Deep Graph Representation Learning and its Application on Graph Clustering

Hua Zheng

Supervisor: Prof. Feng Tian

Department of Creative Technology
Faculty of Science & Technology
Bournemouth University

Submitted in partial fulfillment of the requirements for
the degree of
Doctor of Philosophy

17 April 2023

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognize that its copyright rests with its author and due acknowledgment must always be made of the use of any material contained in, or derived from, this thesis.

Abstract

Graphs like social networks, molecular graphs, and traffic networks are everywhere in the real world. Deep Graph Representation Learning (DGL) is essential for most graph applications, such as Graph Classification, Link Prediction, and Community Detection. DGL has made significant progress in recent years because of the development of Graph Neural Networks (GNNs). However, there are still several crucial challenges that the field faces, including in (semi-)supervised DGL, self-supervised DGL, and DGL-based graph clustering. In this thesis, I proposed three models to address the problems in these three aspects respectively.

GNNs have been widely used in DGL problems. However, GNNs suffer from over-smoothing due to their repeated local aggregation and over-squashing due to the exponential growth in computation paths with increased model depth, which confines their expressive power. To solve this problem, a Hierarchical Structure Graph Transformer called HighFormer is proposed to leverage local and relatively global structure information. I use GNNs to learn the initial graph node representation based on the local structure information. At the same time, a structural attention module is used to learn the relatively global structural similarity. Then, the improved attention matrix was obtained by adding the relatively global structure similarity matrix to the traditional attention matrix. Finally, the graph representation was learned by the improved attention matrix.

Graph contrastive learning (GCL) has recently become the most powerful method in self-supervised graph representation learning (SGL), of which graph augmentation is a critical component to generating different views of input graphs. Most existing GCL methods perform stochastic data augmentation schemes, for example, randomly dropping edges or masking node features. However, uniform transformations without carefully designed augmentation techniques may drastically change the underlying semantics of graphs or graph nodes. I argue that the graph augmentation schemes should preserve the intrinsic semantics of graphs. Besides, existing GCL methods neglect the semantic information that may introduce false-negative samples. Therefore, a novel GCL method with semantic invariance graph augmentation termed SemiGCL is proposed by designing a semantic invariance graph augmentation (SemiAug) and a semantic-based graph contrastive (SGC) scheme.

Deep graph clustering (DGC), which aims to divide the graph nodes into different clusters, is challenging for graph analysis. DGC usually consists of an encoding neural network and a clustering method. Although DGC has made remarkable progress with the development of deep learning, I observed two drawbacks to the existing methods:

1) Existing methods usually overlook learning the global structural information in the node encoding process. Consequently, the discriminative capability of representations will be limited. 2) Most existing methods leverage traditional clustering methods such as K-means and spectral clustering. However, these clustering methods can not simultaneously be trained with the DGL methods, leading to sub-optimal clustering performance. To address these issues, I propose a novel self-supervised DGC method termed Structural Semantic Contrastive Deep Graph Clustering (SECRET). To get a more discriminative representation, I design a structure contrastive scheme (SCS) by contrasting the aggregation of first-order neighbors with a graph diffusion. A consistent loss was also proposed to keep the structure of different views consistent. To jointly optimize the DGL and clustering method, I proposed a novel Self-supervised Deep-learning-based Clustering (SDC) model.

Table of contents

List of figures

List of tables

Acknowledgments

The thesis has finally been completed, indicating that my four-year doctoral life has finished. It was the hardest time but also the best experience in my life. It is hard since no one can finish it by themselves without the help of others. As a rural student, I didn't receive enough good education. Therefore, many things need to be learned before I start my research on artificial intelligence (AI), which makes the work improvement slow. Even though I have faced tremendous difficulties in my studies and life, I made it thanks to the support of the most excellent and generous people. Whenever I face difficulties or even want to give up, they always give me help and remind me that my work has value and that I should try my best to make it. I want to express my gratitude to them from the bottom of my heart.

I will start by acknowledging my excellent parents. They were average Chinese farmers, and they didn't have any other special skills besides farming, but they sent me to the University and even helped me successfully go through this impressive journey with their unconditional love. Throughout my life, they have always provided the backbone for my efforts. I wish I could make them proud and help them live easy lives.

This Ph.D. wouldn't be going so smoothly without the endless support of my supervisor, Feng Tian. At the beginning of my Ph.D., Prof. Tian taught me to work on machine learning, the research area with the most potential, and I am also interested in it. His support not only for the research but also for my daily life. He is an excellent school teacher and a great tutor for my life. I will remain eternally grateful for his kind help.

The most beautiful thing was meeting my wife, Xian Lu, during my Ph.D. Sometimes, research is hard and boring, but love is always sweet. Before I met my wife, I didn't have a specific plan for the future. After we got married, I realized the meaning of my life, and I got the aim to fight. My wife made me grow and become a reliable, courageous, and humble man. She is a gorgeous, kind, intelligent girl who exceeded my expectations. She is such an angel that her appearance completely changed my life.

During these four years, I have made some good friends. Without them, I couldn't complete my Ph.D. Dr. Sultan Mahmud, an optimistic and kind man, always accompanies me. He encouraged and helped me go through the hardest time during my Ph.D. Dr. Kuanishbay, a sincere and hard-working man, set a good example for me. Whenever I doubt whether I can graduate, I will think of him. His hard work and spirit of perseverance inspire me to go forward. Dr. Sadiq, an intelligent and sophisticated man, gave me a lot of advice about my future development. Also, I will thank my fellows from Bournemouth University, Dr. Weilai Xu, Dr. Long Xi, Dr. Yan Zhao, and Dr. Ge Zheng. When I have

dinner with them, I feel like I am back home, and they are just like my family. I will thank my roommates at BU accommodations, Jiange Li, Tobe, and Gary. Even though it has been three years, what happened in our dormitory is still very clear in my memory. The scene in which we cook dinner together every night makes me feel warm, and I also can't forget the happy time we traveled in Northern Europe.

Thanks to my elder sister and younger sister. First, my elder sister, who is a generous and talented woman, started to look after me when I was a little Child. She set a good example for me and always encouraged me to study hard and be a person who is useful to society. I hope that I didn't disappoint her. Then my little sister, nine years younger than me, is very sensible and hardworking. As her elder brother, I should look after her and my parents, but I went far away from home to pursue a Doctor degree, which shaped her into a precocious child who does many things instead of me.

Last but not least, I would like to express my gratitude and appreciation to all the people I met during this four-year journey who made my life more wonderful. Dr. Zhong Ming, who is an exceptional professor, gave me many farsighted suggestions that guided me to have a brilliant career. Dr. Xizhao Wang, who is an outstanding scientist and IEEE fellow, teach me the lesson of machine learning, which is the beginning of my machine learning research. Dr. Zhexue Huang, a distinguished professor, taught me the lesson of big data, which profoundly impacted my subsequent research. Dr. Meikang Qiu, an exceptional professor from the USA, also advised me on the research. Dr. Weipeng Cao helped me join the research life. Dr. Cheng Wen used to discuss many things with me, which was very interesting. Dr. Dugang Liu, who is an expert in deep learning, helps me a lot with deep learning algorithms. Dr. Jamshed, who is a gentle and kind man, is always helpful. Dr. Xiaojun Chen, who is an excellent professor, advised me to work on graph neural networks (GNN). Qiang Liu is my childhood friend. We still keep in touch, and we often share our lives with each other. And Dr. Qinghong Lin, Mingkai He, Guoting Lin, Gaoyang Tang, Shaotian Cai, Zhixuan Liang, Jiangfeng Zhao, and Wenjin Li. I am fortunate to know so many exceptional people in my life who all make me stronger.

Declaration

I state that this thesis is all my own work and does not include the work done in collaboration except as declared in the Preface and specified in the text. There isn't any substantial part of my dissertation that has already been submitted or is being concurrently submitted for any degree, diploma, or other qualification at Bournemouth University or any other University or similar institution except as declared in the Preface and specified in the text.

Hua Zheng
17 April 2023

Nomenclature

$\|\mathbf{v}\|$ The L_2 norm of $\mathbf{v}$

(l, r) The open interval from l to r

$A \odot B$ Hadamard product of A and B

M_{ij} Element (i, j) of matrix $\mathbf{M}$

$[l, r]$ The closed interval from l to r

$|S|$ The length of set S

$\mathbb{E}_{x \sim \mathbb{P}(x)}[f(x)]$ The expected value of $f(x)$ given that x is sampled from $\mathbb{P}(x)$

$\mathbb{P}(x \mid y)$ Probability distribution of a discrete Random Variable x, given y

$\mathbb{P}(x)$ Probability distribution of a Discrete Random Variable x

$\mathbb{R}$ The set of real numbers

$\mathbf{A} \in \mathbb{R}^{N \times N}$ Original adjacency matrix

$\mathbf{A} \in \mathbb{R}^{n \times n}$ The adjacency matrix of a graph

$\mathbf{A}^d \in \mathbb{R}^{N \times N}$ Graph diffusion matrix

$\mathbf{A}^m \in \mathbb{R}^{N \times N}$ Edge-masked adjacency matrix

$\mathbf{D} \in \mathbb{R}^{N \times N}$ Degree matrix

$\mathbf{H} \in \mathbb{R}^{n \times d_r}$ The node representation matrix of a graph

$\mathbf{I}$ Identity matrix

$\mathbf{I}_n$ Identity matrix of shape $n \times n$

$\mathbf{M}$ Matrix

$\mathbf{M}^T$ Transposed matrix $\mathbf{M}$

$\mathbf{P} \in \mathbb{R}^{N \times C}$ Target distribution

$\mathbf{Q} \in \mathbb{R}^{N \times C}$ Soft assignment distribution

$\mathbf{S}^{\mathcal{F}} \in \mathbb{R}^{d \times d}$ Cross-view feature correlation matrix

$\mathbf{S}^{\mathcal{N}} \in \mathbb{R}^{N \times N}$ Cross-view sample correlation matrix

$\mathbf{X} \in \mathbb{R}^{N \times D}$ Attribute matrix

$\mathbf{Z} \in \mathbb{R}^{N \times d}$ Clustering-oriented latent embedding

$\mathbf{Z}^{v_k} \in \mathbb{R}^{N \times d}$ Node embedding in k-th view

$\mathbf{h_i} = [\mathbf{H}]_{v_i}$ The node representation vector of v_i

$\mathbf{h}_{\mathcal{G}}$ The graph representation vector of a graph

$\mathbf{m}_i$ The i-th row vector of matrix $\mathbf{M}$

$\mathbf{v}$ Vector

$\mathbf{x_i} = [\mathbf{X}]_{v_i}$ The node feature of v_i

$\mathcal{C}$ The set of pseudo label

$\mathcal{E}$ The set of edges in a graph

$\mathcal{G}$ A graph

$\mathcal{L}(\cdot)$ Loss function

$\mathcal{MI}(\cdot,\cdot)$ Mutual information function

$\mathcal{N}(v_i)$ The neighbors of node v_i

$\mathcal{R}(\cdot)$ Readout function

$\mathcal{V}$ The set of nodes in a graph

$\mathcal{Y}$ The set of manual label

$\mathrm{Var}(f(x))$ The variance of $f(x)$

$\tilde{\mathcal{G}}$ The augmentation graph of $\mathcal{G}$

$\vec{a} \| \vec{b}$ Concatenation of $\vec{a}$ and $\vec{b}$

$\widehat{\mathbf{A}} \in \mathbb{R}^{N \times N}$ Rebuilt adjacency matrix

$\widehat{\mathbf{X}} \in \mathbb{R}^{N \times D}$ Rebuilt attribute matrix

$\widetilde{\mathbf{A}} \in \mathbb{R}^{N \times N}$ Normalized adjacency matrix

$\widetilde{\mathbf{Z}}^{v_k} \in \mathbb{R}^{d \times K}$ Cluster-level embedding in k-th view

c a pseudo label

d	The dimension of node features
d_r	The dimension of node representation
$e_{i,j}$	An edge betweenv_i and v_j
$f_\theta(\cdot)$	Encoder function parameterized by θ
m	The number of edges in a graph
n	The number of nodes in a graph
$p(x)$	Probability distribution of a continuous Random Variable x
s	Scalar
v_i	A node of a graph
$x \sim \mathbb{P}(x)$	Random variable x sampled from $\mathbb{P}$
y	a manual label

I would like to dedicate this thesis to my parents

Chapter 1

Introduction

Graphs are mathematical abstractions used to represent relationships or connections between entities. They are utilized to model complex systems and phenomena in various fields [86, 171, 90, 63], including computer science, biology, physics, sociology, and many others. For example, in social networks, people are represented as nodes, and their connections are edges, which can be used to analyze the network's structure, identify influential individuals, or predict the spread of information or diseases. The versatility of graphs [9] as a modeling tool has made them a popular choice in many different fields, and their use continues to grow as more complex systems are studied and analyzed.

Different from Euclidean data, such as images [66] and texts [111, 112], which have a grid-like structure and are represented as arrays or vectors, graphs are non-Euclidean structured data [173, 167] that represent a set of objects (nodes) and their relationships (edges). Each node in a graph can contain information, such as attributes or features, that describe the object it represents. Each edge represents a relationship between two nodes, which can be weighted or directed. The structure of a graph can be used to model complex relationships and dependencies between objects. Therefore, analyzing graphs can be challenging because of their complex and irregular structure.

To solve this problem, many Deep Graph Representation Learning (DGL) [72, 50, 155, 166, 203] methods were proposed. The goal of graph representation learning is to learn low-dimensional vector representations, or embeddings, for the nodes or edges of a graph that capture the structural and semantic properties of the graph. Among them, Graph Neural Networks (GNNs) [72, 155, 50] are specifically designed to learn such representations by recursively aggregating and transforming [2, 119, 38] information from neighboring nodes in graphs which have been a promising approach for learning representations of graph-structured data. GNNs learn node embeddings by considering the graph structure and node attributes and using this information to propagate across the graph. This allows them to capture complex dependencies between nodes and to learn rich and informative representations that can be used for various downstream tasks such as node classification[190], link prediction[163], and graph generation[162]. There are many different types of GNNs, including graph convolutional networks (GCNs) [72], graph attention networks (GATs) [157], and GraphSage [50]. Each type of GNN has its

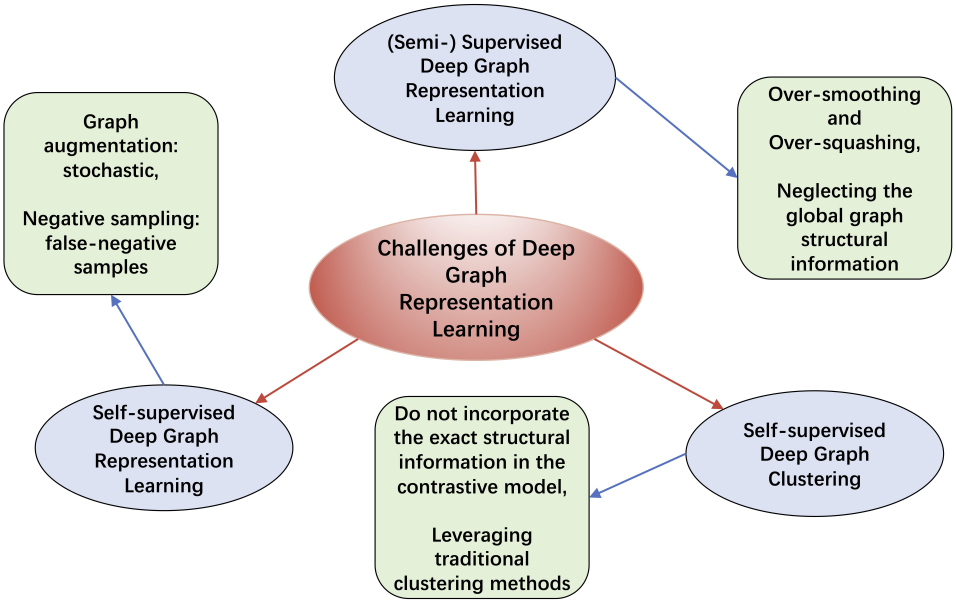

Fig. 1.1 The challenges of Deep Graph Representation Learning(DGL) in three different tasks.

strengths and weaknesses [170], and researchers continue developing new architectures and techniques to improve their performance. DGL has indeed made significant progress in recent years.

DGL is a fundamental component of graph learning, enabling the practical analysis, modeling, and understanding of complex graph-structured data across various domains and applications. Its importance will grow as researchers and practitioners explore new techniques and applications in graph learning. However, there are still several crucial challenges in DGL. In this thesis, I try to improve the performance of DGL so that all the downstream tasks of graph learning can be developed. Through plenty of experiments, I found that both the node feature and graph structure are essential in graph learning, but the most important thing is finding a way to combine these two pieces of information. In my research, I make the most of these two pieces of information, and for structure information, I consider both the local structure and global structure. I have divided my research into three aspects: (semi-)supervised DGL, self-supervised DGL, and the application of DGL on graph clustering. The research problems in the three aspects were concluded in Figure 1.1. In (semi-)supervised DGL, there are three main problems. The first two are the over-smoothing and over-squashing problems, and the third is that existing methods only consider the local structure information but neglect the global graph structural information. In self-supervised DGL, there are two main problems. The first is that existing graph augmentation methods are all stochastic, which may change the semantic information of the graph dramatically, and the second is that existing contrastive methods usually generate false-negative samples, which may degenerate the representation performance. In the application of DGL on graph clustering methods, there are also two main problems. The first is that the existing techniques didn't incorporate the exact structure information in the graph contrastive model, and the second is existing deep graph clustering methods

usually leverage the traditional clustering methods, which are hard to optimize with the DGL network at the same time, so the results may not be optimal.

This thesis has investigated the importance of semantic and structure information in DGL. The primary argument of the research is that both the semantic and structural information, including the local and global structures, are essential for DGL. Semantic information is the meaningful representation of data that captures its underlying meaning or semantics. It goes beyond the raw data and aims to extract higher-level concepts, relationships, and patterns relevant to the task. I designed some simple but efficient algorithms to learn the semantic information first, which is then used as prior knowledge for the DGL method. In graphs, structural information refers to the characteristics, properties, and relationships inherent in the graph's topology or arrangement of nodes and edges. This information provides insights into the organization, connectivity, and patterns present in the graph data. Local and global structure information refers to different levels of granularity when analyzing the connectivity and organization of nodes and edges within a graph. Local structure information focuses on the characteristics and relationships of nodes and edges within the immediate vicinity of a given node or a small subset of nodes in the graph. It involves analyzing a node's local neighborhood, adjacent nodes, and immediate connections to understand its local context and connectivity patterns. Global structure information focuses on the entire graph's overall organization, connectivity, and properties. It involves analyzing the collective behavior, emergent properties, and structural patterns arising from the interactions between nodes and edges across the graph. local structure information focuses on the properties and relationships of nodes within the immediate neighborhood, while global structure information considers the overall organization and properties of the entire graph. Both types of information are essential for understanding the complex interactions and behavior of nodes and edges within graphs and are used in various graph analysis and machine-learning tasks.

However, most existing DGL methods only consider the local structure information but neglect the global structure information, which may degenerate the representation power of DGL. I proposed methods to learn both the global structure information and the local structure information, and then these two structures of information are combined for the DGL training. I chose many commonly used attributed graph datasets to measure the performance of proposed DGL methods. The attributed graph datasets are from domains such as citation graphs, social networks, and biochemical graphs (protein and chemical compounds). The sizes of these graphs are different, so it is efficient to measure the performance of the DGL methods. These datasets are all commonly used in various techniques that are convenient to compare with other methods. For measuring the performance of DGL, both the classification and clustering results of the representations are used. I use deep learning methods in this research but not other machine learning methods. The reason is that Deep learning graph representation methods offer several advantages over traditional machine learning methods, including their capacity to capture complex patterns, end-to-end learning capabilities, scalability to large graphs, flexibility and adaptability to diverse data modalities, and state-of-the-art performance on graph-based tasks. These characteristics

make them highly effective and widely used in graph representation learning and analysis across various fields and applications. I used the Deep Graph Library as my graph network framework in the experiments. Deep Graph Library is a powerful and versatile library for graph-based machine learning and deep learning, offering a comprehensive set of tools and functionalities for building, training, and deploying graph neural network models in various applications and domains. I chose this library because of its ease of use, flexibility, scalability, performance, integration with deep learning frameworks, and active community support.

There are some limitations while conducting my research. 1). Computational Resources and Hardware Limitations: Deep learning models, especially large ones, require substantial computational resources. GPUs are commonly used to accelerate deep learning computations due to their parallel processing capabilities. However, limitations in GPU memory, computational power, and access to high-performance hardware can still pose challenges, particularly for training large models on extensive datasets. 2). Data Availability and Quality: Deep learning models thrive on large, diverse datasets. The availability of high-quality labeled data is crucial for training accurate models. In some domains, obtaining sufficient labeled data can be challenging, and the data quality can significantly impact model performance. 3). Ethical and Bias Concerns: Deep learning models can inherit biases in training data, potentially leading to biased predictions. Ensuring fairness, transparency, and addressing ethical considerations in model development is essential to prevent unintended consequences and reinforce trust. The experiment results proved that my proposed methods are valid and reliable. The validity is because I ran each algorithm ten times and got the average results. The reliability is as I used the augmentation of datasets for each proposed method so that the results are consistent and stable.

Ethical considerations are paramount in developing and deploying DGL models. Here are some key ethical considerations for DGL: 1). Privacy and Data Protection: DGL often involves analyzing sensitive data, such as social networks, healthcare records, or financial transactions. Ensuring the privacy and confidentiality of individuals' data is crucial to prevent unauthorized access, misuse, or unintended disclosure of personal information. The datasets I used in my experiments have already been authorized, and the personal information has been removed. 2). Fairness and Bias: DGL models may inadvertently perpetuate biases in the training data, leading to unfair or discriminatory outcomes. Mitigating biases and ensuring fairness in model predictions is essential, particularly when making decisions that affect individuals' opportunities, resources, or rights. I use augmented data in my methods to reduce the bias. 3). Transparency and Interpretability: DGL models can be complex and opaque, making it challenging to understand how they arrive at specific decisions or predictions. Ensuring transparency and interpretability in model development helps build trust and accountability, enabling users to understand and scrutinize the model's behavior. In my DGL model, I try to interpret how the deep learning network works. 4). Accountability and Responsibility: Developers and researchers working on DGL models are responsible for anticipating and addressing their work's potential ethical implications. They should be accountable for

the consequences of model deployment and strive to minimize harm while maximizing societal benefits. 5). Informed Consent and Data Ownership: When collecting or using graph data for model training, obtaining informed consent from individuals whose data is being used is essential. Additionally, clarifying data ownership and usage rights ensures transparency and empowers individuals to make informed decisions about their data. In my experiment, I obtained informed consent from the owner and declared the data ownership and usage rights. 6). Security and Adversarial Attacks: DGL models may be vulnerable to adversarial attacks, where malicious actors manipulate input data to deceive or compromise the model's performance. Ensuring robustness and resilience against such attacks is critical for model integrity and security. 7). Regulatory Compliance: DGL models may be subject to regulatory requirements and standards, particularly in highly regulated domains such as healthcare, finance, and law enforcement. Compliance with applicable laws, regulations, and ethical guidelines is essential to avoid legal and ethical implications. Addressing ethical considerations in DGL requires a multidisciplinary approach involving collaboration between researchers, developers, policymakers, ethicists, and stakeholders. By prioritizing ethical principles and values throughout the model development lifecycle, we can harness the potential of DGL while minimizing risks and promoting responsible innovation.

After defining the research methodology of this thesis, I will discuss this research in detail in three aspects: (semi-)supervised DGL, self-supervised DGL, and the application of DGL to graph clustering. I will explain both the research problem and solutions.

Firstly, GNNs have gained much attention recently for their strong graph representation capacity. It has become the most popular method in (semi-)supervised DGL. However, GNNs can suffer from over-smoothing [200], and over-squashing [113], which cause many issues in various GNN-based tasks. Over-smoothing is a phenomenon where the node representations in a GNN become too similar after multiple rounds of message passing. This can happen because each node aggregates information from its neighbors. If this process is repeated too often, the information becomes diluted, and the node representations become indistinguishable. This can lead to poor performance in node classification or link prediction tasks. Over-squashing is a related issue where the computational paths in a GNN grow exponentially with the depth of the model. However, when the model gets deeper, a node's receptive field grows exponentially with the number of layers. Therefore, much information is squashed into a single node vector, which may lose important information from the node far away from the target node, and the computational cost becomes prohibitively high. This can lead to poor scalability and inefficiency in GNN-based methods. To address these issues, researchers have developed various techniques such as graph attention mechanisms [157], residual connections [52], and skip connections [99]. Nevertheless, although these techniques can improve the performance of GNNs, they do not completely solve the challenges associated with message passing.

Transformers [154] have been very successful in the field of natural language processing (NLP) [20, 93, 10, 45] and have recently gained popularity in the field of graph representation learning as well. GNNs were previously the go-to method for graph repre-

sentation learning, but some recent works have shown that Transformers [188, 182, 169] can be used to achieve state-of-the-art performance on several graph-related tasks. The success of Transformers in NLP is largely due to their ability to model long-range dependencies in sequential data using self-attention mechanisms. This same self-attention mechanism can also be applied to graph data, where nodes can attend to other nodes based on their structural relationships, allowing for effective modeling of global dependencies in the graph. Plenty of graph transformer works have been proposed to incorporate the graph information into the transformer. To the best of my knowledge, existing graph transformer methods only consider the local graph structural information but neglect the global graph structural information, which degrades the graph transformer's performance. Local graph structural information refers to information directly connected to a particular node in the graph. In contrast, global graph structural information refers to information spread throughout the entire graph. Global structure information is critical for graph learning because it includes semantic information, such as the overall topic of the graph.

To solve this problem, a hierarchical structure graph transformer called HighFormer is proposed that leverages both the local and global structural information. Specifically, firstly, GNN was used to learn the initial graph node representation based on the local structural information. At the same time, a structural attention module is proposed to learn the global structural similarity. Then, the proposed structural attention matrix(SAM) was learned by combining the traditional attention matrix and the global structure similarity matrix.

Secondly, most of the existing DGL models are designed in (semi-)supervised scenarios [57, 72, 157], which require abundant manual labels for training. However, collecting manual labels is costly, especially for large-scale graph datasets like social networks. To address the manual-label issues of supervised DGL, self-supervised graph representation learning (SGL) [27] was proposed to learn graph representations without manual labels. In self-supervised graph representation learning, GNN-based models are trained by pretext tasks that do not require manual labels. The idea behind self-supervised learning is to use the input data structure to generate the supervision signals. The pretext tasks used in SGL are designed to encourage the model to learn important features of the graph structure, For example, masked graph generation [31, 183], which randomly masks some nodes or edges in a graph and then trains the model to predict the missing nodes or edges. This task encourages the model to learn meaningful representations of the graph structure. Also, graph contrastive learning [156, 148] is trained to distinguish between a pair of graphs (positive or negative). These tasks encourage the model to learn discriminative features of the graph structure that are critical for distinguishing between different graphs. Therefore, a well-designed pretext task can help the SGL model learn informative embeddings that improve the performance of downstream tasks.

Graph contrastive learning (GCL) has recently become the most promising technique for learning representations from graph-structured data. GCL aims to learn a representation function that maps graph data points into a low-dimensional vector space such that the Mutual Information (MI) [17] of the positive sample pairs is maximized, and the MI of

negative pairs is minimized. MI is a metric used to measure the statistical dependence between two random variables. Higher MI indicates a stronger relationship or dependency between the two variables. For example, if two points in a graph have a strong relationship, we think that these two points are positive samples. Otherwise, they are negative samples. In GCL, positive and negative samples are generated by leveraging graph augmentation techniques [185]. Graph augmentation involves applying a series of random perturbations to the input graph to generate new similar graphs that are not identical to the original graph. By maximizing the MI of positive pairs and minimizing the MI of negative pairs, GCL can learn to extract meaningful and informative features that capture the underlying structure of the graph. One advantage of GCL is that the learned representations are less sensitive to small changes or disturbances in the input data. This robustness is valuable because it allows the model to generalize well to new, slightly altered input data instances. In graph data, perturbations could refer to changes in the graph structure, node attributes, or other variations. GCL aims to ensure that the learned representations capture the underlying patterns in the data, making them more resistant to noise or small modifications. This can improve the performance in applications where the graph data is noisy or incomplete. Additionally, GCL can learn representations generalizable to unseen graphs, which is important for applications where the input graphs vary in size and structure. However, there are still several challenges and limitations to the existing GCL methods. I argue that there are two most critical problems with the existing GCL approaches. 1) Graph augmentation: Most existing graph augmentation methods perform stochastic transformation schemes [156, 129, 51], such as randomly dropping edges or masking node features. However, uniform transformations without carefully designed augmentation techniques may drastically change the underlying semantics of graphs or graph nodes. 2) Negative sampling: Negative sampling is a common technique used in graph contrastive learning methods. It involves selecting examples that are not similar to the positive example. Negative sampling helps train the model more efficiently, especially when dealing with large graphs, as it avoids considering all possible negative pairs. It focuses on a subset of negative examples, making the training process computationally feasible while providing valuable information for learning robust representations. However, existing graph contrastive methods neglect the semantic information that may introduce false-negative samples since they treat all the other samples except the positive sample as negative, such that some may belong to the same cluster or have similar semantic information to the positive sample.

To solve these issues, I propose a novel graph contrastive learning method with semantic invariance graph augmentation termed SemiGCL by designing a semantic invariance graph augmentation (SemiAug) and a semantic-based graph contrastive (SGC) scheme that leverages a semantic debiasing negative sampling (SDNS) method to generate negative samples. Concretely, a PageRank-based semantic clustering method is proposed to divide the graph into semantic clusters to learn a semantic invariance augmentation. Then, based on the cluster assignment, a semantic invariance augmentation was proposed on both graph structure and node attribute. Specifically, I designed two kinds of augmentations:

structure-level augmentation and attribute-level augmentation. For the structure level, I randomly add edges on the intra-class clusters. For the attribute level, similar to mixup [193], I adopted the operation of linear interpolation that mixes each node's features and their cluster prototypes to get the augmented features of each node. At last, I designed a semantic-based graph contrastive (SGC) method with SDNS, in which the negative samples were selected from other clusters except for the cluster where the positive sample was. Therefore, the semantic information was used to decrease the false-negative samples, improving the discriminative capability of the graph contrastive network.

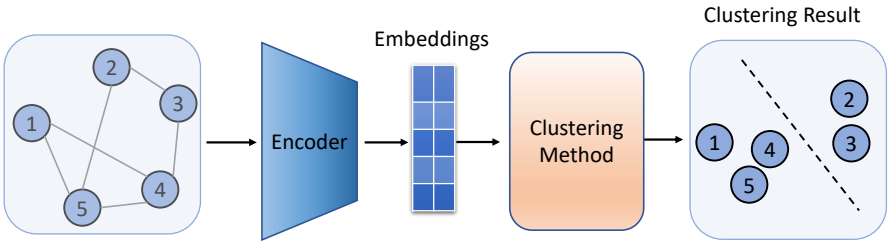

Fig. 1.2 The general pipeline of Self-supervised Deep Graph Clustering (SGC).

Thirdly, benefiting from the rapid development and the strong graph representation performance of self-supervised graph representation learning (SGL)[27], Self-supervised Deep Graph Clustering (SGC) has been more effective in node clustering for a graph than traditional clustering methods. SGC is particularly useful when the graph is large and complex. As the training data increases, the model becomes more accurate. SGC consists of a self-supervised encoder neural network that embeds the nodes into the representation space and a clustering model that separates the node representations into different groups. The general pipeline of the SGC method can be demonstrated in Figure 1.2. More recently, graph contrastive learning (GCL) [156, 129, 51] has become the most promising technique in SGC, benefiting from the powerful capability of capturing implicit supervision information [94].

Although the promising performance of Contrastive Graph Clustering (CGC) has been achieved, I found that the existing method still suffers from many limitations and drawbacks. First of all, existing GCL [206] methods typically focus on node views of graphs but do not explicitly incorporate the global structural information of the graph in the contrastive model. As a result, these methods may suffer from representation collapse [94], which refers to the situation where all nodes are mapped to the same representation. This can be problematic, especially in tasks where the graph structure is important, such as in graph clustering or classification. What's more, in the process of node clustering, most of the existing SGC methods leverage traditional clustering methods, for example, K-means. However, adopting the traditional clustering methods in SGC suffers from many

problems, such as being highly dependent on initialization and unable to optimize the node representation learning and clustering in a unified framework.

To overcome these limitations, I propose a novel contrastive deep graph clustering method called Structural Semantic Contrastive Deep Graph Clustering (SECRET) by designing a structure contrastive scheme (SCS) to keep the structural consistency of two different views and a self-supervised deep learning clustering (SDC) method, which proposed a comprehensive similarity measure criterion and consider the similar node as supervision information. The SCS incorporates the structural information of the graph in the contrastive model that promotes the representation to be more informative. The SDC jointly optimizes the node representation learning and clustering in a unified framework that leverages the strong graph representation capability of GNNs.

1.1 Motivation

Graph-structured data can be found in various real-world applications [89, 202, 181], such as Social networks, Recommender systems, Bioinformatics, Transportation networks, Knowledge graphs, and so on. Graph representation learning aims to learn low-dimensional vector representations of nodes in a graph that capture important structural and semantic information. One of the key developments that led to the popularity of graph representation learning was the introduction of deep learning techniques for graph data, called Deep Graph Representation Learning (DGL). Since then, numerous DGL methods have been proposed, ranging from graph convolutional networks (GCNs) [72] to graph autoencoders (GAEs) [71] to graph attention networks (GATs) [157]. These methods have achieved state-of-the-art performance on graph-based tasks such as node classification, link prediction, and graph clustering.

DGL has made significant progress in recent years. However, there are still several crucial challenges that the field faces, including in (Semi-)Supervised DGL, Self-supervised DGL, and Self-supervised Deep Graph Clustering, such as over-smoothing [200], over-squashing [113], graph augmentation [156], negative sampling [95], and graph clustering [187]. This thesis analyzes these challenges in detail and proposes three methods to solve the problems respectively.

1.2 Research questions and contributions

After providing the background of the DGL and the motivation for this research, the present study endeavors to formulate three research questions that will be answered within this thesis, and specific contributions will also be provided.

