UNIVERSITÉ DU QUÉBEC À MONTRÉAL

PERSPECTIVE GÉOMÉTRIQUE
POUR UNE MEILLEURE COMPRÉHENSION DE L'AUTOSUPERVISION

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

YOUSSEF AMDOUNI

JANVIER 2024

UNIVERSITÉ DU QUÉBEC À MONTRÉAL

TOWARD A BETTER UNDERSTANDING OF SELF SUPERVISION FROM GEOMETRIC PERSPECTIVE

DISSERTATION

PRESENTED

AS PARTIAL REQUIREMENT

TO THE MASTERS IN COMPUTER SCIENCE

BY

YOUSSEF AMDOUNI

JANUARY 2024

ACKNOWLEDGEMENTS

I am grateful for the thesis experience, which provided me with an opportunity for learning and professional growth. I express my heartfelt thanks to those who supported me throughout this project.

I am grateful to Nairouz Mrabah for her guidance, availability, and valuable advice, which were instrumental in the successful completion of my work.

I want to express my sincere appreciation to my colleagues and family for their assistance and encouragement during the development of my project. Their support has been invaluable and has enabled me to overcome the challenges that I faced during my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

RÉSUMÉ

Récemment, l'apprentissage de représentation des graphes est devenu fondamental dans l'analyse des données structurées en graphe. Les travaux précédents ont adopté l'auto-supervision, un paradigme d'apprentissage automatique dans lequel un modèle apprend à partir des données elles-mêmes, sans dépendre d'étiquettes externes, pour extraire des informations sémantiques de haut niveau et des motifs. L'apprentissage auto-supervisé exploite diverses tâches préliminaires, telles que la prédiction des nœuds ou des arêtes manquantes, pour créer des représentations significatives. Ensuite, les chercheurs affinent leurs modèles en utilisant des tâches en aval telles que la classification des nœuds, le regroupement et la prédiction de liens. Cependant, la transition entre les niveaux d'abstraction de l'auto-supervision, qui représentent différentes granularités d'informations dans les données, n'a jamais été étudiée d'un point de vue géométrique. Dans ce mémoire, nous présentons un nouveau cadre d'apprentissage contrastif auto-supervisé conçu pour apprendre des représentations à plusieurs niveaux d'abstraction : au niveaux des nœuds, de la proximité, des clusters et des graphes. En utilisant une stratégie de pré-entraînement suivie d'une étape de raffinement, nous étudions l'évolution de la dimension intrinsèque et de la dimension intrinsèque linéaire lors de la transition entre ces niveaux d'abstraction. Nos résultats expérimentaux démontrent l'existence d'un défi significatif de « Feature Twist » caractérisé par une transformation géométrique abrupte et une détérioration des performances du modèle dans les tâches en aval lors de la transition entre les différents niveaux d'auto-supervision. Pour résoudre ce problème, nous proposons un mécanisme de filtrage qui réduit le « Feature Twist » et facilite les transitions entre les niveaux d'abstraction. Cette amélioration peut atténuer l'effet du problème du « Feature Twist » et améliorer les performances du modèle dans les tâches en aval.

Mots clés: Apprentissage de représentation des graphes, Auto-supervision, Classification, Regroupement, Apprentissage contrastif.

ABSTRACT

Recently, graph representation learning has become fundamental in analyzing graph-structured data. Previous work adopted self-supervision, a machine learning paradigm wherein a model learns from the data without relying on external labels to extract high-level semantic information and patterns. Self-supervised learning leverages various pretext tasks, such as predicting missing nodes or edges, to create meaningful representations. Then, researchers refine their models using downstream tasks such as node classification, clustering, and link prediction. However, the transition between self-supervision abstraction levels, representing different granularities of information in the data, has never been studied from a geometric perspective. In this thesis, we present a novel self-supervised contrastive learning framework designed to learn representations at multiple levels of abstraction: node-, proximity-, cluster-, and graph-levels. By employing a pretraining fine-tuning strategy, we investigate the evolution of intrinsic and linear intrinsic dimensions during the transition between these abstraction levels. Our experimental results demonstrate the existence of a significant Feature Twist challenge characterized by abrupt geometric transformation and deterioration of model performance in downstream tasks when transitioning between different self-supervised levels. To address this issue, we propose an effective filtering mechanism that reduces Feature Twist and facilitates smooth transitions between abstraction levels. This enhancement can mitigate the Feature Twist problem's effect and improve the model's performance in downstream tasks.

Key words: Graph representation learning, self-supervision, Node classification, Clustering, Contrastive learning.

CHAPTER I

INTRODUCTION

1.1 Background Information & Research Context

A graph is a collection of entities or nodes interconnected by edges or links, representing different relationships or interactions between the graph's entities. Graphs can manifest in various forms, including social, biological, transportation, and computer networks. Graphs are highly valuable for complex system modeling because they can capture the unique characteristics of each component of the graph and represent their interconnections and mutual influences. By studying the structure and dynamics of graphs, we gain insight into how these systems behave and function. Graph analysis involves the use of various tools and techniques derived from graph theory. Graphs can take various forms and are used to represent different types of relationships and interactions. In the following section, we will illustrate the most prevalent types of graphs.

- **Directed/Undirected Graphs:** In a directed graph, the links have a direction, indicating a distinction between incoming and outgoing edges. For example, the "follow" graph on Twitter represents one-directional relationships. However, in an undirected graph, the links between the nodes are symmetric and reciprocal. In an undirected graph, the edges between the nodes are symmetric and reciprocal, indicating that the relationship between

the nodes remains the same regardless of the direction. For example, we can represent the relationships between people on Facebook or LinkedIn using an undirected graph, where the links indicate a mutual connection.

- **Static/Dynamic Graphs:** Static and dynamic graphs differ based on the variability of their characteristics or graph topology over time. In static graphs, the node and attribute features remain the same. In dynamic graphs, fluctuations in input attributes or graph structure over time classify the graphs as dynamic. In such instances, the temporal dimension becomes significant, requiring careful consideration when working with dynamic graphs.

- **Attributed graph:** Attributed graph is a graph type in which each node or edge is associated with a set of attributes or features. These attributes can represent various properties of nodes or edges, such as their numerical values, categories, or textual descriptions. We can use attributed graph features to encode rich information about entities and their relationships.

Note that these categories are orthogonal, so we can combine different graph categories. For example, we can work with a dynamic directed attributed graph. Additionally, we distinct various other graph types designed for specific tasks, such as weighted graphs, hypergraphs, and signed graphs. Figure 1.1 illustrates the attributed graph of the Cora data set. In this graph, each node corresponds to a scientific publication, complete with a feature vector and connections to other nodes, representing their coauthors' relationships within the dataset.

Graphs are practical and versatile for making predictions across various domains. By representing datasets as graphs with intricate relational structures, we can effectively address a wide range of downstream tasks, which typically fall into three categories. Node-level categories focus on individual nodes and include tasks like node classification Wu et al. (2019a), which involves predicting node labels

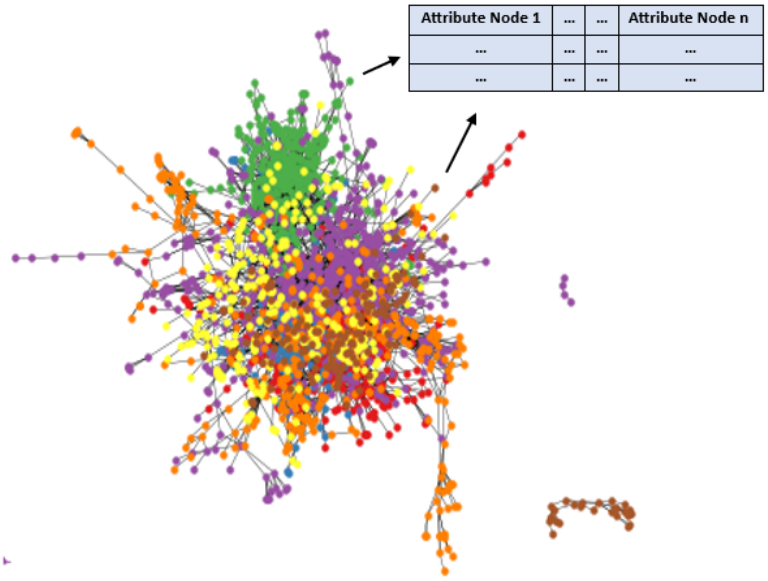

Figure 1.1: Viewing the structure of a graph: Cora network.

by considering their features and connections to other nodes. Unlike traditional supervised machine learning problems, node classification in graph data does not rely on independent identically distributed (i.i.d.) assumptions. Instead, it is based on concepts such as homophily, where nodes tend to share attributes with their neighbors on the graph McPherson et al. (2001), and structural equivalence, which assumes that nodes with similar local neighborhood structures will have similar labels Donnat et al. (2018).

Relation prediction is the second type of downstream task that focuses on the edges. Depending on the application domain, this task has different names, such as link prediction, graph completion, and relational inference. It requires the model to classify edge types or predict whether there is an edge between two given nodes based on their current attributes and connections. It has many real-world applications, such as recommending content to users on social platforms Ying et al. (2018). Finally, graph-level tasks involve classification, regression,

or clustering problems across entire graphs. Each graph is an i.i.d. data point associated with a label (e.g., a molecule structure). The goal is to use a set of graphs to learn a mapping from graphs to labels.

Graph modeling has thus gained significant practical applications in various domains. For example, in e-commerce Li et al. (2020), it has proven to be an invaluable tool to accurately predict user preferences, provide personalized recommendations, and improve crucial metrics such as click-through rate (CTR) and conversion rate (CVR). Spatial-temporal graph modeling has also been used to improve traffic flow Wu et al. (2019b). In the field of medicine and chemistry, graph databases and machine learning techniques have been used to predict clinical trial outcomes Murali et al. (2022), identify drug polypharmacy side effects Lukashina et al. (2022), and determine effective drug-disease treatments Jiang & Huang (2022). Furthermore, graph applications extend to citation and social networks Hu et al. (2020), as well as knowledge bases Ji et al. (2021), which play crucial roles in various natural language preprocessing applications, including relation extraction and entity linking.

Recently, several research studies have focused on unlocking the potential of graph data and addressing its unique challenges, aiming to extend the success of deep learning architectures to graph data. While convolutional neural networks have achieved remarkable results in processing Euclidean spaces like images and text, applying the same concept to graph data is not straightforward due to the non-Euclidean nature of graphs. To tackle this challenge, deep learning methods designed to handle graph data, known as Graph Neural Networks (GNNs), have garnered increasing attention across various network-related fields. The learning of GNNs Scarselli et al. (2008) can be conducted in a supervised or unsupervised manner, depending on our task. In GNNs, each node in a graph can be described by its characteristics and those of its neighbors. GNNs excel at computing com-

plex, high-level functions by aggregating local information from neighborhoods. Since the initial proposal, researchers have devised numerous approaches to tackle the challenge of learning from graphical data. Variations of GNNs have emerged to enhance their ability to map graphs to embedding representation, a process called graph representation learning.

Graph representation learning Hamilton et al. (2017b) is the process of creating compact and meaningful graph representations. The goal is to learn a mapping function that transforms a graph from a discrete domain to a continuous one while preserving its properties and structure. The resulting embeddings can be used as input for various machine learning algorithms, enabling powerful predictive capabilities. The traditional process of graph representation involves extracting statistics (node degree, node centrality, clustering coefficient, etc.) or utilizing specific kernel functions. However, these methods capture only a limited subset of the graph's information. On the contrary, graph representation learning offers a more efficient and comprehensive approach to representation learning. It involves developing algorithms that can learn the mapping function of a graph onto a continuous space. The embedding space should reflect the original graph structure and allow the embeddings to capture the essential properties and features. A simple graph representation approach is shallow encoding Hamilton et al. (2017b), where the encoder is just an aggregation function that integrates information from the local neighborhood of a node. We represent the shallow embedding mathematically as follows:

$$\begin{aligned} ENC(v) &= Z_v \\ Z \in \mathbb{R}^{d \times |v|}, v &\in \mathbb{I}^{|v|} \end{aligned} \tag{1.1}$$

In the above equation, ENC denotes the encoder. Each column of the matrix Z

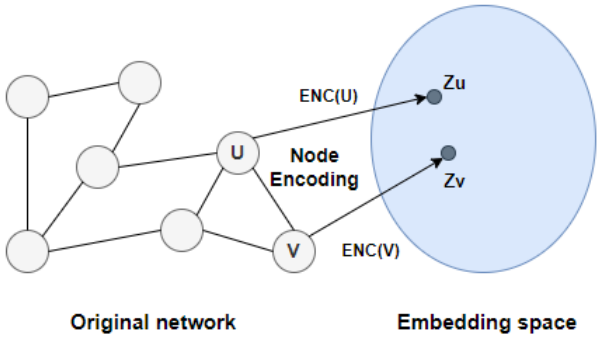

Figure 1.2: A diagram demonstrating the node embedding problem. We aim to learn an encoder (ENC) that maps the nodes to a low-dimensional embedding space. These embeddings are optimized so that distances in the embedding space approximate to node positions in the original graph.

indicates the embedding of a node. The total number of rows in Z is equal to the embedding dimension. v is the indicator vector with all values equal to zero except one in the column indicating node v. Each node is assigned a unique dense vector in shallow encoding. Figure 1.2 illustrates the application of an encoder function to map a graph into a lower-dimensional embedding space. In the embedding space, the distances between the points approximate the positions of nodes in the original graph.

Advanced graph representation learning techniques are built on two types of convolutional operations commonly used in Graph Neural Networks (GNNs): spectral and spatial approaches Zhang et al. (2019).

Spectral approaches are based on spectral graph theory, which examines the properties of a graph in relation to matrices such as characteristic polynomials, eigenvalues, and eigenvectors. In spectral convolution, the operation involves multiplying a signal (representing node characteristics) by a kernel. This operation is defined in the Fourier domain by finding the appropriate decomposition of the graph Laplacian. However, it can be computationally expensive, leading to the development of various approximation methods for efficiency. Recent work in graph

convolutional networks aims to improve the computational efficiency of spectral approaches while maintaining their critical foundation. Defferrard et al. (2016) proposed an approximation method that utilizes smooth filters in the spectral domain. They achieved this by employing Chebyshev polynomials with free parameters that can be learned within a neural network-like model. They achieved promising results on regular domains, such as MNIST, comparable to those of a simple 2D CNN model. In contrast, Kipf & Welling (2017) adopted a similar approach by building upon the framework of spectral graph convolutions. However, they introduced simplifications that allow significantly faster training times and higher predictive accuracy in numerous cases. As a result, this approach has attained state-of-the-art classification results on various benchmark datasets. Overall, this research demonstrates the potential of GCNs to enable efficient learning from graph-structured data. The equation below illustrates how the GCN works.

$$H^{l+1} = f(H^l, A) = \sigma(\tilde{D}^{\frac{-1}{2}}\tilde{A}\tilde{D}^{\frac{-1}{2}}H^l W^l) \tag{1.2}$$

$\tilde{A} = A + I$ is equivalent to adding self-loops to the original graph, where I is the identity matrix and A is the adjacency matrix.
H^l is the node embedding representation at layer l.
σ is a non-linear activation function (Relu).
$\tilde{D}$ is the diagonal node degree matrix of $\tilde{A}$.
W^l is a trainable weight matrix.

