint arXiv:2406.00734, (2024).

[232] K. XU, W. HU, J. LESKOVEC, AND S. JEGELKA, *How powerful are graph neural networks?*, arXiv preprint arXiv:1810.00826, (2018).

[233] K. XU, C. LI, Y. TIAN, T. SONOBE, K.-I. KAWARABAYASHI, AND S. JEGELKA, *Representation learning on graphs with jumping knowledge networks*, in International Conference on Machine Learning, 2018, pp. 5453–5462.

[234] W. XU, P. WANG, Z. ZHAO, B. WANG, X. WANG, AND Y. WANG, *When imbalance meets imbalance: Structure-driven learning for imbalanced graph classification*, in Proceedings of the ACM on Web Conference 2024, 2024, pp. 905–913.

[235] H. XUE, X.-K. SUN, AND W.-X. SUN, *Multi-hop hierarchical graph neural networks*, in 2020 IEEE International Conference on Big Data and Smart Computing (BigComp), IEEE, 2020, pp. 82–89.

[236] S. YAN, Y. XIONG, AND D. LIN, *Spatial temporal graph convolutional networks for skeleton-based action recognition*, in Proceedings of the AAAI conference on artificial intelligence, vol. 32, 2018.

[237] C. YANG, M. SUN, Z. LIU, AND C. TU, *Fast network embedding enhancement via high order proximity approximation.*, in IJCAI, vol. 17, 2017, pp. 3894–3900.

[238] W. YE, H. TIAN, AND Q. CHEN, *Multi-scale wasserstein shortest-path graph kernels for graph classification*, IEEE Transactions on Artificial Intelligence, (2023).

[239] S.-Y. YI, Z. MAO, W. JU, Y.-D. ZHOU, L. LIU, X. LUO, AND M. ZHANG, *Towards long-tailed recognition for graph classification via collaborative experts*, IEEE Transactions on Big Data, (2023).

[240] Y. YIN AND Z. WEI, *Scalable graph embeddings via sparse transpose proximities*, in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1429–1437.

[241] R. YING, R. HE, K. CHEN, P. EKSOMBATCHAI, W. L. HAMILTON, AND J. LESKOVEC, *Graph convolutional neural networks for web-scale recommender systems*, in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 974–983.

[242] Z. YING, J. YOU, C. MORRIS, X. REN, W. HAMILTON, AND J. LESKOVEC, *Hierarchical graph representation learning with differentiable pooling*, in Advances in neural information processing systems, 2018, pp. 4800–4810.

[243] J. YOU, J. M. GOMES-SELMAN, R. YING, AND J. LESKOVEC, *Identity-aware graph neural networks*, in Proceedings of the AAAI conference on artificial intelligence, vol. 35, 2021, pp. 10737–10745.

[244] L. YU, C. TANG, C. YANG, AND J. LV, *Graph-level anomaly detection via deep metric learning*, in 2023 6th International Conference on Algorithms, Computing and Artificial Intelligence, 2023, pp. 88–96.

[245] Z. YU, X. WANG, B. ZHANG, Z. LUO, AND L. DUAN, *Tuaf: Triple-unit-based graph-level anomaly detection with adaptive fusion readout*, in International Conference on Database Systems for Advanced Applications, Springer, 2023, pp. 415–430.

[246] H. YUAN AND S. JI, *Structpool: Structured graph pooling via conditional random fields*, in Proceedings of the 8th International Conference on Learning Representations, 2020.

[247] S. YUN, D. HAN, S. J. OH, S. CHUN, J. CHOE, AND Y. YOO, *Cutmix: Regularization strategy to train strong classifiers with localizable features*, in Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 6023–6032.

[248] S. YUN, K. KIM, K. YOON, AND C. PARK, *Lte4g: Long-tail experts for graph neural networks*, in Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 2434–2443.

[249] Y. ZANG, C. HUANG, AND C. C. LOY, *Fasa: Feature augmentation and sampling adaptation for long-tailed instance segmentation*, in Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 3457–3466.

[250] C. ZHANG, A. SWAMI, AND N. V. CHAWLA, *Shne: Representation learning for semantic-associated heterogeneous networks*, in Proceedings of the twelfth ACM international conference on web search and data mining, 2019, pp. 690–698.

[251] G. ZHANG, Z. YANG, J. WU, J. YANG, S. XUE, H. PENG, J. SU, C. ZHOU, Q. Z. SHENG, L. AKOGLU, ET AL., *Dual-discriminative graph neural network for*

imbalanced graph-level anomaly detection, Advances in Neural Information Processing Systems, 35 (2022), pp. 24144–24157.

[252] H. ZHANG, M. CISSE, Y. N. DAUPHIN, AND D. LOPEZ-PAZ, *mixup: Beyond empirical risk minimization*, arXiv preprint arXiv:1710.09412, (2017).

[253] J. ZHANG, X. SHI, J. XIE, H. MA, I. KING, AND D.-Y. YEUNG, *Gaan: Gated attention networks for learning on large and spatiotemporal graphs*, arXiv preprint arXiv:1803.07294, (2018).