Q1. *In (semi-)supervised DGL, GNNs often suffer from over-smoothing and over-squashing problems caused by the message-passing paradigm, which will lead to poor performance in the following learning tasks. How do we solve these two challenges caused by message-passing?*

In Chapter 3, I propose a hierarchical structure graph transformer called HighFormer that combines local and global structure information with the Transformer. Graph Transformers use self-attention to encode the relationships between nodes in a graph, which allows them to capture long-range dependencies and global graph structure. This makes them particularly effective for tasks that require modeling complex relationships between graph nodes, such as graph and node classification. On the other hand, GNNs use message passing to iteratively update node representations based on the representations of their neighbors in the graph. This allows them to capture local graph structure and node features, making them well-suited for node classification and link prediction tasks. Therefore, these two methods show that both local and global graph structures are critical for node clustering and classification tasks.

In HighFormer, I use GNN to learn the initial graph node representation based on the message passing scheme with the local structure information. At the same time, a structural attention module is used to learn the global structural similarity. Then, I added the softmax attention matrix and the global structure similarity matrix to form the structural attention matrix. Specifically, the original graph was initially input into the GNN to get a local-structure-based representation. On the other hand, the graph structure was input into a structural attention module, which leverages Personalized PageRank to compute the global structural similarity. Then, the softmax attention matrix and the global structural similarity matrix were added to form an improved attention matrix. Finally, I compute the graph representation using the learned improved attention matrix. I use two softmax operations to ensure the final attention matrix considers the graph structure and node feature relationships. In my proposed method, I leverage the personal PageRank to mine the global structure information of the graph instead of the positional encoding. The proposed approach introduces both the local structure information and the global structure information that improve the discrimination of the Transformer. I have theoretically proved that the commonly used positional encoding Laplacian eigenvectors only introduce the local position of each node but can't express the global position information. Extensive experimental evaluation shows that the proposed HighFormer outperforms the positional-encoding-based graph Transformer methods.

Q2. *In self-supervised DGL, existing graph contrastive learning (GCL) methods usually leverage stochastic graph augmentation methods (randomly adding or dropping features and edges), which may change the underlying semantic information of the graph dramatically. Besides, the negative sampling (sampling the negative samples, which are data points dissimilar or unrelated to a given anchor) method may introduce false-negative samples (the model predicts the sample is negative, but the true result is positive) in the existing contrastive scheme. How do we solve these two*

main issues in GCL?

To solve these two critical problems in GCL, I consider the semantic information in graph augmentation and negative sampling. In Chapter 4, I propose a novel graph contrastive learning method with semantic invariance graph augmentation termed SemiGCL by designing a semantic invariance graph augmentation (SemiAug) and a semantic-based graph contrastive (SGC) scheme that leverages a semantic debiasing negative sampling (SDNS) method to generate negative samples. Concretely, a PageRank-based semantic clustering method is proposed to divide the graph into semantic clusters to learn a semantic invariance augmentation. Then, based on the cluster assignment, a semantic invariance augmentation was proposed on both graph structure and node attribute. Specifically, I have designed two kinds of augmentations: structure-level augmentation and attribute-level augmentation. For the structure level, I randomly add edges on the intra-class clusters, and for the attribute level, similar to mixup [193], I adapt the operation of linear interpolation that mixes each node's features and their cluster prototypes to get the augmented features of each node. At last, I designed a semantic-based graph contrastive (SGC) method with SDNS, in which I selected negative samples from other clusters except for the positive sample cluster. This method presents a novel GCL model, SemiGCL, by introducing the SemiAug, which generates semantic invariance augmentations. Graph augmentation is a critical component of GCL, so experimental results show that the proposed SemiAug can improve the representation performance. In this method, I introduce the semantic information to the graph augmentation and the graph contrastive scheme, then proposed SemiAug and SGC, respectively, that the semantic information improves the augmentation performance and decreases the false-negative samples that will enhance the discriminative capability of the graph contrastive network.

Q3. *Existing Contrastive Graph Clustering (CGC) has achieved promising performance, benefiting from its powerful capability of capturing implicit supervision information. However, CGC still suffers from two significant problems. The first issue is representation collapse, in which each node's representations are similar without discrimination. The second issue is that if traditional clustering methods are used in CGC, optimizing representation learning and clustering simultaneously will be challenging. How do we solve these two critical issues in CGC?*

There are many applications for DGl. DGC is one of the most critical applications, which helps summarize large datasets by identifying groups of similar objects or data points. This can help understand the underlying structure of the data and provide insights into the relationships between different variables. CGC has gained attention in recent years for its promising results.

In Chapter 5, I propose a novel contrastive deep graph clustering method called Structural Semantic Contrastive Deep Graph Clustering (SECRET) by designing a structure contrastive scheme (SCS) to keep the structural consistency of two different views and a self-supervised deep-learning-based clustering (SDC) method, which leverages a comprehensive similarity measure criterion considering both the attribute embedding similarity and the structural similarity, which better reveal node relationships and can be seen as supervision information, as similar nodes should be in the same clusters. The SCS incorporates the structural information of the graph in the contrastive model that promotes a more accurate representation. The SDC jointly optimizes the node representation learning and clustering in a unified framework that leverages the strong graph representation capability of GNNs.

1.3 Thesis outline

In this section, I will give a brief introduction of each chapter in this thesis as follows:

- **Chapter 1** introduces the research background, motivation, research problems, and the main contributions of this thesis.
- **Chapter 2** gives a comprehensive review of Deep Graph Representation Learning(DGL), including the (Semi-)Supervised DGL, Self-supervised DGL, and an important application of DGL, the Self-supervised Deep Graph Clustering, as shown in Figure 1.1. I also highlighted the gaps in existing research works, which will be solved in the following three chapters.
- **Chapter 3** introduces the proposed hierarchical structure graph transformer (HighFormer) that leverages local and global structural information, solving the over-smoothing and over-squashing problems caused by the message-passing paradigm.
- **Chapter 4** introduces the proposed graph contrastive learning method with semantic invariance graph augmentation (SemiGCL) by designing a semantic invariance graph augmentation (SemiAug) and a semantic-based graph contrastive (SGC) scheme that leverages a semantic debiasing negative sampling (SDNS) method to generate negative samples. In SemiGCL, the semantic information improves the augmentation performance and decreases the false-negative samples, which improves the discriminative capability of the graph contrastive network.
- **Chapter 5** introduces the proposed Structural Semantic Contrastive Deep Graph Clustering (SECRET) by designing a structure contrastive scheme (SCS) to keep the structural consistency of two different views and a self-supervised deep-learning-based clustering (SDC) method, which leverages a comprehensive similarity measure criterion considering both the attribute embedding similarity and the structural similarity, which better reveal node relationships and can be seen as supervision information, as similar nodes should be in the same clusters. The SCS incorporates

the global structural information of the graph in the contrastive model that promotes a more accurate representation. The SDC jointly optimizes the node representation learning and clustering in a unified framework that leverages the strong graph representation capability of GNNs.

- **Chapter 6** gives the conclusions of this thesis and provides some future works.

1.4 List of publications

Below is a list of the publications I have used to convey the research contributions I have made and documented in this thesis. However, most of the contents in this thesis haven't been published or are still under review.

1. **H. Zheng**, Z. Liang, F. Tian, and Z. Ming, "NMF-based comprehensive latent factor learning with multiview data," in Proc. IEEE Int. Conf. Image Process. (ICIP), Sep. 2019, pp. 489–493.

The list of the papers which haven't been published:

1. **H. Zheng**, F. Tian, "Hierarchical Structure Graph Transformer"
2. **H. Zheng**, F. Tian, "Graph Contrastive Learning with Semantic-invariance Graph Augmentation"
3. **H. Zheng**, F. Tian, "Structural Semantic Contrastive Deep Graph Clustering"

Chapter 2

Deep Graph Representation Learning

In this chapter, I will establish a common notation framework and give a comprehensive review of Deep Graph Representation Learning(DGL) methods, including the (Semi-)Supervised DGL, Self-supervised DGL, and an important application of DGL Deep Graph Clustering, as shown in Figure 1.1. I also highlighted the gaps in existing research works, which will be solved in the following three chapters.

2.1 (Semi-)Supervised Deep Graph Representation Learning

This section will comprehensively survey (Semi-)Supervised DGL methods. DGL refers to the task of learning low-dimensional representations or embeddings of nodes, edges, and entire graphs in a graph-structured data using deep learning techniques. The goal of DGL is to capture the structural, relational, and semantic information encoded in the graph data and to represent it in a continuous vector space, where it can be used for various downstream tasks such as node classification, link prediction, graph classification, and graph generation. DGL methods typically leverage deep neural network architectures, such as graph neural networks (GNNs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), or their variants, to learn expressive representations of graph-structured data. These architectures are designed to operate directly on the graph structure, allowing them to capture the local and global dependencies between nodes, edges, and graph neighborhoods. DGL has emerged as a powerful framework for learning representations of graph-structured data in various domains, including social networks, biological networks, citation networks, knowledge graphs, and recommendation systems. By leveraging deep learning techniques, DGL methods are able to capture complex relationships and patterns in graph data, enabling effective and scalable solutions for a wide range of graph-based tasks. There are two main categories of methods, i.e., Graph Neural Networks(GNNs) and Graph Transformers. GNNs[72] have become the primary strategies for graph representation learning, benefiting from their powerful expressive capabilities. Graph Transformers is a potent method that captures long-range dependencies based on its inherent characteristics.

2.1.1 Graph Neural Networks

In recent years, the success of neural networks has led to increased interest and research in pattern recognition and data mining. Many machine learning tasks that previously required handcrafted feature engineerings, such as object detection [133, 134], machine translation [101, 168], and speech recognition[54], have now been revolutionized by various end-to-end deep learning paradigms like convolutional neural networks (CNNs)[75], recurrent neural networks (RNNs) [42], and autoencoders [158]. It can be attributed to the rapidly developing computational resources, such as GPUs, big training data availability, and deep learning models' ability to extract latent representations from Euclidean data, such as images, text, and videos. For example, an image can be represented as a regular grid in Euclidean space, and a CNN can exploit the shift-invariance, local connectivity, and compositionality of image data. This allows CNNs to extract locally meaningful features shared across the entire dataset for various image analyses.

The traditional machine learning algorithms are only designed to work with Euclidean data. Recently, graphs have become an increasingly popular way of representing and analyzing data in various applications, which presents a challenge for traditional machine learning algorithms [9]. Graph data can be highly complex and variable, making applying standard algorithms that rely on assumptions such as instance independence and fixed-size input vectors difficult. Graph-based learning methods, on the other hand, are specifically designed to handle graph data, allowing them to exploit the rich relationships between nodes to make more accurate predictions. Graph-based learning approaches typically involve algorithms that operate directly on the graph structure rather than treating each node as an independent instance. These algorithms can perform operations such as message passing and graph convolutions, which allow them to propagate information and extract meaningful features from the graph structure. In recent years, a lot of research has focused on developing graph-based learning models, including graph neural networks (GNNs), a type of neural network designed to work with graph data. GNNs are highly effective for graph-related tasks, including node classification, link prediction, and graph clustering. GNNs propagate information across the graph using a set of learnable functions. The information is typically represented as node features, updated iteratively based on the features of their neighboring nodes. The updated features are then used to make predictions about the graph. One of the key challenges in designing GNNs is developing a function that can capture the graph structure meaningfully. Many approaches have been proposed, including spectral methods, message-passing, and attention mechanisms.

However, the message-passing paradigm, commonly used in graph neural networks (GNNs) and related models, can sometimes lead to over-squashing problems, particularly in deep architectures or when dealing with highly interconnected graphs. Over-squashing refers to the phenomenon where the information propagated through the graph becomes distorted or attenuated as it passes through multiple layers, resulting in the loss of important information or gradients. Over-squashing is a related issue where the computational paths in a GNN grow exponentially with the depth of the model. Figure 2.1 shows

that many GNN layers are needed to pass long-range messages. However, when the model gets deeper, a node's receptive field grows exponentially with the number of layers. Therefore, much information is squashed into a single node vector, which may lose important information from the node far away from the target node, and the computational cost becomes prohibitively high. This can lead to poor scalability and inefficiency in GNN-based methods. To address these issues, researchers have developed various techniques such as graph attention mechanisms [157], residual connections [52], and skip connections [99]. Nevertheless, although these techniques can improve the performance of GNNs, they do not completely solve the challenges associated with message passing. It has been proven that the expressive power of GNNs is at most as powerful as the Weisfeiler-Lehman (WL) graph isomorphism test [175]. The WL graph isomorphism test, also known as the WL algorithm or the WL refinement procedure, is a graph isomorphism testing algorithm developed by Boris Weisfeiler and Andréi Lehman in the 1980s. It is one of the most widely used methods for determining whether two graphs are isomorphic, meaning they have the same structure but potentially different node labels. The WL algorithm operates by iteratively refining the labeling of the nodes in the graph based on the local structures of the neighboring nodes. The basic steps of the algorithm are as follows: Initialization: Assign a unique label to each node in the input graphs. Iteration: For each node in the graph, construct a "signature" or "color" by concatenating its label with the sorted list of labels of its neighbors. Apply a hash function to the signatures to obtain a compact representation of the graph structure. Update the labels of the nodes based on the hashed signatures. Termination: Repeat the iteration process until convergence, where no new labels are assigned to any node or until a maximum number of iterations is reached. Comparison: Compare the final labels of the nodes in the two graphs. If the sets of labels are identical for corresponding nodes, then the graphs are considered isomorphic; otherwise, they are not. The WL algorithm is based on graph isomorphism refinement, where the initial labels are refined in each iteration to capture increasingly finer details of the graph structure. By iteratively refining the node labels based on the local neighborhood information, the algorithm can distinguish between non-isomorphic graphs that may have similar initial labeling. The WL algorithm has several desirable properties, including soundness (correctly identifying isomorphic and non-isomorphic graphs), efficiency (polynomial time complexity in most cases), and applicability to a wide range of graph types (including directed and labeled graphs). It is widely used in practice for graph isomorphism testing, subgraph isomorphism testing, and graph classification tasks in various fields such as computer vision, bioinformatics, and network analysis.

Background and Definition

Sperduti et al.[146] were the first to apply neural networks to graphs, which inspired early research on Graph Neural Networks (GNNs). The concept of GNNs was initially introduced by Gori et al. [40] and further developed by Scarselli et al. [139] and Gallicchio et al. [34]. These early studies focused on recurrent GNNs (RecGNNs), which learn

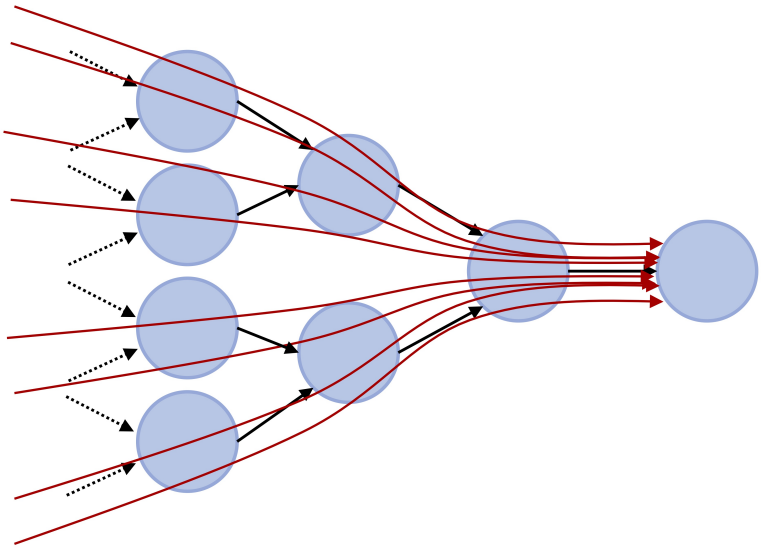

Fig. 2.1 The graphical representation of over-squashing.

a target node's representation by iteratively propagating neighbor information until a stable fixed point is reached. However, this process is computationally expensive, and recent efforts have been made to address these challenges [84, 18]. Following the success of CNNs in the computer vision domain, numerous methods have been developed that redefine the notion of convolution for graph data. These approaches can be divided into two main streams: spectral-based approaches and spatial-based approaches. Specifically, Spectral-based ConvGNNs [11] were first introduced by Bruna et al. and are based on the spectral graph theory. Since then, numerous improvements and extensions have been made on spectral-based ConvGNNs [53, 19, 72, 78]. Spatial-based ConvGNNs, on the other hand, were first addressed by Micheli et al. [109] but were not widely recognized until recently. Many spatial-based ConvGNNs have emerged in recent years [2, 119, 38].

Definition 1 (Graph): A graph can be represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes, and $\mathcal{E}$ is the set of edges. Let $v_i \in \mathcal{V}$ to denote a node and $e_{ij} = (v_i, v_j) \in \mathcal{E}$ to denote an edge pointing from v_j to v_i. The neighbors of node v are defined as $\mathcal{N}(v) = \{u \in \mathcal{V} \mid (v,u) \in \mathcal{E}\}$. The adjacency matrix $\mathbf{A}$ is a $n \times n$ matrix with $A_{ij} = 1$ if $e_{ij} \in E$ and $A_{ij} = 0$ if $e_{ij} \notin E$. A graph may have node attributes $\mathbf{X}$, which is also called the attributed graph, where $\mathbf{X} \in \mathbf{R}^{n \times d}$ is a node feature matrix with $\mathbf{x}_v \in \mathbf{R}^d$ representing the feature vector of a node v. Meanwhile, a graph may have edge attributes $\mathbf{X}^e$, where $\mathbf{X}^e \in \mathbf{R}^{m \times c}$ is an edge feature matrix with $\mathbf{x}^e_{v,u} \in \mathbf{R}^c$ representing the feature vector of an edge (v,u).

Definition 2 (Directed Graph): The directed graph, also known as a digraph, each edge has a direction associated with it. If there is an edge from node i to node j, I can only travel from i to j along that edge and not in the opposite direction. For an undirected graph, if there is an edge from node i to node j, there is also an edge from node j to node i. Therefore, the adjacency matrix of an undirected graph is symmetric. Conversely, if the adjacency matrix of a graph is symmetric, then for any pair of nodes i and j, if there is an

edge from i to j, there is also an edge from j to i. Thus, the graph is undirected. Therefore, a graph is undirected if its adjacency matrix is symmetric.

Recurrent Graph Neural Networks

Recurrent Graph Neural Networks (RecGNNs) are one of the earliest approaches to learning node representations in graph-structured data using recurrent neural architectures. RecGNNs assume that the nodes in a graph communicate or exchange information with their neighbors until a stable state or equilibrium is reached. This is achieved through an iterative process where each node aggregates information from its neighbors and updates its state accordingly. The concept of message passing, central to RecGNNs, has been inherited by more recent approaches to graph neural networks, such as Convolutional Graph Neural Networks (ConvGNNs). ConvGNNs extend the message-passing approach to spatial domains, where a convolution operation replaces the message-passing operation. This allows ConvGNNs to learn local patterns in the graph, just as convolutional neural networks learn local patterns in images.

At a high level, RecGNNs are composed of two main components: a recurrent neural network (RNN) and a graph neural network (GNN). The RNN component is responsible for processing the sequential information within the graph, while the GNN component is responsible for processing the structural information of the graph. The RNN component typically takes the form of a long short-term memory (LSTM) network, a neural network designed to handle sequential data. The LSTM component receives a sequence of node embeddings, which are representations of the nodes in the graph, and uses its recurrent connections to capture the temporal dependencies between them. The GNN component, on the other hand, is responsible for processing the graph structure. This is done by aggregating information from neighboring nodes and edges to update each node's representation. There are many ways to design the GNN component, but one popular approach is to use a message-passing scheme, in which each node sends messages to its neighbors, and the messages are used to update the node embeddings. Once the RNN and GNN components have processed the graph data, their outputs are combined and fed into a final output layer, producing the final predictions for the task.

Limited by the available computational power, during the early stages of research, the primary focus was on directed acyclic graphs [146, 110]. Scarselli et al. [139] proposed a new GNN model(SGNN) based on an information diffusion mechanism that recurrently updates nodes' states by neighborhood information. The diffusion mechanism refers to the process by which particles, molecules, or other entities spread or move from higher concentration areas to lower concentration regions, resulting in a net movement down a concentration gradient. Diffusion is a fundamental concept in physics, chemistry, biology, and various other fields, and it plays a crucial role in many natural processes and phenomena. In the GNN model, the node information can also move from high-quality

information nodes to their neighbors. The update function is as follows:

$$\mathbf{h}_v^{(t)} = \sum_{u \in N(v)} f\left(\mathbf{x}_v, \mathbf{x}_{(v,u)}^{\mathbf{e}}, \mathbf{x}_u, \mathbf{h}_u^{(t-1)}\right) \tag{2.1}$$

Where $f(\cdot)$ is the representation learning function, which maps the original graph features and structural information to a latent space. In graph-based machine learning or deep learning models, this function is typically implemented by a neural network architecture designed to learn meaningful representations of the input graph data. SGNN propagates the node state and computes the parameter gradient alternately to minimize the training objective, and once a convergence criterion is met, the last step of the node hidden states is sent to a readout layer. This approach enables SGNN to handle cyclic graphs effectively. Then, inspired by the concept of Echo State Networks (ESNs), GraphESN [34] is proposed; the main advantage of GraphESN is that it can learn complex nonlinear dynamics of the input graph while retaining a low number of trainable parameters. This makes it particularly suitable for processing large-scale graphs. GraphESN is also computationally efficient, allowing it to process large graphs in real-time.

Gated Graph Neural Network (GGNN) [84] extends LSTM to work on graphs, allowing it to model the complex relationships and dependencies between nodes in a graph. The key innovation of GGNN is the use of gated recurrent units (GRUs) [15] to model the interactions between nodes in a graph. GRUs allow GGNN to selectively update and forget information at each node based on the node's current state and the state of its neighbors. This makes GGNN particularly effective in processing large, complex graphs. The update function is defined as:

$$\mathbf{h}_v^{(t)} = GRU\left(\mathbf{h}_v^{(t-1)}, \sum_{u \in N(v)} \mathbf{W}\mathbf{h}_u^{(t-1)}\right) \tag{2.2}$$

In this function, the representation of a node in one layer is influenced by the representations of its neighboring nodes in the previous layer. This is fundamental to how GGNN operates, as they leverage information from the graph structure to update node representations. where $\mathbf{h}_v^{(0)} = \mathbf{x}_v$ is the initial features of each graph nodes. Compared to SGNN and GraphESN, GGNN uses the back-propagation through time (BPTT) algorithm for learning the model parameters. However, this approach can pose a challenge for large graphs, as GGNN requires running the recurrent function multiple times over all nodes, which necessitates storing the intermediate states of all nodes in memory.

Convolutional Graph Neural Networks

Convolutional graph neural networks (ConvGNNs) generalize the concept of convolution from grid data (e.g., images) to graph data by defining a neighborhood of nodes around each node in the graph. In ConvGNNs, a node's representation is generated by aggregating its features and the features of its neighboring nodes. This aggregation operation is typically performed using a weighted sum of the neighbor features, where the weights are learned

during training. The resulting aggregated feature vector is then passed through a non-linear activation function to update the node's representation. ConvGNNs can be used for various tasks on graph-structured data, such as node classification, link prediction, and graph classification. They have been shown to achieve state-of-the-art performance on several benchmark datasets in these domains.

ConvGNNs differ from recurrent graph neural networks because they do not iterate node states with contractive constraints. Instead, ConvGNNs address the cyclic mutual dependencies by using a fixed number of layers with varying weights in each layer. Graph convolutions are more efficient and convenient to combine with other neural networks, making ConvGNNs increasingly popular. ConvGNNs can be divided into two categories: spectral-based and spatial-based. Spectral-based approaches introduce filters from the perspective of graph signal processing, interpreting the graph convolutional operation as removing noise from graph signals. Spatial-based approaches define graph convolutions by information propagation and inherit ideas from RecGNNs. GCN [15] bridged the gap between spectral-based and spatial-based approaches, and spatial-based methods have recently developed rapidly due to their efficiency, flexibility, and generality.

A. Spectral-based ConvGNNs

The spectral-based ConvGNNs are a class of Graph Neural Networks (GNNs) that operate in the spectral domain of graphs. They use the eigendecomposition of the graph Laplacian matrix to define graph convolutions that can be efficiently computed in the frequency domain. The Laplacian matrix is a square matrix that represents a graph. It is derived from the graph's adjacency matrix and provides valuable information about its structure and properties. The Laplacian matrix is widely used in various fields, including graph theory, spectral graph theory, machine learning, and image processing. Spectral-based ConvGNNs define the convolution operation as a filter function that operates on the eigenvalues of the graph Laplacian matrix. The filter function is usually designed to be localized in the frequency domain, allowing it to capture local graph structure while invariant to graph isomorphism. The use of the Laplacian matrix and its eigenvectors also enables ConvGNNs to incorporate information about the global structure of the graph, making them effective at capturing long-range dependencies and global patterns. Spectral-based techniques have a strong mathematical basis in graph analysis [145, 138, 13]. These methods require graphs to be undirected, and they use the normalized graph Laplacian matrix to represent an undirected graph mathematically. The normalized graph Laplacian can be defined as $\mathbf{L} = \mathbf{I_n} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{D}$ represent the node degree matrix, and it is a diagonal matrix, $\mathbf{D}_{ii} = \sum_j \left(\mathbf{A}_{i,j}\right)$. Given that the normalized graph Laplacian matrix is both real, symmetric, and positive semidefinite, it can be factored in the following way: $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where $\mathbf{U} = [\mathbf{u_0}, \mathbf{u_1}, \cdots, \mathbf{u_{n-1}}] \in \mathbf{R}^{n \times n}$ is the matrix of eigenvectors which ordered by eigenvalues in ascending order and Λ is the diagonal matrix of eigenvalues or the spectrum, $\Lambda_{ii} = \lambda_i$. An orthonormal space is formed by the eigenvectors of the normalized Laplacian matrix., in mathematical words $\mathbf{U}^T\mathbf{U} = \mathbf{I}$. $\mathbf{x} \in \mathbf{R}^n$ represent a graph signal. x_i represent the value of the i^{th} node in a graph. The graph Fourier transform is

defined as $\mathcal{F}(\mathbf{x}) = \mathbf{U}^T\mathbf{x}$, and the inverse graph Fourier transform is $\mathcal{F}^{-1}(\hat{\mathbf{x}}) = \mathbf{U}\hat{\mathbf{x}}$, where $\hat{\mathbf{x}}$ represents the resulted signal from the graph Fourier transform. Using the eigenvectors of the normalized graph Laplacian as its basis, the graph Fourier transform projects the input graph signal onto an orthonormal space. The input signal is $\mathbf{x} = \sum_i \hat{x}_i \mathbf{u}_i$. Therefore, the graph convolution of $\mathbf{x}$ with a filter $\mathbf{g} \in \mathbf{R}^n$ can be defined as:

$$\begin{aligned} \mathbf{x} *_G \mathbf{g} &= \mathcal{F}^{-1}(\mathcal{F}(\mathbf{x}) \odot \mathcal{F}(\mathbf{g})) \\ &= \mathbf{U}\left(\mathbf{U}^T\mathbf{x} \odot \mathbf{U}^T\mathbf{g}\right), \end{aligned} \tag{2.3}$$

where $\odot$ represents the element-wise product. If I define a filter as $\mathbf{g}_\theta = \text{diag}\left(\mathbf{U}^T\mathbf{g}\right)$, then the spectral graph convolution can be defined as

$$\mathbf{x} *_G \mathbf{g}_\theta = \mathbf{U}\mathbf{g}_\theta\mathbf{U}^T\mathbf{x} \tag{2.4}$$

All Spectral-based ConvGNNs adhere to this definition, but the main distinction lies in selecting the filter $\mathbf{g}_\theta$.

The graph convolutional layer of the Spectral Convolutional Neural Network (Spectral CNN)[11] is defined by assuming the filter $\mathbf{g}_\theta = \Theta_{i,j}^{(k)}$ to be a set of learnable parameters, and taking into account graph signals with multiple channels. The layer, denoted by $\mathbf{H}_{:,j}^{(k)}$, is given by the equation:

$$\mathbf{H}_{:,j}^{(k)} = \sigma\left(\sum_{i=1}^{f_{k-1}} \mathbf{U}\Theta_{i,j}^{(k)}\mathbf{U}^T\mathbf{H}_{:,i}^{(k-1)}\right) \quad (j = 1, 2, \cdots, f_k), \tag{2.5}$$

where k represents the layer index, $\mathbf{H}^{(k-1)} \in \mathbf{R}^{n \times f_{k-1}}$ represents the input graph signal, $\mathbf{H}^{(0)} = \mathbf{X}$, f_{k-1} represents the number of input channels, f_k represents the number of output channels, and $\Theta_{i,j}^{(k)}$ is a diagonal matrix with learnable parameters. Spectral CNN encounters some limitations due to the eigendecomposition necessitating a computational complexity of $O\left(n^3\right)$. ChebNet [19] and GCN [72] address these issues in subsequent works by simplifying and making several approximations, thereby reducing the computational complexity to $O(m)$.

ChebNet approximates the filter $\mathbf{g}\theta$ using Chebyshev polynomials of the diagonal matrix of eigenvalues, such that $\mathbf{g}\theta$ is defined as $\mathbf{g}_\theta = \sum_{i=0}^{K} \theta_i T_i(\tilde{\mathbf{\Lambda}})$, where $\tilde{\mathbf{\Lambda}} = 2\mathbf{\Lambda}/\lambda_{\max} - \mathbf{I_n}$, and $\tilde{\Lambda} \in [-1, 1]$. The Chebyshev polynomials are defined as $T_i(\mathbf{x}) = 2\mathbf{x}T_{i-1}(\mathbf{x}) - T_{i-2}(\mathbf{x})$ with $T_0(\mathbf{x}) = 1$ and $T_1(\mathbf{x}) = \mathbf{x}$. Therefore, the convolution of $\mathbf{x}$ with the filter g_θ is

$$\mathbf{x} *_G \mathbf{g}_\theta = \mathbf{U}\left(\sum_{i=0}^{K} \theta_i T_i(\tilde{\mathbf{\Lambda}})\right)\mathbf{U}^T\mathbf{x}, \tag{2.6}$$

where $\tilde{\mathbf{L}} = 2\mathbf{L}/\lambda_{\max} - \mathbf{I_n}$. As $T_i(\tilde{\mathbf{L}}) = \mathbf{U}T_i(\tilde{\mathbf{\Lambda}})\mathbf{U}^T$, ChebNet can be written as

$$\mathbf{x} *_G \mathbf{g}_\theta = \sum_{i=0}^{K} \theta_i T_i(\tilde{\mathbf{L}})\mathbf{x} \tag{2.7}$$

CayleyNet [78] is a type of GCN based on the Cayley transform a mathematical operation that maps a graph structure onto a continuous space. CayleyNet uses this transform to embed the graph structure into a high-dimensional Euclidean space, where convolutional operations can be performed using standard neural network layers. This approach lets CayleyNet capture the graph structure's local and global features.

ChebNet allows for higher-order approximations of graph convolution, while Graph Convolutional Network (GCN)[72] only performs first-order approximations. Assuming $K = 1$ and $\lambda_{\max} = 2$, Equation 2.7 is become

$$\mathbf{x} *_G \mathbf{g}_\theta = \theta_0 \mathbf{x} - \theta_1 \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}} \mathbf{x} \tag{2.8}$$

GCN assumes additional constraints to limit the number of parameters $\theta = \theta_0 = -\theta_1$ So that the definition of graph convolution is

$$\mathbf{x} *_G \mathbf{g}_\theta = \theta \left(\mathbf{I_n} + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}} \right) \mathbf{x} \tag{2.9}$$

To write the equation in a matrix style. Equation 2.9 become

$$\mathbf{H} = \mathbf{X} *_G \mathbf{g}_{\mathbf{\Theta}} = f(\overline{\mathbf{A}} \mathbf{X} \mathbf{\Theta}) \tag{2.10}$$

where $\overline{\mathbf{A}} = \mathbf{I_n} + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$ and $f(\cdot)$ represent an activation function. $\mathbf{I_n} + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$ may cause numerical instability so that GCN leverage a normalization by $\overline{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}$ with $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I_n}$ and $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$

Despite being a spectral-based method, GCN can also be interpreted as a spatial-based method. This is because, from a spatial-based perspective, GCN aggregates feature information from a node's neighborhood. so that Equation 2.10 can be written as

$$\mathbf{h}_v = f\left(\mathbf{\Theta}^T \left(\sum_{u \in \{N(v) \cup v\}} \bar{A}_{v,u} \mathbf{x}_u \right) \right) \quad \forall v \in V. \tag{2.11}$$

Several recent works have explored alternative symmetric matrices in pursuing incremental improvements over GCN [22]. AGCN [82] is an example of such an approach, which learns hidden structural relations not specified by the graph adjacency matrix. This is achieved by constructing a residual graph adjacency matrix through a learnable distance function that inputs the features of two nodes. DGCN [207], on the other hand, introduces a dual graph convolutional architecture that utilizes two parallel graph convolutional layers sharing parameters. These layers employ the normalized adjacency matrix $\overline{\mathbf{A}}$ and the positive pointwise mutual information (PPMI) matrix, which captures nodes' co-occurrence information through random walks sampled from a graph.

B. Spatial-based ConvGNNs

The spatial-based ConvGNNs involve applying convolutional filters to the node features in the graph structure, similar to how CNNs apply filters to the pixels in an image. These filters are typically designed to consider the graph's local structure, such as the nodes and

edges surrounding a given node. By applying multiple convolutional layers, the model can learn hierarchical representations of the graph data, enabling it to perform node or graph classification tasks.