GCN has the advantage of aggregating node representations from their direct neighborhoods, making it a method that bridges the gap between spectral- and spatial-based methods. However, for large-scale graphs, training can be memory-intensive. Additionally, GCN's transduction process can interfere with generalization, making it challenging to learn representations for unseen nodes within the

same graph or nodes from entirely different graphs. To address computational challenges, FastGCN Chen et al. (2018) introduces a novel method for approximating the full-batch gradient computation in GCNs using sampling strategies. This approach enables efficient training on large graphs while maintaining high accuracy. Extensive experiments on several large-scale benchmark datasets demonstrate that FastGCN achieves state-of-the-art performance with significantly faster training time than previous GCN-based methods. The importance of FastGCN lies in its contribution to overcoming the scalability challenges of training GCNs, making it feasible to apply these powerful models to real-world, large-scale graph datasets.

Spacial approaches directly manipulate the graph structure by employing a convolution-like operation on node features. Unlike spectral graph networks that operate in the Fourier domain, spatial graph networks conduct operations in the spatial domain by aggregating information from neighboring nodes in close proximity. It offers a significant advantage through weight sharing, enabling the application of the same set of weights to various sections of the graph. This weight sharing enhances the model's ability to generalize across different graphs. Furthermore, since the convolution operation is conducted locally, it only requires consideration of a subset of nodes at each step, thereby enhancing the computational efficiency of the algorithm.

GraphSAGE Hamilton et al. (2017a) is a popular node representation method for large graphs. It adopts symmetric aggregator functions to generate dense node representations by aggregating information from their local neighborhoods. This approach guarantees that the model can be trained and applied to sets of neighborhood node vectors without any specific ordering requirement. GraphSAGE explores three different aggregator functions: mean aggregator, Long Short-Term Memory (LSTM) aggregator, and pooling aggregator. The mean aggregator is

simple and computationally efficient, but may not capture higher-order proximity information among neighbors. The LSTM aggregator can capture temporal dependencies. However, LSTMs are not symmetric and process their inputs sequentially. To make the LSTM aggregator symmetric, GraphSAGE applies it to a random permutation of the node's neighbors. This process makes the LSTM aggregator invariant to the order of the input nodes, ensuring that the neural network model can be trained and applied to arbitrarily ordered sets of neighborhood node vectors. The pooling aggregator applies a neural network independently to each neighbor vector and performs element-wise max pooling to obtain a fixed-length output vector. This aggregator can capture the most salient information from the neighborhood while preserving the permutation symmetry. Mathematically, we represent the GraphSAGE model and the three aggregation functions: mean aggregator, LSTM aggregator, and pooling aggregator as follows:

$$\begin{gathered} h_v^k = \sigma(W_k AGG(\{h_u^{k-1} \forall u \in N(v)\}), B_k h_v^{k-1}) \\ AGG = \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|} \\ AGG = LSTM(\{h_u^{k-1} \forall u \in \pi(N(v))\}) \\ AGG = \gamma(\{Q h_u^{k-1} \forall u \in (N(v))\}) \end{gathered} \tag{1.3}$$

h_v^k: is the embedding presentation of the last layer.

$|N(v)|$: is the number of neighbors.

σ: activation function to add non-linearity.

γ: element-wise mean/max.

W_k, B_k and Q: are trained parameters.

Graph Attention Networks (GAT) Veličković et al. (2018) is a spatial convolution technique for graphs that incorporates the attention mechanism to assign varying

weights to neighboring nodes. The underlying concept of GAT is that specific nodes carry more crucial information than others, and the model should prioritize those nodes to improve performance. The attention mechanism calculates normalized weight coefficients for each node's neighbors. It learns the weight using a trainable function considering node features and edge connections. These coefficients indicate the significance of each neighbor node's contribution to the central node's representation. Mathematically, we represent the GAT model as follows:

$$h_v^{(k+1)} = \sigma\left(\sum_{u \in \mathcal{N}_v} \alpha_{vu}^{(k)} W^{(k)} h_u^{(k)}\right) \tag{1.4}$$

where $h_v^{(k)}$ is the feature representation of node v at the k-th layer, $\mathcal{N}_v$ is the set of neighboring nodes of node v, $W^{(k)}$ is a weight matrix, $\alpha_{vu}^{(k)}$ is the attention weight coefficient computed for the edge connecting nodes v and u at the k-th layer, and σ is a non-linear activation function.

The attention weight coefficients $\alpha_{vu}^{(k)}$ are computed using an attention mechanism that depends on the node features:

$$\alpha_{vu}^{(k)} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^{(k)\top}\left[\mathbf{W}^{(k)} h_v^{(k)} \| \mathbf{W}^{(k)} h_u^{(k)}\right]\right)\right)}{\sum_{l \in \mathcal{N}_v} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^{(k)\top}\left[\mathbf{W}^{(k)} h_v^{(l)} \| \mathbf{W}^{(k)} h_l^{(k)}\right]\right)\right)} \tag{1.5}$$

Here, $\vec{a}$ is a weight vector to be learned, and LeakyReLU is a leaky rectified linear unit activation function. This attention mechanism allows the model to assign different weights to different neighbors of a node based on their importance in contributing to the representation of the central node. Figure 1.3 summarizes the graph-convolutional neural network (GCN) approaches and highlights their primary real-world applications.

Recent state-of-the-art graph representation learning approaches are based on

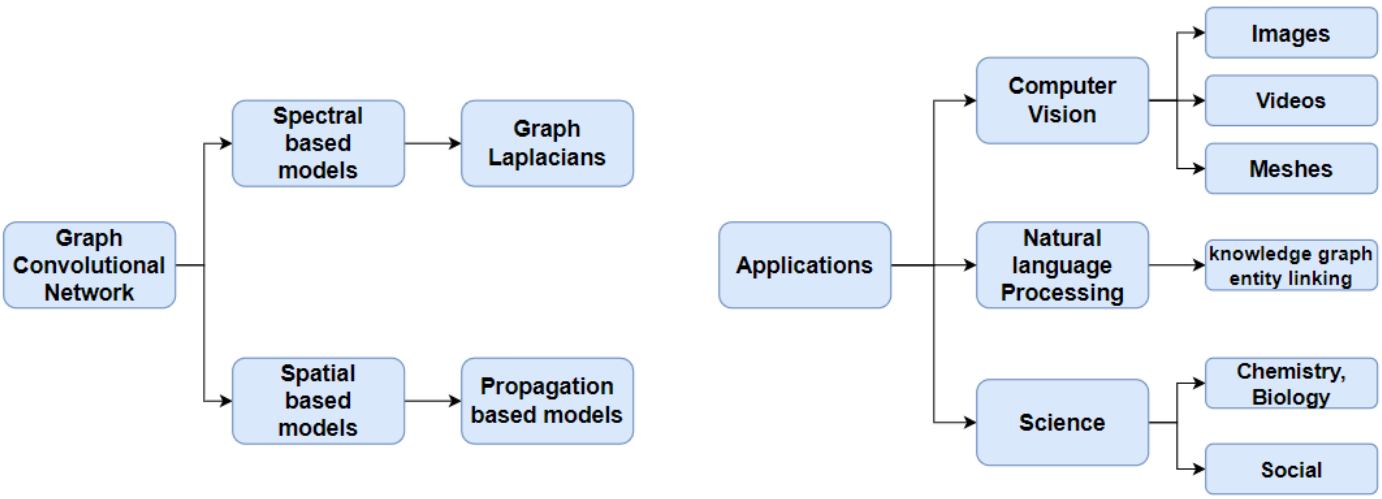

Figure 1.3: An overview of graph convolutional networks.

self-supervised learning You et al. (2020). Unlike supervised learning, which leverages well-defined manual labels to train machine learning models, or unsupervised learning, which focuses on finding patterns or clusters using unlabeled data, self-supervised learning generates supervision signals from the data. In self-supervised learning, models are trained with pretext tasks to achieve better performance and generalization on downstream tasks. The pretext task can learn latent representations or model weights to use them for a defined downstream task. Downstream tasks are the final modeling goal and can be anything like classification or detection with insufficient annotated data samples.

1.2 Motivations

In recent literature on graph modeling, the primary focus has been evaluating the performance of learned embeddings in downstream tasks following self-supervised pretext learning. However, a frequently overlooked issue is Feature Twists (FT). Feature twist denotes sudden geometric transformations observed in latent space representations during the transition from a pretext task to a downstream one.

Previous works have frequently encountered this problem, emphasizing the need to address it. When latent representations capture irrelevant information, they can significantly impair the model's performance in downstream tasks reliant on these embeddings. Therefore, addressing this problem is crucial to ensure the quality and effectiveness of the learned representations.

The approach proposed in Mrabah et al. (2022a) introduces the FT-VGAE model to address the Feature Twist problem. The FT-VGAE employs a Variational Auto-Encoder (VAE) and follows a three-step training process. In the first step, the model minimizes the reconstruction loss to learn low-dimensional curved embedded manifolds. Traditional Euclidean algorithms are not applicable due to the non-linearity of these manifolds. The second step focuses on smoothing the local curved structures while preserving the global ones. The second step achieves this by merging local neighborhoods, promoting uniformity, and helping alleviate the Feature Twist problem. Finally, it uses two objective functions to alternate between clustering-oriented loss and a combination of over-clustering and adjacency reconstruction loss. Over-clustering methods produce more clusters or subgroups than are present in the data. It helps flatten the latent manifolds slowly to avoid twisting the curved structures and ensuring that the learned representations are suitable for downstream tasks. In summary, the FT-VGAE model presents a promising solution to address the Feature Twist problem and enhance the performance of graph-based clustering algorithms.

Building upon prior research, our study generalizes the Feature Twist problem by examining the transitions between various levels of abstraction in self-supervision from a geometric perspective. By adopting an independent training protocol involving pre-training followed by fine-tuning, we aim to identify beneficial transitions that can mitigate the impact of Feature Twist. This research endeavor offers valuable insights into the interplay among different levels of abstraction and

their influence on downstream tasks. It provides a comprehensive understanding of how these transitions impact the performance of graph-based algorithms in various tasks such as node clustering and node classification. The results of our work will serve as a roadmap for future investigations, guiding research efforts toward reducing the effects of Feature Twist. Ultimately, this will enhance the overall performance of graph-based algorithms in various downstream tasks.

1.3 Contributions

In this thesis, we present a novel framework that combines graph neural networks (GCN) and contrastive learning to explore self-supervised learning (SSL) on graphs at various levels of abstraction. Our framework introduces a mathematical formulation for learning SSL models at the node level, proximity level, cluster level, and graph level. Unlike previous research that primarily focuses on generating embeddings for a pretext task and later applying them to a downstream task, our approach enables us to learn and evaluate embeddings across multiple levels of abstraction. This strategy allows us to address the challenge of the Feature Twist problem and achieve better generalization.

To investigate the transition between different levels of SSL, we employ an independent training strategy that takes advantage of the intrinsic dimension (ID) and the linear intrinsic dimension. These geometric measures help us to control the manifolds during the transition process. Moreover, we propose a filtering mechanism that smooths the transitions between different levels of abstraction, with the aim of minimizing the feature twist problem. This mechanism leads to improved performance of the model on downstream tasks. The main contributions of our study can be summarized as follows.

- Develop a mathematical formulation for training self-supervised graph representations at multiple levels of abstraction, including node-, proximity-, cluster-, and graph-levels. The unified proposed formulation can achieve comparable results to the performance of state-of-the-art methods for each level of abstraction.

- Generalize the Feature Twist problem by controlling the geometric transformation of latent representations during the transition between different levels of abstraction.

- Propose a filtering mechanism that alleviates the Feature Twist problems and improves downstream task performance.

1.4 Thesis Plan

The rest of this report is organized as follows:

- **Chapter 2** presents a thorough review of recent advancements in fields closely related to our work. This review encompasses the latest developments, methodologies, and findings in these relevant areas, providing valuable insights and contextualizing our research.

- **Chapter 3** provides a comprehensive and detailed description of our approach. We outline the specific methods we employ to address the challenge of Feature Twist and minimize its impact.

- **Chapter 4** presents our experimental results. We provide a detailed analysis and discussion of the outcomes, highlighting the performance and effectiveness of our approach.

CHAPTER II

LITERATURE REVIEW

Recently, deep learning on graphs has garnered considerable interest. However, much of the research has been centered around semi-supervised learning, resulting in shortcomings such as heavy reliance on labeled data, limited generalization, and weak robustness. To overcome these challenges, self-supervised learning (SSL) has emerged as a promising and trending approach for graph data. SSL leverages well-designed pretext tasks to extract informative knowledge and high-level features from data without performing manual labeling. In contrast to SSL applications in other domains like computer vision and natural language processing, SSL on graphs presents unique backgrounds, design ideas, and taxonomies. To shed light on this evolving area, we provide a comprehensive review of existing approaches that employ SSL techniques for graph data and focus specifically on those relevant to our research problem. We propose a classification of previous works into four categories based on their research focus and methodology. Specifically, our focus lies on self-supervised graph representation learning methods, which we categorize into four levels of abstraction: node-level, proximity-level, cluster-level, and graph-level. Moreover, we provide a comparative summary of existing methods, highlighting their strengths and weaknesses.

2.1 Node level

In the node-level representation learning category, the primary focus is on capturing the essential features of individual nodes within the graph. One notable work in this category is GRACE Zhu et al. (2020), a novel framework for unsupervised graph representation learning that leverages self-supervision through a contrastive objective function at the node level.

Unlike traditional graph representation learning methods that rely on a reconstruction loss function, GRACE takes a different approach by maximizing the Mutual Information (MI) between two generated graph representations. First, GRACE generates two correlated graph views by randomly corrupting the graph topology and node attributes. Graph corruption is accomplished by feature masking and edge removal, which provides diverse contexts for nodes in each graph view. Then, it learns the model to maximize the agreement between node embeddings in the two generated graph views using a contrastive loss function. Using the MI-based contrastive objective, GRACE can learn a more accurate and robust representation of the graph structure and node attributes.