[254] L. ZHANG, X. WANG, H. LI, G. ZHU, P. SHEN, P. LI, X. LU, S. A. A. SHAH, AND M. BENNAMOUN, *Structure-feature based graph self-adaptive pooling*, in Proceedings of The Web Conference 2020, 2020, pp. 3098–3104.

[255] L. ZHANG, J. YUAN, Z. LIU, Y. PEI, AND L. WANG, *A robust embedding method for anomaly detection on attributed networks*, in 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019, pp. 1–8.

[256] M. ZHANG, Z. CUI, M. NEUMANN, AND Y. CHEN, *An end-to-end deep learning architecture for graph classification*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, 2018.

[257] M. ZHANG AND P. LI, *Nested graph neural networks*, Advances in Neural Information Processing Systems, 34 (2021), pp. 15734–15747.

[258] S. ZHANG, S. MUKHERJEE, AND T. K. DEY, *Gefl: extended filtration learning for graph classification*, in Learning on Graphs Conference, PMLR, 2022, pp. 16–1.

[259] X. ZHANG, K. XIE, S. WANG, AND Z. HUANG, *Learning based proximity matrix factorization for node embedding*, in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 2243–2253.

[260] Y. ZHANG, B. KANG, B. HOOI, S. YAN, AND J. FENG, *Deep long-tailed learning: A survey*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2023).

[261] Z. ZHANG, J. BU, M. ESTER, J. ZHANG, Z. LI, C. YAO, H. DAI, Z. YU, AND C. WANG, *Hierarchical multi-view graph pooling with structure learning*, IEEE Transactions on Knowledge and Data Engineering, 35 (2021), pp. 545–559.

[262] Z. ZHANG, J. BU, M. ESTER, J. ZHANG, C. YAO, Z. YU, AND C. WANG, *Hierarchical graph pooling with structure learning*, arXiv preprint arXiv:1911.05954, (2019).

[263] Z. ZHANG, P. CUI, AND W. ZHU, *Deep learning on graphs: A survey*, IEEE Transactions on Knowledge and Data Engineering, (2020).

[264] Z. ZHANG AND L. ZHAO, *Unsupervised deep subgraph anomaly detection*, in 2022 IEEE International Conference on Data Mining (ICDM), IEEE, 2022, pp. 753–762.

[265] J. ZHAO, J. LI, B. ZHOU, F. CHEN, P. TOMCHIK, AND W. JU, *Parallel algorithms for anomalous subgraph detection*, Concurrency and Computation: Practice and Experience, 29 (2017), p. e3769.

[266] L. ZHAO AND L. AKOGLU, *On using classification datasets to evaluate graph outlier detection: Peculiar observations and new insights*, Big Data, 11 (2023), pp. 151–180.

[267] T. ZHAO, C. DENG, K. YU, T. JIANG, D. WANG, AND M. JIANG, *Error-bounded graph anomaly loss for gnns*, in Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 1873–1882.

[268] T. ZHAO, D. LUO, X. ZHANG, AND S. WANG, *Topoimb: Toward topology-level imbalance in learning from graphs*, in Learning on Graphs Conference, PMLR, 2022, pp. 37–1.

[269] T. ZHAO, X. TANG, D. ZHANG, H. JIANG, N. RAO, Y. SONG, P. AGRAWAL, K. SUBBIAN, B. YIN, AND M. JIANG, *Autogda: Automated graph data augmentation for node classification*, in Learning on Graphs Conference, PMLR, 2022, pp. 32–1.

[270] T. ZHAO, X. ZHANG, AND S. WANG, *Graphsmote: Imbalanced node classification on graphs with graph neural networks*, in Proceedings of the 14th ACM international conference on web search and data mining, 2021, pp. 833–841.

[271] Y. ZHENG, M. JIN, Y. LIU, L. CHI, K. T. PHAN, AND Y.-P. P. CHEN, *Generative and contrastive self-supervised learning for graph anomaly detection*, IEEE Transactions on Knowledge and Data Engineering, 35 (2021), pp. 12220–12233.

[272] C. ZHOU, Y. LIU, X. LIU, Z. LIU, AND J. GAO, *Scalable graph embedding for asymmetric proximity*, in Proceedings of the AAAI conference on artificial intelligence, vol. 31, 2017.

[273] M. ZHOU AND Z. GONG, *Graphsr: a data augmentation algorithm for imbalanced node classification*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, 2023, pp. 4954–4962.

[274] S. ZHOU, Q. TAN, Z. XU, X. HUANG, AND F.-L. CHUNG, *Subtractive aggregation for attributed network anomaly detection*, in Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 3672–3676.

[275] H. ZHU AND P. KONIUSZ, *Simple spectral graph convolution*, in International conference on learning representations, 2020.

[276] C. ZHUANG AND Q. MA, *Dual graph convolutional networks for graph-based semi-supervised classification*, in Proceedings of the 2018 World Wide Web Conference, 2018, pp. 499–508.

[277] D. ZOU, Z. HU, Y. WANG, S. JIANG, Y. SUN, AND Q. GU, *Layer-dependent importance sampling for training deep and large graph convolutional networks*, Advances in neural information processing systems, 32 (2019).

[278] M. ZOU, Z. GAN, R. CAO, C. GUAN, AND S. LENG, *Similarity-navigated graph neural networks for node classification*, Information Sciences, 633 (2023), pp. 41–69.