Neural Network for Graphs (NN4G)[109] is a contextual constructive approach that uses context to inform the construction of new nodes and edges in the graph. The algorithm is based on a neural network architecture that can learn from both the graph structure and additional features of the data being analyzed. NN4G computes the node states of each layer by

$$\mathbf{h}_v^{(k)} = f\left(\mathbf{W}^{(k)^T}\mathbf{x}_v + \sum_{u\in N(v)} \Theta^{(k)^T}\mathbf{h}_u^{(k-1)}\right) \tag{2.12}$$

where $f(\cdot)$ denotes the activation function, ReLU, which sets negative input values to zero and leaves positive values unchanged. It has become the most widely used activation function due to its simplicity and effectiveness in deep neural networks. $\Theta^{(k)^T}$ is the weight of the connection between the current node and its neighbors. $\mathbf{W}^{(k)^T}$ is the weight of current node. u is one of the nearest neighbors of node v. Equation 2.12 can transform to the matrix style as

$$\mathbf{H}^{(k)} = f\left(\mathbf{X}\mathbf{W}^{(k)} + \mathbf{A}\mathbf{H}^{(k-1)}\Theta^{(k)}\right), \tag{2.13}$$

Contextual Graph Markov Model (CGMM)[4] is a generative model that learns to represent the probability distribution of a graph using a deep neural network architecture. The key innovation of the CGMM is the use of contextual information, which allows the model to capture the local dependencies between nodes and edges in a graph as well as the global structure of the graph. The model also incorporates a Markovian structure to capture the temporal dependencies in a sequence of graphs. Diffusion Convolutional Neural Network (DCNN) [2], which combines the ideas of graph convolutional neural networks and diffusion processes. The key idea behind DCNNs is to model the diffusion process on the graph as a series of convolutions. In other words, instead of applying convolutional filters directly to the graph, DCNNs use a diffusion process to propagate information across the graph and then apply the convolutional filters. The diffusion graph convolution is defined as

$$\mathbf{H}^{(k)} = f\left(\mathbf{W}^{(k)} \odot \mathbf{P}^k\mathbf{X}\right) \tag{2.14}$$

where $f(\cdot)$ denote the activation function and $\mathbf{P} \in \mathbf{R}^{n\times n}$ is the probability transition matrix and it is computed by $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$. $\mathbf{H}^{(k)}$ denote the hidden representation matrix. At last, the output of the DCNN concatenates $\mathbf{H}^{(1)}, \mathbf{H}^{(2)}, \cdots, \mathbf{H}^{(K)}$ together.

Diffusion Graph Convolution (DGC)[85] architecture consists of two main components: a diffusion convolutional layer that extracts spatial features from the adjacency matrix representing the road network and a recurrent layer that captures temporal dependencies

using a gated recurrent unit (GRU) network. The diffusion graph convolution is defined as

$$\mathbf{H} = \sum_{k=0}^{K} f\left(\mathbf{P}^k \mathbf{X} \mathbf{W}^{(k)}\right) \tag{2.15}$$

where $\mathbf{W}^{(k)} \in \mathbf{R}^{D \times F}$ and $f(\cdot)$ denote the activation function.

Partition Graph Convolution (PGC)[177] consists of multiple layers of spatial-temporal graph convolutional networks, which operate on the graph-structured data to learn spatial-temporal features for action recognition. The authors also introduce a new pooling operation called temporal pyramid pooling, which aggregates features across multiple temporal scales. The update function is

$$\begin{gathered} \mathbf{H}^{(k)} = \sum_{j=1}^{Q} \overline{\mathbf{A}}^{(j)} \mathbf{H}^{(k-1)} \mathbf{W}^{(j,k)}, \\ \text{where } \mathbf{H}^{(0)} = \mathbf{X}, \overline{\mathbf{A}}^{(j)} = \left(\tilde{\mathbf{D}}^{(j)}\right)^{-\frac{1}{2}} \tilde{\mathbf{A}}^{(j)} \left(\tilde{\mathbf{D}}^{(j)}\right)^{-\frac{1}{2}} \text{ and } \tilde{\mathbf{A}}^{(j)} = \mathbf{A}^{(j)} + \mathbf{I}. \end{gathered} \tag{2.16}$$

Message Passing Neural Network (MPNN) [38] considers further information by running K-step message passing iteratively. So that the message-passing function can be defined as

$$\mathbf{h}_v^{(k)} = U_k\left(\mathbf{h}_v^{(k-1)}, \sum_{u \in N(v)} M_k\left(\mathbf{h}_v^{(k-1)}, \mathbf{h}_u^{(k-1)}, \mathbf{x}_{vu}^e\right)\right) \tag{2.17}$$

where $\mathbf{h}_v^{(0)} = \mathbf{x}_v, U_k(\cdot)$. For graph-level tasks, a readout function is defined as

$$\mathbf{h}_G = R\left(\mathbf{h}_v^{(K)} \mid v \in G\right) \tag{2.18}$$

Later, some improved MPNN methods were designed by using different $U_k(\cdot), M_k(\cdot)$, and $R(\cdot)$ [72, 23, 67, 140]. Graph Isomorphism Network (GIN)[174] is a graph neural network architecture that aims to address the drawback of previous message-passing neural network (MPNN)-based methods in distinguishing different graph structures based on the graph embeddings they produce. The key idea behind GIN is to modify the aggregation step in MPNNs by incorporating a learnable parameter $\varepsilon^{(k)}$ that adjusts the weight of the central node during each iteration of message passing. The graph convolutions are defined as

$$\mathbf{h}_v^{(k)} = MLP\left(\left(1 + \varepsilon^{(k)}\right)\mathbf{h}_v^{(k-1)} + \sum_{u \in N(v)} \mathbf{h}_u^{(k-1)}\right) \tag{2.19}$$

where $MLP(\cdot)$ denote the multi-layer perceptron.

One of the main challenges in training GNNs on large graphs is the computational cost associated with processing the full neighborhood of each node. To address this challenge, GraphSage [50] uses a sampling-based approach to obtain a fixed number of neighbors for each node. The basic idea is to randomly select a fixed number of neighbors for each node and then aggregate the information from these neighbors to compute the

node's representation. This approach allows GraphSage to scale to large graphs with millions or billions of nodes and edges while achieving high predictive accuracy. The graph convolutions are defined as

$$\mathbf{h}_v^{(k)} = \sigma\left(\mathbf{W}^{(k)} \cdot f_k\left(\mathbf{h}_v^{(k-1)}, \left\{\mathbf{h}_u^{(k-1)}, \forall u \in S_{\mathcal{N}(v)}\right\}\right)\right), \tag{2.20}$$

where $\mathbf{h}_v^{(0)} = \mathbf{x}_v, f_k(\cdot)$ denote an aggregation function, $S_{\mathcal{N}(v)}$ is a function that sampling node v 's neighbors randomly.

Graph attention networks (GAT) [155] use an attention mechanism based on the graph structure to weigh the contribution of each node's neighbors in the graph convolution operation. The authors argue that this approach not only learns richer representations of nodes but also allows the model to adaptively adjust the importance of different neighbors of a node for different tasks. The graph convolutional of GAT is defined as

$$\mathbf{h}_v^{(k)} = \sigma\left(\sum_{u \in \mathcal{N}(v) \cup v} \alpha_{vu}^{(k)} \mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)}\right) \tag{2.21}$$

where $\mathbf{h}_v^{(0)} = \mathbf{x}_v$. The weight $\alpha_{vu}^{(k)}$ measures the similarity of the node v and its neighbor u :

$$\alpha_{vu}^{(k)} = \text{softmax}\left(g\left(\mathbf{a}^T\left[\mathbf{W}^{(k)}\mathbf{h}_v^{(k-1)} \| \mathbf{W}^{(k)}\mathbf{h}_u^{(k-1)}\right)\right)\right), \tag{2.22}$$

where $g(\cdot)$ is the LeakyReLU activation function and $\mathbf{a}$ is the parameter vector. GAT also employs multi-head attention to increase the model's expressive capability. Gated Attention Network (GAAN)[194]extends the basic attention mechanism to include a gating mechanism that computes an additional attention score for each attention head.

GeniePath[96] further extends this approach by proposing an LSTM-like gating mechanism to control the flow of information across graph convolutional layers. This mechanism, called the adaptive receptive field, selectively allows or blocks the flow of information from neighboring nodes based on their relevance to the task at hand. By adopting the receptive field based on the node's representation and the current task, the model can better capture relevant information and ignore irrelevant information.

Mixture Model Network (MoNet)[115] adopts a different approach to assign weights to a node's neighbors. MoNet introduces node pseudo-coordinates to determine the relative position between a node and its neighbor. These pseudo-coordinates can be thought of as virtual positions that do not depend on the actual positions of nodes in space but rather on the graph's connectivity. MoNet provides a flexible framework that can generalize several existing approaches for manifolds and graphs by constructing non-parametric weight functions. Specifically, MoNet generalizes approaches such as Geodesic CNN (GCNN)[106], Anisotropic CNN (ACNN)[8], Spline CNN[32], GCN[72], and DCNN[2] as special instances of MoNet. These approaches typically define weight functions based on the geometrical properties of the manifold or graph, such as distances between nodes or curvatures of the manifold. By constructing non-parametric weight functions, MoNet

can incorporate these geometric properties and learn personalized weight vectors for each node.

PATCHY-SAN[119] is a method of achieving weight sharing across different locations in a graph using a learnable weight associated with the ranking of a node's neighbors based on certain criteria. Specifically, PATCHY-SAN orders a node's neighbors based on their graph labelings, derived from node degree, centrality, and Weisfeiler-Lehman color. These labelings are essentially node scores, and the top q neighbors are selected for each node. The ordered neighbors can then be treated as a fixed sequence, which allows the graph-structured data to be converted into grid-structured data. This enables using a standard 1D convolutional filter to aggregate neighborhood feature information, where the order of the filter's weights corresponds to the order of a node's neighbors. Large-scale Graph Convolutional Network (LGCN) [35] is a powerful method for processing large-scale graphs. By sorting the feature matrix of a node's neighborhood based on feature information, LGCN can effectively capture local structural information and enhance the node representation. This can be especially useful in applications such as recommendation systems or social network analysis, where understanding the relationships between nodes is critical.

Spatial-temporal Graph Neural Networks

Spatial-temporal graph neural networks (STGNNs) are a type of neural network architecture designed to learn from spatial-temporal graphs. These graphs represent data that vary over both space and time, such as traffic patterns [102], human movements [177], or weather conditions [172]. STGNNs leverage graph convolutions, a convolutional operation that operates on graphs instead of grids or images. Graph convolutions allow the network to capture spatial dependencies between nodes in the graph, which is crucial for tasks such as traffic speed forecasting or human action recognition. In addition to graph convolutions, STGNNs typically incorporate recurrent neural networks (RNNs) or convolutional neural networks (CNNs) to model the temporal dependencies between data points over time. This allows the network to learn how patterns evolve and change over time and to make accurate predictions or classifications based on these patterns.

Many real-world applications involve dynamic graphs in terms of their structures and inputs, and STGNNs are a powerful tool to capture this dynamicity. As you mentioned, these methods often assume interdependencies between connected nodes and aim to model the dynamic node inputs over time. In the case of traffic speed forecasting, for example, it's important to consider the spatial dependencies between roads to predict future speeds accurately. STGNNs typically fall into two categories: RNN-based methods and CNN-based methods. The former often uses a recurrent neural network (RNN) architecture to model the temporal dependencies between node inputs, while the latter uses a convolutional neural network (CNN) to capture the spatial dependencies between nodes. Some STGNNs combine these two approaches to model the dynamic graph structure and

inputs comprehensively. Overall, STGNNs are an exciting area of research with great potential for a wide range of dynamic graph applications.

RNN-based approaches for STGNNs often incorporate graph convolutions [194, 142, 85] to capture spatial-temporal dependencies in the input and hidden states. Graph convolutions are particularly useful because they allow the model to consider the node's features and neighborhood relationships. By combining graph convolutions with an RNN architecture, these methods can capture both spatial and temporal dependencies in the graph data. Suppose a simple RNN is defined as

$$\mathbf{H}^{(t)} = \sigma\left(\mathbf{W}\mathbf{X}^{(t)} + \mathbf{U}\mathbf{H}^{(t-1)} + \mathbf{b}\right) \tag{2.23}$$

where $\mathbf{X}^{(t)} \in \mathbf{R}^{n\times d}$ denote the node feature matrix at the time of t. After introducing the graph convolution, Equation 2.23 can be written as

$$\mathbf{H}^{(t)} = \sigma\left(G\mathrm{con}\,v\left(\mathbf{X}^{(t)}, \mathbf{A}; \mathbf{W}\right) + G\mathrm{conv}\left(\mathbf{H}^{(t-1)}, \mathbf{A}; \mathbf{U}\right) + \mathbf{b}\right) \tag{2.24}$$

where $G\mathrm{con}\,v(\cdot)$ denote the graph convolutional layer.

Graph Convolutional Recurrent Network (GCRN)[142] combines an LSTM network with ChebNet [19], a type of graph convolutional network that uses Chebyshev polynomials to approximate the graph Laplacian. In GCRN, the ChebNet layer captures the spatial dependencies between nodes, and the LSTM captures temporal dependencies in the input data. By combining these two types of layers, GCRN can effectively model spatial and temporal dependencies. Diffusion Convolutional Recurrent Neural Network (DCRNN)[85], on the other hand, incorporates a diffusion graph convolutional layer into a GRU network. This diffusion graph convolutional layer applies a diffusion process to Laplacian graph to capture the spatial dependencies between nodes.

Structural-RNN [62] proposes a recurrent framework that uses node-level and edge-level RNNs to predict node labels at each time step. This approach considers the graph data's temporal and spatial dependencies by processing the temporal information of each node and edge separately using different RNNs. The node-RNN is used to capture the temporal information for each node, while the edge-RNN captures temporal information for each edge. To incorporate the spatial information between nodes, the output of the edge-RNN is used as input to the node-RNN. To make the model more computationally feasible, the authors split nodes and edges into semantic groups, where nodes or edges in the same group share the same RNN model. By using both node-level and edge-level RNNs, Structural-RNN can capture complex temporal and spatial dependencies within a graph.

The CGCN [186] model combines 1D convolutional layers and graph convolutional layers, which capture spatial and temporal dependencies in graph data. The model consists of a spatial-temporal block comprising three layers: a gated 1D convolutional layer, a graph convolutional layer, and another gated 1D convolutional layer. The gated 1D convolutional layers capture temporal dependencies in the data, while the graph convolutional layer captures spatial dependencies between nodes. CGCN can effectively model spatial and

temporal dependencies by combining these two types of layers. CGCN also incorporates a gated mechanism to control the flow of information between layers, helping the model to better capture temporal dependencies in the data.

Traditional graph convolutional networks rely on a fixed adjacency matrix to represent the relationships between nodes. However, in many real-world applications, the relationships between nodes can change over time, making using a fixed adjacency matrix difficult. Graph WaveNet[171] addresses this issue using a self-adaptive adjacency matrix learned from the input data. The self-adaptive adjacency matrix is generated using a dilated causal convolution similar to the one in WaveNet. This convolution is applied to the input feature matrix, generating the filter coefficients to construct the self-adaptive adjacency matrix. Using a dilated causal convolution, Graph WaveNet can capture spatial and temporal dependencies in graph data. The self-adaptive adjacency matrix is

$$\mathbf{A}_{adp} = \text{SoftMax}\left(\text{ReLU}\left(\mathbf{E}_1\mathbf{E}_2^T\right)\right), \tag{2.25}$$

where **E1** is the source node embedding and **E2** is the target node embedding with learnable parameters. The multiplication of **E1** with **E2** generates the adjacency matrix, representing the dependency weights between the source node and the target node.

Gated Attention Networks (GaAN) [194] is a graph neural network architecture that utilizes attention mechanisms to learn dynamic spatial dependencies in graph data. GaAN uses an RNN-based approach to model temporal dependencies in the data, while the attention mechanism is used to update the edge weights between pairs of connected nodes. The attention function in GaAN considers both the node features and the edge weights between pairs of nodes to dynamically update the relationships between nodes, allowing the model to adapt to changes in the graph structure over time. Another attention-based method, the Attention-Based Spatial-Temporal Graph Convolutional Network(ASTGCN)[46], is designed to learn both spatial and temporal dependencies in graph data through a CNN-based approach. The model includes spatial and temporal attention functions that allow it to capture latent dynamic and temporal dependencies, respectively. One common drawback of learning latent spatial dependencies is that it can be computationally expensive, especially when dealing with large graphs. To address this issue, ASTGCN utilizes a localized spatial attention function that only considers a subset of neighboring nodes within a local region of each target node. This reduces the computational complexity from $O\left(n^2\right)$ to $O\left(kn\right)$ where k is a hyperparameter representing the local region's size.

2.1.2 Graph Transformers

Transformer-based methods for graphs have been proposed to improve the expressive power of graph encoders. These methods incorporate the graph structure and node features into the Transformer architecture to enable better modeling of the complex dependencies and interactions between nodes in the graph.

One such method is the Graph Transformer, which extends the self-attention mechanism used in the Transformer to work with graphs. In the Graph Transformer, each node is

represented as a feature vector, and the self-attention mechanism is used to aggregate information from neighboring nodes to update the node features. This enables the model to capture the graph's local structure and learn representations sensitive to the connectivity patterns in the graph.

Graph Transformer is a graph neural network that utilizes the transformer architecture, originally designed for natural language processing tasks, to process graph-structured data. Compared to traditional graph neural networks (GNNs), Graph Transformer has several potential advantages, such as processing efficiency, encoding long-range dependencies, handling directed and weighted graphs, and avoiding over-smoothing.

Transformer Architecture

Before introducing the detailed methods of Graph Transformer, I will briefly introduce the vanilla Transformer architecture. Transformer [154] is a novel encoder-decoder architecture for neural machine translation, which utilizes a self-attention mechanism to attend to all positions in an input sequence. It is primarily used for natural language processing (NLP) tasks such as machine translation, language modeling, and text generation. The Transformer architecture consists of encoder and decoder layers, each including multiple attention mechanisms. The encoder layers are responsible for encoding the input sequence into a set of hidden representations, while the decoder layers generate the output sequence based on these representations. At the heart of the Transformer architecture is the attention mechanism, which allows the model to selectively focus on different parts of the input sequence when computing the output. The Transformer architecture consists of a series of repeating blocks containing two main components: a self-attention mechanism and a feedforward neural network. The self-attention mechanism allows the model to attend to all positions in the input sequence rather than just a fixed window of positions. This improves the model's ability to capture long-range dependencies in the input. Let $\mathbf{X} \in \mathbb{R}^{n\times d}$ as the input features of the Transformer model, where n is the number of samples, d is the dimension, then one block layer of the Transformer can be formulated as $f_\theta : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ with $f_\theta(\mathbf{X}) =: \mathbf{Z}$ which can be defined by:

$$\mathbf{A} = \frac{1}{\sqrt{d}}\mathbf{XQ}(\mathbf{XK})^\top \tag{2.26}$$

$$\widetilde{\mathbf{X}} = \mathrm{SoftMax}(\mathbf{A})(\mathbf{XV}) \tag{2.27}$$

$$\mathbf{M} = \mathrm{LayerNorm}_1(\widetilde{\mathbf{X}}\mathbf{O} + \mathbf{X}) \tag{2.28}$$

$$\mathbf{F} = \sigma\left(\mathbf{M}\mathbf{W}_1 + \mathbf{b}_1\right)\mathbf{W}_2 + \mathbf{b}_2, \tag{2.29}$$

$$\mathbf{Z} = \mathrm{LayerNorm}_2(\mathbf{M} + \mathbf{F}) \tag{2.30}$$

where Equation 2.26, Equation 2.27, and Equation 2.28 are the equation of computing the attention values; while Equation 2.29 and Equation 2.30 are the feed-forward network (FFN) layers. Here, Softmax $(\cdot)$ is the row-wise softmax function, LayerNorm $(\cdot)$ is the layer normalization function [3], and σ is the activation function. $\mathbf{Q},\mathbf{K},\mathbf{V},\mathbf{O} \in \mathbb{R}^{d\times d}, \mathbf{W}_1 \in \mathbb{R}^{d\times d_f}, \mathbf{b}_1 \in \mathbb{R}^{d_f}, \mathbf{W}_2 \in \mathbb{R}^{d_f\times d}, \mathbf{b}_2 \in \mathbb{R}^{d}$ refer to trainable parameters. Multi-head self-attention is an extension of the self-attention mechanism used in the Transformer architecture. In multi-head self-attention, the model computes multiple sets of attention weights or "heads", each independently applied to the input sequence. Specifically, $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ are divided into H heads with $\mathbf{Q}^{(h)}, \mathbf{K}^{(h)}, \mathbf{V}^{(h)} \in \mathbb{R}^{d\times d_h}$ with $d = \sum_{h=1}^{H} d_h$, and the softmax attention matrices of each head $\widetilde{\mathbf{X}}^{(h)} \in \mathbb{R}^{n\times d_h}$ can be concatenated to obtain the final softmax attention matrix $\widetilde{\mathbf{X}}$. Therefore, in the case of multi-head self-attention, Equation **??** and Equation **??** can be rewrite as:

$$\mathbf{A}^{(h)} = \frac{1}{\sqrt{d}}\mathbf{X}\mathbf{Q}^{(h)}\left(\mathbf{X}\mathbf{K}^{(h)}\right)^{\top} \tag{2.31}$$

$$\widetilde{\mathbf{X}} = \|_{h=1}^{H}\left(\text{SoftMax}\left(\mathbf{A}^{(h)}\right)\mathbf{X}\mathbf{V}^{(h)}\right). \tag{2.32}$$

In multi-head self-attention, the self-attention mechanism is applied multiple times in parallel, each time with different weights. This allows the model to attend to different parts of the input sequence with different sets of parameters, effectively enabling the model to capture multiple levels of abstraction and learn richer representations.

The vanilla Transformer regards the input data as a fully connected graph and calculates the attention weights for each input sequence element. The graph Transformer methods incorporate the graph structure and node features into the Transformer architecture. There are roughly three ways to combine the graph information with the Transformer: 1) GNNs as the graph information learner that directly combines the GNNs with the Transformer. 2) Designing a new positional encoding for graphs based on the structure information. 3) Improving the attention matrices by the structure information of the graph. Then, I talk about these three kinds of methods in detail.

GNNs as the graph information learner

There are plenty of works in this category. For example, GraphTrans [169] is a graph neural network (GNN) architecture that combines a standard GNN layer with a Transformer subnetwork on top. The GNN layer propagates information through the graph, while the Transformer subnetwork is used to refine the node representations. Specifically, in GraphTrans, the GNN layer is used to compute initial node representations by aggregating information from neighboring nodes. The resulting node features are then passed through the Transformer subnetwork, which consists of multiple layers of self-attention and feedforward neural networks. The self-attention mechanism in the Transformer enables the model to capture long-range dependencies and interactions between nodes, while the feedforward neural networks enable the model to learn more complex functions.

Grover [136] Grover is a graph neural network (GNN) architecture that uses a single GTransformer module to encode the graph structure and node features. The GTransformer module in Grover is similar to the Transformer architecture used in natural language processing tasks. It consists of multiple layers of self-attention and feedforward neural networks, which encode the graph structure and node features into a set of node embeddings. Specifically, in Grover, each node is represented as a feature vector, and the self-attention mechanism in the GTransformer module is used to compute a weighted sum of the feature vectors of neighboring nodes. The resulting aggregated feature vector is then passed through a feedforward neural network to obtain a new node feature vector. This process is repeated multiple times to enable the model to capture complex interactions between nodes in the graph.

GraphiT [108] GraphiT (Graph Isomorphism Transformer) is a graph neural network (GNN) architecture that uses a single Graph Isomorphism Network (GIN) layer with a modified self-attention mechanism to encode the graph structure and node features. The GIN layer in GraphiT is a graph convolutional neural network (GCN) layer that applies a series of graph convolutional kernel functions to aggregate information from neighboring nodes. The resulting node representations are then passed through a feedforward neural network to obtain a refined node feature vector. In addition to the GIN layer, GraphiT also includes a modified self-attention mechanism that operates on the node embeddings to enable the model to capture global information from the graph. Specifically, the self-attention mechanism in GraphiT uses a modified attention score that considers the similarity between the node embeddings and the similarity between the corresponding graph structures. The output of the GIN layer and the modified self-attention mechanism are combined to obtain a set of node embeddings that can be used for various downstream tasks.

Mesh Graphormer [87] is designed for 3D human pose estimation and is specifically tailored for processing 3D meshes, and it introduces a new type of block called the Mesh Residual Block (MRB) instead of the Graph Residual Block (GRB) mentioned in your statement. The Mesh Graphormer architecture consists of a series of MRBs stacked on each other with a single MHSA layer between each pair of MRBs. The MRB is designed to capture both local and global interactions among 3D mesh vertices and body joints, and it consists of a mesh convolutional neural network to model local interactions and a graph convolutional neural network to model global interactions. The graph convolution in each Transformer block is defined as

$$\mathbf{M}' = \text{GraphConv}\left(\mathbf{A}^G, \mathbf{M}; \mathbf{W}^G\right) = \sigma\left(\mathbf{A}^G \mathbf{X} \mathbf{W}^G\right). \tag{2.33}$$

where $\mathbf{A}^G \in \mathbb{R}^{n \times n}$ is the adjacency matrix and $\mathbf{W}_G$ is the trainable parameters. $\sigma(\cdot)$ is the non-linear activation function.

Graph-BERT[195] utilizes a stacked self-attention mechanism, similar to the Transformer architecture, but with the addition of a graph residual term in each attention layer. The graph residual term enables the model to capture higher-order or longer-range depen-

dencies in graph data by maintaining the previous attention layer's output and combining it with the current layer's output, thereby preserving the original node embeddings and avoiding the over-smoothing effect. The update function is defined as

$$\mathbf{M}' = \mathbf{M} + \mathrm{G} - \mathrm{Res}\left(\mathbf{X}, \mathbf{X}_r, \mathbf{A}^G\right), \tag{2.34}$$

where $\mathrm{G} - \mathrm{Res}\left(\mathbf{X}, \mathbf{X}_r, \mathbf{A}^G\right)$ denote the graph residual term and $\mathbf{X}_r$ denote the raw features.

Designing a new positional encoding for graphs

Using graph neural networks (GNNs) combined with Transformers has proven effective in modeling graph-structured data. However, the best architecture for combining these two approaches can be difficult to determine and often requires extensive hyperparameter tuning. To address this issue, researchers have proposed using a graph-encoding strategy that does not require modifications to the Transformer architecture. One approach involves compressing the graph structure into positional encoding (PE) vectors, similar to how positional encoding is used for sequential data like sentences. PE vectors can be created by assigning each node in the graph a unique position based on its connectivity to other nodes. This positional information is then encoded as a vector and added to the input features of each node. This allows the Transformer to incorporate information about the relative position of nodes within the graph when making predictions:

$$\widetilde{\mathbf{X}} = \mathbf{X} + f_{\mathrm{map}}\left(\mathbf{P}\right) \tag{2.35}$$

where $\mathbf{X} \in \mathrm{R}^{n \times d}$ is the input embeddings, $\mathbf{P} \in \mathrm{R}^{n \times d_p}$ is the graph embeddings, and $f_{\mathrm{map}} : \mathrm{R}^{d_p} \to \mathrm{R}^{d}$ is a transformation neural network. The graph PE $\mathbf{P}$ is usually learned from the adjacency matrix $\mathbf{A}^G \in \mathrm{R}^{n \times n}$. Dwivedi et al. [24] leverage Laplacian eigenvectors as $\mathbf{P}$ in Graph Transformer. They first compute the Laplacian eigenvectors:

$$\mathbf{U}^T \Lambda \mathbf{U} = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A}^G \mathbf{D}^{-1/2} \tag{2.36}$$

where D is the degree matrix, and Λ, U refers to the eigenvalues and eigenvectors. The eigenvectors corresponding to the k smallest non-trivial eigenvalues are used as the positional encoding, with $\mathbf{P} \in \mathbb{R}^{n \times k}$. When computing eigenvectors from the Laplacian matrix to generate graph PE, it is common to encounter eigenvectors with arbitrary signs, which can result in numerical instability during training. One common approach to address this issue is randomly flipping the eigenvectors' sign during training. Hussain et al. [61] proposed a graph embedding method called Edge Positional Encodings (EPE), which uses the singular value decomposition (SVD) of the adjacency matrix to generate positional encoding $\mathbf{P}$ for the edges in a graph. Specifically, the EPE method first computes the SVD of the adjacency matrix to obtain the left singular vectors and values. The left singular vectors are then used to generate embeddings for the edges in the graph, with each edge being represented by the concatenation of its two endpoint node embeddings. The singular

values are used to scale the edge embeddings, which helps capture each edge's importance in the graph.

$$\begin{aligned}&\mathbf{A}^G \overset{\text{SVD}}{\approx} \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = (\mathbf{U}\sqrt{\mathbf{\Sigma}}) \cdot (\mathbf{V}\sqrt{\mathbf{\Sigma}})^T = \hat{\mathbf{U}}\hat{\mathbf{V}}^T, \\ &\mathbf{P} = \hat{\mathbf{U}} \| \hat{\mathbf{V}},\end{aligned} \tag{2.37}$$

where $\mathbf{U}, \mathbf{V} \in \mathrm{R}^{n \times r}$ contain the r left and right singular vectors corresponding to the top r singular values in the diagonal matrix $\Sigma \in \mathbb{R}^{r \times r}$, $\|$ refers to concatenation operator.

Graph-BERT [195] is a graph neural network model that combines the Transformer architecture with graph attention mechanisms to perform node classification and link prediction tasks on graph-structured data. Graph-BERT introduces three types of positional encoding (PE) to incorporate positional information into the model: 1)Absolute positional encoding: This type of PE is similar to the positional encoding used in the original Transformer model for sequential data. It assigns a unique fixed vector to each position in the graph, which is added to each node's input features to encode its position. 2)Relative positional encoding: This type of PE captures the relative position of each pair of nodes in the graph. It is computed by taking the difference between the absolute positional encodings of the two nodes and then applying a linear transformation to obtain the final embedding vector. 3)Distance positional encoding: This type of PE encodes the distance between each pair of nodes in the graph. It is computed by taking the shortest path distance between the two nodes and applying a linear transformation to obtain the final embedding vector.

Improving the attention matrices by the structure information of the graph

While using node positional encoding is a common practice to incorporate graph information into Transformer architectures, this approach suffers from information loss due to the compression of the graph structure into fixed-sized vectors. In recent years, several works have explored alternative approaches to improving attention matrix computation based on graph information:

$$\mathbf{A} = f_{\mathrm{G_{att}}}\left(\mathbf{X}, \mathbf{A}^G, \mathbf{E}; \mathbf{Q}, \mathbf{K}, \mathbf{W}_1\right) \tag{2.38}$$

$$\mathbf{M} = f_{\mathbf{M}}\left(\mathbf{X}, \mathbf{A}, \mathbf{A}^G, \mathbf{E}; \mathbf{V}, \mathbf{W}_2\right), \tag{2.39}$$

where $\mathbf{X}$ refers to the input attributes, $\mathbf{A}^G$ refers to the adjacent matrix of the graph, $\mathbf{E} \in \mathbb{R}^{n \times n \times d_e}$ refers to the edge features if available, $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ refers to the attention parameters, $\mathbf{W}_1, \mathbf{W}_2$ refers to extra graph encoding parameters.

A series of models that adapt the self-attention mechanism to GNN-like architectures is based on restricting each node to only attend to its local node neighbors in the graph. This can be achieved by incorporating an attention masking mechanism, which computes attention weights for each node only concerning its neighboring nodes:

$$\mathbf{A} = \left(\frac{1}{\sqrt{d}}\mathbf{X}\mathbf{Q}(\mathbf{X}\mathbf{K})^\top\right) \odot \mathbf{A}^G, \tag{2.40}$$

where $\mathbf{A}^G \in \mathbb{R}^{n\times n}$ refers to the adjacent matrix. In [24], $\mathbf{A}^G_{ij} = 1$ if there exist an edge between i-th node and j-th node. If there exist the edge features in the graph, the Equation 2.40 can be written as:

$$\mathbf{A} = \left(\frac{1}{\sqrt{d}}\mathbf{XQ}(\mathbf{XK})^\top\right) \odot \mathbf{A}^G \odot \mathbf{E}'\mathbf{W}_E, \tag{2.41}$$

where $\mathbf{W}_E \in \mathbb{R}^{d_e\times d}$ refers to the parameter matrix, $\mathbf{E}'$ is the edge-embedding matrix that outputted from the previous layer which is learned from $\mathbf{A}$.

Yao et al. [180] introduces a graph neural network architecture for Abstract Meaning Representation (AMR) parsing, which involves extracting semantic representations from natural language texts. The architecture utilizes the extended Levi graph, a heterogeneous graph containing different types of edges, to capture the different types of relations between nodes in AMR. To enable the processing of this heterogeneous graph, the authors group all edge types into a single one to obtain a homogeneous subgraph referred to as a connected subgraph, which contains the complete connected information from the original graph. Similarly, Min et al. [114] propose a graph neural network architecture for click-through rate (CTR) prediction tasks. To capture neighborhood relations among nodes, they design four types of interaction graphs: the induced subgraph, the similarity graph, the cross-neighborhood graph, and the complete graph. The induced subgraph represents direct interactions between nodes, while the similarity graph captures similarities between nodes based on their features. The cross-neighborhood graph captures interactions between neighborhoods of nodes, and the complete graph captures all possible connections between nodes in the graph.

GraphiT [108] is a recent model that extends the adjacency matrix to a kernel matrix, allowing the model to capture more complex dependencies between nodes in a graph. The basic idea behind GraphiT is to compute a kernel matrix for the graph, which represents the similarity between pairs of nodes in the graph. This kernel matrix is then used to define a weighted adjacency matrix, which encodes the structural information of the graph. GraphiT uses a random feature map to compute the kernel matrix, which maps each node in the graph to a high-dimensional space. This feature map is defined by a random projection matrix learned during training. The update function is as follows:

$$\mathbf{A} = \left(\frac{1}{\sqrt{d}}\mathbf{XQ}(\mathbf{XQ})^\top\right) \odot \mathbf{K}_r \tag{2.42}$$

$$\widetilde{\mathbf{X}} = \text{SoftMax}(\mathbf{A})(\mathbf{XV}) \tag{2.43}$$

$$\mathbf{M} = \text{LayerNorm}\left(\mathbf{D}^{-\frac{1}{2}}\widetilde{\mathbf{X}} + \mathbf{X}\right), \tag{2.44}$$

where $\mathbf{D} \in \mathbb{R}^{n\times n}$ refers to the diagonal matrix of node degrees, $\mathbf{K}_r \in \mathbb{R}^{n\times n}$ refers to the kernel matrix on the graph, i.e., diffusion kernel or p-step random walk kernel.