This approach enables GRACE to capture complex and subtle relationships between nodes in the graph, leading to improved performance in downstream tasks. It enhances the ability of the node-level learning representations to encode crucial information and uncover hidden patterns within the graph, making it highly effective in various applications. GRACE demonstrates the effectiveness of leveraging a contrastive objective function and Mutual Information for unsupervised graph representation learning. Its ability to capture intricate relationships between nodes enables it to achieve superior performance in downstream tasks, making it a valuable addition to the arsenal of techniques for graph data analysis. Figure 2.1 provides a straightforward description of contrastive learning to create a graph

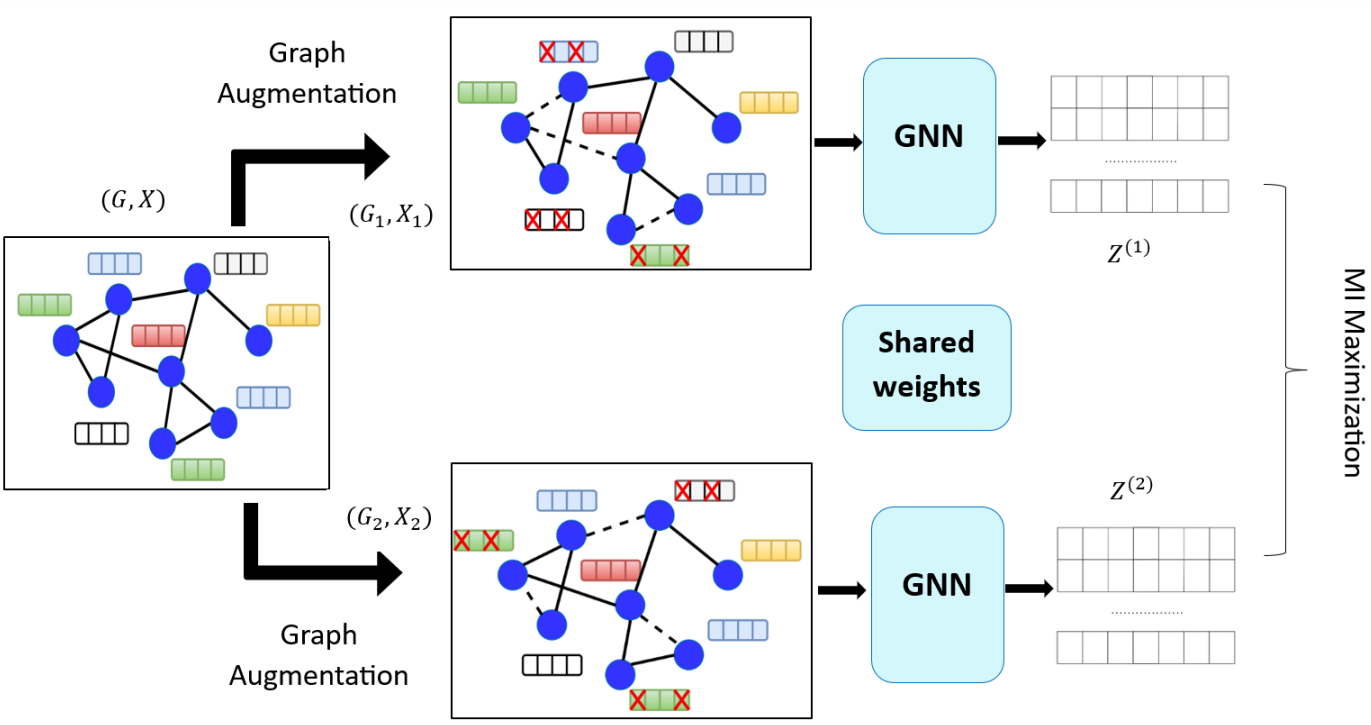

Figure 2.1: A high-level overview of node level self supervision.

representation at the node level.

GCA Zhu et al. (2021) is a notable graph representation learning method that employs a contrastive learning approach to maximize Mutual Information (MI) between node embeddings from two correlated graph views. A key aspect of GCA's approach is its focus on data augmentation design, as corrupting graphs must preserve intrinsic patterns while avoiding the removal of crucial nodes or edges that may degrade the quality of the learned embeddings. To address this challenge, GCA introduces a data augmentation strategy based on node centrality. Using node centrality as a measure, GCA identifies important edges and nodes within the graph. GCA's data augmentation involves two steps: first, it augments the initial graph by removing unimportant edges, and second, it adds noise to the features of less critical nodes. This strategy ensures that the overall graph

structure remains intact, while enhancing the diversity of the generated views.

The data augmentation technique employed by GCA leads to a more robust and accurate graph representation than previous methods. By preserving the essential graph structure and node features while incorporating sufficient diversity in the augmented views, GCA effectively captures the underlying patterns and relationships within the graph data. Empirical results Zhu et al. (2021) demonstrate the effectiveness of GCA's data augmentation strategy, as it outperforms state-of-the-art graph representation learning methods.

Unlike previous methods that often rely on large numbers of negative examples and complex graph augmentations, which can be computationally expensive, especially for large graphs, BGRL Thakoor et al. (2022) takes a different approach to learning graph representation. BGRL adopts a more efficient and scalable design by learning through predicting alternative augmentations of the input, using only simple graph augmentations.

Inspired by recent advances in self-supervised learning in vision, BGRL learns node representations by encoding two augmented versions of a graph with two distinct graph encoders: an online encoder and a target encoder. The online encoder is trained to predict the representation of the target encoder, while the target encoder is updated as an exponential moving average of the online network. A critical advantage of BGRL is that it does not require contrasting with negative examples, making it highly scalable even for large graphs. By eliminating the need for extensive negative sampling and complex augmentation techniques, BGRL significantly reduces computational overhead, making the model more feasible to apply on large-scale graph datasets.

2.2 Proximity level

In contrast to node-level contrastive learning methods, proximity-level techniques do not rely on data augmentation. Instead, they capture similarities based on the relationships between neighboring nodes, avoiding the use of perturbed representations of the same node. One commonly used proximity-level technique is Random Walk Perozzi et al. (2014), which offers an efficient and expressive approach to define node similarity.

Random Walk leverages a stochastic process to determine node similarity, considering local and higher-order neighborhood information by exploring pairs of nodes that co-occur during the random walk. The algorithm starts from a given node in the graph and proceeds by randomly selecting a neighbor, moving to that neighbor and then choosing another neighbor from that point, generating a sequence of nodes. Node embeddings are created by estimating the probability of visiting a node from another neighbor using the random walk strategy. The optimization mechanism aims to maximize random walk co-occurrence probability, ensuring that the latent representations preserve the similarity observed in the original graph. In other words, neighboring nodes in the graph should also be close to each other in the latent space representation. The loss function used in this approach is calculated as follows:

$$\begin{gathered} \mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} - \log P(v|Z_u) \\ \log P(v|Z_u) = \frac{z_u^T z_v}{\sum_{n \in V} z_u^T z_n} \end{gathered} \tag{2.1}$$

To optimize random latent representations, we need to find Z_u embedding that minimizes $\mathcal{L}$. But to do this naively without any changes is too costly. A new formulation corresponds to the use of logistic regression to distinguish the target node v from nodes n_i sampled from the distribution P such that:

$$\frac{z_u^T z_v}{\sum_{n \in V} z_u^T z_n} \approx \log(\sigma(z_u^T z_v)) - \sum_{i=1}^{k} \log(z_u^T z_{n_i}), n_i \approx P_v \tag{2.2}$$

P_v denotes random distribution across all nodes.

Instead of normalizing with respect to all nodes, we normalize to k random n_i. By leveraging proximity-level techniques like Random Walk, we can effectively capture meaningful similarities between nodes within the graph without performing data augmentation.

DeepWalk Perozzi et al. (2014) is a graph node embedding technique that leverages a random walk strategy combined with the Skip-Gram model and Hierarchical Softmax to generate meaningful vector representations for nodes in a graph. The Skip-Gram model Mikolov et al. (2013), firstly designed for natural language processing, is employed as an essential component of DeepWalk. The Skip-Gram model is an unsupervised deep learning technique to generate vector representations for words in a language. It operates by training a simple neural network with a hidden layer to maximize the co-occurrence probability of words in the same window.

Applying the DeepWalk algorithm to graphs starts by generating random walks within the graph. These random walks are not constrained to have the same length, but there is a fixed upper bound that cannot be surpassed. Each random walk represents a sequence of traversed nodes starting from a particular node in

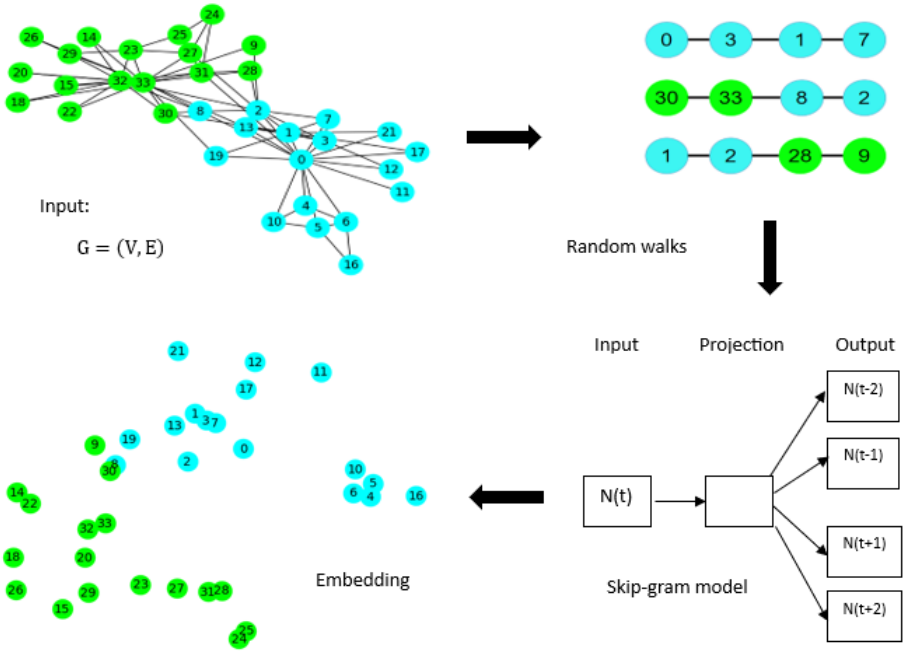

Figure 2.2: All the steps used by the DeepWalk Perozzi et al. (2014) algorithm to generate the node embedding of a given graph where $N(t)$ represents the t^{th} node in the random walk sequence.

the graph. Then, it train a Skip-Gram model using the entire set of previously generated random walk steps. During training, the Skip-Gram model learns to predict the context nodes based on a given target node in each random walk sequence. This process effectively captures the local structure of the graph and the relationships between nodes. The hidden layers of the trained Skip-Gram model contain valuable information that represents the learned embeddings for each node in the graph. These embeddings, derived from the Skip-Gram model, serve as the final representation of each node, effectively encoding its characteristics and relationships with other nodes in the graph. Figure 2.2 shows the main steps of embedding generation using the deepwalk model.

DeepWalk adopts fixed-length unbiased random walks from each node, which results in a narrow-node similarity representation. The Node2Vec algorithm Grover

& Leskovec (2016) is an extension of DeepWalk that introduces a more flexible notion of node neighborhoods, leading to richer node representations. Node2Vec achieves this by generating biased random walks on the graph, combining the Breadth-First Sampling (BFS) and Depth-First Sampling (DFS) algorithms to exchange local and global graph information. The algorithm guides the generation of biased random walks using two parameters, p and q. The parameter p governs the probability that a random walk returns to the previous node, while q controls the probability that the random walk explores an unknown part of the graph. Similar to DeepWalk Perozzi et al. (2014), Node2Vec uses the skip-gram approach to generate node embeddings.

GMI (Graphical Mutual Information) Peng et al. (2020) is another approach to learning graph representation that introduces a more straightforward way to consider mutual information (MI) within graphical structures, eliminating the need for readout and corruption functions. Instead, GMI directly derives MI by comparing the input (i.e., the sub-graph consisting of the input neighborhood) with the output (i.e., the hidden representation of each node) of the encoder. Interestingly, theoretical derivations show that the directly formulated MI can be broken down into a weighted sum of local MIs between each neighborhood feature and the hidden vector. This decomposition makes the MI computation tractable and enables the input features decomposition. Additionally, this form of MI can satisfy the symmetric property by adjusting the weight values.

2.3 Cluster level

Node- and proximity-based graph construction methods often struggle to leverage imprecise clustering labels effectively and may require post-processing operations to obtain accurate clustering results. The Structural Deep Clustering Network

(SDCN) Bo et al. (2020) is a powerful approach that addresses clustering-oriented tasks with proximity-level information by exploiting high-order structure and adjacency reconstruction. It achieves this by integrating structural information into deep clustering, employing a delivery operator to transfer representations learned by an autoencoder to the corresponding Graph Convolutional Network (GCN) layer. Additionally, SDCN adopts a dual self-supervised mechanism, effectively unifying these two distinct deep neural architectures and guiding the model update. Experimental results demonstrate the superiority of SDCN's strategy, yielding significant improvements in clustering-oriented tasks. Incorporation of structural information and smooth interplay between the autoencoder and the GCN contribute to performance enhancement and robustness of the model.

Self-supervised Contrastive Attributed Graph Clustering (SCAGC) Xia et al. (2021) proposes an innovative approach that incorporates node and cluster information for attributed graph clustering. SCAGC adopts graph augmentation techniques to create two graph views: attribute masking, where random noise is added to node attributes, and edge perturbation, where edges are randomly added or dropped in the topological graph. These two representations interact and evolve jointly within an end-to-end framework. The model is optimized using a joint loss function, which carefully balances contrastive cluster-level loss, contrastive node-level loss, and a regulatory term. The self-supervised contrastive loss aims to maximize the similarity among nodes within the same cluster and simultaneously minimize the similarity between nodes from different clusters. By leveraging inaccurate clustering labels, SCAGC effectively maximizes Mutual Information between the two graph representations. Figure 2.3 illustrates the cluster-level contractive approach in generating node embeddings.

Deep Attentional Embedding Graph Clustering (DAEGC) Wang et al. (2019) presents an innovative framework that synergistically optimizes graph cluster-

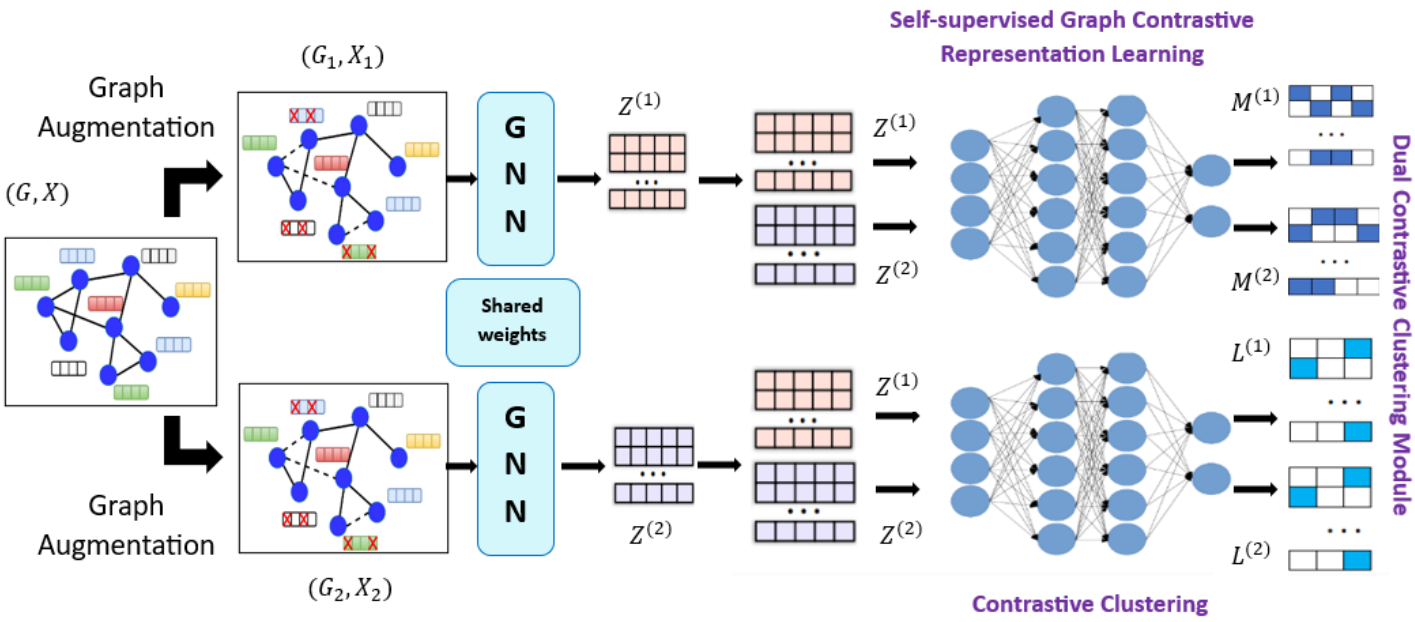

Figure 2.3: Self-supervised Contrastive Attributed Graph Clustering.

ing and embedding learning. By combining these two tasks, DAEGC achieves mutual benefit and improved performance. The core of DAEGC is a graph attentional autoencoder, which learns a powerful latent representation by minimizing the reconstruction loss function. The graph attentional autoencoder consists of an encoder that effectively captures the graph structure and the node content. This is achieved by incorporating a graph attention network, enabling the model to identify crucial relationships and features within the graph. The attention mechanism helps the encoder focus on the most informative parts of the graph during the learning process. On the other hand, the decoder in DAEGC reconstructs the topological graph information, ensuring the integrity of the learned graph representation. In addition to the autoencoder, DAEGC includes a self-training clustering module, a crucial component that further enhances clustering performance. This module leverages confident clustering assignments to guide the optimization procedure, leading to more accurate and robust clustering results.

2.4 Graph level

In the field of graph representation learning, there exists a category of approaches focused on learning graph-level representations. Among these techniques, Deep Graph Infomax (DGI) Velickovic et al. (2019) stands out as one of the most prominent. DGI is an unsupervised graph learning framework that aims to maximize mutual information between global graph representations and local node embeddings. To achieve this, it uses the scalable estimation of mutual information through Mutual Information Neural Estimation (MINE) Belghazi et al. (2018).

DGI builds on the foundation laid by Deep InfoMax (DIM) Hjelm et al. (2019), which focuses on maximizing mutual information between high-level global representations and local-level representations of the input to learn embeddings of high-dimensional data. DGI employs a graph neural network to generate node embeddings, and an injective readout function is applied to create graph-level embeddings. The graph-level representations are learned via contrastive learning, achieved by discriminating between nodes in the original graph and nodes in a corrupted graph view. To generate corrupted graph views, DGI employs feature shuffling as part of its graph augmentation scheme. However, it is worth noting that the injective property used in the readout function might be overly restrictive in many cases. Figure 2.4 presents the process of generating the embedding using a constrained loss at the graph level.

Graph InfoClust (GIC) Mavromatis & Karypis (2020) is an extension of the Deep Graph Infomax (DGI) Velickovic et al. (2019) model that incorporates the assumption that nodes tend to belong to clusters. This assumption implies that nodes within the same cluster should demonstrate a high similarity to each other, whereas nodes in different clusters should exhibit a low similarity. To effectively capture graph-level and cluster-level information, GIC utilizes a differentiable K-

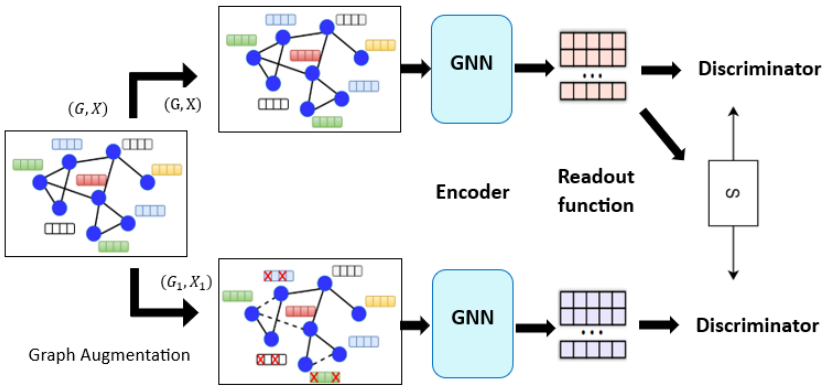

Figure 2.4: A high-level overview of graph level self supervision.

means method, enabling end-to-end model training. Differentiable K-means is a technique that combines the K-means algorithm with backpropagation, allowing clustering to be learned in a differentiable manner. The algorithm enables the model to be trained using gradient descent, which is more efficient than traditional K-means algorithms. With differentiable K-means, GIC can learn graph-level and cluster-level representations within a single framework, maximizing mutual information for both levels. As a result, GIC provides a more effective and efficient clustering method for graph data.

2.5 Conclusion

This chapter provided a comprehensive overview of self-supervised graph representation learning. Additionally, it included a review of the literature of recent studies that investigated different levels of abstraction in graphs. In the upcoming chapters, we will offer a detailed description of our approach to studying graph self-supervision from a geometric perspective, along with our experimental results.

CHAPTER III

PROPOSED APPROACH

This chapter describes the proposed approach. We start by providing a detailed overview of our proposed method, which encompasses contrastive objectives and graph augmentation techniques. Subsequently, we delve into the independent training strategy and the metrics we employ to evaluate Feature Twist. In order to enhance the transition between self-supervised abstraction levels and enhance performance in downstream tasks, we introduce our filtering strategy. Its primary objective is to achieve smoother transitions between two abstraction levels, thereby improving the overall performance of our models.

3.1 Notations

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ a non-directed attributed graph, where:

$\mathcal{V} = \{v_1, ..., v_N\}$ is a set of node with $|\mathcal{V}| = N$, and N is the number of nodes.

$\mathcal{E} = \mathcal{V} \times \mathcal{V}$ is the edge set, denoting connections between nodes.

$X \in \mathbb{R}^{N \times J}$ is the feature matrix, where, $x_i \in \mathbb{R}^J$ represents the feature vector associated with the i^{th} node, and J is the dimension of the feature vector.

Let $A = (a_{ij})$ be the adjacency matrix, which describes the topological structure of the graph, such that $a_{ij} = 1$ if there is an edge between nodes v_i and v_j $(i.e.(v_i, v_j) \in \mathcal{E})$ and $a_{ij} = 0$ otherwise.

These preliminary notations lay the foundation for our subsequent discussions on the proposed method. Understanding the graph representation and the feature matrix is crucial to grasp the techniques and strategies we employ to leverage this graph structure effectively. In the following sections, we will delve into the details of our framework, including contrastive objectives and graph augmentation techniques, as well as the independent training strategy and the metrics used to measure Feature Twist. Additionally, we will introduce a novel filtering strategy designed to enhance the transition between self-supervised abstraction levels and improve performance in downstream tasks.

3.2 Contrastive learning framework

3.2.1 Graph views generation

Data augmentation is a valuable technique that enables the generation of additional training data without the need for extensive data collection or labeling efforts. In fields such as Computer Vision and Natural Language Processing, data augmentation typically involves cropping, flipping, and masking to create modified versions of existing data or synthetic data based on existing samples. However, when dealing with non-Euclidean and irregular data structures such as graphs, data augmentation becomes more challenging. Unlike regular and Euclidean data represented by grids or sequences, graph data relies on node connectivity to encode its structure. Consequently, the structured augmentation operations used in traditional settings cannot be easily applied to graph data. This presents a significant challenge in the graph machine learning domain, as designing effective

graph view generation to enhance the self-supervision learning process remains an open problem.

In graph data augmentation context, we can categorize the techniques into two main categories, namely structure augmentations and feature augmentations. Structure augmentations focus on modifying the graph's connectivity by adding or removing edges or by adding or removing nodes from the initial graph view. On the other hand, feature augmentations focus on modifying or creating raw node features. Common approaches include masking or shuffling the original node features.

In this work, we propose a hybrid scheme for generating graph views by considering corruptions at both the topology and node attribute levels. Specifically, we randomly remove edges, mask features, and shuffle nodes (feature vectors). This approach aims to provide diverse contexts for nodes in different views, thereby boosting the optimization process of the contrastive objective. In the following, we detail the adopted graph augmentation strategy:

- **Dropping-Edges:** Edge-dropping methods involve randomly removing a certain fraction of links from the graph. To achieve this, we start by sampling a random masking matrix, where each entry in the matrix is drawn from a Bernoulli distribution with a probability of removing each edge. By applying the Hadamard product (element-wise multiplication) between the original adjacency matrix and the masking matrix, we obtain the resulting adjacency matrix of the corrupted graph.

- **Node Feature Masking:** Node feature masking methods randomly mask a fraction of dimensions in the node feature vectors with zeros. To implement this, we begin by sampling a random masking vector, where each entry is drawn from a Bernoulli distribution with a probability of masking each

dimension independently. The resulting node feature is then constructed as the concatenation of the original node feature vectors, with each element-wise multiplied by the corresponding element in the masking vector.

- **Node Shuffling:** Node shuffling methods involve randomly shuffling the node feature vectors of the original graph. This operation provides a way to alter the order of node features, thereby creating various views of the graph data.

To address the limitations of uniform graph augmentation schemes, such as uniformly dropping edges, randomly masking features, and shuffling nodes, which may inadvertently destroy intrinsic structures and attributes of graphs, we integrate an adaptive data augmentation scheme proposed by Zhu et al. (2021) into our framework.

Adaptive graph data augmentation strategies encompass various graph priors related to topological and semantic aspects. These augmentations are guided by specific graph characteristics, such as node centrality measures, which identify crucial edges and feature dimensions. This approach results in a higher probability of corruption for unimportant links and features. By doing so, the augmentation process focuses on perturbing less critical elements of the graph while preserving its essential structures and attributes. For the topology level, we can employ edge centrality measures to assess the influence of edges based on the centrality of the two connected nodes. Also, for node centrality at the topology level, we can use three effective and straightforward measures of centrality: degree centrality in Eq. 3.1, eigenvector centrality in Eq. 3.2, and PageRank centrality Eq. 3.3. For node features, we can utilize previously determined node centrality measures to identify essential and unimportant nodes. Consequently, we assign a higher masking probability to less important nodes during adaptive data augmen-

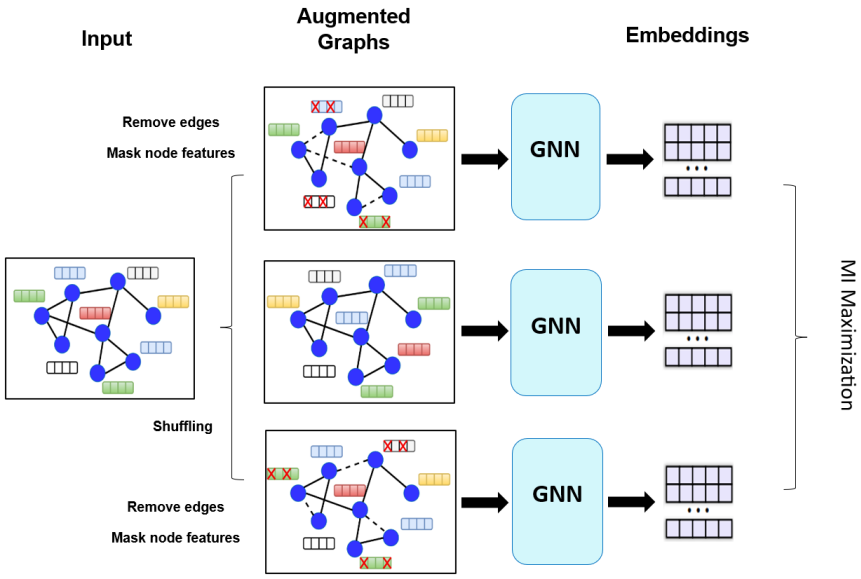

Figure 3.1: Our proposed framework for graph representation learning using contrastive loss.

tation. Figure 3.1 presents a high-level overview of our framework for training graph representations at various abstraction levels.

$$C_{\text{degree}}(v_i) = \frac{\sum_j A_{ij}}{N-1} \tag{3.1}$$

$$x_i = \frac{1}{\lambda} \sum_{j=1}^{N} A_{ij} x_j \tag{3.2}$$

x_i is the eigenvector centrality score of the i^{th} node in the network and λ is the largest eigenvalue of the adjacency matrix A.

$$Pr(v) = \frac{1-d}{N} + d \sum_{u \in \mathcal{V}} \frac{Pr(u)}{L(u)} \tag{3.3}$$

$Pr(v)$ represents the PageRank centrality of node v, d is the damping factor

(typically set to a value like 0.85) and $L(u)$ is the degree of node u.

3.2.2 Graph representation learning

In the field of unsupervised graph representation learning, the contrastive paradigm has been a prominent approach. Previous research in this field has mainly focused on local contrastive patterns that encourage nearby nodes to have similar embeddings. In particular, nodes that appear in the same random walk are considered positive samples. Pioneering works like DeepWalk Perozzi et al. (2014) and Node2vec Grover & Leskovec (2016) use contrastive estimation to model probabilities of node co-occurrence pairs through random walks and biased random walks, respectively, and learn node embeddings using skip-gram models. Although these techniques show potential, they overly prioritize structural information captured in graph proximities, such as adjacency matrix transformations. This limitation results in scalability issues, particularly with large datasets, and poses challenges in appropriately tuning hyperparameters, potentially compromising their effectiveness.

Recent advances in graph neural networks (GNNs) have introduced more powerful graph convolutional encoders, surpassing traditional approaches in terms of performance. An influential model, proposed by Kipf & Welling (2017), introduced a robust spectral graph convolution framework. Another notable model is GraphSAGE Hamilton et al. (2017a), which combines objectives similar to those of DeepWalk to enhance its performance. Additionally, the concept of graph attention model has emerged as an effective mechanism for capturing relational dependencies within graphs. In this work, our primary goal is to understand the transition between different abstraction levels. Therefore, we have adopted the graph convolutional encoder from Kipf & Welling (2017) as our base model for

learning graph representations. However, it is worth noting that our framework is flexible and can accommodate other GNNs, such as GraphSAGE and the graph attention model.

3.2.3 Contrastive loss

In contrastive learning, the primary objective is to learn meaningful representations by contrasting similar and dissimilar pairs of samples within a latent space. The aim is to maximize the similarity between positive pairs (samples from the same class or context) while minimizing the similarity between negative pairs (samples from different classes or contexts). This is typically achieved by employing a graph neural network to map the samples into a latent space and then comparing their representations using a similarity metric, such as contrastive loss.

In self-supervised learning, contrastive learning has found extensive use in various domains, including computer vision and natural language processing. For instance, in self-supervised visual representation learning, contrastive methods are employed to acquire discriminative representations through contrasting positive and negative samples. Negative samples can be created using image augmentation techniques such as cropping, rotation, color distortion, and more to introduce diversity into the training data. Contrastive learning represents a revitalization of the traditional Information Maximization strategy and has shown significant success in the graph neural networks field. By leveraging contrastive loss, the model can effectively capture underlying semantic information in a pretext task, which is crucial for learning informative representations in self-supervised learning settings.