Graphormer [182] aims to incorporate soft graph bias into attention scores. One of the key features of Graphormer is its Spatial Encoding mechanism. This mechanism encodes the spatial location of each node in the graph as a multi-dimensional vector, which is then added to the node embedding. This enables the model to capture the spatial relationships between nodes in the graph, which is especially important in applications such as molecular modeling. Concretely, they consider a distance function $\phi\left(v_i, v_j\right)$ used to compute the spatial distance between nodes v_i and v_j in the graph. They leverage $\phi\left(v_i, v_j\right)$ as the shortest path distance (SPD) between v_i and v_j. If there is no path between them, the output of ϕ is set to -1. The update function is as follows:

$$\mathbf{A} = \left(\frac{1}{\sqrt{d}}\mathbf{XQ}(\mathbf{XK})^\top\right) + \mathbf{B}^s. \tag{2.45}$$

$\mathbf{B}^s$ refers to the bias matrix, where $\mathbf{B}^s_{ij} = b_{\phi(v_i,v_k)}$ is the learnable scalar indexed by $\phi\left(v_i, v_k\right)$.

In addition to the Spatial Encoding mechanism, Graphormer also proposed an edge feature bias term. Specifically, for each ordered node pair $\left(v_i, v_j\right)$, they search (one of) the shortest path $\mathrm{SP}_{ij} = (e_1, e_2, \ldots, e_N)$ from v_i to v_j, and then compute an average of dot-products of the edge features and a learnable embedding along the path. The overall attention score can be written as follows:

$$\mathbf{A} = \left(\frac{1}{\sqrt{d}}\mathbf{XQ}(\mathbf{XK})^\top\right) + \mathbf{B}^s + \mathbf{B}^c \tag{2.46}$$

where $\mathbf{B}^c$ refers to the edge feature bias matrix. $\mathbf{B}^c_{ij} = \frac{1}{N}\sum_{n=1}^{N} x_{e_n}\left(w_n^E\right)^\top$, where x_{e_n} refers to the feature of the n-th edge e_n in $\mathrm{SP}_{ij}, w_n^E \in \mathbb{R}^{d_e}$ refers to the n-th weight embedding, and d_e refers to the dimensionality of edge feature.

Gophormer [199] utilizes a novel attention mechanism called Proximity-Enhanced Multi-Head Attention (PE-MHA) to encode and propagate multi-hop graph information effectively. PE-MHA is designed to capture and utilize both local and global context in the graph data by taking advantage of the proximity between nodes. At each model layer, the PE-MHA module computes multi-head attention over the input embeddings with an additional proximity-enhancing term considering the distance between nodes. By incorporating this proximity-enhancing term, the model can effectively encode multi-hop dependencies and capture long-range interactions in the graph. The attention score with the proximity-enhancing term can be defined as:

$$\mathbf{A}_{ij} = \left(\frac{1}{\sqrt{d}}\mathbf{x}_i\mathbf{Q}\left(\mathbf{x}_j\mathbf{K}\right)^\top\right) + \phi^{ij}\mathbf{b}^\top \tag{2.47}$$

where $\mathbf{b} \in \mathbb{R}^M$ is the bias of the structure information, and the proximity encoding is calculated by

$$\phi_{ij} = \mathrm{Concat}\left(\Phi_m\left(v_i, v_j\right) \mid m \in 0, 1, \ldots, M-1\right), \tag{2.48}$$

where $\Phi_m(\cdot)$ is the structural encoding function.

Propagation Learning over Asymmetric Neighbors(PLAN) [68] is a graph neural network architecture designed to model the tree structure of rumor propagation in social media. To capture the asymmetric nature of the tree structure, PLAN introduces a structure-aware self-attention mechanism that considers the level and position of neighboring nodes in the tree. The attention score can be calculated by

$$\begin{aligned} \mathbf{A}_{ij} &= \frac{1}{\sqrt{d}}\left(\mathbf{x}_i\mathbf{Q}\left(\mathbf{x}_j\mathbf{K}\right)^{\top}\right) + a_{ij}^{K} \\ \mathbf{M}_i &= \sum_{j=1}^{n} \mathrm{SoftMax}\left(\mathbf{A}_{ij}\right)\left(\mathbf{x}_j\mathbf{V} + a_{ij}^{V}\right) \end{aligned} \tag{2.49}$$

Wherea_{ij}^{V} and a_{ij}^{K} denote one of the structural relationships, i.e., parent, child, before, after, and self.

Graph Transformer is a special graph neural network that utilizes the transformer architecture. Compared to traditional graph neural networks (GNNs), Graph Transformer has several potential advantages:

- **Processing Efficiencies :** Traditional GNNs like GCN and GAT rely on a message-passing algorithm to update node representations, which can be computationally expensive for large graphs. In contrast, the self-attention mechanism used by Graph Transformer allows for more parallel computing, making it more efficient for processing large-scale graphs.

- **Encoding Long-Range Dependencies:** The self-attention mechanism in Graph Transformer allows it to capture long-range dependencies in the input, which GNNs with a fixed neighborhood size may miss. This can be especially useful in applications with important long-range dependencies, such as social network analysis.

- **Handling Directed and Weighted Graphs:** Traditional GNNs are designed for undirected and unweighted graphs and may not be easily adapted to handle directed or weighted graphs. Graph Transformer, on the other hand, can handle directed and weighted graphs by using a directed attention mechanism and weighting the edges.

- **Avoiding Over-Smoothing:** One common issue with traditional GNNs is that they may over-smooth the node representations after multiple message passing steps, resulting in information loss. Graph Transformer addresses this issue using a position-wise feedforward neural network to prevent over-smoothing and preserve more information. Overall, Graph Transformer has the potential to be a powerful tool for processing a wide range of graph-structured data, especially for large-scale graphs with long-range dependencies and complex node relationships.

Plenty of Graph Transformer works have been proposed to incorporate the graph information into the Transformer, and many graph Transformer methods have achieved state-of-the-art. To the best of my knowledge, existing graph transformer methods only consider the local graph information but neglect the global graph information, degrading

the graph transformer's performance. i.e., The first category method uses GNN, which is based on the local structural information; the second category uses the improved positional encoding, which is learned based on the local structural information. The third category also improves the attention matrices by the local structural information. However, the global structure information is important for some graph tasks, such as graph classification or clustering. To solve this problem, a hierarchical structure graph transformer called HighFormer is proposed that leverages both the local and global structure information. It will be described in Chapter 3.

2.2 Self-supervised Deep Graph Representation Learning

Self-supervised representation learning (SSL), which was regarded as the key to human-level intelligence, has achieved great success in computer vision (CV) and natural language processing (NLP). Following the great success of SSL in CV and NLP, self-supervised graph representation learning (SGL) has recently been increasingly popular.

The early SGL was started with unsupervised graph embedding [131, 44]. This kind of method learns node representations by random walk, which converts a graph into a random node sequence and then maximizes the agreement of contextual nodes in the truncated random paths (Word2Vec). Then, a classical unsupervised learning method graph autoencoder (GAE)[71] was proposed, also a kind of SGL method that learns the representation by reconstructing the graph structure. After that, a series of similar methods were proposed. Particularly, graph autoencoder (GAE) and variational graph autoencoder (VGAE) [71] first learn the graph embeddings by a two-layer GCN encoder and then reconstruct the adjacency matrix by the decoder method. Marginalized graph autoencoder (MGAE) [162] first learns the graph embeddings via a three-layer GCN encoder and then applies the marginalized denoising autoencoder method to reconstruct the graph. Adversarially regularized graph autoencoder (ARGE) and adversarially regularized variational graph autoencoder (ARVGE) [126] are based on GAE and VGAE but use generative adversarial networks to enforce that graph embeddings match a prior distribution. Random walk and GAE-based methods are known to overemphasize proximity information at the cost of structural information, and the performance of the random-walk-based method is highly dependent on how hyperparameters are set [131].

Contrastive learning (CL) has recently received considerable attention due to its impressive performance. It is an unsupervised approach for capturing the similarity between different samples. The goal is to maximize the similarity of positive pairs and minimize that between negative pairs [14]. Positive pairs generally augment the same instance, while those from different instances are regarded as negative pairs. CL has achieved great success in computer vision [14] and natural language processing [192], and some graph CL methods have been proposed recently. Like the CV and the NLP domain, Graph CL was formed from three key components, i.e., graph augmentations, contrastive pretext tasks, and contrastive objective functions.

Graph augmentation refers to enhancing or enriching a given graph dataset by introducing additional nodes, edges, or attributes or applying transformations to the existing graph structure. Graph augmentation aims to improve the quality, diversity, or representativeness of the graph data, thereby enhancing the performance of downstream tasks such as graph classification, node classification, link prediction, and graph generation. Here are some common techniques used in graph augmentation: 1). Node Addition: New nodes are added to the graph dataset, often by sampling from a predefined distribution or by generating synthetic nodes using generative models. Node addition increases the size and complexity of the graph, allowing the model to learn from a more diverse set of examples. 2). Edge Addition: Additional edges are inserted between existing nodes in the graph, either randomly or based on certain criteria such as node similarity or spatial proximity. Edge addition enriches the graph connectivity and captures additional relationships between nodes. 3). Attribute Augmentation: Additional attributes or features are added to the nodes or edges in the graph, providing supplementary information about the entities and their relationships. Attribute augmentation can include textual features, numerical features, categorical features, or embeddings derived from external sources. 4). Graph Transformation: The existing graph structure is transformed or modified using graph coarsening, sparsification, or graph perturbation. These transformations alter the topology or properties of the graph while preserving its essential characteristics. 5). Graph Sampling: Subgraphs are randomly sampled from the original graph dataset or using sampling strategies designed to capture specific structural patterns or properties of interest. Graph sampling allows for the creation of smaller, more manageable datasets for training or analysis. 6). Generative Models: Generative models such as graph autoencoders, variational graph autoencoders, or graph generative adversarial networks (GANs) are used to generate synthetic graphs that mimic the characteristics of the original graph dataset. Generative models learn to capture the underlying distribution of the graph data and generate new instances accordingly. Graph augmentation techniques are commonly employed in machine learning and data mining tasks involving graph-structured data to address challenges such as data scarcity, imbalance, or heterogeneity. By augmenting the graph dataset with additional examples or variations, researchers can improve graph-based models' robustness, generalization, and performance across various applications.

Pretext tasks, also known as pretext learning or self-supervised learning, are auxiliary tasks designed to provide supervised signals for training deep learning models in an unsupervised or semi-supervised manner. Unlike traditional supervised learning tasks, where labeled data is provided for a specific target task (e.g., classification or regression), pretext tasks involve constructing artificial supervisory signals from the input data. These pretext tasks are typically chosen such that solving them requires capturing helpful information or learning meaningful representations from the data. The term "pretext" implies that the task serves as a pretext or surrogate for the target task of interest. The model is trained to solve the pretext task first, and the representations learned during pretext task training are then transferred or fine-tuned for downstream tasks, which may include classification, regression, clustering, or other tasks. Pretext tasks are particularly useful when labeled

data for the target task is scarce, expensive, or unavailable. By leveraging the inherent structure or properties of the data to construct pretext tasks, researchers can effectively leverage large amounts of unlabeled data for pretraining deep learning models. Among them, contrastive pretext tasks are commonly used. The model is trained to discriminate between pairs of augmented versions of the same input (positive pairs) and pairs of different inputs (negative pairs). By maximizing agreement between positive pairs and minimizing agreement between negative pairs, the model learns to extract useful features or embeddings from the data.

The success of CL in CV heavily relies on well-designed image augmentations. It is the same in Graph CL. However, because of the non-Euclidean nature of graph data, the augmentation of CV can not be applied to graph data. For a graph, there are two kinds of information: node attribute and topological structure. Therefore, there are two types of graph augmentations: attributive-based augmentation and topological-based augmentation. There are two types of attributive-based augmentation. The first one is feature masking (FM)[59, 205, 185, 65, 64], which randomly masks a part of node features with zeros. The second one is feature shuffle (FS) [156, 123], randomly placing the nodes in different positions from the original graph to act as a negative example.

The contrastive pretext tasks tend to maximize the mutual information (MI) between the augmented counterpart of the same object (positive pairs), i.e., node, subgraph, and graph, and at the same time, minimize the MI between the augmented instances of different objects (negative samples). MI is a measure of the amount of information that one random variable contains about another random variable. It quantifies the degree of dependence or association between the two variables. MI is widely used in various fields, including information theory, statistics, machine learning, and data science. It is a fundamental measure for quantifying relationships between variables, feature selection, clustering, and dimensionality reduction. In machine learning, mutual information is often used to select the most informative features for predictive modeling tasks. There are two kinds of contrastive schemes: node-global (or patch-global) contrast and node-node contrast. The node-global contrastive scheme tends to preserve the MI between node representations with the global graph summary. For example, Deep Graph InfoMax (DGI) [156] maximizes the MI between node-level embeddings and the graph-level representation, encouraging the encoder to learn localized and global semantic information. Unlike the node-global contrastive scheme, the node-node contrastive scheme only contrasts the augmented node representations of the same objects. GraphCL [185] maximizes the MI between two target nodes from different views. Graph augmentation can be viewed from various perspectives, so we call the different augmented graph different views. Graphical Mutual Information (GMI) [129] measures the MI of input and edges with the representation of nodes and edges, respectively. Multi-View Graph Representation Learning (MVGRL) [51] learns node and graph level representations by contrasting the embeddings of first-order neighbors and graph diffusion. Graph diffusion, also known as network diffusion or random walk-based diffusion, refers to spreading information, influence, or signals across a graph structure through iterative propagation. In graph diffusion, a "seed" or initial signal is

typically injected into one or more nodes in the graph, and this signal then spreads to neighboring nodes according to predefined rules or algorithms. The diffusion process is often modeled as a random walk on the graph, where at each step, a random walker moves from its current node to one of its neighbors with a certain probability distribution. The choice of neighbors and probabilities depends on the specific diffusion algorithm used. Graph diffusion has various applications in network analysis, social network analysis, recommendation systems, information retrieval, and machine learning. Graph Contrastive representation learning with Adaptive augmentation (GCA) [206] contrasts each node's embedding with its adaptive augmentation. Unlike the above methods, which can only be used in specific graph types, Graph Contrastive Coding (GCC) [132] leverages contrastive learning with GNNs to learn intrinsic and transferable topological structural representations from multiple networks.

The different pretext tasks usually correspond to the different MI estimation methods. For estimating the MI, there are three commonly used contrastive losses, i.e., InfoNCE [185], BYOL [43] loss, and Jensen-Shannon divergence (JSD) [156] loss. The InfoNCE tends to find a lower-bound MI estimation by pulling together a positive pair while pushing away all the negative pairs, for example, GRACE [205] and GCC [132]. The contrastive model with InfoNCE loss is heavily dependent on constructing the negative examples, as the performance will be improved when the number of negative pairs increases. In addition, some false negative pairs in the original InfoNCE loss function could affect the representation performance. Inspired by BYOL [43], a contrastive method that does not require negative pairs, some SGL methods were proposed by introducing the BYOL loss, for example, BGRL [151]. The node-node contrastive scheme usually uses the InfoNCE and BYOL losses. In contrast, the node-global contrastive scheme prefers to use the JSD loss, which derives from the DGI [156] method. The following research adopted Many works on the JSD loss, such as in [51, 148, 149].

2.2.1 Graph Augmentation

Data augmentations have become an effective method to improve the representation performance of images and text. A lot of transformation methods have been proposed for images and text, such as flipping, rotation, color shifting [74], back translation [141], and positional swaps. However, these data augmentation methods can only be used in Euclidean data. Graphs are non-Euclidean structural data that are formed with nodes and edges, and they are relational data. The nodes are connected by edges, which is an entirety. Some little structural modifications may cause a great change in semantic information. Therefore, it's hard to generate semantic-invariant augmentations for graphs.

Most of the existing graph augmentation methods are stochastic schemes, for example, randomly dropping edges. However, the random methods neglect the semantic information of each node, which may change the underlying semantics of the graph drastically. Graph augmentation techniques can be roughly classified into two categories: the feature level and the structure level.

Feature-level Augmentations

A feature-level graph augmentation concentrates on the transformation of the node features. There are many methods for the feature-level class. One of them aims to add noises to the node features [30] or the embedding of nodes [178], which is termed feature corruption. There are two kinds of noises, i.e., random noises [156] or the noises learned by adversarial learning [30, 178]. Velickovic et al. [156] proposed a transformation method called feature shuffling, which randomly shuffles the rows of the feature matrix or each node item of the graph. Another commonly used method is feature masking [185, 184, 150], which masks a part of the feature by setting it to 0. In real-world applications, the graph is always noisy or sometimes even incomplete. Therefore, Wang et al. designed a feature-rewriting [165] method, which rewrites the noisy or incomplete features by their neighbors. Feature propagation [21], which propagates the feature of a node to other nodes across the structure of the graph, is also a method of graph feature augmentation. Liu et al. [94] leverage graph diffusion [51] method to generate a new augmented graph.

Structure-level Augmentations

The structure-level augmentation concentrates on the transformation of the topological structure or the adjacency matrix of the graph. There are many methods for the structure-level transformation class. Among them, the most commonly used is edge perturbation [123, 184, 185, 206], which modifies a part of the edges in the graph, for example, randomly adding or dropping some edges. The next method is graph rewiring, which is trying to improve the structure of the graph by rewiring the edges. Another effective method is graph diffusion, which exploits the global structural similarity of each node. There are two commonly used graph diffusion algorithms, i.e., the personalized PageRank (PPR) [125] and the heat kernel [73]. Besides, node dropping and node insertion are two opposite augmentation methods. Node dropping just masks some nodes in the graph, and the node insertion adds some virtual nodes [38] as well as some edges in the original graph. The last important structure-level augmentation method is graph sampling. It is a method that aims to find subgraphs that can preserve the desired properties [201, 100].

2.2.2 Graph Contrastive Learning

Self-supervised representation learning (SSL) has achieved great success in computer vision (CV) and natural language processing (NLP) since their promising performance. Recently, following the CV and NLP, self-supervised graph representation learning (SGL) has become increasingly popular, and a lot of research has been conducted.

The early traditional unsupervised graph representation method [131, 44] leverages the contrastive paradigm. This kind of method learns node representations by random walk, which converts a graph into a random node sequence and then forces the contextual nodes to have similar representations. But these methods overly emphasize the similarity of the structural information [205]. After the proposal of graph neural networks (GNN),

graph convolutional encoders have become more powerful than previous methods. Then, a classical unsupervised learning method graph autoencoder (GAE)[71] was proposed that learns the representation by reconstructing the graph structure. After that, a series of similar methods were proposed. Particularly, graph autoencoder (GAE) and variational graph autoencoder (VGAE) [71] first learn the graph embeddings by a two-layer GCN encoder and then reconstruct the adjacency matrix by the decoder method. Marginalized graph autoencoder (MGAE) [162] first learns the graph embeddings via a three-layer GCN encoder and then applies the marginalized denoising autoencoder method to reconstruct the graph. Adversarially regularized graph autoencoder (ARGE) and adversarially regularized variational graph autoencoder (ARVGE) [126] are based on GAE and VGAE but use generative adversarial networks to enforce that graph embeddings match a prior distribution. Random walk and GAE-based methods are known to overemphasize proximity information at the cost of structural information, and the performance of the random-walk-based method is highly dependent on how hyperparameters are set [131].

Contrastive learning (CL) has recently received considerable attention due to its impressive performance. It is an unsupervised approach for capturing the similarity between different samples. The goal is to maximize the similarity of positive pairs and minimize that between negative pairs [14]. Positive pairs generally augment the same instance, while those from different instances are regarded as negative pairs. CL has achieved great success in computer vision [14] and natural language processing [192], and some graph CL methods have been proposed recently. Like the CV and the NLP domain, Graph CL was formed from three key components, i.e., graph augmentations, contrastive pretext tasks, and contrastive objective functions.

The success of CL in CV heavily relies on well-designed image augmentations. It is the same in graph contrastive learning (GCL). GCL is the CL method used in graph data. However, because of the non-Euclidean nature of graph data, the augmentation of CV can not be applied to graph data. For a graph, there are two kinds of information, node attribute, and topological structure, therefore, there are two types of graph augmentations: attributive-based augmentation and topological-based augmentation. There are two types of attributive-based augmentation. The first one is feature masking (FM)[59, 205, 185, 65, 64], which randomly masks a part of node features with zeros. The second one is feature shuffle (FS) [156, 123], which randomly places the nodes in different positions from the original graph to act as a negative example.

The contrastive pretext tasks tend to maximize the mutual information (MI) between the augmented counterpart of the same object (positive pairs), i.e., node, subgraph, and graph, and at the same time, minimize the MI between the augmented instances of different objects (negative samples). There are two kinds of contrastive schemes: node-global (or patch-global) contrast and node-node contrast. The node-global contrastive scheme tends to preserve the MI between node representations with the global graph summary. For example, Deep Graph InfoMax (DGI) [156] maximizes the MI between node-level embeddings and the graph-level representation, which encourages the encoder to learn both localized and global semantic information. Different from the node-global contrastive scheme, the node-

node contrastive scheme only contrasts the augmented node representations of the same objects. GraphCL [185] maximizes the MI between two target nodes from different views. Graphical Mutual Information (GMI) [129] measures the MI of input nodes and edges with the representation of nodes and edges, respectively. Multi-View Graph Representation Learning (MVGRL) [51] learns node and graph level representations by contrasting the embeddings of first-order neighbors and graph diffusion, each of which is regarded as a different view. Graph Contrastive representation learning with Adaptive augmentation (GCA) [206] contrasts each node's embedding with its adaptive augmentation. Different from the above methods, which can only be used in specific graph types, Graph Contrastive Coding (GCC) [132] leverages contrastive learning with GNNs to learn intrinsic and transferable topological structural representations from multiple networks.

The different pretext tasks usually correspond to the different MI estimation methods. For estimating the MI, there are three commonly used contrastive losses, i.e., InfoNCE [185], BYOL [43] loss, and Jensen-Shannon divergence (JSD) [156] loss. The InfoNCE tends to find a lower-bound MI estimation by pulling together a positive pair while pushing away all the negative pairs, for example, GRACE [205] and GCC [132]. The contrastive model with InfoNCE loss is heavily dependent on the construction of the negative examples, as the performance will be improved when the number of negative pairs increases. In addition, there are some false negative pairs in the original InfoNCE loss function that could affect the representation performance. Inspired by BYOL [43], a contrastive method that does not require negative pairs, some SGL methods were proposed by introducing the BYOL loss, for example, BGRL [151]. In general, the node-node contrastive scheme usually uses the InfoNCE and BYOL losses. In contrast, the node-global contrastive scheme prefers to use the JSD loss, which derives from the DGI [156] method. A lot of works were adopted on the JSD loss in the following research, such as in [51, 148, 149].

2.3 Deep Graph Clustering

Graph clustering, which aims to divide a whole graph into several node clusters, has been studied for decades. Early methods utilize various shallow approaches to group graphs. The algorithm of Newman et al. [118] cluster vertices with higher-than-average density based on graph Laplacian eigenmaps. Nikolentzos et al. [120] proposed a method based on the eigenvalue decomposition of the adjacency matrix. Guo et al. [47] devised a co-clustering method that leverages valuable topology relationships between instances to boost clustering performance. Liu et al. [88] introduced the principle of content propagation to integrate the structure and content in a network for community clustering. Li et al. [83] applied a matrix factorization-based method to learn an embedding of node attributes and network topology that can produce a consistent cluster partition. The limitations of these shallow approaches are that they only capture either part of the graph information or shallow relationships between the node features and the topological structure. As a result, these methods cannot effectively exploit the graph structure or the relationships between that structure and the nodes' content.

Deep graph clustering (DGC) has grown explosively in recent years and considerably improved the clustering performance. The general pipeline of DGC consists of two parts: an encoding neural network and a clustering method. Specifically, a self-supervised graph representation method trained the encoding neural network to embed the original graph nodes into the latent representation space. Subsequently, a clustering method was designed to divide the latent node representations into several disjoint clusters.

In the beginning, Tian et al. [152] proposed a simple method, which first leverages the stacked autoencoder to learn the node embeddings of the graph and then performs K-means to obtain the clustering result. Subsequently, DNGR [12] was proposed using a random surfing model to capture the graph structure directly instead of using Deepwalk [131], a sampling-based method. However, the previous methods can only learn the graph structural information but ignore the node features. Motivated by the great success of the graph convolutional encoder (GCN) [72], which can learn both node features and structural information, GAE/VGAE [71] was proposed. Concretely, GAE/VGAE adopts GCN as an encoder and leverages a simple inner product to reconstruct the adjacent matrix as a reconstruction decoder. Subsequently, Motivated by GAE/VGAE, which acts as a self-supervised graph representation (SGL) method, MGAE [162] was proposed that learn node representation with GAE, and then leverage spectral clustering algorithm to group the latent node representations into distinct clusters. Similarly, Zhang et al. [197] proposed a graph convolution method termed AGC, which exploits adaptive graph convolution to capture more global structure information and then adopts spectral clustering.

Subsequently, motivated by the generative adversarial networks (GAN) [39], several GAN-related DGL methods were designed. For example, ARGA/ARVGA [127, 126] was proposed by adopting GAN to enforce the representations of graph nodes aligned with a prior distribution.

Then, Wang et al. [161] proposed a goal-directed deep clustering method termed DAEGC [161], which jointly optimizes graph embedding and graph clustering instead of the previous two-step methods. Concretely, DAEGC adopts an attention-based graph encoder and a self-training clustering module that performs clustering on the representation, and the clustering result also guides the representation learning. Another similar method termed GMMVGAE [60] was proposed by jointly optimizing a variational graph auto-encoder(VGAE) and Gaussian mixture models (GMM). Instead of the K-means clustering method, GMM can effectively discover the inherent complex data distributions. Subsequently, Bo et al. proposed a structural deep clustering network (SDCN) [7], which integrates structural information and attribute information into a deep clustering model.

Many contrastive deep graph clustering (CDGC) methods have recently been proposed. Firstly, Hassani et al. proposed MVGRL [51], which adopts graph diffusion as graph augmentation and contrasts node embeddings with the graph embeddings from another view and vice versa. Many efforts have been made to solve the open problems on CDGC. Among them Zhao et al. [198] indicated that existing methods often introduce false-negative samples as they overlooked the cluster information, therefore, GDCL [198] was proposed to correct the negative samples. Specifically, GDCL optimizes graph representation

learning and clustering jointly so that the optimized representations can promote clustering. Precise clustering results can also improve the representation by introducing suitable negative samples. Besides, Lee et al. [77] argue that the performance of the existing contrastive learning methods is highly dependent on designing the augmentation scheme, which is very difficult. Therefore, AFGRL [77] is proposed to generate an alternative view without data augmentation. Moreover, Liu et al. [94] indicate that existing node encoding methods may suffer from representation collapse [94]. They proposed a Dual Correlation Reduction Network (DCRN) in deep graph clustering to address this issue. Although the CDGC has achieved excellent performance, there are still some issues. Firstly, in graph representation, existing GCL methods only contrast the node views of graphs but neglect the structural information. I argue that learning the graph structure and node features is equally important. Secondly, in the process of node clustering, most of the existing DGC methods leverage traditional clustering methods, i.e., K-means and spectral clustering. Therefore, the traditional clustering methods can not leverage the strong graph representation capability of GNNs. Besides, the traditional clustering methods can not be jointly optimized with graph representation learning.

2.4 Graph Datasets

In this section, I will introduce the commonly used graph datasets, which can be divided into four groups, i.e., citation networks, biochemical graphs, social networks, and others. Table 2.1 summarizes the benchmark datasets. The details of each dataset are as follows:
Citation Networks is a type of graph that represents the relationships between papers, authors, and their citations, co-authorships, and authorships. Although these networks are typically directed, they are often treated as undirected to evaluate model performance in node classification, link prediction, and node clustering tasks. The three popular datasets for paper citation networks are Cora, Citeseer, and Pubmed. Cora contains 2708 machine learning publications grouped into seven classes, while Citeseer contains 3327 scientific papers grouped into six classes. In both Cora and Citeseer, each paper is represented by a one-hot vector indicating the presence or absence of specific words from a dictionary. On the other hand, Pubmed contains 19717 diabetes-related publications, with each paper represented as a term frequency-inverse document frequency (TF-IDF) vector. DBLP is a massive citation dataset that contains millions of papers and authors collected from computer science bibliographies. The raw dataset can be found at https://dblp.uni-trier.de, and a processed version of the paper-citation network is continuously updated at https://aminer.org/citation.

Biochemical Graphs, also known as chemical graphs, represent molecules and compounds using atoms as nodes and chemical bonds as edges. These graphs are often used for evaluating graph classification performance. Several popular datasets contain biochemical graphs, including NCI-1 and NCI-9: These datasets contain 4110 and 4127 chemical compounds, respectively, labeled as to whether they are active in hindering the growth of human cancer

cell lines. MUTAG: This dataset contains 188 nitro compounds labeled as to whether they are aromatic or heteroaromatic. D&D and Protein: These datasets represent proteins as graphs labeled whether they are enzymes or non-enzymes. PTC: This dataset consists of 344 chemical compounds labeled as to whether they are carcinogenic for male and female rats. QM9: This dataset records 13 physical properties of 133885 molecules with up to 9 heavy atoms. Alchemy: This dataset records 12 quantum mechanical properties of 119487 molecules comprising up to 14 heavy atoms. Protein-Protein Interaction (PPI): This dataset contains 24 biological graphs with nodes represented by proteins and edges represented by the interactions between proteins. In PPI, each graph is associated with one human tissue, and each node is labeled with its biological state. These datasets are commonly used for evaluating the performance of graph neural network models in various tasks, such as node classification and link prediction, and have been instrumental in advancing the field of graph representation learning in the context of biochemistry and molecular biology.

Social Networks are formed by user interactions from online services. they are a useful tool for studying online behavior and social phenomena. Some of the most popular social network datasets include BlogCatalog, a social network comprising bloggers and their social relationships. The classes of bloggers represent their interests. Reddit: This dataset is an undirected graph formed by posts collected from the Reddit discussion forum. Two posts are linked if they contain comments by the same user. Each post has a label indicating the community to which it belongs. These datasets are commonly used to evaluate the performance of graph neural network models in various tasks related to social network analysis, such as node classification, link prediction, and community detection. They provide important insights into the structure and dynamics of online social networks and help researchers and practitioners design effective and efficient strategies for engaging and supporting online communities.

Other Graphs Several other datasets are used in graph neural network research. MNIST: This dataset contains 70000 images of size $28 * 28$ labeled with ten digits. An MNIST image can be represented as a graph by constructing an 8-nearest-neighbors graph based on pixel locations. METR-LA: This spatial-temporal graph dataset contains four months of traffic data collected by 207 sensors on the highways of Los Angeles County. The graph's adjacency matrix is computed by the sensor network distance with a Gaussian threshold. NELL: This is a knowledge graph from the Never-Ending Language Learning project. It consists of facts represented by a triplet involving two entities and their relation. These datasets study different aspects of graph representation learning, including spatial-temporal, image, and knowledge graphs. They are useful for evaluating the performance of graph neural network models in various tasks, including node classification, link prediction, and path analysis. Overall, these datasets help advance the development of graph neural networks and understanding graph-structured data.

Table 2.1 Summary of the commonly used graph datasts

Category	Dataset	Graphs	Nodes	Edges	Features	Classes
Citation Networks	Cora	1	2708	5429	1433	7
	Citeseer	1	3327	4732	3703	6
	Pubmed	1	19717	44338	500	3
	DBLP	1	4107340	36624464	-	-
Biochemical Graphs	PPI	24	56944	818716	50	121
	NCI-1	4110	29.87	32.30	37	2
	MUTAG	188	17.93	19.79	7	2
	D&D	1178	284.31	715.65	82	2
	PROTEIN	1113	39.06	72.81	4	2
	PTC	344	25.5	-	19	2
	QM9	133885	-	-	-	-
	Alchemy	119487	-	-	-	-
Social Networks	Reddit	1	232965	11606919	602	41
	BlogCatalog	1	10312	333983	-	39
Other Graphs	MNIST	70000	784	-	1	10
	METR-LA	1	207	1515	2	-
	Nell	1	65755	266144	61278	210

Chapter 3

Hierarchical Structure Graph Transformer

3.1 Introduction

The graph is a kind of structural data that consists of node objects and edges that reflect the relationships of the nodes. Recently, graph neural networks (GNNs) [72] have become the primary strategy for graph representation learning, benefiting from their powerful expressive capabilities. Most GNNs learn the graph representation by a message-passing [38] scheme that aggregates the neighbors of each node. Different kinds of message-passing architectures have been designed, such as GCN [72], GAT[157], and GraphSage[50]. However, there are many limitations in the message-passing paradigm, such as over-smoothing [49] and over-squashing [1] problems. Specifically, the over-smoothing problem will occur when the GNNs model gets deeper; as a result, the node features may become similar everywhere in the graph. Another major problem, over-squashing, will happen when GNNs propagate information between two distant nodes in the graph. When GNNs aggregate messages from a distant path, the long-distance nodes will be squashed into very limited information. As a result, GNNs can not propagate messages from long distant nodes even though sometimes these long-range nodes are critical to the learning tasks. Based on these inherent limitations of the message-passing paradigm, it has been proven that the expressive power of message-passing is bounded by the Weisfeiler-Lehman (WL) graph isomorphism test [105, 116, 174].

Many attempts [200, 137, 176] have made tried to solve the above issues. Nevertheless, it seems that none of them can entirely resolve these problems caused by the message-passing paradigm. On the other hand, attention mechanisms have become the most effective method in sequence-based tasks [5, 37], for example, natural language processing (NLP). The attention mechanisms can learn the most relevant part of the input features and deal with variable sizes of the input. Dealing with a single sequence is commonly called self-attention. Inspired by the previous works on the attention mechanism, the graph attention networks (GAT) [155] algorithm was proposed by introducing an attention-based architecture that leverages a self-attention strategy to calculate the representations

of each node on their neighbors. GAT can specify different weights to the neighbors, which improves the discrimination of the representation learning. Thanks to the attention mechanism, GAT is more resistant to over-squashing than GNNs, which aggregate the neighbors equally. However, GAT calculates limited attention, which only considers their neighbors in the graph but can not deal with long-distance relationships.