Our proposed framework adopts the standard graph-contrastive learning paradigm, wherein the primary objective is to enhance the agreement of representations

across diverse views. Initially, we generate three distinct graph views through stochastic graph augmentation applied to the original graph. This process enhances the diversity of the data representations. Subsequently, we utilize a contrastive objective to enforce the encoded embeddings of each node in the various views to align with one another while remaining distinguishable from embeddings of other nodes. This approach not only improves the discriminative power of our model, but also enables generalization across different abstraction levels, such as node, proximity, cluster, and graph level. This formulation plays a crucial role in enhancing the effectiveness of our contrastive loss function across multiple abstraction levels, leading to more robust and comprehensive representations. In Eq. 3.4 we provide a detailed mathematical exposition of our framework which serves as a robust mechanism for investigating self-supervision across varying levels of abstraction.

$$\mathcal{L}(u_i, g(v_i)) = \frac{e^{\theta(u_i, g(v_i))/\tau}}{\sum_{s_j \in \mathcal{S}_u \setminus g(u_i)} e^{\theta(u_i, s_j)/\tau} + \sum_{s_j \in \mathcal{S}_v \setminus g(v_i)} e^{\theta(u_i, s_j)/\tau} + \sum_{s_j \in \mathcal{S}^- \setminus g(v^-)} e^{\theta(u_i, s_j)/\tau}} \tag{3.4}$$

Let M be the embedding space dimension we define:

$U = GCN(f^+(X, A)) \in \mathcal{R}^{N,M}$ as the node embedding matrix of the first generated positive view where u_i is the embedding vector of i^{th} node.

$V = GCN(f^+(X, A)) \in \mathcal{R}^{N,M}$ as the node embedding of the second generated positive view where v_i is the embedding vector of i^{th} node.

$V^- = GCN(f^-(X, A)) \in \mathcal{R}^{N,M}$ as the node embedding of the generated negative view where v_i^- is the embedding vector of i^{th} node.

$g^{(abstraction-level)}(u_i)$ is a flexible function that can take on different roles depending

on the self-supervision level being used in the framework. The $g^{(abstraction-level)}(u_i)$ function can serve as a readout, proximity function, cluster function, or identity as defined below:

$$
\begin{aligned}
g^{(node)}(u_i) &= Id(u_i) = u_i \\
g^{(proximity)}(u_i) &= \frac{1}{|\mathcal{N}(u_i)|} \sum_{j \in Neighbors(u_i)} u_j \\
g^{(cluster)}(u_i) &= \frac{1}{|Cluster(u_i)|} \sum_{j \in Cluster(u_i)} u_j \\
g^{(graph)}(u_i) &= \frac{1}{|\mathcal{N}(u_i)|} \sum_{j \in Graph(u_i)} u_j
\end{aligned}
\tag{3.5}
$$

Let $\mathcal{S}$ represent the output set of the function g as:

$$
\begin{aligned}
\mathcal{S}_u &= g^{(\text{abstraction-level})}(\{u_1, u_2, \ldots, u_n\}) \\
\mathcal{S}_v &= g^{(\text{abstraction-level})}(\{v_1, v_2, \ldots, v_n\}) \\
\mathcal{S}^- &= g^{(\text{abstraction-level})}(\{v_1^-, v_2^-, \ldots, v_n^-\})
\end{aligned}
\tag{3.6}
$$

We denote $H = GCN(X, A)$ as the learned representations of nodes, where h_i is the latent space representation of node i.

We define the critic $\theta(x, y) = sim(k(x), k(y))$ where $sim(.,.)$ is the cosine similarity and $k(.)$ is a non-linear projection to enhance the expression power of the critic function θ Chen et al. (2020); Tschannen et al. (2019). The projection function $k(.)$ is implemented with a two-layer multilayer perceptron (MLP). We define also τ as a temperature parameter.

In our framework, for each iteration we generate three graph views by randomly corrupting the original graph, denoted as $\tilde{G}_1$, $\tilde{G}_2$ and $\tilde{G}_3$. We distinguish between

two types of graph data augmentation namely, positive augmentation denoted as f^+, and negative augmentation f^-. We define positive augmentation as a corruption technique that does not harm graph structure or node features by creating unexciting edges or fake node features. For positive augmentation, we adopt stochastic edge dropping and node feature masking. In contrast, we define negative augmentation as a corruption technique that may alter or harm the structure or node features. In this case, we employ random node shuffling for negative augmentation. Then, we apply a graph neural network encoder to generate latent space representation for each view. Next, we adopt a contrastive objective to maximize mutual information between the generated graph views. The adaptability of our function g to different roles depending on the self-supervision level allows the proposed framework to effectively tackle various tasks and abstraction levels within the graph representation learning process. As we have symmetric views, we update the model parameter by maximizing the objective in Eq. 3.7. The learning algorithm is summarized in Algorithm 1.

$$\mathcal{J} = \frac{1}{2N}\sum_{i=1}^{N}(\mathcal{L}(u_i, g^{(\text{abstraction-level})}(v_i)) + \mathcal{L}(v_i, g^{(\text{abstraction-level})}(u_i)) \tag{3.7}$$

Algorithm 1: The Proposed Training Algorithm

for $i \leftarrow 1$ **to** n **do**

Generate three graph views $\tilde{G}_1$, $\tilde{G}_2$ and $\tilde{G}_3$ by performing stochastic corruption on G.

Generate latent representation of each view using the GCN encoder.

Compute the contrastive objective $\mathcal{J}$ Eq. 3.7.

Update parameters by applying stochastic gradient ascent to maximize $\mathcal{J}$ with Eq. 3.7.

end

Node level:

At the node-level, we begin by creating two positive augmentation views through edge dropping and node feature masking. Additionally, we generate a negative augmentation view using node shuffling. Next, we use a graph neural network to encode these views and produce node embeddings. To optimize the model, we employ a contrastive loss function. This function ensures that the encoded embeddings of each node in the two positive views agree with each other, while also being distinguishable from embeddings of other nodes in both the positive and negative augmented views.

Proximity level:

In the proximity-level abstraction, we aim to enhance the similarity between a node and its close neighbors while reducing the similarity with more distant nodes. To achieve our goal, we train our proximity-based model using three corrupted graph views created using our graph augmentation strategies. Then, we input those graph views into our GCN encoder. The key to optimizing the model lies in the contrastive loss objective. This objective maximizes the similarities between a node and its neighbors while minimizing the similarities with nodes that are farther away. This process helps the model to capture and emphasize the local structure and relationships within the graph. To define node proximity or neighbors, we can use either a random walk approach or a more straightforward method like node i^{th} order node connection.

Cluster level:

In the cluster-level abstraction, our goal is to enhance the similarity among nodes within the same cluster while reducing the similarity between nodes belonging to other clusters. To achieve this goal, we utilize our graph augmentation techniques to create three distinct data perspectives. Subsequently, we generate node embeddings by leveraging a graph neural network. The contrastive objective comes

into play to maximize the similarities among nodes within the same cluster and, at the same time, minimize the similarities between nodes from different clusters. This process helps to create more distinct and informative cluster representations. To obtain the clustering labels, we can employ traditional clustering algorithms such as K-means or spectral clustering.

Graph level:
In the graph-level abstraction, we repeat the same procedures by creating corrupted graph views and generating node embedding. Then, we employ a contrastive loss function to maximize the similarity between a node and its positive augmented graph and minimize the similarity with its negative augmented graph.

3.3 Independent training

To study the transition between different self-supervision abstraction levels from geometric perspective, we adopt an independent training protocol consisting of two main steps: pretraining and fine-tuning. This protocol, commonly employed in transfer learning scenarios, enables us to take advantage of learned information effectively. During pretraining, we specifically train our model's embeddings on a given pretext task at different abstraction levels, including node-, proximity-, cluster-, and graph-levels. Subsequently, we fine-tune the pretrained model to adapt it to another level of abstraction.

To evaluate the geometric configuration of the manifolds, we compute their Intrinsic Dimension (ID) and Linear Intrinsic Dimension (LID). The ID metric measures the dimension of the embedded manifolds, providing an estimate of the minimum number of parameters required to accurately represent the latent space's underlying structure. However, estimating ID becomes more complex in the presence of curved manifolds and non-uniform point distributions. To address this challenge,

we use the approach presented in Facco et al. (2017) to effectively estimate ID and LID such that:

Let $X = \{x_i\}_1^N$ a set of N points uniformly sampled from a data manifold, whose intrinsic dimension is equal to d. Let $\mu_i = \frac{r_2(i)}{r_1(i)}$ where $r_1(i)$ and $r_2(i)$ are the distances between the sample x_i and its first and second nearest neighbors, respectively, among the set X. We can derive the intrinsic dimension d from the Pareto distribution $F(\mu_i|d)$ as follow:

$$d = \frac{\log(1-F(\mu_i))}{\log(\mu_i)}, F(\mu_i|d) = (1 - \mu_i^{-d})\mathbb{I}_{[1,+\infty[} \tag{3.8}$$

To estimate LID, we can use PCA (Principal Component Analysis) to identify the principal components (eigenvectors of the data's covariance matrix) that spans the subspace with the minimal projection error as Ansuini et al. (2019). Next, we evaluate the performance of our model after the pretraining and fine-tuning steps on downstream tasks, specifically node classification and clustering. These evaluations help us gauge the quality and effectiveness of the embeddings produced by our trained model in practical application scenarios.

3.4 Filtering mechanism

In the next chapter, we will delve into the experimental results that demonstrate the transition between different levels of self-supervision. We will provide further details about the Feature Twist problem, which arises during this transition and adversely affects the quality of the generated embeddings, consequently impacting the performance of downstream tasks. The Feature Twist problem is characterized by an abrupt geometric transformation when moving from one level of self-supervision to another. This sudden shift in the geometric configuration of the embeddings can be detrimental to the model's ability to capture meaningful

patterns and relationships, leading to lower performance in downstream tasks.

We propose a filtering mechanism to address and mitigate the negative effects of Feature Twist. This mechanism aims to smooth Feature Twist, particularly during the transition to proximity level and cluster level. By applying this filtering technique, we intend to enhance the quality of the embeddings across different abstraction levels, ultimately improving the model's performance and adaptability in diverse tasks.

3.4.1 Proximity level filtering strategy

In the pretraining step, we use the graph information to define the neighbors of the nodes using a random walk strategy to optimize the proximity level objective function. This process helps us capture the local connectivity and relationships within the graph. In the subsequent fine-tuning step, specifically during the transition to the proximity level, we can utilize the embeddings learned in the pretraining phase to define the node proximity on the latent space. To achieve this, we create a notion of node proximity that incorporates both the original graph structure and the proximity in the learned latent space. One way to define node proximity is by using the Euclidean distance.

To further improve the model's performance and address the Feature Twist problem characterized by an abrupt geometric transformation and deterioration of model performance, we adopt a more powerful filtering strategy. This strategy employs a threshold-based filter in which nodes with Euclidean distance below a specific threshold are supposed to close or be trusted neighbors of the target node as described in Eq. 3.9. By implementing this filtering process, we can split the node set into two categories: nodes with trusted neighbors in the latent space and nodes without trusted neighbors. Then, we train nodes with trusted neighbors us-

ing a proximity level loss while the remaining node uses the same loss function as the pre-training step. Our approach is grounded in the understanding that not all nodes interconnected within a topological graph structure demonstrate uniform behavior. In scenarios where we incorporate information from untrusted neighbors, there is a risk of incorporating misleading information to the target node during training. So, we perform a proximity loss for only nodes with confident neighbors.

$$Neighbors(u) = \{v \in \mathcal{V}, \|\mathbf{u} - \mathbf{v}\|_2 \leq \lambda\} \tag{3.9}$$

λ: Threshold to identify confident neighbors.

$\mathbf{u}$ and $\mathbf{v}$ are the latent space representation of the target node u and node v.

3.4.2 Cluster level filtering strategy

Previously, to fine-tune models using cluster-level loss, we applied the K-means algorithm on the generated embeddings to identify centroids and cluster nodes. K-means is a simple yet efficient machine-learning clustering algorithm. However, its performance on high-dimensional data is often unsatisfactory, mainly when we have a curved embedding structure. Thus, to improve the transition to the cluster level, we made a significant change to the previous method. Firstly, instead of using K-means to determine clusters and centroids, we now employ a deep clustering strategy based on the Student's t-distribution method. The key idea behind this soft clustering mechanism is to learn both the cluster assignments and the cluster centroids simultaneously through an iterative process. So, we initialize the cluster centers μ_j in the latent space randomly or by applying K-means on the pre-trained embeddings from the previous self-supervision level. Then, the similarity between the embedded point h_i and cluster centers μ_j is measured by

Student's t-distribution, as follows:

$$q_{ij} = \frac{(1 + \|\mathbf{h}_i - \mu_j\|^2)^{-1}}{\sum_k (1 + \|\mathbf{h}_i - \mu_k\|^2)^{-1}} \tag{3.10}$$

where q_{ij} denotes the soft clustering assignments, it is the probability of assigning an embedded point $\mathbf{h}_i$ to the the centroid μ_j.

Subsequently, we integrated a filtering strategy inspired by the concept presented in Mrabah et al. (2022b). The challenge lies in the fact that during the self-supervision training process, we cannot precisely define the correct cluster centroids and the nodes within each cluster. As a result, there is an accumulation of errors due to learning with noisy clustering assignments, which can degrade the effectiveness and robustness of the clustering model. To address this issue, we apply a filtering strategy to identify nodes with reliable clustering assignments. The filtering strategy involves dividing the node set into two subsets: trusted nodes and untrusted nodes. Trusted nodes refer to those whose clustering assignments at a specific iteration t are considered accurate enough to confidently decide to which cluster they belong. On the other hand, untrusted nodes have uncertain clustering assignments and are not used for the next training iteration. By training our model based only on the subset of trusted nodes, we focus on more reliable and accurate cluster assignments, reducing the impact of noisy clustering information.

To establish trusted nodes within each cluster, we implement a strategy centered around two essential hyperparameters. The first parameter sets an upper limit on the allowable distance between a node's representation and the centroid representation of the cluster in the embedding space. This distance criterion determines whether a node qualifies for membership in a given cluster. In contrast, the sec-

ond threshold serves a distinct purpose: it acts as a penalty for nodes situated in close proximity to two different cluster centroids simultaneously, thereby addressing conflicts that might arise. Let $\Omega(t)$ represent the set of confident nodes at iteration t.

$$\Omega(t) = \{i \; in \mathcal{V} \quad | \quad \lambda_i^1 \geq \alpha_1 \quad and \quad (\lambda_i^1 - \lambda_i^2) \geq \alpha_2\} \tag{3.11}$$

where: $\alpha_1 \, and \, \alpha_2$ are two hyper-parameters.

$\lambda_i^1 = \max_{j \in \{1 \ldots K\}}(q_{ij})$ is the first high-confidence clustering assignment.

$\lambda_i^2 = \max_{j \in \{1 \ldots K\}}(q_{ij} \,|\, q_{ij} < \lambda_i^1)$ is the second high-confidence clustering assignment.

Algorithm 2 summarizes all the steps used to perform the fine-tuning with a filtering strategy during the transition from node level to proximity level and the transition from node level to cluster level.

3.5 Conclusion

This chapter provided a comprehensive overview of our proposed framework for studying self-supervision using different abstraction levels. We began by presenting our graph augmentation techniques, followed by the strategies for graph representation learning and the proposed contrastive objective. Additionally, we discussed the training protocols and evaluation metrics used to address the feature twist problem. Finally, we introduced our filtering solution designed to enhance the transition and mitigate the Feature Twist effect.

Algorithm 2: The Proposed Fine-Tuning Algorithm with Filtering

Load the saved training weights from the node-level model

for $i \leftarrow 1$ **to** n **do**

Generate three graph views $\tilde{G}_1$, $\tilde{G}_2$ and $\tilde{G}_3$ by performing stochastic corruption on G.

Generate latent representation of each view using the GCN encoder.

if *model == cluster-level* **then**

Select trusted nodes using Eq. (3.11).

Compute the contrastive objective $\mathcal{J}$ Eq. (3.7) using trusted nodes.

end

else if *model == proximity-level* **then**

Split the node set into nodes with trusted neighbors and nodes without trusted neighbors using Eq. (3.9).

Apply a node-level loss for nodes without trusted neighbors and a proximity-level loss for nodes with trusted neighbors Eq. (3.5).

Compute the contrastive objective $\mathcal{J}$ Eq. (3.7).

end

Update parameters by applying stochastic gradient ascent to maximize $\mathcal{J}$ with Eq (3.7).

end

CHAPTER IV

EVALUATION OF THE PROPOSED APPROACH

In this chapter, we conduct an experimental evaluation to assess the quality of the node embeddings generated at different abstraction levels. We apply these embeddings to node classification and clustering tasks using four well-known benchmark datasets: Cora, CiteSeer, PubMed, and DBLP. Then, we present the independent training protocol results that enable us to gain insights into the behavior of manifolds during the transition between various abstraction levels. Finally, we show our employed filtering technique results used to mitigate the effects of the Feature Twist problem.

4.1 Dataset details

In this study, we employ four citation network datasets Sen et al. (2008): Cora, CiteSeer, DBLP, and PubMed. These datasets form the basis for our investigation into the transition between abstraction levels and the evaluation of the model's performance on downstream tasks.

- Cora: The Cora dataset is a citation network for scientific publications in computer science. It comprises papers as nodes, with edges denoting citation relationships among them. Each node in the graph is linked to a bag-of-

words representation of its content, rendering it well suited for text-based classification tasks.

- CiteSeer: Similarly to Cora, CiteSeer is another citation network dataset focused on scientific literature. It includes documents from various disciplines covering computer science, biology, and other domains. The dataset's structure consists of papers connected by citation links, and, like Cora, it is often used for document classification tasks.

- DBLP: The DBLP dataset is a citation network sourced from the Digital Bibliography & Library Project (DBLP). It encompasses computer science-related publications, including conference papers and journal articles. The dataset provides citation relationships between different publications. It is widely used for various research tasks in the computer science domain.

- PubMed: The PubMed dataset is a citation network that primarily comprises biomedical publications. It is specifically related to the field of life sciences and medicine. It encompasses articles sourced from PubMed, a widely used database within the biomedical research community. The dataset contains citation links between scientific publications. It is often used for tasks such as document classification and citation recommendation.

These datasets have become popular benchmarks in the research community for studying graph-based representation learning, citation analysis, and other machine learning tasks. They offer diverse domains, sizes, and characteristics, making them valuable resources for evaluating the performance and generalizability of machine learning models. Table 4.1 below provides us with summary statistics about the datasets, such as the number of nodes, feature dimensions, edges, classes, and node connectivity.

Table 4.1 Statistics of datasets used in experiments.

Data	**Nodes**	**Edges**	**Features**	**Classes**	**Isolated nodes**	**Avg node degree**
Cora	2,708	5,429	1,433	7	NO	3.90
CiteSeer	3,327	4,732	3,703	6	YES	2.74
DBLP	17,716	105,734	1,639	4	NO	5.97
PubMed	19,717	88,648	500	3	NO	4.50

4.2 Hardware and software configurations and hyperparameters settings

All experiments are performed on a Linux server under the same hardware and software environments. The specification of the software libraries and frameworks as well as the hardware are provided in Table 4.2. The hyperparameters for our models include the embedding dimension (128 or 256), graph augmentation scheme (uniform or adaptive augmentation), graph augmentation probability, activation function, optimizer (we use Adam for all experiments), learning rate (0.01 for small datasets and 0.001 for larger datasets), and the activation function (ReLU or PReLU). For the proximity-level model, we also need to define the number of neighbors to consider.

4.3 Pre-training results

Our primary objective is to investigate self-supervision from geometric perspectives, focusing on exploring the transition behavior between different levels of abstraction. To accomplish this goal, we start by pretraining the four abstraction levels, node-level, proximity-level, cluster-level, and graph-level, using the previously defined contrastive objective. Ensuring the effectiveness of each model during the pretraining phase is crucial before advancing to the subsequent steps. This step will establish a robust foundation for the remainder of our experimentations. To facilitate a fair and unbiased comparison later, we will train all four

Table 4.2 Hardware and software used for all the conducted experiments.

Hardware	
RAM	132GB
CPU model	Intel(R) Xeon(R) CPU E5-2620 V4 @ 2.10GHz
Number of CPUs	32
GPU model	GeForce RTX 2080 Ti
GPU memory	11 GB
Number of GPUs	2
Software	
Operating System	Ubuntu 18.04.5 LTS
Python	3.9.16
Pytorch	1.13.0+cu117
Sklearn	1.3.0

Table 4.3 Evaluation results of our pre-training step for the four levels of abstraction: C-Acc stands for classification accuracy, and K-Acc stands for clustering accuracy.

Method	**Cora**		**CiteSeer**		**DBLP**		**PubMed**	
	C-Acc	**K-Acc**	**C-Acc**	**K-Acc**	**C-Acc**	**K-Acc**	**C-Acc**	**K-Acc**
Node Level	**0.8425**	0.7160	**0.7292**	0.6931	**0.8427**	0.7977	0.8433	0.6834
Proximity Level	0.8400	**0.7433**	0.7168	**0.6946**	0.8402	**0.8026**	0.8276	**0.6982**
Cluster Level	0.7957	0.5542	0.6945	0.5822	0.7956	0.6990	0.7772	0.6565
Graph Level	0.8178	0.5561	0.7212	0.6913	0.8311	0.7257	0.8040	0.6047
GRACE	0.8347	0.7023	0.7135	0.6705	0.8287	0.7934	**0.8523**	0.6843
DGI	0.8261	0.713	0.6872	0.688	0.8321	0.7752	0.768	0.5337

models under identical conditions, namely the same augmentation techniques and GCN encoders.

To conduct the experimental evaluation, we adopt a linear evaluation scheme which involves two key steps. Firstly, each model undergoes unsupervised training on a "pretext task" to learn latent representations. Subsequently, we utilize these

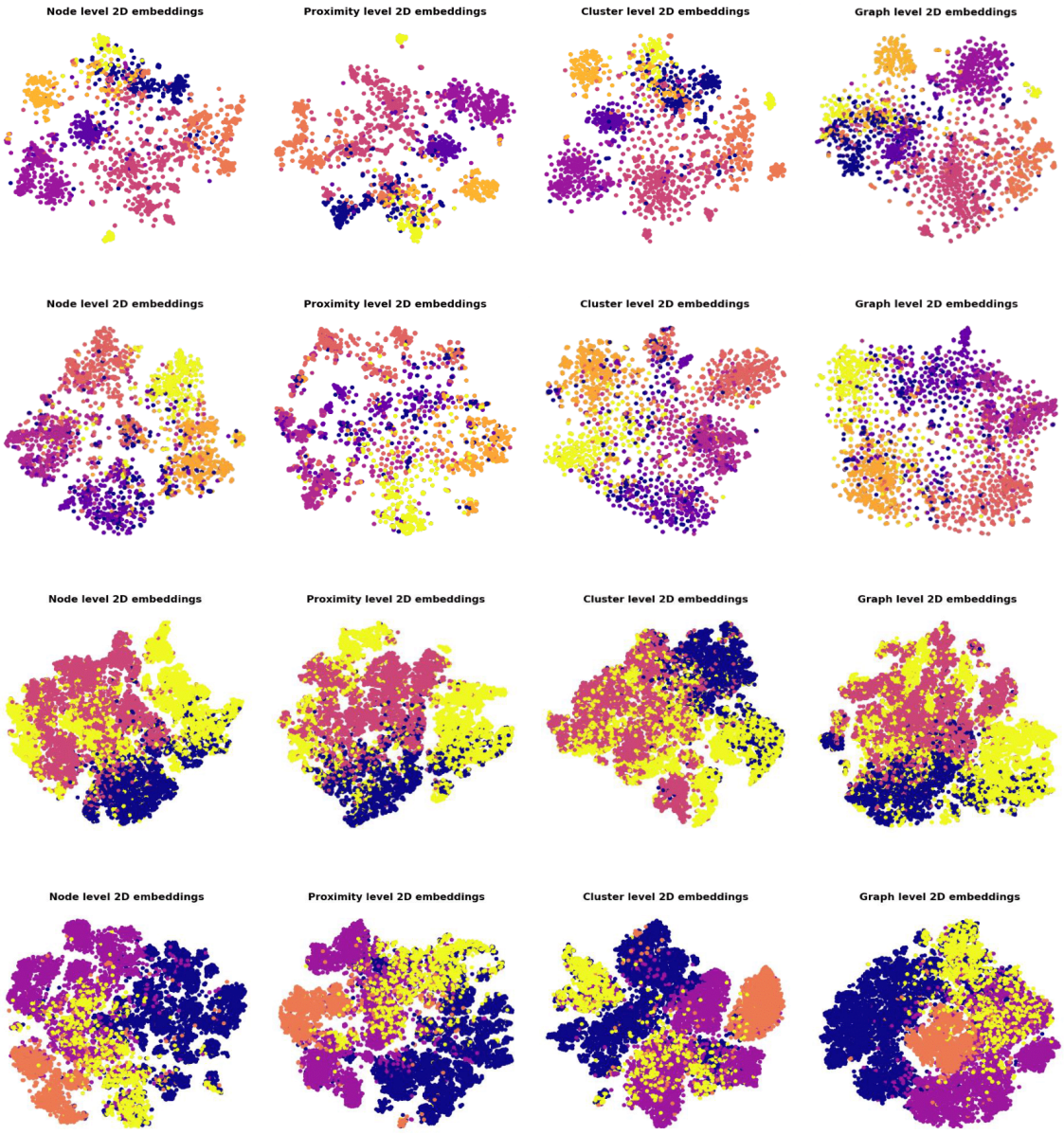

Figure 4.1: 2D t-SNE projection of the latent space representation of Cora, CiteSeer, PubMed and DBLP Datasets. The legend corresponds to 4 rows: Cora, CiteSeer, PubMed and DBLP with each row containing four columns, from left to right: node-level, proximity-level, cluster-level and graph-level. The embeddings are colored according to the different classes of each dataset.

generated latent representations to train and evaluate a logistic regression classifier for the downstream task. We assess the model's performance using classification accuracy and clustering accuracy.

At the proximity-level abstraction, we leverage random walk techniques to capture local node relationships and contextual information. By generating node proximities through random walks, we create informative neighborhood structures. Our training objective function aims to maximize the similarities between a node and its neighboring nodes while minimizing the similarities with unrelated nodes. Moving on to the cluster-level model, we adopt the K-means algorithm to group nodes into meaningful clusters. This clustering step enables us to capture higher-order graph patterns and distinct node clusters. During training, we seek to maximize the similarities between a node and its corresponding cluster centroid while minimizing the similarities with nodes belonging to the extra clusters. Finally, to train the graph-level model, we introduce a readout function to generate a comprehensive graph representation. This readout function aggregates information from all nodes to create a global graph representation.

As illustrated in Table 4.3, the results underscore the efficacy of our proposed framework in effectively modeling constructive loss at various levels of abstraction. Particularly noteworthy is the performance of our graph-level model, which closely rivals that of the state-of-the-art Deep Graph Infomax (DGI) model. Furthermore, our node-level model showcases comparable performance to the leading model in its category by consistently demonstrating slight improvements over the state-of-the-art across three datasets: Cora, CiteSeer, and DBLP. Furthermore, our proximity-level model has shown promising results and gives us the best clustering performance on the four datasets, which validates the effectiveness of our approach in capturing local node relationships. However, we acknowledge that our cluster-level model's performance falls short compared to previous models. We anticipated

this outcome due to the inherent challenge of effectively defining cluster centroids during the pretraining phase. Consequently, the model may suffer from cumulative errors arising from noisy label assignments. Despite the limitations observed in the cluster-level model, our overall results demonstrate the potential and efficacy of our self-supervised contrastive learning framework.

Figure 4.1 shows the t-SNE projection of the latent space representation of the Cora, CiteSeer, PubMed, and DBLP datasets for our four models. The node-level and proximity-level models provide good embedding representation, particularly for Cora, CiteSeer, and DBLP, where we can observe coherence between nodes with the same label. The embedding quality of the graph-level and cluster level is slightly worse compared to the two previous models. Mainly, we have a closer distance between nodes from different clusters.

4.4 Feature Twist

In the following experiments, we delve into the transition across various levels of abstraction from a geometric perspective. Our exploration is based on the average Intrinsic Dimension (ID) and Local Intrinsic Dimension (LID) estimations derived from principal component analysis. Additionally, we evaluate the efficacy of our models in classification as a downstream task using accuracy as a metric.

As we can see in Figure 4.2, the node-level model demonstrates high accuracy in downstream classification tasks, accompanied by a consistent decrease in the average ID and LID values throughout the pretraining phase. The LID consistently surpasses the ID, indicating the presence of underlying linear structures in the embedding space that the ID alone fails to capture. However, as we transit from node level to higher levels, a concerning deterioration of geometric manifolds becomes evident. This degradation is reflected in an increasingly abrupt increase

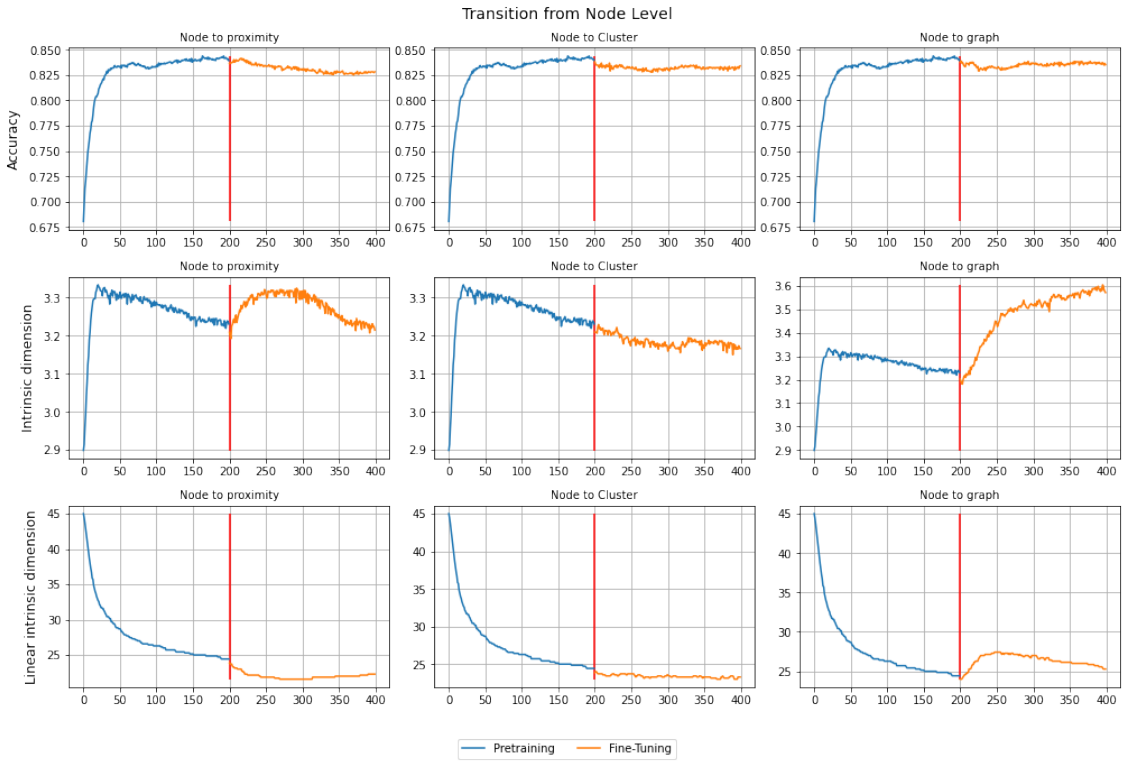

Figure 4.2: Evaluation of the self-supervised transition from the node level to other levels of abstraction for the Cora dataset.

in the ID, resulting in a stark contrast between the ID and the LID. Consequently, our observed low-dimensional manifolds remain curved. This transition adversely affects the performance of the downstream tasks, particularly at the proximity and cluster level. We posit that the application of filtering techniques holds the potential to enhance these outcomes. We intend to delve deeper into these techniques and their impact on the results, which we will present in subsequent sections.