Recently, some works have started to leverage Transformers [154] instead of GNNs to solve the graph representation problem. Since the development of Transformers, the field of NLP has been a tremendous success that Transformers have been the most powerful and best-performing neural network for dealing with long-range sequential data [20, 93, 10, 45]. Based on the attention mechanism [5], a word attends to other words in a sentence and then combines the received weighted information to generate the representation. Later, Transformer has also shown its great potential in computer vision (CV) [22, 97], for example, Vision Transformer (ViT) that directly applies transformer to sequences of image patches. Very recently, many graph transformer models have been proposed. Graph Transformers use self-attention to encode the relationships between nodes in a graph, which allows them to capture long-range dependencies and global graph structure. This makes them particularly effective for tasks that require modeling complex relationships between graph nodes, such as graph and node classification. On the other hand, GNNs use message passing to iteratively update node representations based on the representations of their neighbors in the graph. This allows them to capture local graph structure and node features, making them well-suited for node classification and link prediction tasks and so on.

The existing graph transformer methods make many attempts to incorporate the graph information into the original Transformer. There are roughly three ways to combine the graph structural information with the Transformer: 1) GNNs are the graph structural information learner that directly combines the GNNs with the Transformer. For example, GraphTrans [169] consists of a standard GNN layer and a transformer layer that the GNN layer performs as a local structure representation learner. In contrast, the Transformer layer as a global relationship learner computes the pairwise node interactions information. 2) Designing a new positional encoding for graphs based on the structure information. For example, learning a positional embedding to express the structure information and then, similar to the vanilla Transformer, adding the positional embedding to the input graph node features. The graph positional embedding is usually learned from the structural information, i.e., the adjacent matrix. For example, Dwivedi et al. [24] leverage the Laplacian eigenvectors as the positional embedding, which is very similar to the sinusoidal positional embeddings in NLP. 3) Improving the attention matrices by the structure information of the graph. The vanilla Transformer learns the attention matrices by computing the similarity of the embedding of each sample pair. The structural information of the graph data can be used to improve the attention matrices. For example, Graphormer [182] added the edge embedding and spatial embedding into the original attention matrix to form a new improved attention matrix where the spatial embedding is learned from the graph structure. Although plenty of Graph Transformer works have been proposed that attempt to

incorporate the graph information into the Transformer, many graph Transformer methods have achieved state-of-the-art. To the best of my knowledge, existing graph Transformer methods only consider the local graph information but neglect the global graph information, degrading the graph Transformer's performance. i.e., The first category of methods uses GNN, which is based on the local structural information. The second category of methods uses an improved positional embedding, which is learned based on the local structural information, and the third category of methods Improves the attention matrices also by the local structural information. However, the global structure information is quite important for some graph tasks such as graph classification or graph clustering.

To solve this problem, a hierarchical structure graph transformer called HighFormer is proposed that leverages both the local and global structure information. I use GNN to learn the initial graph node representation based on the message passing scheme with the local structure information, and at the same time, a structural attention module is used to learn the global structural similarity. Then, I added the softmax attention matrix and the global structure similarity matrix to form an improved attention matrix. Specifically, as illustrated in Figure 3.1, the original graph was first input the GNN to get a local-structure-based representation. On the other hand, the graph structure was input into a structural attention module, which leverages Personalized PageRank to compute the global structural similarity. Then, the softmax attention matrix and the global structural similarity matrix were added to form an improved attention matrix. Finally, I compute the graph representation using the learned improved attention matrix. Here, I use two softmax operations to ensure the graph structure and node feature relationships are considered in the final attention matrix. Moreover, I have theoretically proved that the commonly used positional embedding (PE) Laplacian eigenvectors only introduce the local position of each node but can't introduce more rich structural information. In the proposed method, I leverage the personal PageRank to mine more global structure information of the graph instead of the PE.

The key contributions of this method are summarized as follows:

- In this method, a novel Graph Transformer method, Hierarchical Structure Graph Transformer shorted by HighFormer, was proposed. The proposed approach introduces both the local structure information and the relatively global structure information that improves the discrimination of the Transformer.

- To introduce the relatively global structural information, I leverage the personalized PageRank to compute the relatively global structural similarity, and then I directly added it to the softmax attention matrix to form an improved attention matrix. Compared to positional embedding, the HighFormer approach contains more structural information.

- I have theoretically proved that the commonly used positional embedding Laplacian eigenvectors only introduce the local position of each node but can't express the relatively global position information.

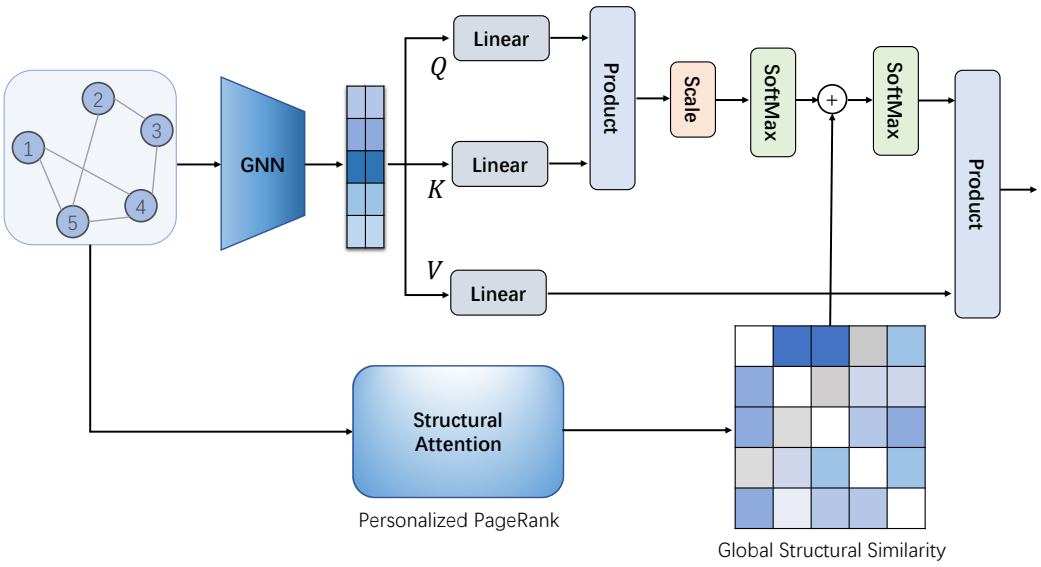

Fig. 3.1 The proposed HighFormer method's general pipeline which consists of two structure information learning modules. The first one is the GNN module, which is used to learn the initial graph node representation based on the local structure information. The second is the structural attention module, which is used to learn the relatively global structural similarity. Then, I added the softmax attention matrix with the relatively global structure similarity matrix to form an improved attention matrix. At last, the graph representation was computed using the learned improved attention matrix.

- Extensive experimental evaluation shows that the proposed HighFormer clearly outperforms the positional-embedding-based graph Transformer methods.

3.2 Hierarchical Structure Graph Transformer

This section will present the Hierarchical Structure Graph Transformer (HighFormer) for graph learning. Different from Euclidean data, such as images and text, the graphs are non-Euclidean data that contain structural information. The structural information can be seen as the relationships between graph nodes. Therefore, the graph transformer methods make many attempts to incorporate the graph information into the original Transformer.

The vanilla Transformer learns the attention matrices by computing the similarity of the embedding of each sample pair. The architecture of the Transformer comprises a series of Transformer layers [154], each of which contains two components: a self-attention module and a position-wise feed-forward network (FFN). Let $H = \left[h_1^\top, \cdots, h_n^\top\right]^\top \in \mathbb{R}^{n \times d}$ is the input of self-attention module where d denote the representation dimension and $h_i \in \mathbb{R}^{1 \times d}$ denote the representation at position i. H is projected by three matrices $W_Q \in \mathbb{R}^{d \times d_K}, W_K \in \mathbb{R}^{d \times d_K}$ and $W_V \in \mathbb{R}^{d \times d_V}$ to the corresponding representations Q, K, V. The self-attention can be defined as:

$$\begin{aligned} Q = HW_Q, \quad K = HW_K, \quad V = HW_V, \\ S = \frac{QK^\top}{\sqrt{d_K}}, \quad \text{Attn}(H) = \text{softmax}(S)V, \end{aligned} \tag{3.1}$$

Here, the matrix S represents the similarity between the queries and the keys. Specifically, I focus on single-head self-attention and assume that the dimensions of the keys and values are equal ($d_K = d_V = d$). The extension to the multi-head attention is standard and straightforward, and the bias terms are omitted for simplicity.

There are about three methods to incorporate the graph structural information into the original Transformer, i.e., the first one is GNNs as the graph structural information learner that directly combines the GNNs with the Transformer. The second method is designing a new graph positional embedding based on the structure information. The third method is improving the attention matrices by the structural information of the graph. However, existing graph Transformer methods only consider the local graph structural information but neglect the global structural graph information degrades the performance of the graph Transformer. i.e., The first method uses GNN which is based on the local structural information, the second method uses the improved positional embedding which is learned based on the local structural information and the third category improves the attention matrices by the local structural information. However, for some graph tasks such as graph classification or graph clustering, the global structure information is quite essential. To solve the above issues, the hierarchical structure graph transformer (HighFormer) is proposed that leverages both the local and global structural information as illustrated in Figure 3.1.

3.2.1 The GNN Module

For the local level, an effective approach is to leverage a GNN model to extract the local structural information. More formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph where $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$, $n = |\mathcal{V}|$ is the number of nodes. The feature vector of node v_i is denoted as x_i, and the graph node feature matrix is denoted as $\mathbf{X}$.

GNNs generally have the objective of acquiring the representation of nodes and graphs. This is achieved through a learning process that involves the iterative updating of a node's representation by combining representations of its first or higher-order neighbors. In this context, the notation $\boldsymbol{h}_i^{(l)}$ is used to refer to the representation of node v_i in the l-th layer, with $\boldsymbol{h}_i^{(0)}$ defined as x_i. The AGGREGATE-COMBINE step characterizes the l-th iteration of the aggregation process, which can be defined as:

$$\begin{aligned} a_i^{(l)} &= \text{AGGREGATE}^{(l)}\left(\left\{\boldsymbol{h}_j^{(l-1)} : j \in \mathcal{N}(v_i)\right\}\right), \\ h_i^{(l)} &= \text{COMBINE}^{(l)}\left(\boldsymbol{h}_i^{(l-1)}, a_i^{(l)}\right) \end{aligned} \tag{3.2}$$

where $\mathcal{N}(v_i)$ denotes a set of node's first or higher-order neighbors v_i. The AGGREGATE function gathers information from neighbors and is commonly employed in various architectures of GNNs using aggregation functions such as MEAN, MAX, and SUM

[72, 50, 155, 174]. On the other hand, the COMBINE function aims to merge information from neighbors into the node representation.

Each node v in the set $\mathcal{V}$ is assumed to have an initial feature vector $\boldsymbol{h}_v^0$ of dimension $\mathbb{R}^{d_0}$. Since HighFormer is a flexible framework that can be combined with different graph neural networks (GNNs), I have minimal assumptions regarding the GNN layers that provide input to the Transformer subnetwork. To combine the AGGREGATE-COMBINE, the general GNN layer can be defined as:

$$\boldsymbol{h}_v^\ell = f_\ell\left(\boldsymbol{h}_v^{\ell-1}, \left\{\boldsymbol{h}_u^{\ell-1} \mid u \in \mathcal{N}(v)\right\}\right), \quad \ell = 1, \ldots, L_{\text{GNN}} \tag{3.3}$$

where L_{GNN} denote the number of GNN layers for a GNN model, $\mathcal{N}(v) \subseteq \mathcal{V}$ denote the neighborhoods of node v, and $f_\ell(\cdot)$ denote the AGGREGATE-COMBINE function which parameterized by the neural network. It should be noted that although many GNN layers support the incorporation of edge features, I will not discuss them in detail here to avoid complicating the notation.

3.2.2 The Structural Attention Module

The GNN module has incorporated the local structural information into the node embeddings. The neighborhoods used in GNN are very limited (usually first-order neighbors). Neighborhood aggregation in GNNs involves passing messages between neighboring nodes and aggregating them to update the node representations. However, as the range of the neighborhood increases, more and more distant nodes are included in the aggregation process, and the information from the original node representation can become diluted or lost. This problem is related to the concept of over-smoothing [81, 176], which occurs when the node representations become too similar after multiple rounds of neighborhood aggregation. This can result in a loss of discriminative power and hinder the ability of the model to distinguish between nodes with different properties or labels.

To address this issue, I proposed a structural attention module based on personalized PageRank (PPR), which incorporates a chance to teleport back to the root node. This feature guarantees that the PageRank score captures the local neighborhood of each root node [125]. By adjusting the teleport probability, I can balance maintaining locality (i.e., staying close to the root node to prevent over-smoothing) and utilizing information from a wider neighborhood. With this propagation scheme, I demonstrate that it is possible to perform an increased number of propagation steps, even infinitely many, without the risk of over-smoothing. PPR uses neighborhood information more widely than GNN, so it can obtain more global-level structural information. Specifically, the adjacency matrix was fed into the structural attention module, which leverages Personalized PageRank (PPR) [125] to compute the global structural similarity, which is formulated as:

$$\mathbf{P} = \gamma\left(\mathbf{I}_n - (1-\gamma)\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}\right)^{-1} \tag{3.4}$$

Where γ is the teleport probability. Then, the softmax attention matrix and the global structural similarity matrix were added to form an improved attention matrix.

Recent works on GNN [117, 183, 147, 25, 80] have explored the issue of positional embeddings (PEs), to learn both structural and positional features. Dwivedi et al. [25] used available graph structures to pre-compute Laplacian eigenvectors [6] and utilized them as node positional information. As Laplacian PEs generalize the PEs used in the original transformers [154] to graphs and can better encode distance-aware information, many existing methods use Laplacian eigenvectors as PEs in Graph Transformer. The Laplacian eigenvectors of all graphs in the dataset are pre-computed, which are defined through the factorization of the graph Laplacian matrix:

$$\Delta = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} = \mathbf{U}^T\Lambda\mathbf{U}, \tag{3.5}$$

where $\mathbf{W}$ denote the $n \times n$ adjacency matrix, $\mathbf{D}$ denote the degree matrix, and $\Lambda, \mathbf{U}$ denote the eigenvalues and eigenvectors respectively. The k smallest non-trivial eigenvectors of a node are utilized as its positional embedding, with λ_i representing the value associated with node i.

I have theoretically proved that the commonly used Laplacian PEs only introduce the local position information of each node but can't introduce more rich structural information. I also proved that the Laplacian PEs have the same effect as the Spectral Clustering. The detailed proof process is as follows:

Graph Laplacian

The degree matrix D is a diagonal matrix defined as:

$$\mathbf{D} = \begin{pmatrix} d_1 & \cdots & \cdots \\ \cdots & d_2 & \cdots \\ \vdots & \vdots & \ddots \\ \cdots & \cdots & d_n \end{pmatrix} \tag{3.6}$$

Assuming the graph G is weighted, the non-negative weight $w_{ij} \geq 0$ is carried by each edge connecting two vertices v_i and v_j. The weighted adjacency matrix, denoted by $W = \left(w_{ij}\right) i, j = 1, \ldots, n$, represents the graph. A weight of 0 for w_{ij} indicates that the vertices v_i and v_j are not connected. $w_{ij} = w_{ji}$ is required in an undirected graph. The degree of a vertex $v_i \in V$ is defined as:

$$d_i = \sum_{j=1}^{n} w_{ij} \tag{3.7}$$

Given a subset of vertices $B \subset V$, its complement $V \backslash B$ is defined as $\bar{B}$. The size of a subset $B \subset V$ can be defined in two ways:

$$\begin{aligned} |B| &:= \text{ the number of vertices in } B \\ \text{vol}(B) &:= \sum_{i \in B} d_i. \end{aligned} \tag{3.8}$$

The first way to measure the size of B is by its number of vertices, and the second way is by the degree of its edges.

The graph Laplacian matrix is defined as:

$$L = D - W \tag{3.9}$$

For a arbitrary vector $f \in \mathbb{R}^n$, the Laplacian matrix have

$$f'Lf = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(f_i - f_j\right)^2. \tag{3.10}$$

The detailed process to proof the Equation 3.10 by the definition of d_i is as follows:

$$\begin{aligned} f'Lf &= f'Df - f'Wf = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij} \\ &= \frac{1}{2} \left(\sum_{i=1}^{n} d_i f_i^2 - 2 \sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} d_j f_j^2 \right) = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(f_i - f_j\right)^2. \end{aligned} \tag{3.11}$$

There are two normalized graph Laplacians, i.e., the symmetric normalized graph Laplacian and the random walk normalized graph Laplacian. They are defined as:

$$\begin{aligned} L_{\text{sym}} &:= D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2} \\ L_{\text{rw}} &:= D^{-1} L = I - D^{-1} W. \end{aligned} \tag{3.12}$$

Graph Partitioning

Clustering intuition involves segregating points into various groups based on their similarities. When data is presented in the form of a similarity graph, the goal is to identify a partition of the graph where the edges between different groups have minimal weight (indicating dissimilarity between points in different clusters). In contrast, the edges within a group have high weight (indicating similarity among points within the same cluster). Then, I will explore how spectral clustering can be used as an approximation to solve such graph partitioning problems. For two disjoint subsets $A, B \subset V$, I can define

$$\text{cut}(A, B) = \sum_{i \in A, j \in B} w_{ij}. \tag{3.13}$$

To construct a partition from a similarity graph with adjacency matrix W, the most straightforward approach is to solve the mincut problem. This involves selecting the partition

$A_1,\ldots,A_k$ that minimizes a certain criterion:

$$\text{cut}\left(A_1,\ldots,A_k\right) := \sum_{i=1}^{k} \text{cut}\left(A_i,\bar{A}_i\right) \tag{3.14}$$

Two of the most prevalent objective functions used to encode this are RatioCut [48] and the normalized cut Ncut [144]. In RatioCut, the size of a graph subset A is defined by its vertex count $|A|$, whereas Ncut uses the edge weights to determine the subset size through the $\text{vol}(A)$ function. The definitions are:

$$\begin{aligned} \text{RatioCut}\left(A_1,\ldots,A_k\right) &= \sum_{i=1}^{k} \frac{\text{cut}\left(A_i,\bar{A}_i\right)}{|A_i|} \\ \text{Ncut}\left(A_1,\ldots,A_k\right) &= \sum_{i=1}^{k} \frac{\text{cut}\left(A_i,\bar{A}_i\right)}{\text{vol}\left(A_i\right)}. \end{aligned} \tag{3.15}$$

RatioCut

The principle for relaxing the RatioCut minimization problem for a general value of k is similar to the one explained above. To achieve this, I start by partitioning the set V into k sets, denoted by $A_1,\ldots,A_k$, and define k indicator vectors, namely $h_i = (h_{1,i},\ldots,h_{n,i})'$, where n is the total number of elements in the set. The indicator vector is defined as:

$$h_{i,j} = \begin{cases} 1/\sqrt{|A_i|} & \text{if } i \in A_j \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

We can set the matrix $H \in \mathbb{R}^{n\times k}$ to include the k indicator vectors as columns. It is worth noting that the columns in H are orthogonal to one another, meaning that $H'H = I$. By performing similar calculations to those presented in the previous, we can see that

$$h_i' L h_i = 2\frac{\text{cut}\left(|A_i|,\left|\bar{A}_i\right|\right)}{|A_i|}. \tag{3.17}$$

Moreover, we have

$$h_i' L h_i = \left(H'LH\right)_{ii}. \tag{3.18}$$

Combining them we have:

$$\text{RatioCut}\left(A_1,\ldots,A_k\right) = \frac{1}{2}\sum_{i=1}^{k} h_i' L h_i = \frac{1}{2}\sum_{i=1}^{k}\left(H'LH\right)_{ii} = \frac{1}{2}\text{Tr}\left(H'LH\right), \tag{3.19}$$

where Tr is the trace of a matrix. Then, the problem of minimizing $\text{RatioCut}\left(A_1,\ldots,A_k\right)$ can be write as:

$$\min_{A_1,\ldots,A_k} \text{Tr}\left(H'LH\right) \text{ subject to } H'H = I. \tag{3.20}$$

The problem can be relaxed by permitting the entries of the matrix H to have any real value. As a result, the relaxed problem would be:

$$\min_{H\in\mathbb{R}^{n\times k}} \operatorname{Tr}\left(H'LH\right) \text{ subject to } H'H = I. \tag{3.21}$$

This becomes the standard form of the trace minimization problem, according to some variation of the Rayleigh-Ritz theorem [104], the solution involves selecting H as the matrix comprising the first k eigenvectors of L arranged as columns.

Ncut

The indicator vectors $h_i = (h_{1,i},\ldots,h_{n,i})'$ for Ncut is defined as:

$$h_{i,j} = \begin{cases} 1/\sqrt{\operatorname{vol}(A_i)} & \text{if } i \in A_j \\ 0 & \text{otherwise.} \end{cases} \tag{3.22}$$

Then, we can write the problem of minimizing Ncut by setting the matrix H as the one containing the k indicator vectors as columns. Observe that $H'H = I$, $h_i'Dh_i = 1$, and $h_i'Lh_i = 2\operatorname{cut}\left(A_i,\bar{A}_i\right)/\operatorname{vol}(A_i)$. So, the problem of minimizing Ncut can be defined as:

$$\min_{A_1,\ldots,A_k} \operatorname{Tr}\left(H'LH\right) \text{ subject to } H'DH = I. \tag{3.23}$$

The problem can be relaxed by relaxing the discreteness condition and substituting $U = D^{1/2}H$ to obtain the relaxed problem.

$$\min_{U\in\mathbb{R}^{n\times k}} \operatorname{Tr}\left(U'D^{-1/2}LD^{-1/2}U\right) \text{ subject to } U'U = I \tag{3.24}$$

The given problem of minimizing the trace can be solved using matrix U, which comprises the initial k eigenvectors of L_{sym} as its columns. By replacing H with $D^{-1/2}U$, it can be observed that the resulting solution H comprises the first k eigenvectors of the matrix L_{rw} or the first k generalized eigenvectors of $Lv = \lambda Dv$. This technique leads to the normalized spectral clustering algorithm proposed by Shi and Malik [144].

From the above proof process, we know that the normalized spectral clustering algorithm has the same process as the calculation of Laplacian PEs. In Transformers, the PEs should indicate the precise position relationships for each pair of nodes in a graph. However, The Laplacian PEs can only indicate the approximate class information of a node, which may degenerate the discriminate capacity of the graph Transformers. Therefore, I use the structural attention module in the proposed method to calculate the structural attention scores instead of the PEs in the traditional graph Transformers.

3.2.3 The Transformer Module

After obtaining the definitive GNN encodings and the relatively global structural similarity $\boldsymbol{h}_v^{L_{\text{GNN}}}$ and $S_{v,u}$ respectively, the next step involves sending them to the Transformer subnetwork within HighFormer. To accomplish this, I begin by performing a linear projection of

the $\boldsymbol{h}_v^{L_{\text{GNN}}}$ onto the Transformer dimension. Additionally, I apply a Layer Normalization to ensure that the embedding is properly normalized:

$$\overline{\boldsymbol{h}}_v^0 = \text{LayerNorm}\left(\boldsymbol{W}^{\text{Proj}}\boldsymbol{h}_v^{L_{\text{GNN}}}\right) \tag{3.25}$$

where $\boldsymbol{W}^{\text{Proj}} \in \mathbb{R}^{d_{\text{TF}} \times d_{L_{\text{GNN}}}}$ denote a learnable weight matrix, and d_{TF} and $d_{L_{\text{GNN}}}$ denote the dimension of Transformer and the dimension of the GNN embeddings, respectively. I input the projected node embeddings $\overline{\boldsymbol{h}}_v^0$ into a standard Transformer layer stack without additive positional embeddings. This is because I assume that the GNN has already incorporated the local structural information into the node embeddings, and I use the PPR to denote the relatively global structural similarity, which is directly added to the softmax attention matrix:

$$\begin{aligned} a_{v,u}^{\ell} &= \left(\boldsymbol{W}_{\ell}^{Q}\overline{\boldsymbol{h}}_v^{\ell-1}\right)^{\top}\left(\boldsymbol{W}_{\ell}^{K}\overline{\boldsymbol{h}}_u^{\ell-1}\right)/\sqrt{d_{\text{TF}}} \\ \alpha_{v,u}^{\ell} &= \underset{w\in\mathcal{V}}{\text{softmax}}\left(a_{v,w}^{\ell}\right) + P_{v,w} \\ \beta_{v,u}^{\ell} &= \underset{w\in\mathcal{V}}{\text{softmax}}\left(\alpha_{v,w}^{\ell}\right) \\ \overline{\boldsymbol{h}}_v^{\prime\ell} &= \sum_{w\in\mathcal{V}} \beta_{v,w}^{\ell}\boldsymbol{W}_{\ell}^{V}\overline{\boldsymbol{h}}_w^{\ell-1} \end{aligned} \tag{3.26}$$

where $\boldsymbol{W}_{\ell}^{Q}, \boldsymbol{W}_{\ell}^{K}, \boldsymbol{W}_{\ell}^{V} \in \mathbb{R}^{d_{\text{TF}}/n_{\text{head}} \times d_{\text{TF}}/n_{\text{head}}}$ denote the learned query, key, and value matrices, respectively, for a single attention head in layer ℓ. $P_{v,w}$ denote the structural similarity between node v and node w which calculated by the PPR. The standard practice involves running n_{head} parallel attention heads and concatenating their resulting per-head encodings $\overline{\boldsymbol{h}}_v^{\prime\ell}$. The concatenated encodings are fed into a fully connected subnetwork within the Transformer architecture. This subnetwork consists of a sequence of operations including Dropout, Layer Norm, FC (fully connected), nonlinearity, more Dropout, more FC, yet more Dropout, and another Layer Norm, with residual connections from $\overline{\boldsymbol{h}}_v^{\ell-1}$ to after the first dropout and from before the first fully-connected sublayer to after the dropout immediately following the second fully-connected sublayer.

A singular embedding vector that characterizes the entire graph is necessary to perform whole-graph classification. In Graph Neural Networks (GNN), the module responsible for reducing the embeddings of each node and/or edge to a singular embedding is referred to as the "readout" module. The most prevalent readout modules include straightforward mean or max pooling or a solitary "virtual node" that maintains connections to every other node within the network.

The proposed method follows the readout method used in GraphTrans [169]. To generate the prediction, I take the transformed per-node embeddings $\bar{h}v^0$ and add a learnable embedding $h_{\langle CLS\rangle}$ to the sequence. The first embedding $\bar{h}_{\langle \text{CLS}\rangle} \in \mathbb{R}^{d_{\text{TF}}}$ from the transformer output is then used as the representation of the entire graph. It is important to note that since I do not use positional encodings, the placement of the special token at the "beginning"

of the sentence does not hold any special computational significance and is chosen by convention. Finally, I generate the prediction by:

$$y = \text{softmax}\left(\boldsymbol{W}^{\text{out}}\, \overline{\boldsymbol{h}}_{<\text{CLS}>}^{L_{\text{TF}}}\right). \tag{3.27}$$

where L_{TF} denote the number of layers in the Transformer architecture.

3.3 Experiments

The proposed Transformer model HighFormer was evaluated on graph classification tasks on four datasets, namely NCI1, NCI109, OGBG-PPA, and OGBG-CODE2. The results showed that the proposed framework consistently outperformed all benchmarks, demonstrating its effectiveness and generalizability. To train the proposed model, the Adam optimizer [70] was used. The Transformer modules utilized in the experiments had an embedding dimension of 128 and a hidden dimension 512 in the feedforward subnetwork. It's worth noting that the Transformer baselines were trained solely with the sequence of node embeddings while ignoring the underlying graph structure.

3.3.1 Datasets

I evaluate the performance of the proposed method on four benchmark datasets for graph classification, including NCI1 [160], NCI109 [160], OGBG-PPA [58], and OGBGCODE2 [58]. Then, I will give an introduction to each dataset:

- **NCI1** is a benchmark graph classification dataset in machine learning and graph analysis. It is a subset of the NCI database of chemical compounds and is commonly used for evaluating the performance of graph classification algorithms. The dataset contains 4,110 graphs, with each graph representing a chemical compound. The compounds are classified into two categories based on their ability to inhibit cancer cell growth. The graphs have varying sizes, with the largest graph containing 111 nodes and 159 edges. The nodes represent atoms in the compounds, while the edges represent chemical bonds between the atoms. The NCI1 dataset is commonly used in research to evaluate the performance of graph-based machine learning models, such as graph convolutional neural networks, in predicting the inhibitory activity of chemical compounds. The dataset poses several challenges for these models, including handling large and complex graphs, dealing with noisy data, and overcoming class imbalance.

- **NCI109** is a benchmark graph classification dataset commonly used in machine learning and cheminformatics research. It is a subset of the NCI database of chemical compounds and is designed to evaluate the ability of machine learning models to predict the anticancer activity of compounds. The dataset contains 4,127 graphs, each representing a chemical compound. The compounds are classified into one

of two categories based on their ability to inhibit the growth of cancer cells. The graphs have varying sizes, with the largest graph containing 164 nodes and 343 edges. The nodes represent atoms in the compounds, while the edges represent chemical bonds between the atoms. The NCI109 dataset presents several challenges for machine learning models, including handling large and complex graphs, dealing with noisy data, and overcoming class imbalance. The dataset has been used as a benchmark for evaluating the performance of graph-based machine learning models, such as graph convolutional neural networks, in predicting the anticancer activity of chemical compounds.

- **OGBG-PPA** (OGB Graph Challenge: Prediction of Property of Antibiotics) dataset is a benchmark graph classification dataset designed to evaluate the performance of machine learning models in predicting the antibacterial spectrum of antibiotics. The dataset consists of 1,715 graphs, each representing an antibiotic molecule. The graphs are labeled with one of 45 different functional categories, indicating the spectrum of activity of the antibiotic against different bacteria strains. The graphs have varying sizes, with the largest graph containing 121 nodes and 238 edges. The nodes represent atoms in the molecules, while the edges represent chemical bonds between the atoms. The OGBG-PPA dataset presents several challenges for machine learning models, including dealing with large and complex graphs, handling noisy and incomplete data, and overcoming class imbalance. The dataset has been used as a benchmark for evaluating the performance of graph-based machine learning models, such as graph convolutional neural networks, in predicting the antibacterial spectrum of antibiotics.

- **OGBGCODE2** is a graph classification dataset consisting of code snippets in the programming language Python. It is an extension of the original OGBG-Code dataset and is designed to be a benchmark for machine-learning models that classify code snippets according to their intended functionality. The dataset contains 1,833 graphs, with each graph representing a code snippet. The graphs are labeled with one of 11 functional categories: sorting, searching, and string manipulation. Each graph contains up to 500 nodes and 1,000 edges, and the nodes correspond to Python code tokens, while the edges represent data flow dependencies between the tokens. The OGBGCODE2 dataset is unique because it presents several challenges for machine learning models, including dealing with large and complex graphs, capturing the semantic meaning of code, and handling class imbalance.

3.3.2 Experimental Setup

I trained HighFormer on two biology datasets, NCI1 and NCI109, for 100 epochs using a batch size of 256. Each experiment was run 20 times with different random seeds, and the average and standard deviation of the test accuracies were calculated. The model used an architecture with 4 transformer layers and a dropout ratio 0.1 for both the GNNs and Trans-

former modules. The GNN module width and depth in the HighFormer model were taken from the simple baseline with a hidden dimension of 128 and 3 GNN layers, as described in [76]. Additionally, I implemented a cosine annealing schedule [98] for learning rate decay. For the OGB Graph Challenge datasets, OGBG-PPA and OGBGCODE2, I evaluate the HighFormer approach and compare it to various GNNs such as GCN [72], GIN [174], and DeeperGCN [79]. Additionally, I examine two recently proposed graph Transformers, including the original Transformer with RWPE [26] and GraphTrans [169], which uses the vanilla Transformer on top of a GNN. I take most of the results for the comparison methods from the original papers. If the original paper does not provide the necessary information, the comparison method results are sourced from Dwivedi et al. [25].

3.3.3 Comparison to State-of-the-Art Methods

Table 3.1 and 3.2 demonstrate the performance of HighFormer compared to other GNNs and Transformers. The HighFormer models exhibit superior performance to the state-of-the-art (SOTA) methods on these datasets, indicating their capacity to amalgamate the advantages of GNNs, structural information, and Transformers. Notably, the HighFormer models exhibit significantly better results on the CODE2 dataset than the SOTA methods despite having fewer parameters and minimal hyperparameter tuning.

3.3.4 Analysis of the Experiment Results

In this section, I will analyze the graph classification results on four different datasets. I evaluate the HighFormer model versus several state-of-the-art methods, including GNNs and Transformers. From the experiment results, I have found that the hierarchical framework achieves SOTA performance on the graph classification tasks, outperforming other graph Transformers and GNNs. I have noticed that integrating the structure using the proposed structure-aware attention leads to a significant enhancement compared to the plain Transformer with RWPE. The plain Transformer utilizes only node attribute similarity without taking into account structural similarity not to mention relatively global structural similarity. Although selecting an appropriate absolute positional encoding and readout technique can enhance performance, their impact is significantly lower than integrating the structure into the approach.

Table 3.1 presents the results for both NCI1 and NCI109, including the simple baselines such as GCN Set2Set, SortPool, and SAGPool, sourced from [76], along with the strong baselines [28] and the FA layer [1]. The HighFormer model, which has the same architecture as the simple baseline, significantly improves the average accuracy by 8.1% for NCI1 and 5.5% for NCI109. The experiment results also indicate that the transformer module's attention mechanism can effectively grasp long-range information that may be challenging for the GNN module to learn.

The complexity of the proposed HighFormer is the same with GraphTrans [169]. I replaced the positional encoding learning in GraphTrans with Structural Attention, which

was calculated by Personalized PageRank. The complexity of Personalized PageRank is nearly linear. So, compared to GraphTrans, no extra complexity is introduced.

Table 3.1 The experiment results on NCI biological datasets: HighFormer outperforms all the listed baselines on both NCI1 and NCI109 test accuracy

Model	GNN Type	GNN layer count	NCI1 ↑	NCI109 ↑
Set2Set [76, 159]	GCN	3	0.686 ± 0.019	0.698 ± 0.012
SortPool [76, 196]	GCN	3	0.738 ± 0.01	0.74 ± 0.012
SAGPool$_h$[76]	GCN	3	0.675 ± 0.011	0.679 ± 0.014
SAGPool$_g$[76]	GCN	3	0.742 ± 0.012	0.741 ± 0.008
Errica et al. [28]	GIN	8	0.8 ± 0.014	-
Alon and Yahav [1]	GIN	8	0.815 ± 0.012	-
Transformer [154]	-	-	0.685 ± 0.026	0.701 ± 0.023
HighFormer	GCN	3	$\mathbf{0.819 \pm 0.015}$	$\mathbf{0.795 \pm 0.013}$

Table 3.2 The experiment results on OGBG-PPA and OGBG-CODE2.