As depicted by Figure 4.3, the proximity-level model delivers notable classification outcomes in downstream tasks, even in the presence of intricate curved low-dimensional manifolds. Notably, we observe a gradual shift from proximity to nodes, which appears to have a marginal impact on reducing the ID. However, as we progress to higher abstraction levels like graph and cluster, the average

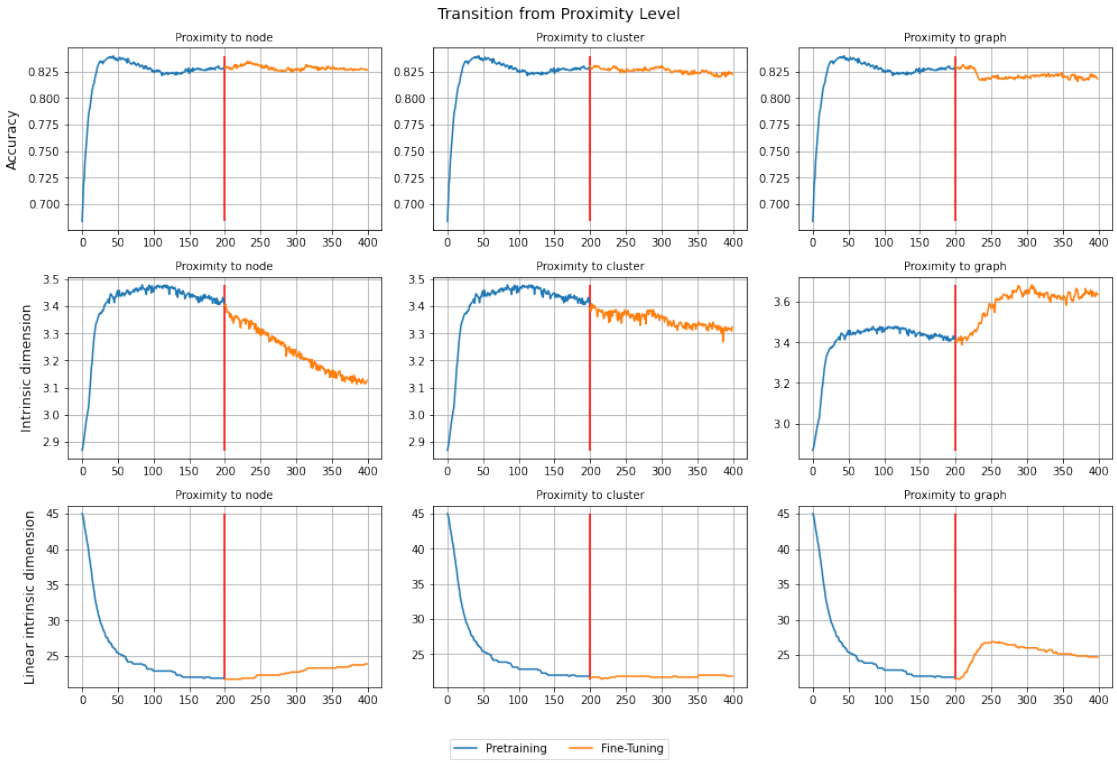

Figure 4.3: Evaluation of the self-supervised transition from the proximity level to other levels of abstraction for the Cora dataset.

ID tends to escalate. Overall, transitioning from proximity to other abstraction levels does not enhance performance in downstream tasks, and the curved low-dimensional manifold persists, which indicates the presence of a Feature Twist problem.

The cluster-level model exhibits lower accuracy results compared to the node- and proximity-level models, as shown in Figure 4.4. Additionally, it demonstrates a higher Linear Intrinsic Dimension compared to the preceding models. This difference is due to the issue of error propagation that arises from incorrect clustering assignments. We generated cluster centroids using the K-means model applied to untrained embeddings, which cannot provide precise cluster centroids. We then aimed to maximize the similarity between each node and its respective cluster

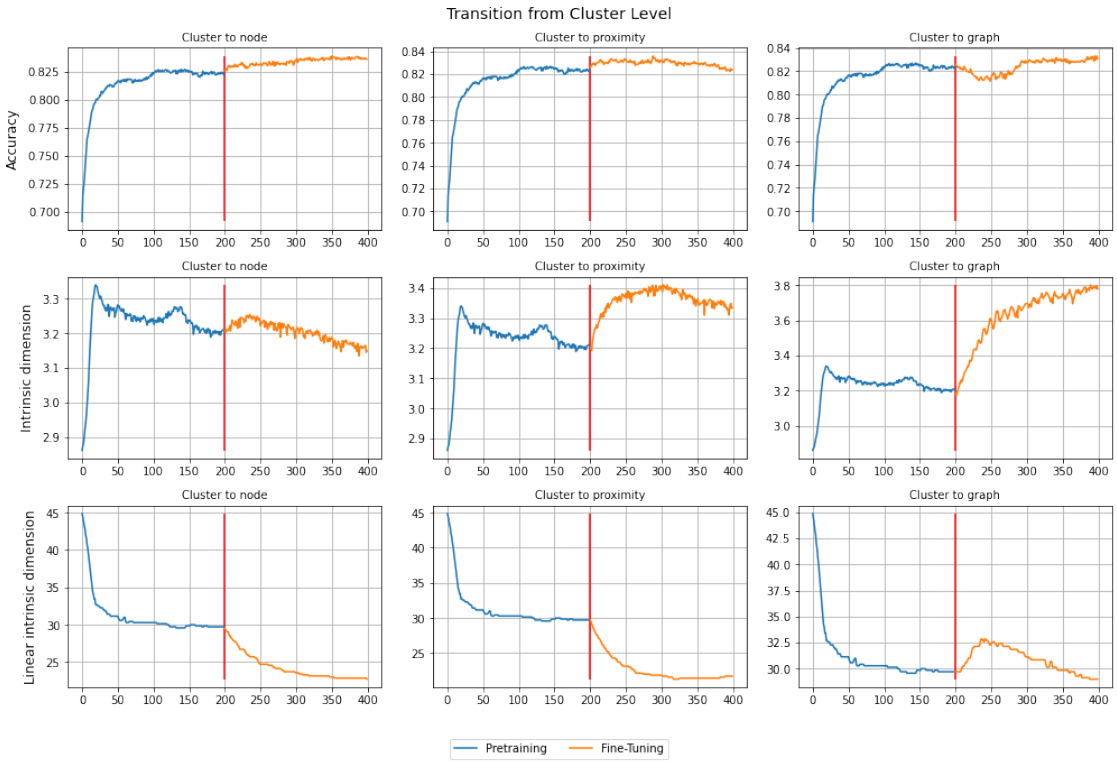

Figure 4.4: Evaluation of the self-supervised transition from the cluster level to other levels of abstraction for the Cora dataset.

centroid while minimizing the similarity with other cluster centroids. Transitioning from the cluster level to other levels can enhance model accuracy, but it still lags behind the results of the other models. Also, we observe a sudden geometric transformation in ID and LID. Overall, it is better to apply clustering loss after a pre-training step to mitigate the adverse effects of inaccurate clustering assignment propagation.

The performance of the graph-level model, as depicted in Figure 4.5, exhibits a slight lag behind both node-level and proximity-level models in the classification tasks. In particular, an intriguing observation emerges from the higher LID value within this model compared to the LID values of node-level and proximity-level models. Importantly, this LID value shows a consistent decrease during the

pretraining phase. In contrast, the Intrinsic Dimensionality during graph-level training displays a less discernible pattern. It fluctuates around a relatively constant value without showing a clear monotonic trend. During the transition from the graph level to distinct levels, an evident potential for enhancing classification accuracy becomes clear. However, despite this potential, the achieved results still fall short compared to the performance of node-level and proximity-level models. Also, an abrupt transition of average ID becomes evident in the initial iterations, accompanied by a consistent decrease in LID. This observation underscores the evolving nature of the model's latent space representation. Interestingly, the transition to proximity-level stands out due to their closely aligned ID and LID values compared to the previous models. This alignment suggests a more flattened and evenly distributed latent space representation during the transition to proximity level.

In summary, the experimental findings indicate the existence of a prominently curved latent representation within the models. Furthermore, a distinct Feature Twist phenomenon is evident, marked by a sudden geometric shift between different levels of abstraction. Although some transitions exhibit a smoother geometric transformation, they do not yield noticeable enhancements in downstream task performance. In the following section, we discuss our proposed filtering mechanism performance. This mechanism aims to enhance the results and address the Feature Twist problem, offering a potential solution to mitigate the observed challenges.

4.5 Protection mechanism against Feature Twist

In this section, we present the results of our transition process from the node level to the proximity and cluster levels, employing a filtering mechanism. The

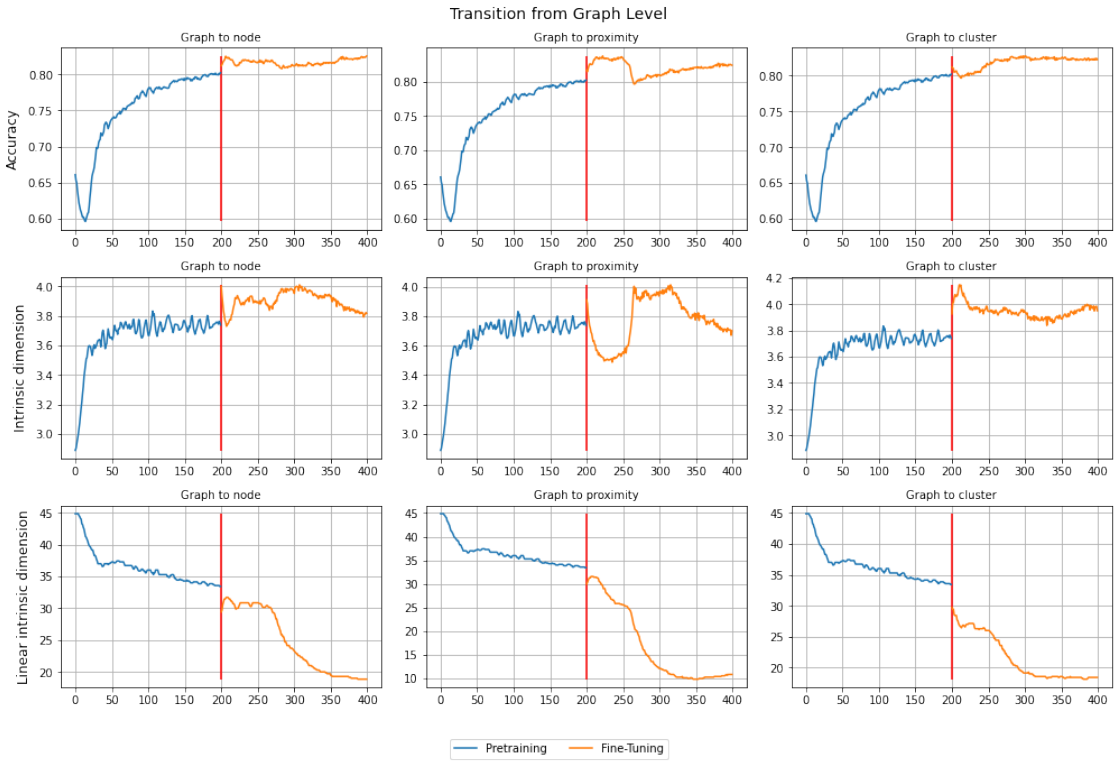

Figure 4.5: Evaluation of the self-supervised transition from the graph level to other levels of abstraction for the Cora dataset.

choice of starting with the node level as a pretraining step is embedded in its demonstrated superiority in both classification and clustering, as highlighted in the previously shared results.

As a high-level overview of our intuition behind the filtering mechanisms, our aim is to use the performance achieved during the pretraining phase to effectively improve the fine-tuning step. To address the potential of encountering the Feature Twist effect during the transition from pretraining to fine-tuning, we use a filtering strategy that involves dividing the initial node set into two categories. This strategy keeps training the nodes in the first category utilizing the loss used in the pertaining step. The second category of nodes is trained using a loss function of different abstraction levels. This differentiation arises from the understanding

that not all nodes are well suited to contribute to the cluster or proximity level losses. Therefore, it is better to use node level loss in the pretraining step because it is not worth training the first category using an abstraction level that does not yield the best results.

4.5.1 Node level to proximity level improvement

Previously, we did not leverage the learned embeddings during the transition from node level to cluster and proximity levels. To optimize our previous findings, we introduced a filtering approach. This filter selects nodes based on their Euclidean distances in latent space by employing a predetermined threshold. Subsequently, we direct the training of selected nodes towards the proximity level objective, while nodes not chosen undergo training using the node level objective. This strategy helps us to enhance the previous results, as shown in Figure 4.6. It yields a smoother transition in both ID and LID values between the two self-supervised phases, concurrently boosting clustering and classification accuracy. A closer examination in Figure 4.7 accentuates the accuracy improvements achieved through our filtering methodology. Our filtering technique outperforms not only the proximity-level model but also the non-filtered node-to-proximity transition model and the node-level model. Also, when examining the 2D t-SNE embedding representation in Figure 4.9 of the node-to-proximity model post the application of the filtering mechanism, a discernible enhancement becomes evident. Within this improved representation, nodes sharing the same label exhibit a heightened proximity to each other, establishing a distinct cluster, while nodes with differing labels maintain a greater distance from one another. This refinement is notably more pronounced compared to both the original proximity model and the node-to-proximity representation without applying the filtering mechanism.