METRIC	OGBG-PPA ↑ ACCURACY	OGBG-CODE2 ↑ F1 SCORE
GCN	0.684 ± 0.008	0.151 ± 0.002
GCN-VIRTUAL NODE	0.686 ± 0.006	0.159 ± 0.002
GIN	0.689 ± 0.010	0.149 ± 0.002
GIN-VIRTUAL NODE	0.704 ± 0.011	0.158 ± 0.002
DEEPERGCN	0.771 ± 0.007	-
TRANSFORMER	0.645 ± 0.003	0.167 ± 0.001
GRAPHTRANS	-	0.183 ± 0.002
HighFormer	$\mathbf{0.773 \pm 0.004}$	$\mathbf{0.194 \pm 0.003}$

3.4 Conclusion

GNNs have been widely used in DGL problems. However, GNNs suffer from over-smoothing and over-squashing problems, which confine their expressive power. To solve these problems, a Hierarchical Structure Graph Transformer called HighFormer is proposed to leverage both local and relatively global structure information. I use GNNs to learn the initial graph node representation based on the local structure information. At the same time, a structural attention module is used to learn the relatively global structural similarity. Then, I added the softmax attention matrix and the relatively global structure similarity matrix to form an improved attention matrix. Finally, I compute the graph representation using the learned improved attention matrix. An experimental evaluation of four commonly used benchmark datasets shows that the proposed HighFormer clearly outperforms the state-of-the-art methods.

Chapter 4

Graph Contrastive Learning with Semantic-invariance Graph Augmentation

4.1 Introduction

In recent years, Deep Graph Representation Learning (DGL) [72, 157, 174, 170, 92] has become increasingly popular since graph data are everywhere in real-world applications such as traffic [171], social networks [124], biology [33] and knowledge graphs [63]. DGL based on Graph Neural Networks (GNN) aims to transform raw graphs into low-dimensional vector embeddings that preserve the graph node features and graph structural information. DGL is critical as the quality of the learned representations will affect the downstream tasks such as graph clustering, graph classification, and link prediction.

Most of the existing DGL models are designed in (semi-)supervised scenarios [57, 72, 157], which require abundant manual labels for training. However, collecting manual labels is costly, especially for large-scale graph datasets, for example, social networks. To address the manual-label issues of supervised DGL, self-supervised graph representation learning (SGL) was proposed to learn graph representations without manual labels. In SGL, GNN-based models were trained by solving some handcrafted pretext tasks such as masked graph generation and graph contrastive learning. During the training, the data can generate the supervision signals by itself automatically, which does not need to provide manual labels. For SGL, a well-designed pretext task will help the learning model to get informative embeddings that promote the performance of the downstream tasks.

Recently, graph contrastive learning (GCL), which aims to maximize the mutual information (MI) of the positive sample pairs and minimize the MI of negative pairs, achieves promising performance and can learn robust and generalizable representations. Among the GCL, Graph augmentation is a critical component, which generates multi-view graphs for contrast. Unlike the data augmentation method for texts and images, graphs are non-Euclidean data that cannot adopt the augmentation methods for Euclidean data [128, 121]. Most existing graph augmentation methods perform stochastic transformation

schemes [156, 129, 51], such as randomly dropping edges or masking node features. However, uniform transformations without carefully designed augmentation techniques may change the underlying semantics of graphs or graph nodes drastically. For example, in social networks, dropping the edges between two people may separate them into two different groups. Some approaches, for example, GCA [206], improve the stochastic transformation schemes by considering the node centrality [206] or the importance of nodes or edges. In this method, they thought the nodes or edges with high degrees (more neighbors) were more important, so they preferred to drop the edges of the nodes with few neighbors. Nevertheless, Liu et al. [91] argue that the nodes with few neighbors will restrict the expressive performance of GNN, but the GCA will aggravate this situation. So that designing augmentation schemes based on the importance of the node is not a good way. The edges connected with unimportant nodes may be crucial for identifying the cluster of the node. Therefore, the graph augmentation should not be determined by the simple node centrality, for example, the node degrees, but the inherent semantics of nodes should be considered. I argue that the graph augmentation schemes should preserve the intrinsic semantics of each graph node or subgraph.

Besides graph augmentation, the MI estimation method is another critical component that is determined by the contrastive objective. Among them, the most effective and promising one is the noise-contrastive estimator (InfoNCE). InfoNCE can be seen as a lower-bound MI estimation, which intends to distinguish the representation of the same node in two different augmented views from other node representations. Therefore, for one node, there is only one positive pair and 2N negative pairs from the inter-view or intra-view node samples. However, existing graph contrastive methods neglect the semantic information that may introduce false-negative samples since they treat all the other samples except the positive sample as negative samples, yet some of them may belong to the same cluster or have similar semantic information with the positive sample. This issue is termed sampling bias [16], which will limit the discriminative capability of the graph representation. I argue that the negative sampling should also consider the semantic information of each node to avoid the sampling bias problem.

To solve the aforementioned issues, I propose a novel graph contrastive learning method with semantic invariance graph augmentation termed SemiGCL by designing a semantic invariance graph augmentation (SemiAug) and a semantic-based graph contrastive (SGC) scheme which leverages a semantic debiasing negative sampling (SDNS) method to generate negative samples. Concretely, to learn a semantic invariance augmentation, a PageRank-based semantic clustering method is proposed to divide the graph into semantic clusters. The PageRank [125] values refer to the importance of each node in a graph. Therefore, similar to the density-based clustering method [135] that cluster centers should have a higher PageRank value than their neighbors and should have a relatively long distance from other points with higher PageRank values. After finding the cluster centers, the graph nodes can be divided into different semantic clusters according to the distance of the shortest path, or geodesic path, to the semantic centers. Then, based on the cluster assignment, a semantic invariance augmentation was proposed on both graph

structure and node attribute. Specifically, two kinds of augmentations were designed, structure-level augmentation and attribute-level augmentation. For the structure level, I randomly added edges on the intra-class clusters, and for the attribute level, similar to mixup [193], the operation of linear interpolation was adapted that mixes each node's features and their cluster prototypes to get the augmented features of each node. At last, a semantic-based graph contrastive (SGC) method was designed with SDNS, which selects negative samples from other clusters except for the positive sample cluster. Therefore, the semantic information can be used for decreasing the false-negative samples that improve the discriminative capability of the graph contrastive network. The general pipeline of the proposed SemiGCL method is illustrated in Figure 4.1.

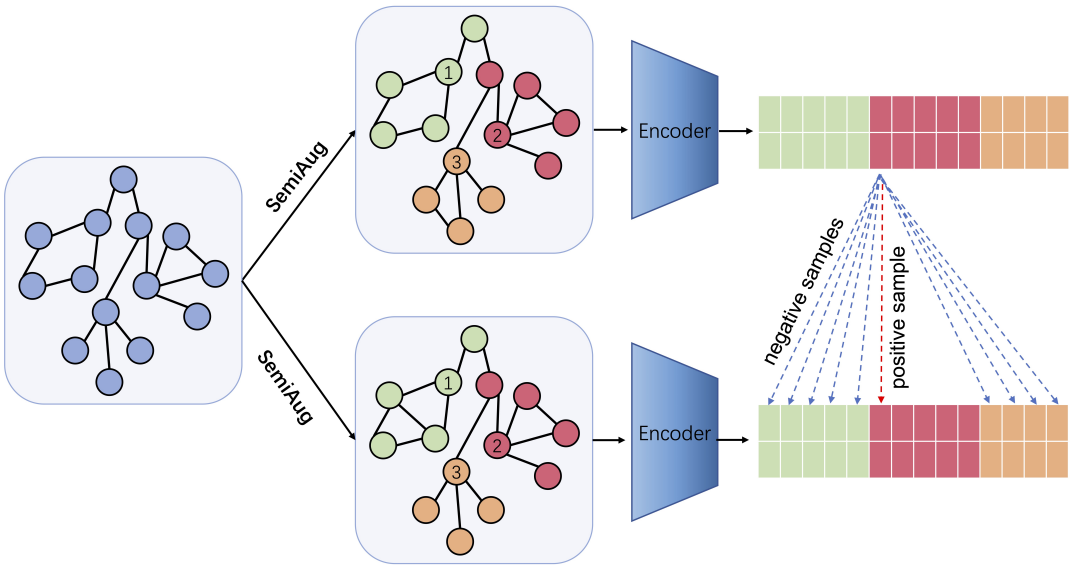

Fig. 4.1 The general pipeline of the proposed SemiGCL method, which consists of two parts. (a) The semantic invariance graph augmentation (SemiAug) first leverages a PageRank-based semantic clustering method to divide the graph into semantic clusters where the node with numbers are the cluster centers, then based on the semantic clusters, a semantic invariance augmentation was proposed on the structure level and attribute level. (b) A semantic-based graph contrastive (SGC) scheme leverages a semantic debiasing negative sampling (SDNS) method that selects negative samples from other clusters except for the positive sample's cluster. The proposed SemiGCL takes advantage of the semantic information of the graph that improves the discriminative capability of the graph contrastive network

The key contributions of this proposed method are summarized as follows:

- In this method, a novel graph augmentation method is presented, which is used to generate semantic invariance augmentations. As graph augmentation is a critical component of GCL that the proposed SemiAug improves the representation performance.
- To decrease the sampling bias problem that the existing graph contrastive methods may introduce false-negative samples, I proposed an SGC method that introduces

SDNS to select negative samples from other clusters except for the positive sample's cluster, which can improve the discriminative capability of the graph contrastive network.

- To learn a semantic partition, a simple but effective PageRank-based semantic clustering method is proposed, which first learns cluster centers by the PageRank values and the shortest path distance, then according to the shortest path distance to each cluster center, the graph can be divided into different semantic clusters efficiently.

4.2 Graph Contrastive Learning with Semantic-invariance Graph Augmentation

Most existing graph augmentation methods perform stochastic transformation schemes [156, 129, 51], such as randomly dropping edges or masking node features. However, uniform transformations without carefully designed augmentation techniques may change the underlying semantics of graphs or graph nodes drastically. Based on the experience of image augmentation, a good augmentation should keep the original intrinsic semantic information of the image. This principle also applies to graph augmentation. So that I argue that the graph augmentation schemes should preserve the intrinsic semantics of each graph node or subgraph. Besides graph augmentation, the MI estimation method is another critical component that is determined by the contrastive objective. Among them, the most effective and promising one is the noise-contrastive estimator (InfoNCE). However, existing graph contrastive methods neglect the semantic information that may introduce false-negative samples since they treat all the other samples except the positive sample as negative samples, yet, some of them may belong to the same cluster or have similar semantic information to the positive sample. This issue is termed sampling bias [16], which will limit the discriminative capability of the graph representation. I argue that the negative sampling should also consider the semantic information of each node to avoid the sampling bias problem.

To solve the aforementioned issues, I propose a novel graph contrastive learning method with semantic invariance graph augmentation termed SemiGCL by designing a semantic invariance graph augmentation (SemiAug) and a semantic-based graph contrastive (SGC) scheme which leverages a semantic debiasing negative sampling (SDNS) method to generate negative samples. Then, I will give a detailed introduction to the SemiGCL model in two parts the SemiAug and the SGC model.

4.2.1 Semantic Invariance Graph Augmentation

To learn a semantic invariance augmentation, a PageRank-based semantic clustering (PSC) method is proposed to divide the graph into semantic clusters. The PageRank [125] values refer to the importance of each node in a graph. Therefore, similar to the density-based

clustering method [135] that cluster centers should have a higher PageRank value than their neighbors and should have a relatively long distance from other points with higher PageRank values. After finding the cluster centers, the graph nodes can be divided into different semantic clusters according to the distance of the shortest path, or geodesic path, to the semantic centers. First, I will give a detailed introduction to the proposed PageRank-based semantic clustering method.

PageRank-based semantic clustering

The PageRank-based semantic clustering (PSC) method is designed based on the density-based clustering method[135, 29]. Density-based clustering algorithms like DBSCAN [29] can be very useful for identifying clusters in data sets where the number and shape of clusters are not known in advance. However, due to their sensitivity to parameter settings, it is important to carefully choose the appropriate values to use for the parameters. This may require some experimentation with different values and evaluating the resulting clustering results to determine the optimal parameter settings. Despite this challenge, density-based clustering can be a powerful tool for uncovering underlying patterns in a data set.

The density peaks clustering (DPC) algorithm is similar to DBSCAN and the mean-shift method in that it can detect non-spherical clusters and does not require a priori knowledge of the number of clusters. In DPC, the algorithm identifies cluster centers based on their higher density compared to their neighbors and their relatively large distance from points with higher densities. One advantage of DPC is that it is robust with respect to the choice of a single parameter, d_c, which represents the cutoff density for defining cluster centers. This means that DPC can provide reliable and consistent clustering results across various data sets. Additionally, ongoing research around DPC suggests that it has the potential for further improvement and application in various fields.

In the PSC method, I introduce the DPC algorithm to the graph domain by redefining the density of the graph based on the PageRank values. The PageRank algorithm uses a mathematical formula to calculate the importance or relevance of a web page based on the links it has with other pages on the internet. The basic idea behind PageRank is that a page is more important if it is linked to many other important pages, and less important if it is not linked to by many other important pages. The computational formula for PageRank involves iterating through a large matrix of web pages and calculating the importance of each page based on the links between pages. The equation for calculating the PageRank of a web page A was defined as follows:

$$PR(A) = (1-d) + d \cdot \sum_i \frac{PR(T_i)}{C(T_i)} \tag{4.1}$$

where $PR(T_i)$ is the PageRank value of page T_i which is connected with page A; $C(T_i)$ is the number of outbound links on page T_i, d denotes a damping factor in $(0,1)$

From Equation 4.1, we can observe that the PageRank of a web page A is recursively defined by the PageRank of web pages that link to A. Within the algorithm, the PageRank

score of web pages T_i is always weighted by the number of outbound links, denoted as $C(T_i)$. This leads to a smaller PageRank value being transferred from T_i to the recipient page A. The algorithm also assumes that any additional inbound link to a recipient page A will always increase its page rank score. So that the PageRank can also be defined as follows:

$$PR(A) = \frac{(1-d)}{N} + d \cdot \sum_i \frac{PR(T_i)}{C(T_i)} \tag{4.2}$$

where N defines the number of pages. The PageRank of a web page can be thought of as the probability that a surfer will visit that page after clicking on many links. The damping factor, denoted as d, represents the probability that a surfer will keep clicking on links and is typically set between 0 and 1. Because the surfer may stop clicking on links and jump to another page at random, the complement of d, which is $(1-d)$, is incorporated into the algorithm.

Because of the immense size of the web, an iterative computation is typically used to approximate the calculation of PageRank. In this approach, each web page is assigned an initial starting value, and the PageRank scores of all pages are computed in iterative rounds, based on Equation 4.1 or 4.2. This process continues until the scores converge or reach a predetermined stopping condition, at which point the final PageRank scores are obtained. This iterative approach allows the algorithm to handle a large number of interconnected pages on the web, while still generating accurate PageRank scores.

The PageRank score measures the importance of a node in a graph based on the quality and quantity of links pointing to it. The scored result then reflects the influence and authority of a node in relation to other nodes on the graph. This score is also related to the density of the graph nodes. This is because if a node's PageRank score is large, the inbound link to this node is corresponding to a lot, so the local density of the subgraph around the node is also large. Therefore, I consider the PageRank score of a node as the subgraph density or local density around this node. Then based on the density peaks clustering (DPC) [135] algorithm we can obtain a fast partition for the graph.

The DPC algorithm computes two metrics, namely the local density ρ_i and the distance δ_i, for every data point x_i. The distance between two points x_i and x_j is represented by d_{ij}. A parameter called the 'cut-off' distance, denoted by d_c, is defined as the value obtained by sorting all the distances in ascending order after computing them between every two points and selecting the value at the threshold.

The local density ρ_i can be calculated by 4.3:

$$\begin{aligned} \rho_i &= \sum_j \chi(x) = \sum_j \chi\left(d_{ij} - d_c\right) \\ \chi(x) &= \begin{cases} 1, x < 0 \\ 0, x \geq 0 \end{cases} \end{aligned} \tag{4.3}$$

If the dataset is not that big, ρ_i can be defined as the Gaussian kernel function:

$$\rho_i = \sum_j \exp\left(-\frac{{d_{ij}}^2}{d_c^2}\right) \tag{4.4}$$

The δ_i is defined as:

$$\delta_i = \begin{cases} \min_{j:\rho_j>\rho_i}\left(d_{ij}\right), \text{ if } \exists \text{ js. t. } \rho_j > \rho_i \\ \max_j\left(d_{ij}\right), \text{ otherwise} \end{cases} \tag{4.5}$$

In the DPC, the PageRank score is used as the local density ρ_i. For the distance of two nodes in the graph, it combined both the feature distance and the structural distance. Concretely, the feature distance is calculated by the Euclidean metric and the structural distance is calculated by the shortest path. The distance is defined as follows:

$$d_{ij} = Eu(x_i, x_j) + SP(x_i, x_j) \tag{4.6}$$

where $Eu()$ denote the euclidean distance and $SP()$ denote the shortest path.

Once the DPC algorithm calculates the local density ρ_i and distance δ_i for each data point, it selects the points with higher values of ρ_i and δ_i as the cluster centers. This selection is made by plotting the decision graph with ρ_i on the x-axis and $\delta_i$on the y-axis. Next, the DPC algorithm assigns the remaining points to the nearest neighbor classes. It does so by locating each data point to the class of its nearest point with an equal or higher density. Finally, the algorithm removes the outliers whose density is less than the boundary threshold of the current category.

Semantic Invariance Graph Augmentation

Two kinds of semantic-invariance graph augmentation (SemiAug) have been proposed in the SemiGCL method, i.e., feature-level augmentation and structure-level augmentation, which are jointly performed in the SemiGCL model.

Feature-level Augmentation. The SemiAug assumes that there are M latent clusters in the graph. After the PSC algorithm the latent variable $c_i \in \{1, 2, \ldots, M\}$ was introduced to indicate the belonged clusters of a node v_i. SemiAug creates an enhanced version of node v_i at the feature level for SemiGCL, using linear interpolation:

$$\tilde{\mathbf{h}}_i = \alpha \mathbf{h}_i + (1-\alpha)\mathbf{w}_{c_i} \tag{4.7}$$

where $\mathbf{w} = \{\mathbf{w}_m\}_{m=1}^M$ is the cluster prototypes. $\tilde{\mathbf{h}}_i$ includes information from v_i and it's cluster prototype. The augmentation makes the distant positive pair closer and the near negative pair away, which retains the nodes' cluster distributions. α is the parameter to control the augmentation strength.

Structure-level Augmentation. The traditional structure-level augmentation methods, also called edge perturbation [156, 185, 184, 206], always generate a new graph by randomly

removing existing edges from the input graph and randomly adding new edges to it. However, uniform transformations without carefully designed augmentation techniques may change the underlying semantics of graphs or graph nodes drastically. Especially for randomly removing edges, if a crucial edge is removed, the semantics of the graph will be changed a lot. A good augmentation method should not change the semantic information of the original data. Therefore, to maximize the preservation of the original semantic meaning, the proposed structure-level SemiAug method only adds some new edges in the intra-class. The process of edge perturbation in a mathematical context preserves the initial order of nodes and modifies a portion of the entries in the adjacency matrices provided. This can be defined as follows:

$$\tilde{\mathbf{A}} = \mathbf{A} \oplus \mathbf{C}, \tag{4.8}$$

where $\mathbf{C}$ denote the corruption matrix and $\oplus$ is the XOR (exclusive OR) operation. Typically, the corruption matrix $\mathbf{C}$ is generated by independent and identically distributed (i.i.d.) sampling from a prior distribution. The value of each entry $\mathbf{C}_{ij}$ determines whether corruption will be applied to the corresponding position (i, j) in the adjacency matrix. For the proposed SemiAug, according to the initial cluster assignment $c_i \in \{1, 2, \ldots, M\}$, I randomly add edges in the intra-class of the node group. For example, given a corruption rate ρ, In the intra-class of the node group, I can define the corruption matrix as $\mathbf{C}_{ij} \sim \text{Bernoulli}(\rho)$. The elements in $\mathbf{C}$ are set to 1 with the probability ρ and 0 with the probability $1-\rho$.

4.2.2 Semantic-based Graph Contrastive Scheme

For the traditional graph contrastive method. The contrastive objective is as a discriminator which distinguishes the embeddings of the same node in two different views from other node embeddings. In this approach, for each node v_i I consider the embedding generated in one view, denoted as $\boldsymbol{u}_i$, as the anchor, and the corresponding embedding produced in the other view, i.e., $\boldsymbol{v}_i$, as the positive sample. Moreover, all the other node embeddings in the two views are considered negative samples. This definition of the pairwise objective for each positive pair $(\boldsymbol{u}_i, \boldsymbol{v}_i)$ echo the InfoNCE objective [122], adapted to the multi-view graph CL model. The objective function is defined as:

$$\ell(\boldsymbol{u}_i, \boldsymbol{v}_i) = \log \frac{e^{\theta(\boldsymbol{u}_i, \boldsymbol{v}_i)/\tau}}{e^{\theta(\boldsymbol{u}_i, \boldsymbol{v}_i)/\tau} + \sum_{k \neq i} e^{\theta(\boldsymbol{u}_i, \boldsymbol{v}_k)/\tau} + \sum_{k \neq i} e^{\theta(\boldsymbol{u}_i, \boldsymbol{u}_k)/\tau}} \tag{4.9}$$

where τ denotes the temperature parameter. θ is a critic function that is defined as $\theta(\boldsymbol{u}, \boldsymbol{v}) = s(g(\boldsymbol{u}), g(\boldsymbol{v}))$, where $s(\cdot, \cdot)$ denote the cosine similarity and $g(\cdot)$ denote the nonlinear projection [14, 153] which commonly used in the contrastive learning. We can define negative samples for a positive pair as all other nodes in the two views. This means that negative samples can come from two sources: inter-view and intra-view nodes, which correspond to the second and third terms in the denominator of Equation 4.9, respectively. As these two views are symmetric that the overall loss is defined as

$$\mathcal{J} = \frac{1}{2N}\sum_{i=1}^{N}\left[\ell\left(\boldsymbol{u}_i, \boldsymbol{v}_i\right) + \ell\left(\boldsymbol{v}_i, \boldsymbol{u}_i\right)\right] \tag{4.10}$$

However, the existing graph contrastive model neglects the semantic information that may introduce false-negative samples since they treat all the other samples except the positive pairs as negative samples, yet some of them may belong to the same cluster or have similar semantic information with the positive samples. This issue is termed sampling bias [16], which will limit the discriminative capability of the graph representation. Therefore, the negative sampling should also consider the semantic information of each node to avoid the sampling bias problem.

To solve the sampling bias problem, the proposed SemiGCL method first leverages the PSC method to obtain the initial cluster assignment. Assume there are M latent clusters in the graph. The initial cluster indicating variable $c_i \in \{1, 2, \ldots, M\}$ can be used as the supervision information for the graph contrastive model. Then, a semantic debiasing negative sampling (SDNS) method is proposed to generate negative samples. Different from the traditional negative sampling method which defined all the other nodes except the positive pairs as negative samples, SDNS defines the negative samples from other clusters except for the positive sample's cluster. The SGC objective function is defined as:

$$\ell\left(\boldsymbol{u}_i, \boldsymbol{v}_i\right) = \log \frac{e^{\theta(\boldsymbol{u}_i, \boldsymbol{v}_i)/\tau}}{\sum_{v_k \notin c_i} e^{\theta(\boldsymbol{u}_i, \boldsymbol{v}_k)/\tau} + \sum_{v_k \notin c_i} e^{\theta(\boldsymbol{u}_i, \boldsymbol{u}_k)/\tau}} \tag{4.11}$$

Comparing the traditional objective of the GCL Equation 4.9 with the SGC objective Equation 4.11, we can see that, the positive pairs are the same, but for the negative samples, in the SGC objective, the samples that are in the same cluster with the positive pairs are all removed from the negative samples. Therefore, in the SGC objective, the positive samples are very similar, and the negative samples are usually quite different from the positive ones which follows the principle of contrastive learning. The proposed SGC model with SDNS decreases the false-negative samples by using the learned semantic information that improves the discriminative capability of the graph contrastive network. As the two views of the graph are symmetric the overall loss is the same as Equation 4.10.

4.3 Experiments

This section presents the experiments to evaluate the proposed SemiGCL model. First, I will briefly introduce the experimental setup and then I will provide details of the experiment process and the experimental results as well as their analysis.

Table 4.1 Summary of the datasets used in the node classification experiments.

Dataset	Nodes	Edges	Features	Classes
Wiki-CS	11,701	216,123	300	10
Amazon-Computers	13,752	245,861	767	10
Amazon-Photo	7,650	119,081	745	8
Coauthor-CS	18,333	81,894	6,805	15
Coauthor-Physics	34,493	247,962	8,415	5

4.3.1 Experimental Setup

Datasets

To conduct a comprehensive comparison, my study includes the use of five commonly used datasets: Wiki-CS[1], Amazon-Computers[2], AmazonPhoto[3], Coauthor-CS[4], and Coauthor-Physics[5]. These datasets come from real-world networks in various domains, and their detailed statistics are summarized in Table 4.1. Our focus is to analyze the performance of transductive node classification on these datasets. Then I will give an introduction to each dataset:

- **Wiki-CS** [107] is a reference network that is constructed based on Wikipedia. Its nodes correspond to articles about computer science, and the edges represent hyperlinks between the articles. Each node is labeled with one of ten classes, representing a specific branch of the field. The features of each node are calculated as the average of pretrained GloVe[130] word embeddings of the words used in the article.

- **Amazon-Computers** and **Amazon-Photo** [143] are two graphs that represent co-purchase relationships constructed by Amazon. In these networks, each node represents a good or product, and two goods are connected when they are frequently purchased together. Additionally, every node is labeled with its category and has a sparse bag-of-words feature encoding product reviews.

- **Coauthor-CS** and **Coauthor-Physics** [143] are two academic networks that contain co-authorship graphs based on the Microsoft Academic Graph from the KDD Cup 2016 challenge. These graphs represent nodes as authors and edges as co-authorship relationships between them; that is, two nodes are connected if they have co-authored a paper. Additionally, each node has a sparse bag-of-words feature based on the paper keywords of the corresponding author. The label of an author corresponds to their most active research field.

[1]https://github.com/pmernyei/wiki-cs-dataset/raw/master/dataset
[2]https://github.com/shchur/gnn-benchmark/raw/master/data/npz/amazon_electronicscomputers.npz
[3]https://github.com/shchur/gnn-benchmark/raw/master/data/npz/amazon_electronics_photo.npz
[4]https://github.com/shchur/gnn-benchmark/raw/master/data/npz/ms_academic_cs.npz
[5]https://github.com/shchur/gnn-benchmark/raw/master/data/npz/ms_academic_phy.npz

Among the five datasets, the Wiki-CS has dense numerical features, while the other four datasets only contain sparse one-hot features. In the case of the Wiki-CS dataset, the evaluation of models is conducted using the public splits shipped with the dataset [107]. However, for the other four datasets, there are no public splits available. In order to evaluate these datasets, I randomly split them into training, validation, and test sets, where 10%, 10%, and 80% of nodes are selected for each respective set.

Evaluation protocol

I follow the linear evaluation scheme outlined in DGI [156] for each experiment. The scheme first trains each model in an unsupervised manner, then uses the resultant embeddings to train and evaluate a basic ℓ_2 regularized logistic regression classifier. To ensure a fair evaluation, I replicate this procedure twenty times with different data splits and report the mean performance on each dataset. Furthermore, accuracy is the metric used to measure performance in these experiments.

Baselines

I evaluate representative baseline methods that can be categorized into two types: (1) traditional methods such as DeepWalk [131] and node2vec [44]; and (2) deep learning methods including Graph Autoencoders (GAE, VGAE) [71], Deep Graph Infomax (DGI) [156], Graphical Mutual Information Maximization (GMI) [129], Multi-View Graph Representation Learning (MVGRL) [51] and Graph Contrastive representation learning with Adaptive augmentation (GCA)[206]. Additionally, I present results obtained by applying a logistic regression classifier on raw node features and using DeepWalk with embeddings concatenated with input node features. In all cases, the performance metrics are reported based on the official implementations of the baseline methods.

Implementation details

I use a two-layer GCN [72] as the backbone network for all the deep learning baselines. The encoder architecture is defined as:

$$\begin{aligned} \mathrm{GCN}_i(X,A) &= \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}XW_i\right), \\ f(X,A) &= \mathrm{GCN}_2\left(\mathrm{GCN}_1(X,A),A\right). \end{aligned} \tag{4.12}$$

where $\hat{A} = A + I$ denote the adjacency matrix with self-loops, $\hat{D} = \sum_i \hat{A}_i$ denote the degree matrix, $\sigma(\cdot)$ denote the nonlinear activation function and $\boldsymbol{W}_i$ denote the weight matrix.

4.3.2 Experiment Results of Graph Node Classification

The summary of the experiment performance of the different deep learning baselines is in Table 4.2. Overall, the table shows that the proposed model exhibits strong performance across all five datasets. SemiGCL consistently outperforms unsupervised baselines by

significant margins on both transductive tasks, demonstrating that the proposed contrastive learning framework is superior. On the two coauthor datasets where existing baselines have already achieved high performance, SemiGCL can still push the performance boundary further.

Furthermore, I make the following observations. Firstly, traditional contrastive learning methods like DeepWalk have inferior performance compared to simple logistic regression classifiers that only use raw features on some datasets (Coauthor-CS and Coauthor-Physics), indicating their lack of effectiveness in utilizing node features. In contrast, GCN-based methods, such as GAE, are able to incorporate node features when learning embeddings. However, on certain datasets (Wiki-CS), their performance is still worse than DeepWalk+feature, which we attribute to their simplistic method of selecting negative samples based on edges only. This demonstrates the importance of selecting negative samples based on augmented graph views in contrastive representation learning. Moreover, compared to existing baselines DGI, GMI, and MVGRL, the proposed method performs better due to its semantic invariance graph augmentation and debiasing negative sampling in the graph contrastive model, which improves the discriminative capability of the graph contrastive network. Even though MVGRL incorporates global information through diffusion, it fails to consider the effects of semantic information on the input graphs.

Secondly, I observe that GCA [206] improves the stochastic transformation schemes by considering the node centrality [206] or the importance of nodes or edges. In this method, they thought the nodes or edges with high degrees (more neighbors) were more important, so they preferred to drop the edges of the nodes with few neighbors. Nevertheless, Liu et al. [91] argue that the nodes with few neighbors will restrict the expressive performance of GNN, but the GCA will aggravate this situation. So, designing augmentation schemes based on the node's importance is not a good way. The edges connected with unimportant nodes may be crucial for identifying the node's cluster. Therefore, the graph augmentation should not be determined by the simple node centrality, for example, the node degrees, but the inherent semantics of nodes should be considered.

In summary, the superior performance of SemiGCL compared to existing state-of-the-art methods verifies the effectiveness of the proposed SemiGCL approach that performs data augmentation based on the semantic invariance principle and performs the graph contrastive model with debiasing negative sampling.

Then, I will analyze the complexity of the proposed SemiGCL. Compared to (GCA)[206], the PageRank-based semantic clustering is learning first. The PageRank-based semantic clustering is a fast clustering method based on PageRank, and its time complexity is $\mathcal{O}(n)$. So, compared to GCA, no extra complexity is introduced.

4.4 Conclusion

Graph Contrastive Learning (GCL) has become the most successful and powerful method for self-supervised graph representation learning (SGL). Graph augmentation is a critical component of GCL, which is used to generate different views of input graphs. Most

Table 4.2 The summary of the experiment performance of the different deep learning baselines.

Method	Training Data	Wiki-CS	Amazon-Computers	Amazon-Photo	Coauthor-CS	Coauthor-Physics
Raw features	$\boldsymbol{X}$	71.98±0.00	73.81±0.00	78.53±0.00	90.37±0.00	93.58±0.00
node2vec	$\boldsymbol{A}$	71.79±0.05	84.39±0.08	89.67±0.12	85.08±0.03	91.19±0.04
DeepWalk	$\boldsymbol{A}$	74.35±0.06	85.68±0.06	89.44±0.11	84.61±0.22	91.77±0.15
DeepWalk + features	$\boldsymbol{X},\boldsymbol{A}$	77.21±0.03	86.28±0.07	90.05±0.08	87.70±0.04	94.90±0.09
GAE	$\boldsymbol{X},\boldsymbol{A}$	70.15±0.01	85.27±0.19	91.62±0.13	90.01±0.71	94.92±0.07
VGAE	$\boldsymbol{X},\boldsymbol{A}$	75.63±0.19	86.37±0.21	92.20±0.11	92.11±0.09	94.52±0.00
DGI	$\boldsymbol{A},\boldsymbol{A}$	75.35±0.14	83.95±0.47	91.61±0.22	92.15±0.63	94.51±0.52
GMI	$\boldsymbol{X},\boldsymbol{A}$	74.85±0.08	82.21±0.31	90.68±0.17	OOM	OOM
MVGRL	$\boldsymbol{A},\boldsymbol{A}$	77.52±0.08	87.52±0.11	91.74±0.07	92.11±0.12	95.33±0.03
GCA-DE	$\boldsymbol{X},\boldsymbol{A}$	78.30±0.00	87.85±0.31	92.49±0.09	93.10±0.01	95.68±0.05
GCA-PR	$\boldsymbol{X},\boldsymbol{A}$	78.35±0.05	87.80±0.23	92.53±0.16	93.06±0.03	95.72±0.03
GCA-EV	$\boldsymbol{X},\boldsymbol{A}$	78.23±0.04	87.54±0.49	92.24±0.21	92.95±0.13	95.73±0.03
SemiGCL	$\boldsymbol{X},\boldsymbol{A}$	**78.95±0.06**	**88.05±0.20**	**92.69±0.26**	**93.53±0.21**	**95.89±0.12**

existing GCL methods perform stochastic data augmentation schemes. However, uniform transformations without carefully designed augmentation techniques may change the underlying semantics of graphs or graph nodes drastically. Besides, existing graph contrastive methods neglect the semantic information that may introduce false-negative samples. Therefore, a novel graph contrastive learning method with semantic invariance graph augmentation termed SemiGCL is proposed by designing a semantic invariance graph augmentation (SemiAug) and a semantic-based graph contrastive (SGC) scheme. SemiGCL outperforms the state-of-the-art methods due to its semantic invariance graph augmentation and debiasing negative sampling in the graph contrastive model, which improves the discriminative capability of the graph contrastive network.

Chapter 5

Structural Semantic Contrastive Deep Graph Clustering

5.1 Introduction

The widespread adoption of networked applications such as citation networks [71], social networks [124], and protein-protein interaction networks [33], has resulted in a massive amount of graph data. Graph data mining plays an important role in analyzing the structures of these networks [170]. However, the complexity of real-world graph structures poses a significant challenge for graph learning tasks, especially graph clustering [161].

Graph clustering [161] aims to group node objects in a graph that the objects from a group are more similar to each other than those from other groups in an unsupervised manner. Graph clustering has been used in numerous applications such as community detection [83], customer segmentation[191], and protein interaction prediction[33].

Early graph clustering methods leverage various approaches to group the graph nodes, such as density-based clustering [118], co-clustering [47] and non-negative matrix factorization [164, 83]. However, these shallow approaches only capture either part of the graph information or shallow relationships between graph topology and node features, which leads to their sub-optimal performance.

In recent years, benefiting from the rapid development and the strong graph representation performance of deep graph neural networks (GNNs), i.e., Graph Convolutional Networks (GCN)[72], Graph Attention Networks (GAT)[155] and Message Passing Neural Networks (MPNN)[38], Deep Graph Clustering(DGC) has become the state-of-the-art graph clustering methods. To learn the representation of both the topology structure and node features, Graph Autoencoders (GAE/VGAE) [71] is proposed with GCN and a reconstruction decoder. More recently, contrastive learning[156, 129, 51] has become the most promising technique in DGC, benefiting from the powerful capability of capturing implicit supervision information [94].

Although the promising performance of contrastive graph clustering has been achieved, I found that the existing method still suffers from many problems. Firstly, in the process of node encoding, existing graph contrastive learning (GCL) methods which can be divided

into two kinds: node–global contrastive scheme and node-node contrastive scheme [206], only contrast the node views of graphs but neglect the important structural information. Therefore, without considering the global graph structure information in GCL, existing methods may suffer from representation collapse [94], which may map all nodes to the same representation. Consequently, the discriminative capability of representation will degenerate. Secondly, in the process of node clustering, most of the existing DGC methods leverage traditional clustering methods, for example, K-means. However, the traditional clustering methods have some drawbacks as follows: 1) Firstly, these methods are center-based clustering algorithms that are highly dependent on the cluster center initialization so the clustering results of the traditional clustering methods are not stable. 2) Besides, As the representation and the clustering process are two separate stages, the traditional clustering methods can not leverage the strong graph representation capability of GNNs. 3) lastly, these methods cannot jointly optimize the node representation learning and clustering in a unified framework.

To solve the aforementioned problems, I propose a novel contrastive deep graph clustering method called Structural Semantic Contrastive Deep Graph Clustering (SECRET) by designing a structure contrastive scheme (SCS) to keep the structural consistency of two different views and a new self-supervised deep graph clustering method that leverages a comprehensive similarity measure criterion considering both the attribute embedding similarity and the structural similarity, which better reveals node relationships and can be seen as supervision information, as similar nodes should be in the same clusters. Concretely, to get a more discriminative representation, the SCS is proposed by contrasting the aggregation of first-hop neighbors and a graph diffusion, as the original graph and its graph diffusion can be seen as two different structural views. I design the SCS to keep the structure of different views consistent. A new cross-view structural consistency objective function is proposed to enhance the discriminative capability of the representation network. For each view, I adopt the node–global contrastive scheme, which maximizes mutual information (MI) between node and graph representations. Therefore, the designed new GCL approach includes both node-level contrastive and structure-level contrastive, which will enhance the discriminative capability of the learned network. To alleviate the aforementioned problem of traditional clustering methods, I first calculate the nearest neighbors of each node in the embedding space by the comprehensive similarity. Then, I propose a self-supervised deep-learning-based clustering (SDC) model by adding a cluster head in the representation networks. Concretely, in the SDC method, the cluster head and the representation networks are jointly optimized by assigning a node and one of its nearest neighbors to the same cluster, according to the homophily property assumption of real-world networks. The general pipeline of the proposed SECRET method is illustrated in Figure 5.1.

The key contributions of this proposed method are summarized as follows:

- In this approach, a novel GCL model is presented by introducing the SCS, which is used to keep the structural consistency of two different views of a graph. Our proposed GCL method leverage both node-level contrastive and structure-level

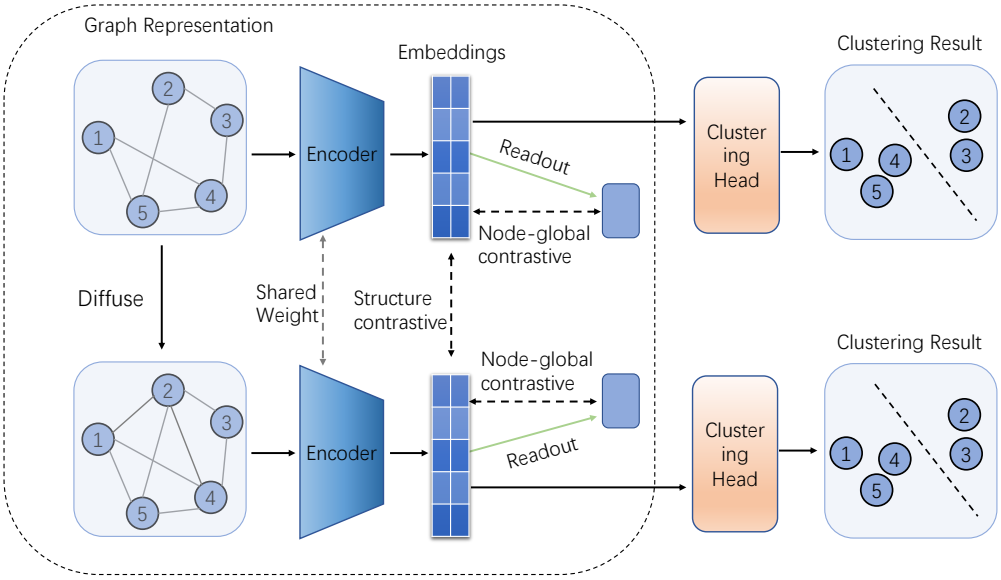

Fig. 5.1 The general pipeline of the proposed SECRET method, which consists of two parts. (a) The graph structure contrastive representation learning (GSC-RL) method is proposed to learn the structural-semantic embeddings, which include a structure contrastive scheme (SCS) to keep the consistency of the local structure and the global structure and a node-global contrastive scheme to preserve the MI between node representations with the global graph summary. (b) The self-supervised deep-learning-based clustering (SDC) combined the graph encoder networks with a clustering head (MLP). I leverage the proposed comprehensive similarity on the learned embeddings as prior and then encourages the SDC to output the same labels for similar instances.

contrastive, which will prompt the neural network to learn an accurate graph structure and a more discriminative representation.

- Instead of leveraging the traditional clustering methods, like K-means, I proposed a self-supervised deep-learning-based clustering (SDC) model, which can leverage the strong graph representation capability of GNNs by jointly optimizing the cluster head and the representation networks through the gradient descent algorithms.
- To train the SDC model, I calculate the nearest neighbors of each node in the embedding space as supervision information. And to better reveal the node relationships in a graph, a comprehensive similarity measure criterion is proposed, which consists of the attribute embedding similarity and the structural similarity.

5.2 Structural Semantic Contrastive Deep Graph Clustering

In this section, I will introduce the proposed Structural Semantic Contrastive Deep Graph Clustering (SECRET) model which includes a Graph Structure Contrastive Representation Learning (GSC-RL) method with a structure contrastive scheme (SCS) to keep the

structural consistency of two different views and a Self-supervised Deep-learning-based Clustering (SDC) method to leverage the strong graph representation capability of GNNs. As illustrated in Figure 5.1, SECRET mainly consists of two components, i.e., a graph representation module and a graph clustering module. The following subsections will explain each component of the SECRET model and network objectives in detail.

5.2.1 Graph Structure Contrastive Representation Learning

In this section, I will introduce the Graph Structure Contrastive Representation Learning (GSC-RL) method, which includes a structure contrastive scheme (SCS) to keep the consistency of the local structure and the global structure and a node-global contrastive scheme to preserve the MI between node representations with the global graph summary. The GSC-RL leverages both node-level contrastive and structure-level contrastive, which will enhance the discriminative capability of the learned network and obtain structural-semantic embeddings.

Concretely, the SCS is proposed by contrasting the aggregation of first-hop neighbors and graph diffusion to get a more discriminative representation. The original graph and its graph diffusion are two different structural views. I designed the SCS to keep the structure of different views consistent. A new cross-view structural consistency objective function is proposed to enhance the discriminative capability of the representation network. I adopt the node–global contrastive scheme for each view, which maximizes mutual information (MI) between node and graph representations. Therefore, the new GSC-RL approach includes both node-level contrastive and structure-level contrastive, enhancing the learned network's discriminative capability.

Problem Definition

Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. There are C categories of nodes in the graph. $\mathcal{V}$ and $\mathcal{E}$ are the node set and edge set, respectively. Among them, $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ and $\mathcal{E} = \{e_1, e_2, \ldots, e_M\}$. The attribute of the graph node is $\mathbf{X} \in \mathbb{R}^{N \times D}$ and the adjacency matrix is $\mathbf{A} = \left(a_{ij}\right)_{N \times N}$, where $a_{ij} = 1$ if $\left(v_i, v_j\right) \in \mathcal{E}$, otherwise $a_{ij} = 0$. The degree matrix is $\mathbf{D} = \operatorname{diag}(d_1, d_2, \ldots, d_N) \in \mathbb{R}^{N \times N}$ and $d_i = \sum_{\left(v_i, v_j\right) \in \mathcal{E}} a_{ij}$. The normalized adjacency matrix is denoted by $\widetilde{\mathbf{A}} \in \mathbb{R}^{N \times N} = \mathbf{D}^{-1}(\mathbf{A} + \mathbf{I})$, where $\mathbf{I} \in \mathbb{R}^{N \times N}$ denote the identity matrix.

Graph Diffuse

The structure contrastive scheme (SCS) aims to keep local and global structures consistent. In this method, the global structure views are generated by the diffusion matrices [36]. Diffusion matrices are mathematical constructs that model and analyze the spread of information, influence, or other entities through a network or system. They are particularly relevant in network science, social network analysis, epidemiology, and physics. In the context of network science and social networks, diffusion matrices are often associated

with diffusion processes, which describe how a phenomenon (such as information, behavior, or disease) spreads across nodes or edges of a network. These matrices represent the probabilities or rates of transition between different states or conditions within the network. As the adjacency and diffusion matrices represent the graph's local structure and global structure, respectively, maximizing the agreement of these two views of structures allows the neural networks to learn rich information from both local and global structures. Diffusion is defined as follows:

$$\mathbf{A}^d = \sum_{k=0}^{\infty} \Theta_k \mathbf{T}^k \in \mathbb{R}^{n \times n} \tag{5.1}$$

where $\mathbf{T} \in \mathbb{R}^{n \times n}$ denote the generalized transition matrix, Θ denote the weighting coefficient and it can determine the proportion of local and global information. To guarantees convergence, $\sum_{k=0}^{\infty} \theta_k = 1, \theta_k \in [0,1]$, and $\lambda_i \in [0,1]$ are eigenvalues of $\mathbf{T}$. Diffusion can be calculated by sparsification and fast approximation approaches [36].

Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. The adjacency matrix is $\mathbf{A} \in \mathbb{R}^{n \times n}$ and their diagonal degree matrix is $\mathbf{D} \in \mathbb{R}^{n \times n}$. There are two kinds of commonly used generalized graph diffusion. They are Personalized PageRank (PPR) [125] based and heat-kernel [73] based generalized graph diffusion. They set $\mathbf{T} = \mathbf{A}\mathbf{D}^{-1}$, and $\theta_k = \alpha(1-\alpha)^k$ and $\theta_k = e^{-t}t^k/k$!, respectively, where α is the teleport probability of random walk and t is the time for diffusion [36]. Therefore, The heat and PPR diffusion is defined as

$$\mathbf{A}^d_{\text{heat}} = \exp\left(t\mathbf{A}\mathbf{D}^{-1} - t\right) \tag{5.2}$$

$$\mathbf{A}^d_{\text{PPR}} = \alpha\left(\mathbf{I}_n - (1-\alpha)\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)^{-1} \tag{5.3}$$

As shown in the first step of Figure 5.1, I transform a sample graph into a correlated view of the same graph and I only apply the transformation to the structure of the graphs and not the initial node features. So that the two views of the graph for the structure contrastive are generated. In this method, I use $\mathbf{A}^d$ to represent the heat and PPR diffusion uniformly. Finally, I denote $\mathcal{G}^d = \left(\mathbf{X}, \mathbf{A}^d\right)$ as the diffusion of the graph.

Graph Augmentation

Self-supervised graph representation learning has shown promising results in learning rich representations for nodes in a graph. By leveraging graph augmentation techniques [51, 185], such as adding or removing edges or nodes, the self-supervised model can learn from different contexts and enhance the quality of the learned representations. These techniques enable the model to capture complex patterns and structures present in the graph, which would be difficult to learn using traditional supervised learning approaches.

Data augmentations have become an effective method to improve the representation performance of images and text. A lot of transformation methods have been proposed for images and text, such as flipping, rotation, color shifting [74], back translation [141],

and positional swaps. However, these data augmentation methods can only be used in Euclidean data. Graphs are non-Euclidean structural data formed with nodes and edges and are relational data. The nodes are connected by edges, which is an entirety. Some little structural modifications may cause a great change in semantic information.

I leverage the adaptive augmentation method proposed in [206] on both the original and diffusion graphs. A graph consists of two parts of information, i.e., node features and topology structure. Intuitively, two kinds of graph augmentations, i.e., feature masking and edge perturbation, are jointly performed in the proposed SECRET model.

Feature Masking. For the attribute-level augmentation, I first calculate the masking matrix $\mathbf{N} \in \mathbb{R}^{N \times D}$ by the adaptive augmentation method [206]. The adaptive augmentation method is a graph-augmented algorithm that tries to keep the important feature and structure information and perturb the unimportant one. Then the augmented attribute matrix $\widetilde{\mathbf{X}} \in \mathbb{R}^{N \times D}$ can be defined as:

$$\widetilde{\mathbf{X}} = \mathbf{X} \odot \mathbf{N} \tag{5.4}$$

where $\odot$ is the Hadamard product [56].

Edge Perturbation. For structure-level augmentation, similar to feature masking, I first generate a masked matrix $\mathbf{M} \in \mathbb{R}^{N \times N}$ according to the adaptive augmentation method [206], then, the edge-masked adjacency matrix $\mathbf{A}^m \in \mathbb{R}^{N \times N}$ can be calculated and normalized by

$$\mathbf{A}^m = \mathbf{D}^{-\frac{1}{2}}((\mathbf{A} \odot \mathbf{M}) + \mathbf{I})\mathbf{D}^{-\frac{1}{2}} \tag{5.5}$$

Finally, I denote the augmentation of the original graph $\mathcal{G}$ and the diffusion graph $\mathcal{G}^d$ as $\mathcal{G}^\alpha = \left(\widetilde{\mathbf{X}}, \mathbf{A}^m\right)$ and $\mathcal{G}^\beta = \left(\widetilde{\mathbf{X}}, \mathbf{A}^{dm}\right)$ respectively.

The Graph Structure Contrastive Framework

In this method, I designed two kinds of contrastive learning methods. the SCS to keep the structure of different views consistent and for each view, I adopt the node–global contrastive scheme which maximizes mutual information (MI) between node and graph representations. Therefore, the designed GSC-RL approach includes both node-level contrastive and structure-level contrastive. Then I will introduce these two kinds of contrastive schemes in detail.

a. Structure Contrastive Scheme

The SCS is proposed by contrasting the aggregation of first-hop neighbors and graph diffusion to get a more discriminative representation. The original graph and its graph diffusion are two different structural views. I design the SCS to keep the structure of different views consistent. A new cross-view structural consistency objective function is proposed to enhance the discriminative capability of the representation network.

The proposed framework provides flexibility in selecting the network architecture, as it does not impose any constraints. I choose to prioritize simplicity by utilizing the commonly utilized graph convolution network (GCN) [72] as the base graph encoder.

For each view, I employ a dedicated graph encoder. i.e., $g_\theta(), .g_\omega() : .\mathbb{R}^{n\times d_x} \times \mathbb{R}^{n\times n} \longmapsto \mathbb{R}^{n\times d_h}$. I regard adjacency and diffusion matrices as two structurally consistent views. The GCN layers are defined as $\sigma(\mathbf{A}^m\widetilde{\mathbf{X}}\mathbf{\Theta})$ and $\sigma(\mathbf{A}^{dm}\widetilde{\mathbf{X}}\mathbf{\Theta})$. The objective is to learn two distinct sets of node representations, where each set of representations corresponds to one of the views, respectively. $\tilde{\mathbf{A}} = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2} \in \mathbb{R}^{n\times n}$ denote the symmetrically normalized adjacency matrix, $\hat{\mathbf{D}} \in \mathbb{R}^{n\times n}$ denote the degree matrix of $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I_N}$ where $\mathbf{I_N}$ denote the identity matrix, $\mathbf{A}^d \in \mathbb{R}^{n\times n}$ denote the diffusion matrix, $\mathbf{X} \in \mathbb{R}^{n\times d_x}$ denote the node features, $\mathbf{\Theta} \in \mathbb{R}^{d_x\times d_h}$ denote the network parameters. At last, the output is two sets of graph node representations $\mathbf{H}^\alpha, \mathbf{H}^\beta \in \mathbb{R}^{n\times d_h}$ which from two views of a graph.

The proposed SCS aims to keep the structure of different views consistent so that the model maximizes the agreement of the cross-view [206, 179, 205, 204]. Specifically, $\mathcal{L}_1$ is the Mean Squared Error (MSE) loss between the two views of the graph. It is defined as:

$$\mathcal{L}_1 = \frac{1}{N}\sum_{i=1}^{N}\left\|\vec{h}_i^\alpha - \vec{h}_i^\beta\right\|_2^2 \tag{5.6}$$

where $\vec{h}_i^\alpha, \vec{h}_i^\beta$ denote the representations of node i from views α, β, respectively.

b. Node-global Contrastive Scheme

I follow the intuitions from DGI [156] to learn a graph convolutional encoder that maximizes local-global mutual information to obtain node representations that capture the globally relevant information of the entire graph.

The global graph representation $\vec{h}_g$ is obtained by a readout function, $\vec{h}_g = \mathcal{R}(\mathbf{H})$. DGI employs a discriminator to maximize the local mutual information. The proposed method utilizes the deep InfoMax [55] algorithm and maximizes the MI between the representation of each node and their corresponding global graph representation. The objective is defined as:

$$\mathcal{L}_2 = \frac{1}{N}\sum_{i=1}^{N}\left[\mathrm{MI}\left(\vec{h}_i^\alpha, \vec{h}_g^\alpha\right) + \mathrm{MI}\left(\vec{h}_i^\beta, \vec{h}_g^\beta\right)\right] \tag{5.7}$$

where N is the number of nodes in the graph. $\vec{h}_i^\alpha, \vec{h}_i^\beta$ denote the representations of node i from views α, β, respectively, and $\vec{h}_g^\alpha, \vec{h}_g^\beta$ denote the representations of graph $\mathcal{G}$ from views α, β, respectively.

5.2.2 The Self-supervised Deep-learning-based Clustering

This section proposes a self-supervised deep-learning-based clustering (SDC) model by adding a cluster head in the representation networks. Concretely, in the SDC method, the cluster head and the representation networks are jointly optimized by assigning a node and one of its nearest neighbors to the same cluster, according to the homophily property assumption of real-world networks.

After the GSC-RL, a shared projection head receives the learned representations as input. $f_\psi() : .\mathbb{R}^{n\times d_h} \longmapsto \mathbb{R}^{n\times d_h}$. The projection head is an MLP with two hidden layers.

The GSC-RL has generated two sets of graph node representations $\mathbf{H}^{\alpha}, \mathbf{H}^{\beta}$. I employ a linear combination operation to merge the latent embeddings obtained from two different views:

$$\mathbf{H} = \frac{1}{2}\left(\mathbf{H}^{\alpha} + \mathbf{H}^{\beta}\right) \tag{5.8}$$

To train the SDC model, I calculate the nearest neighbors of each node in the embedding space as supervision information. According to the homophily property assumption, a node and one of its nearest neighbors always have the same label. And to better reveal the node relationships in a graph, a comprehensive similarity (*com_sim*) measure criterion is proposed which is consist of the attribute embedding similarity and the structural similarity.

The attribute embedding similarity applies cosine similarity as the criterion and the structural similarity leverages the Personalized PageRank (PPR) as the criterion. Let

$$\cos\_\mathrm{sim}(\vec{h}_i, \vec{h}_j) = \frac{\vec{h}_i^{\top}\vec{h}_j}{\|\vec{h}_i\|\|\vec{h}_j\|} \tag{5.9}$$

denote the dot product between ℓ_2 normalized graph node embeddings $\vec{h}_i$ and $\vec{h}_j$ (cosine similarity), and the graph similarity computed by PPR is formulated as:

$$\mathbf{S}^{\mathrm{PPR}} = \gamma\left(\mathbf{I}_n - (1-\gamma)\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)^{-1} \tag{5.10}$$

where γ is the teleport probability. If I use $\mathbf{S}^{\cos}$ denote the similarity matrix of the graph node embeddings. Then, the comprehensive similarity is formulated as:

$$\mathbf{S}^{\mathrm{com}} = \mathbf{S}^{\cos} + \mathbf{S}^{\mathrm{PPR}} \tag{5.11}$$

Then, according to the comprehensive similarity matrix $\mathbf{S}^{\mathrm{com}}$ found k-nearest neighbors (KNN) for each node and regard them as the supervision information to train the projection head in a self-supervised manner.

After constructing KNNs for each $\vec{h}_i$, that I use $\vec{h}_i^k$ to denote the k neighbors where $K = \{1, \cdots, k\}$. I plan to learn a clustering function $f_{\psi}()$, where ψ is the parameters of the clustering model. The clustering function $f_{\psi}()$ ends with a Softmax to obtain a soft assignment over the clusters $\mathcal{C} = \{1, \cdots, c\}$, with $f_{\psi}(\vec{h}_i) \in [0,1]^c$. Let $f_{\psi}^c(\vec{h}_i)$ denote the probability of node sample $\vec{h}_i$ being assigned to the cluster c. I train the SDC model by minimizing the following objective function:

$$\begin{aligned} \mathcal{L}_3 &= -\frac{1}{n}\sum_{\vec{h}_i \in \mathcal{V}} \sum_{\vec{h}_i^k \in \mathcal{N}(\vec{h}_i)} \log\left(f_{\psi}(\vec{h}_i) \cdot f_{\psi}(\vec{h}_i^k)\right) + \lambda \sum_{c \in \mathcal{C}} f_{\psi}'^c() \log f_{\psi}'^c() \\ &\text{where } f_{\psi}'^c() = \frac{1}{n}\sum_{\vec{h}_i \in \mathcal{V}} f_{\psi}^c(\vec{h}_i) \end{aligned} \tag{5.12}$$

The first term in Eq. 5.12 aims to make a consistent prediction for $\vec{h}_i$ and one of its KNNs sample $\vec{h}_i^k$, which is randomly selected from $\mathcal{N}\left(\vec{h}_i\right)$. Here, the dot product is used to implement the consistent of two predictions, that the dot product will be maximal

when the prediction $f_\psi(\vec{h}_i)$ and $f_\psi(\vec{h}_i^k)$ are one-hot and get the same prediction. To avoid assigning all samples to a single cluster, an entropy term [41] is used, that is, the second term in Eq. 5.12. The term "entropy" originates from thermodynamics but has been widely adopted in various fields, including information theory, statistics, and machine learning, with different interpretations in each context. Entropy generally represents the uncertainty or randomness associated with a random variable or a probability distribution. Here, the entropy term is used to restrain the predictions uniformly across the clusters $\mathcal{C}$, and λ is the weight of this term.

5.3 Experiments

This section presents the experiments to evaluate the proposed SECRET model. First, I briefly introduce the experimental setup and elaborate on the process and results. Then, an ablation study is presented to assess the effectiveness of different parts of SECRET and give an in-depth analysis. Finally, a sensitivity analysis on critical hyperparameters is conducted.

5.3.1 Experimental Setup

Datasets

SECRET was evaluated on four benchmark datasets. These four datasets are extensively used, so comparing the proposed method with other benchmark methods is convenient. They are Cora [1], Citesser [2], Pubmed [3] and Wiki [4]. Cora, Citeseer, and Pubmed [71] are citation networks where nodes (papers) are connected if one paper cites another. Wiki [197] is a webpage network where nodes represent webpages, and a link indicates that two pages are connected. Statistical information about the datasets is provided in Table 5.1.

Table 5.1 Summary of the datasets used in the experiments.

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,312	4,732	3,703	6
Pubmed	19,717	44,338	500	3
Wiki	2,405	17,981	4,973	17

Baselines and Evaluation Metrics.

SECRET is compared with three types of clustering methods. 1) Methods that only use node features: K-means. 2) Structural clustering methods that only use graph structures:

[1] https://linqs-data.soe.ucsc.edu/public/lbc/cora.tgz
[2] https://linqs-data.soe.ucsc.edu/public/lbc/citeseer.tgz
[3] https://linqs-data.soe.ucsc.edu/public/Pubmed-Diabetes.tgz
[4] https://github.com/thunlp/TADW/tree/master/wiki

Spectral Clustering, which takes the node adjacency matrix as the similarity matrix. DeepWalk [131] utilizes local information obtained from random walks to learn representations by treating walks as the equivalent of sentences. Instead of using sampling-based random walks for generating linear sequences, Deep Neural Networks for Graph Representations (DNGR) [12] adopts a random surfing model to capture graph structural information. 3) Attributed graph clustering methods exploit both graph structures and node features. There are two kinds of methods in this category. The two-step methods include GAE, VGAE [71], ARGAE, ARVGAE [127] and AGC [197]. The end-to-end methods include DAEGC [161], SDCN [7] and GMM-VGAE [60].

Three widely used clustering evaluation metrics are applied [161] i.e., Accuracy (ACC), Normalized Mutual Information (NMI), and Adjusted Rand Index (ARI) to evaluate the clustering performance.

Implementation Details

SECRET is implemented as follows. A two-layer GCN [72] is employed as the encoder for graph representation learning. In the representation learning stage, similar to DGI [156], a Graph Structure Contrastive method is utilized to obtain node representations that capture both the local and globally relevant information of the graph. Representation learning is performed until the contrastive loss stops decreasing.

In the clustering stage, the k-nearest neighbors are identified in the learned comprehensive similarity matrix. The KNNs can act as supervision information for the clustering model. For each node $\vec{h}_i$, a random neighbor $\vec{h}_i^k$ in KNNs is sampled, then Eq. 5.12 is employed as the loss function to make consistent predictions for $\vec{h}_i$ and $\vec{h}_i^k$. The clustering model is performed until the SDC loss stops decreasing.

Parameter Settings.

All the network weights are updated through Adam [69]. In the first stage, the encoders are set with a 512-neuron embedding layer, and the representation learning model is trained for a maximum of 2000 epochs. In the second stage, the clustering task is performed for a maximum of 2000 epochs. I summarize all hyperparameter configurations in Table 5.2.

For the baselines, the settings are those from the corresponding papers. For those methods utilizing K-means, I randomly initialized the centroids twenty times and selected the best result for comparison.

Table 5.2 Hyperparameter Configurations

Dataset	Representation Learning			Clustering			
	p_{e1}	p_{n1}	lr	p_{e2}	p_{n2}	lr	λ
Cora	0.1	0.2	0.0005	0.4	0.1	0.001	2.0
Citeseer	0.1	0.3	0.0001	0.3	0.5	0.001	3.0
Pubmed	0.0	0.2	0.0005	0.3	0.3	0.0005	2.0
Wiki	0.3	0.4	0.0005	0.0	0.5	0.0005	3.0

5.3.2 Experimental Results

Each method was run ten times on four benchmark datasets. The average results are listed in Table 5.3, where bold values indicate the best performance. There are three types of input: feature information only, structure information only, or both. As the experiment setup is the same as in the AGC paper [197], a part of the baseline results are taken from that paper, though some of them lack ARI results. For other baselines without published results, the source codes of the corresponding original papers were run on the datasets (except GMM-VGAE [60] whose source is unavailable).

It can be observed in Table 5.3 that the methods using both feature and structure information of the graph usually outperform those using only one type of information. This observation demonstrates that both the feature and structure information are essential for graph clustering. The proposed SECRET model clearly outperforms existing attributed graph clustering methods across most of the benchmark datasets. On the Cora and Citeseer dataset, for example, SECRET shows a relative increase of 17.8% and 14.8% w.r.t. ARI against GMM-VGAE and 26.8% and 16.8% against AGC. The reasons for this performance gain are that (1) compared to the end-to-end method GMM-VGAE, SECRET can extract high-level semantically meaningful features; (2) compared to the two-step method AGC, which uses spectral clustering to obtain a cluster partition, SECRET utilizes KNNs to train a classifier to get a cluster assignment which can better mine the structure information. However, SECRET is only comparable to GMM-VGAE on Pubmed. This is probably because Pubmed is more sparse than other datasets, and the structure information is limited. Since SECRET puts more emphasis on structure information than other methods, the performance of SECRET on Pubmed is not as good as on other datasets.

Then, I will analyze the complexity of the proposed SECRET. Compared to other methods like DAEGC [161] and GMM-VGAE [60]. SECRET calculates a comprehensive similarity matrix as supervision information. The comprehensive similarity matrix combines attribute embedding similarity and structural similarity, and their time complexity are all $\mathcal{O}(n)$. So, no extra complexity is introduced compared to DAEGC and GMM-VGAE.

5.3.3 Ablation Study

This section reports results from the ablation study to evaluate the effectiveness of different parts of the proposed method on three citation datasets, that are Cora, Citesser, and Pubmed. The ablation study evaluates two parts: the representation part and the clustering part.

Representation

The effect of using different graph augmentations on graph contrastive learning (GCL) [156] was also studied. GCL aims to maximize the mutual information between each node embedding (local) and the global embedding of the whole graph (global). In particular, three different cases were considered: 1) GCL without graph augmentation (WOA); 2) GCL with uniform random graph augmentation (URA); 3) GCL with adaptive graph

Table 5.3 Average clustering performance (%) for the present models on Cora, Citeseer and Pubmed datasets.

METHOD	INPUT	Cora			Citeseer			Pubmed			Wiki		
		ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
K-means[103]	Feature	35.9	17.2	10.3	43.3	20.8	16.2	59.5	31.1	28.0	29.2	28.6	6.1
Spectral[189]	Graph	39.8	29.7	17.4	30.8	9.0	8.2	49.6	14.7	9.8	31.9	35.5	7.5
DeepWalk [131]	Graph	52.9	38.4	29.1	39.0	13.1	13.7	64.7	23.8	25.5	38.5	32.4	-
DNGR [12]	Graph	41.9	31.8	14.2	32.6	18.0	4.3	46.8	15.3	5.9	37.6	35.9	-
GAE [71]	Both	61.3	44.4	38.1	48.2	22.7	19.2	64.2	22.5	22.1	17.3	11.9	-
VGAE [71]	Both	64.7	43.4	37.5	51.9	24.9	23.8	69.9	28.6	31.7	28.7	30.3	-
ARGAE [127]	Both	64.0	44.9	35.2	57.3	35.0	34.1	68.1	27.6	29.1	41.4	39.5	-
ARVGAE [127]	Both	63.8	45.4	40.1	54.4	26.1	24.5	63.5	23.2	22.5	41.6	40.0	-
AGC [197]	Both	68.9	53.7	44.8	67.0	41.2	41.7	69.8	31.6	31.1	47.7	45.3	15.1
DAEGC [161]	Both	70.4	52.8	49.6	67.2	39.7	41.0	67.1	26.6	27.8	43.5	41.2	14.3
SDCN [7]	Both	58.2	42.6	32.8	66.3	39.0	40.6	56.2	15.7	13.1	32.0	26.0	13.0
GMM-VGAE [60]	Both	71.5	54.4	48.2	67.4	42.3	42.4	**71.0**	30.3	**33.0**	-	-	-
SECRET	Both	**75.9**	**59.2**	**56.8**	**72.0**	**47.3**	**48.7**	68.6	**32.7**	30.9	**51.3**	**48.4**	**32.9**

augmentation [206] (AGA). To compare the representation performance of these methods, K-means were applied to the learned embeddings. Table 5.4 shows the clustering results with different augmentation methods. The results indicate that the designed AGA method outperforms WOA and URA, which proved the effectiveness of the proposed augmentation method.

Table 5.4 Clustering results with K-means on different augmentation methods which are 1) GCL without augmentation (WOA). 2) GCL with uniform random augmentation (URA). 3) GCL with adaptive graph augmentation (AGA).

METHOD	Cora			Citeseer			Pubmed		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
WOA	70.9±0.7	55.8±0.6	50.1±1.2	68.1±0.5	43.4±0.7	44.0±0.7	64.2±0.1	25.4±0.3	23.4±0.2
URA	71.0±0.4	56.1±0.3	51.2±0.4	68.4±0.1	43.4±0.1	44.4±0.2	64.2±0.2	25.9±0.3	23.8±0.3
AGA	**72.9±0.6**	**56.9±0.2**	**52.6±0.6**	**69.3±0.3**	**44.0±0.4**	**45.2±0.5**	**65.7±0.3**	**28.3±0.6**	**26.3±0.6**

Clustering

The effect of different components in the SDC loss was also assessed. Eq. 5.12 is the loss function of the designed SDC model which has two terms i.e., the consistent prediction term (CPT) and the entropy term (ET). The CPT is used to enforce coherent predictions between a sample and its KNNs and the ET is for avoiding the cluster degeneracy problem, which ensures that predictions are uniform across the clusters $\mathcal{C}$. To evaluate the importance of each term in the SDC loss, an ablation study experiment was designed. The clustering results are summarized in Table 5.5. It can be seen that: 1) using CPT as the loss function

only leads to cluster degeneracy, i.e., all samples being assigned to a single cluster, resulting in poor performance; 2) using ET as the loss function only also leads to bad results, as it only restricts the predictions uniformly across the clusters; 3) using both CPT and ET yields the best results.