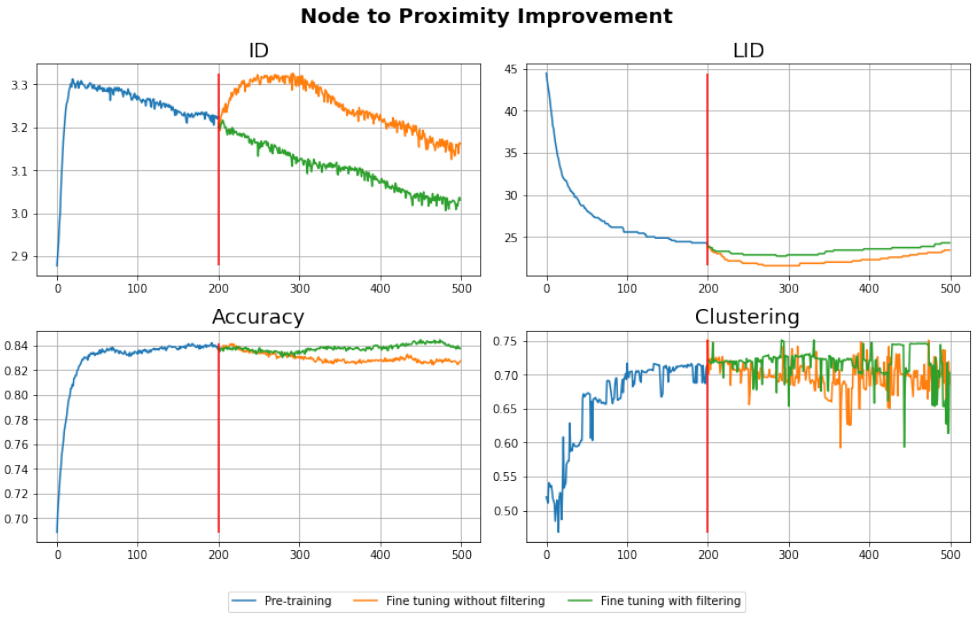

Figure 4.6: Node to proximity evaluation on Cora using our proposed filtering strategy.

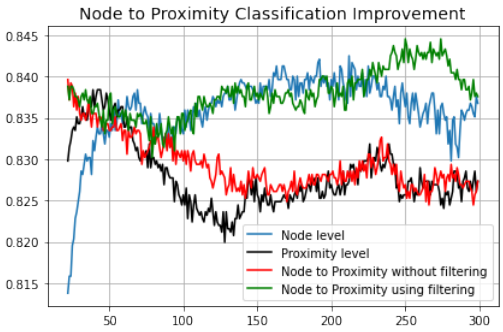

Figure 4.7: Node to proximity classification improvement on Cora using our proposed filtering strategy. The y-axis represents the classification accuracy, and the x-axis denotes the number of iterations.

In summary, the efficacy of our filtering strategy lies in its ability to acknowledge that not all nodes possess crucial informative neighbors for propagating meaningful information. As a result, our process discerns which nodes are better suited for training with proximity loss, while others are more aptly trained solely using the node-level loss.

4.5.2 Node level to cluster level improvement

To enhance the efficacy of the transition from node to cluster level, we make some refinements to our previous methodology. Departing from the K-means used for cluster determination and centroid identification, we use a soft clustering approach rooted in deep clustering employing the Student's t-distribution method. Furthermore, we introduced a filtering mechanism inspired by the concept expounded in Mrabah et al. (2022b), aiming to intelligently select the most relevant nodes for the clustering objective. The integration of this strategy can improve the previous results, as illustrated in Figure 4.8.

During the initial epochs of the transition process, we observed a slight dip in performance compared to our previous methods. This temporary decline can be attributed to the fact that during this early phase of transition, we lack a substantial number of trusted nodes to effectively train the model. As a result, the model performance may not be at its optimal level at this stage. However, as the transition progresses, our approach begins to showcase its strengths. Over time, our strategy demonstrates the capacity to surpass the performance of the previous cluster-level models. Furthermore, when observing the 2D embedding representation of t-SNE in Figure 4.9 of the node-to-cluster model after applying the filter, a distinct enhancement becomes apparent compared to the cluster-level model and node-to-cluster before filtering. In this refined representation, nodes sharing the same label are closer to each other, effectively increasing the cohesion within cluster groups. At the same time, these label-aligned nodes are noticeably more distant from dissimilar label nodes.

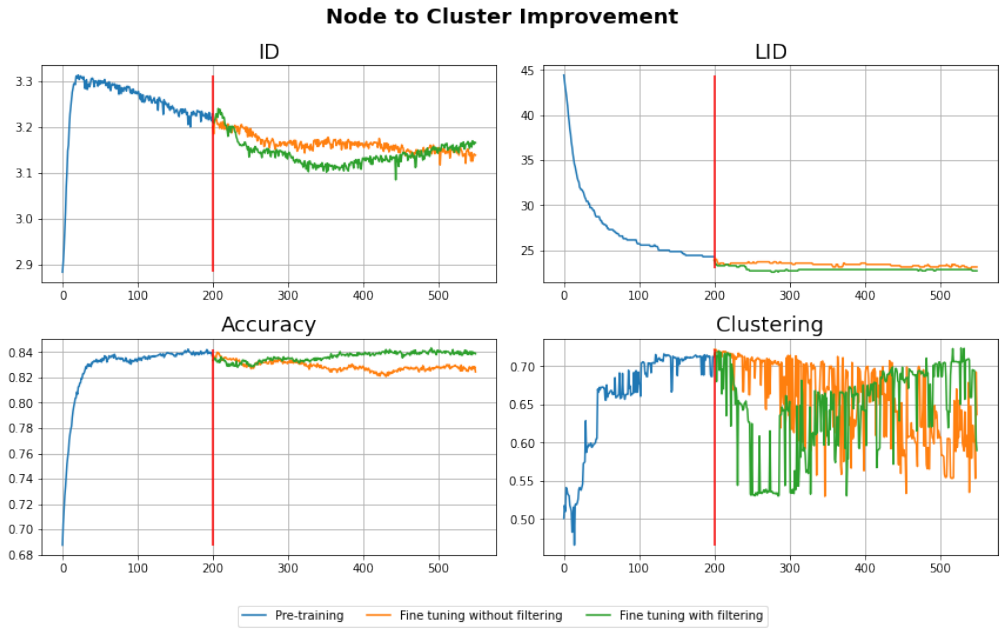

Figure 4.8: Node to cluster evaluation on the Cora using our proposed filtering strategy.

4.6 Conclusion

In this chapter, we provided a comprehensive overview of our training procedures. We presented the results of our pretraining phase, showcasing the effectiveness of our framework in modeling the different abstraction levels. Through experimentation, we demonstrated the existence of the Feature Twist problem, which poses challenges during the transition between abstraction levels. Finally, we introduced our filtering mechanism, which effectively improves the quality of the results.

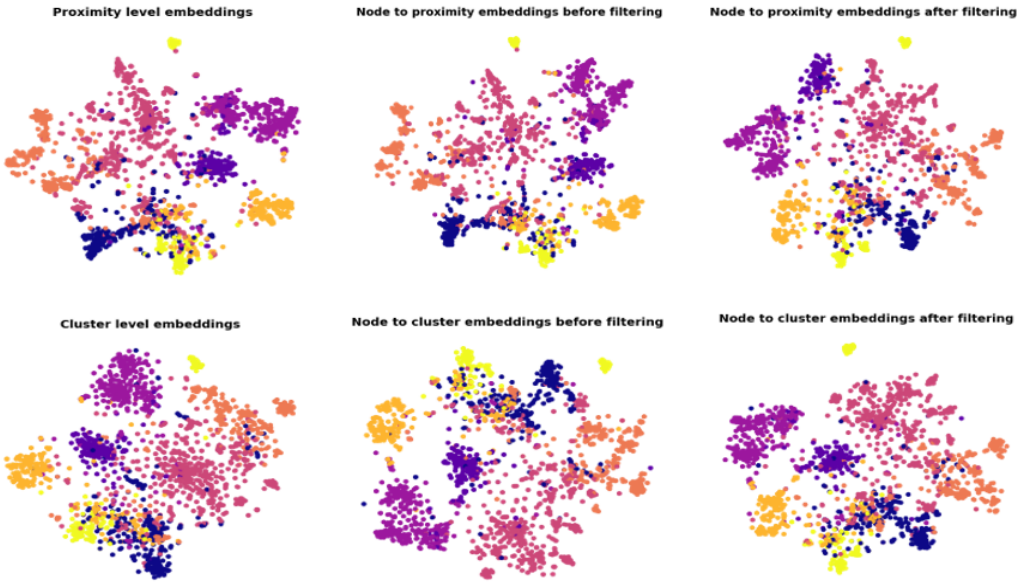

Figure 4.9: 2D t-SNE projection of the latent space representation of Cora. The first row present from left to right the proximity level model, the node to proximity model before applying filter and the node to proximity after applying filter. The second row present from left to right the cluster level model, the node to cluster model before applying filter and the node to cluster after applying filter.

CHAPTER V

CONCLUSION

This work established the first geometric exploration of the self-supervision paradigm. Our novel graph contrastive learning framework can train models at different abstraction levels, including node, proximity, cluster, and graph levels. The framework utilizes graph augmentation techniques to generate corrupted graph views, and by maximizing the agreement of representations between these views, we improve the quality of our learned embeddings.

To explore the transition between different abstraction levels in self-supervision, we adopted a pretraining-finetuning protocol. The performance of our models during the pretraining phase yielded competitive results, surpassing state-of-the-art methods. Using intrinsic dimension and linear intrinsic dimension as metrics, we identified the presence of the Feature Twist issue problem characterized by a sudden and abrupt shift between abstraction levels. To address this issue, we introduced a filtering mechanism to improve the transition, particularly at proximity and cluster levels.

To improve the transition to the proximity level, we adopted a threshold-based approach considering only nodes with distances below a certain threshold as proximity to the target node. To enhance the transition to the cluster level, we categorized nodes into trusted and untrusted nodes based on their distances to cluster

centroids. Then, we gradually trained the model using only a set of trusted nodes. These filtering techniques can help improve the model's performance on downstream tasks. By addressing the Feature Twist problem, we enhance the models' adaptability and effectiveness, leading to more reliable and robust embeddings. This research opens new avenues for understanding the interplay between self-supervision and geometric properties in graph representation learning.

As part of a future perspective, we aim to investigate the competition between self-supervision abstraction levels from a geometric perspective using a joint training strategy. By adopting a joint training approach, we can explore how different levels of self-supervision interact and complement each other in the learning process. This investigation will help us better understand the interplay between the various abstraction levels and how they collectively contribute to the overall model performance.

BIBLIOGRAPHY

Ansuini, A., Laio, A., Macke, J. H. & Zoccolan, D. (2019). Intrinsic dimension of data representations in deep neural networks. *Advances in Neural Information Processing Systems*, *32*.

Belghazi, M. I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A. & Hjelm, D. (2018). Mutual information neural estimation. In *International Conference on Machine Learning*, pp. 531–540. PMLR.

Bo, D., Wang, X., Shi, C., Zhu, M., Lu, E. & Cui, P. (2020). Structural deep clustering network. In *Proceedings of the Web Conference 2020*, pp. 1400–1410.

Chen, J., Ma, T. & Xiao, C. (2018). FastGCN: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations (ICLR)*.

Chen, T., Kornblith, S., Norouzi, M. & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pp. 1597–1607. PMLR.

Defferrard, M., Bresson, X. & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems*, *29*.

Donnat, C., Zitnik, M., Hallac, D. & Leskovec, J. (2018). Spectral graph wavelets for structural role similarity in networks.

Facco, E., d'Errico, M., Rodriguez, A. & Laio, A. (2017). Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific Reports*, *7*(1), 12140.

Grover, A. & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864.

Hamilton, W., Ying, Z. & Leskovec, J. (2017a). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, *30*.

Hamilton, W. L., Ying, R. & Leskovec, J. (2017b). Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, *40*(3), 52–74.

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

Hu, Z., Dong, Y., Wang, K., Chang, K.-W. & Sun, Y. (2020). Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1857–1867.

Ji, S., Pan, S., Cambria, E., Marttinen, P. & 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.

Jiang, H. & Huang, Y. (2022). An effective drug-disease associations prediction model based on graphic representation learning over multi-biomolecular network. *BMC Bioinformatics*, *23*, 1–17.

Kipf, T. N. & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, pp. 1–14.

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

Lukashina, N., Kartysheva, E., Spjuth, O., Virko, E. & Shpilman, A. (2022). Simvec: predicting polypharmacy side effects for new drugs. *Journal of Cheminformatics*, *14*(1), 1–12.

Mavromatis, C. & Karypis, G. (2020). Graph infoclust: Leveraging cluster-level node information for unsupervised graph representation learning. *arXiv preprint arXiv:2009.06946*.

McPherson, M., Smith-Lovin, L. & Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, *27*(1), 415–444.

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013). Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations (ICLR) Workshop*.

Mrabah, N., Bouguessa, M. & Ksantini, R. (2022a). Escaping feature twist: A variational graph auto-encoder for node clustering. *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3351–3357.

Mrabah, N., Bouguessa, M., Touati, M. F. & Ksantini, R. (2022b). Rethinking graph auto-encoder models for attributed graph clustering. *IEEE Transactions on Knowledge and Data Engineering*.

Murali, V., Muralidhar, Y. P., Königs, C., Nair, M., Madhu, S., Nedungadi, P., Srinivasa, G. & Athri, P. (2022). Predicting clinical trial outcomes using drug bioactivities through graph database integration and machine learning. *Chemical Biology & Drug Design*, *100*(2), 169–184.

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

Perozzi, B., Al-Rfou, R. & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. & Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80.

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

Thakoor, S., Tallec, C., Azar, M. G., Azabou, M., Dyer, E. L., Munos, R., Veličković, P. & Valko, M. (2022). Large-scale representation learning on graphs via bootstrapping. In *International Conference on Learning Representations (ICLR)*.

Tschannen, M., Djolonga, J., Rubenstein, P. K., Gelly, S. & Lucic, M. (2019). On mutual information maximization for representation learning. In *International Conference on Learning Representations (ICLR)*.

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y. & Hjelm, R. D. (2019). Deep graph infomax. *International Conference on Learning Representations (ICLR) (Poster)*, *2*(3), 4.

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

Wang, C., Pan, S., Hu, R., Long, G., Jiang, J. & Zhang, C. (2019). Attributed graph clustering: A deep attentional embedding approach. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3670–3676.

Wu, J., He, J. & Xu, J. (2019a). Demo-net: Degree-specific graph neural networks for node and graph classification. In *Proceedings of the 25th ACM International Conference on Knowledge Discovery and Data Mining*, pp. 406–415.

Wu, Z., Pan, S., Long, G., Jiang, J. & Zhang, C. (2019b). Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1907–1913. AAAI Press.

Xia, W., Gao, Q., Yang, M. & Gao, X. (2021). Self-supervised contrastive attributed graph clustering. *arXiv preprint arXiv:2110.08264*.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L. & Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 974–983.

You, Y., Chen, T., Wang, Z. & Shen, Y. (2020). When does self-supervision help graph convolutional networks? In *International Conference on Machine Learning*, pp. 10871–10880. PMLR.

Zhang, S., Tong, H., Xu, J. & Maciejewski, R. (2019). Graph convolutional networks: a comprehensive review. *Computational Social Networks*, *6*(1), 1–23.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S. & Wang, L. (2020). Deep Graph Contrastive Representation Learning. In *ICML Workshop on Graph Representation Learning and Beyond*. Retrieved from `https://arxiv.org/abs/2006.04131`

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S. & Wang, L. (2021). Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, pp. 2069–2080.