The effect of the proposed comprehensive similarity (COS) for calculating KNNs was also studied by comparing it with the commonly used graph node embeddings similarity (NES). The clustering results are summarized in table 5.6. It can be seen that: 1) using NES KNNs as supervision information to train the clustering model cannot provide the best performance, as it fails to reflect the relatively global graph structure; 2) using COS KNNs as supervision information to train the clustering model yield the best clustering performance because it considers information not only from the global feature but also from the relatively global structure.

Table 5.5 The ablation study of the SECRET clustering loss which consists of two terms the consistent prediction term (CPT) and the entropy term (ET).

Model	Terms		Cora			Citeseer			Pubmed		
	CPT	ET	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
1	✓	✗	13.0±0.0	0.0±0.0	0.0±0.0	7.9±0.0	0.0±0.0	0.0±0.0	20.8±0.0	0.0±0.0	0.0±0.0
2	✗	✓	41.1±3.1	24.5±1.6	17.6±2.8	23.2±0.7	3.4±0.2	0.2±0.1	42.6±0.3	2.9±0.3	3.3±0.2
3	✓	✓	**74.3±0.1**	**56.6±0.2**	**52.8±0.3**	**70.9±0.0**	**45.5±0.1**	**47.2±0.1**	**67.9±0.1**	**31.2±0.2**	**29.3±0.2**

Table 5.6 The clustering results are based on different similarity measures to learn KNNs: 1) graph node embeddings similarity (NES). 2) comprehensive similarity (COS).

Structures	Cora			Citeseer			Pubmed		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
NES	73.9±0.2	56.2±0.1	51.7±0.3	70.4±0.1	45.1±0.1	46.6±0.1	66.8±0.1	29.2±0.3	27.2±0.2
COS	**74.3±0.1**	**56.6±0.2**	**52.8±0.3**	**70.9±0.0**	**45.5±0.1**	**47.2±0.1**	**67.9±0.1**	**31.2±0.2**	**29.3±0.2**

5.3.4 Sensitivity Analysis

Lastly, sensitivity analysis was performed for the critical hyperparameters in SECRET, namely two probabilities p_e and p_n, which determine the level of graph augmentation on edges and nodes for the two steps, i.e., Graph Structure Contrastive Representation Learning (GSC-RL) and Self-supervised Deep-learning-based Clustering (SDC). Attributed graph clustering was conducted while varying these parameters from 0 to 1, with other parameters unchanged.

The GSC-RL results on the Cora dataset are shown in Figure 5.2. It can be observed that the performance of GSC-RL evaluated by quantifying the degree to which the mined nearest neighbors are the instances of the same semantic cluster, yields the best results under low probability ($p_e < 0.5$, $p_n < 0.5$). The SDC results on the Cora dataset are shown

in Figure 5.3. It was found that the performance of clustering in terms of accuracy was the best when the parameters were close to 0.5 and were relatively stable.

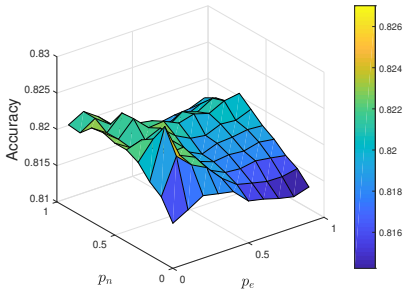

Fig. 5.2 The performance of GSC-RL with varied hyperparameters p_e and p_n on the Cora dataset in terms of the accuracy of KNN (percentage of correct pairs %).

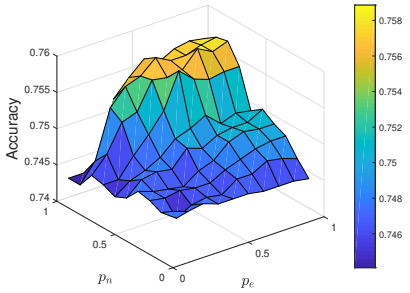

Fig. 5.3 The performance of SDC with varied hyperparameters p_e and p_n on the Cora dataset in terms of node clustering accuracy (%).

5.4 Conclusion

Although the DGC methods have made remarkable progress, I observe that there are two drawbacks to the existing methods. 1) In the node encoding process, existing methods usually overlook the learning of structural information. Consequently, the discriminative capability of representations will be limited. 2) Most of the existing methods leverage traditional clustering methods, i.e., K-means. Thus the clustering process can not benefit from the strong graph representation ability of GNNs, which leads to sub-optimal clustering performance. To address these issues, I propose a novel self-supervised DGC method termed **S**tructural S**e**mantic **C**ont**r**astive D**e**ep Graph Clus**t**ering (SECRET). To get a more discriminative representation, I design a structure contrastive scheme (SCS) by contrasting the aggregation of first-order neighbors and a graph diffusion, and I propose a consistent loss to keep the structure of different views consistent. To benefit from the strong graph representation ability of GNNs, I proposed a novel Self-supervised Deep-learning-based Clustering (SDC) model, which jointly optimizes the cluster head and the representation networks through the gradient descent algorithms. To better reveal the node relationships in a graph, I also proposed a comprehensive similarity measure

criterion which is calculated by considering both the attribute embedding similarity and the structural similarity. An experimental evaluation of four commonly used benchmark datasets shows that the proposed SECRET clearly outperforms the state-of-the-art methods.

Chapter 6

Conclusions and Future Works

This thesis has investigated the importance of global structural and semantic information for Deep Graph Representation Learning. The primary argument is that for graph data, both the node feature and structural information, including the local and global structures, are important for graph representation learning.

6.1 Conclusion

Graph Neural Networks (GNNs) as a Deep Graph Representation Learning (DGL) method have attracted more and more attention for their convincing performance. DGL has indeed made significant progress in recent years. However, there are still several crucial challenges that the field faces, including (semi-)supervised DGL, self-supervised DGL, and DGL-based graph clustering. In this thesis, I proposed three models to address previous issues, respectively, which demonstrate a significant outperformance to the state-of-the-art methods.

GNNs have been widely used in (semi-)supervised DGL problems. However, GNNs suffer from over-smoothing and over-squashing problems. To solve this problem, a Hierarchical Structure Graph Transformer called HighFormer is proposed to leverage local and relatively global structure information. I use GNN to learn the initial graph node representation based on the local structure information. At the same time, a structural attention module is used to learn the relatively global structural similarity. Then, I added the softmax attention matrix and the relatively global structure similarity matrix to form an improved attention matrix.

The existing graph contrastive learning (GCL) methods often neglect the semantic information on augmentation schemes and contrastive frames. Therefore, a novel graph contrastive learning method with semantic invariance graph augmentation termed SemiGCL is proposed by designing a semantic invariance graph augmentation (SemiAug) and a semantic-based graph contrastive (SGC) framework. First, a density-based approach is proposed to find the potential semantic centers of the graph and then divide the graph nodes into different semantic clusters according to the distance (structure and feature) to the semantic centers. Then, I generate the augmented graphs on the structure and

attribute levels according to the learned semantic cluster assignment. At last, I design a semantic-based graph contrastive (SGC) framework with a semantic debiasing negative sampling (SDNS) that selects negative samples from other clusters except for the positive sample's cluster.

Although the existing deep graph clustering (DGC) methods have made remarkable progress, I observe that there are two drawbacks to the existing methods. 1) existing methods usually overlook learning structural information in the node encoding process. Consequently, the discriminative capability of representations will be limited. 2) Most existing methods leverage traditional clustering methods, e.g., K-means. Thus, the clustering process can not benefit from the strong graph representation ability of GNNs, which leads to sub-optimal clustering performance. To address these issues, I propose a novel self-supervised DGC method termed **S**tructural S**e**mantic **C**ont**r**astive D**e**ep Graph Clus**t**ering (SECRET). To get a more discriminative representation, I design a structure contrastive scheme (SCS) by contrasting the aggregation of first-order neighbors with a graph diffusion. I propose a consistent loss to keep the structure of different views consistent. To benefit from the strong graph representation ability of GNNs, I proposed a novel Self-supervised The deep-learning-based clustering (SDC) model jointly optimizes the cluster head and the representation networks through the gradient descent algorithms.

6.2 Future works

Some potential future works for Deep Graph Representation Learning and Deep Graph Clustering could include the following:

- Graph Transformer models have shown promising performance in various tasks involving graph-structured data. Graph Transformer architectures are inspired by the Transformer model originally introduced for natural language processing(NLP) tasks but adapted to handle graph data. For example, in chapter 3, I have proposed a new graph Transformer that can learn more information than the GNN module because the attention mechanism used in the Transformer architecture can capture long-range information. How to incorporate the graph information into the Transformer architecture is still an open issue. Specifically, for NLP, positional encoding is a technique used in transformer models to inject positional information into the input embeddings of tokens in a sequence. Since transformers do not inherently understand the order of tokens in a sequence, positional encoding helps the model learn to capture sequential relationships and understand the relative positions of tokens within the input sequence. However, the graph structure is not sequence data, so it's hard to encode it into positional encoding. In future work, I will try to design a more proper and efficient way to encode the graph structure.

- Graph contrastive learning has gained significant attention and popularity in recent years as a powerful approach for learning representations of graph-structured data. This method is inspired by contrastive learning techniques in computer vision and

natural language processing and has been adapted to handle graph data. In chapter 4 and chapter 5, I have proposed two new contrastive learning models with promising performance. However, recent graph contrastive learning methods still suffer from sample selection bias. The effectiveness of contrastive learning methods heavily depends on the quality and diversity of the negative examples used during training. Biased or poorly chosen negative examples can lead to suboptimal representations and degraded performance. In future work, I will explore more accurate negative sampling methods to improve the representation performance.

- Joint learning of graph representation and clustering methods is a promising approach to effectively capture the underlying structure of graph data and perform clustering simultaneously. The model can effectively leverage both the intrinsic structure of the graph data and the clustering objectives to produce informative and meaningful cluster assignments. This approach can potentially improve clustering performance, particularly in scenarios where the underlying structure of the data is complex or high-dimensional. In chapter 5, I have proposed a novel contrastive deep graph clustering method by joint training a structure contrastive scheme (SCS) and a new self-supervised deep graph clustering method with a promising clustering performance. However, the computational complexity of the proposed method is relatively high. I will explore a more efficient and effective method to learn graph representation and clustering objectives jointly in future work.

- The advancement of graph representation learning has led to many promising applications across various domains. Graph representation learning in AI pharmaceuticals is an emerging field that leverages techniques from GNNs and other machine learning methods to model and analyze molecular structures, biological interactions, chemical compounds, and other graph-structured data relevant to drug discovery and development. Despite the significant research efforts in AI pharmaceuticals, several challenges persist. For example, model interpretability and explainability: Many AI models used in pharmaceuticals, such as deep learning models, are often considered "black boxes" due to their complex architectures and high-dimensional representations. Understanding and interpreting the decision-making process of these models is crucial for gaining insights into drug mechanisms, safety, and efficacy. In future work, I will explore interpretable AI pharmaceuticals models to improve the safety and efficacy of drug designed by AI.

References

[1] Alon, U. and Yahav, E. (2020). On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*.

[2] Atwood, J. and Towsley, D. (2016). Diffusion-convolutional neural networks. *Advances in neural information processing systems*, 29.

[3] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[4] Bacciu, D., Errica, F., and Micheli, A. (2018). Contextual graph markov model: A deep and generative approach to graph processing. In *International Conference on Machine Learning*, pages 294–303. PMLR.

[5] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[6] Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396.

[7] Bo, D., Wang, X., Shi, C., Zhu, M., Lu, E., and Cui, P. (2020). Structural deep clustering network. In *Proceedings of The Web Conference 2020*, pages 1400–1410.

[8] Boscaini, D., Masci, J., Rodolà, E., and Bronstein, M. (2016). Learning shape correspondence with anisotropic convolutional neural networks. *Advances in neural information processing systems*, 29.

[9] Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. (2017). Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.

[10] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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

[12] Cao, S., Lu, W., and Xu, Q. (2016). Deep neural networks for learning graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

[13] Chen, S., Varma, R., Sandryhaila, A., and Kovačević, J. (2015). Discrete signal processing on graphs: Sampling theory<? pub _newline=""? *IEEE transactions on signal processing*, 63(24):6510–6523.

[14] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

[15] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[16] Chuang, C.-Y., Robinson, J., Lin, Y.-C., Torralba, A., and Jegelka, S. (2020). Debiased contrastive learning. *Advances in neural information processing systems*, 33:8765–8775.

[17] Cover, T. M. (1999). *Elements of information theory*. John Wiley & Sons.

[18] Dai, H., Kozareva, Z., Dai, B., Smola, A., and Song, L. (2018). Learning steady-states of iterative algorithms over graphs. In *International conference on machine learning*, pages 1106–1114. PMLR.

[19] Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29.

[20] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[21] Ding, K., Xu, Z., Tong, H., and Liu, H. (2022). Data augmentation for deep graph learning: A survey. *ACM SIGKDD Explorations Newsletter*, 24(2):61–77.

[22] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[23] Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28.

[24] Dwivedi, V. P. and Bresson, X. (2020). A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*.

[25] Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. (2020). Benchmarking graph neural networks.

[26] Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. (2021). Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875*.

[27] Ericsson, L., Gouk, H., Loy, C. C., and Hospedales, T. M. (2022). Self-supervised representation learning: Introduction, advances, and challenges. *IEEE Signal Processing Magazine*, 39(3):42–62.

[28] Errica, F., Podda, M., Bacciu, D., and Micheli, A. (2019). A fair comparison of graph neural networks for graph classification. *arXiv preprint arXiv:1912.09893*.

[29] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press.

[30] Feng, F., He, X., Tang, J., and Chua, T.-S. (2019). Graph adversarial training: Dynamically regularizing based on graph structure. *IEEE Transactions on Knowledge and Data Engineering*, 33(6):2493–2504.

[31] Feng, W., Zhang, J., Dong, Y., Han, Y., Luan, H., Xu, Q., Yang, Q., Kharlamov, E., and Tang, J. (2020). Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33:22092–22103.

[32] Fey, M., Lenssen, J. E., Weichert, F., and Müller, H. (2018). Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 869–877.

[33] Fout, A., Byrd, J., Shariat, B., and Ben-Hur, A. (2017). Protein interface prediction using graph convolutional networks. In *Advances in Neural Information Processing Systems*, volume 30, pages 6530–6539.

[34] Gallicchio, C. and Micheli, A. (2010). Graph echo state networks. In *The 2010 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE.

[35] Gao, H., Wang, Z., and Ji, S. (2018). Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1416–1424.

[36] Gasteiger, J., Weißenberger, S., and Günnemann, S. (2019). Diffusion improves graph learning. *Advances in neural information processing systems*, 32.

[37] Gehring, J., Auli, M., Grangier, D., and Dauphin, Y. N. (2016). A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611.02344*.

[38] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR.

[39] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, volume 27, pages 2672–2680.

[40] Gori, M., Monfardini, G., and Scarselli, F. (2005). A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE.

[41] Grandvalet, Y. and Bengio, Y. (2006). Entropy regularization. *Semi-Supervised Learning*, pages 151–168.

[42] Graves, A. and Graves, A. (2012). Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45.

[43] Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284.

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

[45] Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., et al. (2020). Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*.

[46] Guo, S., Lin, Y., Feng, N., Song, C., and Wan, H. (2019). Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 922–929.

[47] Guo, T., Pan, S., Zhu, X., and Zhang, C. (2018). Cfond: consensus factorization for co-clustering networked data. *IEEE Transactions on Knowledge and Data Engineering*, 31(4):706–719.

[48] Hagen, L. and Kahng, A. B. (1992). New spectral methods for ratio cut partitioning and clustering. *IEEE transactions on computer-aided design of integrated circuits and systems*, 11(9):1074–1085.

[49] Hamilton, W. L. (2020). Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159.

[50] Hamilton, W. L., Ying, R., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035.

[51] Hassani, K. and Khasahmadi, A. H. (2020). Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pages 4116–4126. PMLR.

[52] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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

[54] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97.

[55] Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. (2019). Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*.

[56] Horn, R. A. (1990). The hadamard product. In *Proc. Symp. Appl. Math*, volume 40, pages 87–169.

[57] Hu, F., Zhu, Y., Wu, S., Wang, L., and Tan, T. (2019a). Hierarchical graph convolutional networks for semi-supervised node classification. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16*.

[58] Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. (2020). Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133.

[59] Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. (2019b). Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*.

[60] Hui, B., Zhu, P., and Hu, Q. (2020). Collaborative graph convolutional networks: Unsupervised learning meets semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4215–4222.

[61] Hussain, M. S., Zaki, M. J., and Subramanian, D. (2021). Edge-augmented graph transformers: Global self-attention is enough for graphs. *arXiv preprint arXiv:2108.03348*.

[62] Jain, A., Zamir, A. R., Savarese, S., and Saxena, A. (2016). Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 5308–5317.

[63] Ji, S., Pan, S., Cambria, E., Marttinen, P., and Philip, S. Y. (2021). A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2):494–514.

[64] Jin, M., Zheng, Y., Li, Y.-F., Gong, C., Zhou, C., and Pan, S. (2021). Multi-scale contrastive siamese networks for self-supervised graph representation learning. In *International Joint Conference on Artificial Intelligence IJCAI*.

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

[66] Jing, L. and Tian, Y. (2020). Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):4037–4058.

[67] Kearnes, S., McCloskey, K., Berndl, M., Pande, V., and Riley, P. (2016). Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30:595–608.

[68] Khoo, L. M. S., Chieu, H. L., Qian, Z., and Jiang, J. (2020). Interpretable rumor detection in microblogs by attending to user interactions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8783–8790.

[69] Kingma, D. and Ba, J. (2014a). Adam: A method for stochastic optimization. *Computer Science*.

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

[71] Kipf, T. N. and Welling, M. (2016). Variational graph auto-encoders. In *NIPS Workshop on Bayesian Deep Learning*.

[72] Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

[73] Kondor, R. I. and Lafferty, J. (2002). Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th international conference on machine learning*, volume 2002, pages 315–322.

[74] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

[75] LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

[76] Lee, J., Lee, I., and Kang, J. (2019). Self-attention graph pooling. In *International conference on machine learning*, pages 3734–3743. PMLR.

[77] Lee, N., Lee, J., and Park, C. (2022). Augmentation-free self-supervised learning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7372–7380.

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

[79] Li, G., Xiong, C., Thabet, A., and Ghanem, B. (2020a). Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*.

[80] Li, P., Wang, Y., Wang, H., and Leskovec, J. (2020b). Distance encoding–design provably more powerful gnns for structural representation learning. *arXiv preprint arXiv:2009.00142*.

[81] Li, Q., Han, Z., and Wu, X.-M. (2018a). Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*.

[82] Li, R., Wang, S., Zhu, F., and Huang, J. (2018b). Adaptive graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

[83] Li, Y., Sha, C., Huang, X., and Zhang, Y. (2018c). Community detection in attributed graphs: An embedding approach. In *Thirty-second AAAI conference on artificial intelligence*, pages 338–345.

[84] Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. (2015). Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

[85] Li, Y., Yu, R., Shahabi, C., and Liu, Y. (2017). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.

[86] Li, Z., Shen, X., Jiao, Y., Pan, X., Zou, P., Meng, X., Yao, C., and Bu, J. (2020c). Hierarchical bipartite graph neural networks: Towards large-scale e-commerce applications. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1677–1688. IEEE.

[87] Lin, K., Wang, L., and Liu, Z. (2021). Mesh graphormer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12939–12948.

[88] Liu, L., Xu, L., Wangy, Z., and Chen, E. (2015). Community detection based on structure and content: A content propagation perspective. In *2015 IEEE international conference on data mining*, pages 271–280. IEEE.

[89] Liu, M., Wang, Z., and Ji, S. (2021a). Non-local graph neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 44(12):10270–10276.

[90] Liu, Q., Allamanis, M., Brockschmidt, M., and Gaunt, A. (2018). Constrained graph variational autoencoders for molecule design. *Advances in neural information processing systems*, 31.

[91] Liu, S., Ying, R., Dong, H., Li, L., Xu, T., Rong, Y., Zhao, P., Huang, J., and Wu, D. (2022a). Local augmentation for graph neural networks. In *International Conference on Machine Learning*, pages 14054–14072. PMLR.

[92] Liu, Y., Jin, M., Pan, S., Zhou, C., Zheng, Y., Xia, F., and Yu, P. (2022b). Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*.

[93] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019a). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[94] Liu, Y., Tu, W., Zhou, S., Liu, X., Song, L., Yang, X., and Zhu, E. (2022c). Deep graph clustering via dual correlation reduction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7603–7611.

[95] Liu, Y., Yang, X., Zhou, S., Liu, X., Wang, Z., Liang, K., Tu, W., Li, L., Duan, J., and Chen, C. (2022d). Hard sample aware network for contrastive deep graph clustering. *arXiv preprint arXiv:2212.08665*.

[96] Liu, Z., Chen, C., Li, L., Zhou, J., Li, X., Song, L., and Qi, Y. (2019b). Geniepath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4424–4431.

[97] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021b). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022.

[98] Loshchilov, I. and Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

[99] Lukovnikov, D. and Fischer, A. (2021). Improving breadth-wise backpropagation in graph neural networks helps learning long-range dependencies. In *International Conference on Machine Learning*, pages 7180–7191. PMLR.

[100] Luo, D., Cheng, W., Yu, W., Zong, B., Ni, J., Chen, H., and Zhang, X. (2021). Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 779–787.

[101] Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

[102] Lv, M., Hong, Z., Chen, L., Chen, T., Zhu, T., and Ji, S. (2020). Temporal multi-graph convolutional network for traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, 22(6):3337–3348.

[103] MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

[104] Magnus, J. R. (1998). Handbook of matrices: H. lütkepohl, john wiley and sons, 1996. *Econometric Theory*, 14(3):379–380.

[105] Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. (2019). Provably powerful graph networks. *Advances in neural information processing systems*, 32.

[106] Masci, J., Boscaini, D., Bronstein, M., and Vandergheynst, P. (2015). Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45.

[107] Mernyei, P. and Cangea, C. (2020). Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*.

[108] Mialon, G., Chen, D., Selosse, M., and Mairal, J. (2021). Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*.

[109] Micheli, A. (2009). Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511.

[110] Micheli, A., Sona, D., and Sperduti, A. (2004). Contextual processing of structured data by recursive cascade correlation. *IEEE Transactions on Neural Networks*, 15(6):1396–1410.

[111] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[112] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

[113] Min, E., Chen, R., Bian, Y., Xu, T., Zhao, K., Huang, W., Zhao, P., Huang, J., Ananiadou, S., and Rong, Y. (2022a). Transformer for graphs: An overview from architecture perspective. *arXiv preprint arXiv:2202.08455*.

[114] Min, E., Rong, Y., Xu, T., Bian, Y., Zhao, P., Huang, J., Luo, D., Lin, K., and Ananiadou, S. (2022b). Masked transformer for neighhourhood-aware click-through rate prediction. *arXiv preprint arXiv:2201.13311*.

[115] Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124.

[116] Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. (2019). Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609.

[117] Murphy, R., Srinivasan, B., Rao, V., and Ribeiro, B. (2019). Relational pooling for graph representations. In *International Conference on Machine Learning*, pages 4663–4673. PMLR.

[118] Newman, M. E. (2006). Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104.

[119] Niepert, M., Ahmed, M., and Kutzkov, K. (2016). Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023. PMLR.

[120] Nikolentzos, G., Meladianos, P., and Vazirgiannis, M. (2017). Matching node embeddings for graph similarity. In *Thirty-first AAAI conference on artificial intelligence*.

[121] Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI*, pages 69–84. Springer.

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

[123] Opolka, F. L., Solomon, A., Cangea, C., Veličković, P., Liò, P., and Hjelm, R. D. (2019). Spatio-temporal deep graph infomax. *arXiv preprint arXiv:1904.06316*.

[124] Opsahl, T., Agneessens, F., and Skvoretz, J. (2010). Node centrality in weighted networks: Generalizing degree and shortest paths. *Social networks*, 32(3):245–251.

[125] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

[126] Pan, S., Hu, R., Fung, S.-f., Long, G., Jiang, J., and Zhang, C. (2019). Learning graph embedding with adversarial training methods. *IEEE transactions on cybernetics*, 50(6):2475–2487.

[127] Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., and Zhang, C. (2018). Adversarially regularized graph autoencoder for graph embedding. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI*.

[128] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. (2016). Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544.

[129] Peng, Z., Huang, W., Luo, M., Zheng, Q., Rong, Y., Xu, T., and Huang, J. (2020). Graph representation learning via graphical mutual information maximization. In *Proceedings of The Web Conference 2020*, pages 259–270.

[130] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[131] Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710.

[132] Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., and Tang, J. (2020). Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1150–1160.

[133] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

[134] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

[135] Rodriguez, A. and Laio, A. (2014). Clustering by fast search and find of density peaks. *science*, 344(6191):1492–1496.

[136] Rong, Y., Bian, Y., Xu, T., Xie, W., Wei, Y., Huang, W., and Huang, J. (2020a). Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems*, 33:12559–12571.

[137] Rong, Y., Huang, W., Xu, T., and Huang, J. (2020b). Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*.

[138] Sandryhaila, A. and Moura, J. M. (2013). Discrete signal processing on graphs. *IEEE transactions on signal processing*, 61(7):1644–1656.

[139] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.

[140] Schütt, K. T., Arbabzadah, F., Chmiela, S., Müller, K. R., and Tkatchenko, A. (2017). Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8(1):13890.

[141] Sennrich, R., Haddow, B., and Birch, A. (2015). Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

[142] Seo, Y., Defferrard, M., Vandergheynst, P., and Bresson, X. (2018). Structured sequence modeling with graph convolutional recurrent networks. In *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I 25*, pages 362–373. Springer.

[143] Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. (2018). Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

[144] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905.

[145] Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98.

[146] Sperduti, A. and Starita, A. (1997). Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735.

[147] Srinivasan, B. and Ribeiro, B. (2019). On the equivalence between node embeddings and structural graph representations. *arXiv preprint arXiv:1910.00452*.

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

[149] Sun, Q., Li, J., Peng, H., Wu, J., Ning, Y., Yu, P. S., and He, L. (2021). Sugar: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism. In *Proceedings of the Web Conference 2021*, pages 2081–2091.

[150] Thakoor, S., Tallec, C., Azar, M. G., Azabou, M., Dyer, E. L., Munos, R., Veličković, P., and Valko, M. (2021a). Large-scale representation learning on graphs via bootstrapping. *arXiv preprint arXiv:2102.06514*.

[151] Thakoor, S., Tallec, C., Azar, M. G., Munos, R., Veličković, P., and Valko, M. (2021b). Bootstrapped representation learning on graphs. In *ICLR 2021 Workshop on Geometrical and Topological Representation Learning*.

[152] Tian, F., Gao, B., Cui, Q., Chen, E., and Liu, T.-Y. (2014). Learning deep representations for graph clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

[153] Tschannen, M., Djolonga, J., Rubenstein, P. K., Gelly, S., and Lucic, M. (2019). On mutual information maximization for representation learning. *arXiv preprint arXiv:1907.13625*.

[154] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

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

[156] Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. (2019). Deep graph infomax. In *International Conference on Learning Representations*.

[157] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations*.

[158] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A., and Bottou, L. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12).

[159] Vinyals, O., Bengio, S., and Kudlur, M. (2015). Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*.

[160] Wale, N., Watson, I. A., and Karypis, G. (2008). Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14:347–375.

[161] Wang, C., Pan, S., Hu, R., Long, G., Jiang, J., and Zhang, C. (2019). Attributed graph clustering: A deep attentional embedding approach. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16*.

[162] Wang, C., Pan, S., Long, G., Zhu, X., and Jiang, J. (2017a). Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 889–898.

[163] Wang, P., Agarwal, K., Ham, C., Choudhury, S., and Reddy, C. K. (2021). Self-supervised learning of contextual embeddings for link prediction in heterogeneous networks. In *Proceedings of the Web Conference 2021*, pages 2946–2957.

[164] Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., and Yang, S. (2017b). Community preserving network embedding. In *Thirty-first AAAI conference on artificial intelligence*.

[165] Wang, Y., Wang, W., Liang, Y., Cai, Y., Liu, J., and Hooi, B. (2020). Nodeaug: Semi-supervised node classification with data augmentation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 207–217.

[166] Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019a). Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR.

[167] Wu, L., Lin, H., Tan, C., Gao, Z., and Li, S. Z. (2021a). Self-supervised learning on graphs: Contrastive, generative, or predictive. *IEEE Transactions on Knowledge and Data Engineering*.

[168] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

[169] Wu, Z., Jain, P., Wright, M., Mirhoseini, A., Gonzalez, J. E., and Stoica, I. (2021b). Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, 34:13266–13279.

[170] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24.

[171] Wu, Z., Pan, S., Long, G., Jiang, J., and Zhang, C. (2019b). Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16*.

[172] Xiao, X., Jin, Z., Wang, S., Xu, J., Peng, Z., Wang, R., Shao, W., and Hui, Y. (2022). A dual-path dynamic directed graph convolutional network for air quality prediction. *Science of The Total Environment*, 827:154298.

[173] Xie, Y., Xu, Z., Zhang, J., Wang, Z., and Ji, S. (2022). Self-supervised learning of graph neural networks: A unified review. *IEEE transactions on pattern analysis and machine intelligence*.

[174] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018a). How powerful are graph neural networks? In *International Conference on Learning Representations, ICLR*.

[175] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How powerful are graph neural networks? In *International Conference on Learning Representations*.

[176] Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. (2018b). Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462. PMLR.

[177] Yan, S., Xiong, Y., and Lin, D. (2018). Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

[178] Yang, L., Zhang, L., and Yang, W. (2021). Graph adversarial self-supervised learning. *Advances in Neural Information Processing Systems*, 34:14887–14899.

[179] Yang, X., Hu, X., Zhou, S., Liu, X., and Zhu, E. (2022). Interpolation-based contrastive learning for few-label semi-supervised learning. *IEEE Transactions on Neural Networks and Learning Systems*.

[180] Yao, S., Wang, T., and Wan, X. (2020). Heterogeneous graph transformer for graph-to-sequence learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7145–7154.

[181] Yi, H.-C., You, Z.-H., Huang, D.-S., and Kwoh, C. K. (2022). Graph representation learning in bioinformatics: trends, methods and applications. *Briefings in Bioinformatics*, 23(1):bbab340.

[182] Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. (2021). Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888.

[183] You, J., Ying, R., and Leskovec, J. (2019). Position-aware graph neural networks. In *International conference on machine learning*, pages 7134–7143. PMLR.

[184] You, Y., Chen, T., Shen, Y., and Wang, Z. (2021). Graph contrastive learning automated. In *International Conference on Machine Learning*, pages 12121–12132. PMLR.

[185] You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. (2020). Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823.

[186] Yu, B., Yin, H., and Zhu, Z. (2017). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*.

[187] Yue, L., Jun, X., Sihang, Z., Siwei, W., Xifeng, G., Xihong, Y., Ke, L., Wenxuan, T., Wang, L. X., et al. (2022). A survey of deep graph clustering: Taxonomy, challenge, and application. *arXiv preprint arXiv:2211.12875*.

[188] Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J. (2019). Graph transformer networks. *Advances in neural information processing systems*, 32.

[189] Zelnik-Manor, L. and Perona, P. (2004). Self-tuning spectral clustering. *Advances in neural information processing systems*, 17.

[190] Zeng, J. and Xie, P. (2021). Contrastive self-supervised learning for graph classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10824–10832.

[191] Zhang, C., Zhang, H., and Wang, J. (2018a). Personalized restaurant recommendation method combining group correlations and customer preferences. *Information Sciences*, 454:128–143.

[192] Zhang, D., Nan, F., Wei, X., Li, S., Zhu, H., McKeown, K., Nallapati, R., Arnold, A., and Xiang, B. (2021). Supporting clustering with contrastive learning. In *"Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies"*.

[193] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2018b). mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations, ICLR*.

[194] Zhang, J., Shi, X., Xie, J., Ma, H., King, I., and Yeung, D.-Y. (2018c). Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*.

[195] Zhang, J., Zhang, H., Xia, C., and Sun, L. (2020). Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*.

[196] Zhang, M., Cui, Z., Neumann, M., and Chen, Y. (2018d). An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

[197] Zhang, X., Liu, H., Li, Q., and Wu, X.-M. (2019). Attributed graph clustering via adaptive graph convolution. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16*.

[198] Zhao, H., Yang, X., Wang, Z., Yang, E., and Deng, C. (2021a). Graph debiased contrastive learning with joint representation clustering. In *International Joint Conference on Artificial IntelligenceIJCAI*, pages 3434–3440.

[199] Zhao, J., Li, C., Wen, Q., Wang, Y., Liu, Y., Sun, H., Xie, X., and Ye, Y. (2021b). Gophormer: Ego-graph transformer for node classification. *arXiv preprint arXiv:2110.13094*.

[200] Zhao, L. and Akoglu, L. (2019). Pairnorm: Tackling oversmoothing in gnns. *arXiv preprint arXiv:1909.12223*.

[201] Zheng, C., Zong, B., Cheng, W., Song, D., Ni, J., Yu, W., Chen, H., and Wang, W. (2020). Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*, pages 11458–11468. PMLR.

[202] Zhou, F., Cao, C., Zhang, K., Trajcevski, G., Zhong, T., and Geng, J. (2019). Meta-gnn: On few-shot node classification in graph meta-learning. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2357–2360.

[203] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI open*, 1:57–81.

[204] Zhu, Y., Xu, Y., Liu, Q., and Wu, S. (2021a). An empirical study of graph contrastive learning. *arXiv preprint arXiv:2109.01116*.

[205] Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. (2020). Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*.

[206] Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. (2021b). Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, pages 2069–2080.

[207] Zhuang, C. and Ma, Q. (2018). Dual graph convolutional networks for graph-based semi-supervised classification. In *Proceedings of the 2018 world wide web conference*, pages 499–508.