**Distribution Agreement**

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

Zheng Zhang Date

Representation Learning on Physical and Information Networks

By

Zheng Zhang
Doctor of Philosophy

Computer Science and Informatics

---

Liang Zhao, Ph.D.
Advisor

---

Andreas Züfle, Ph.D.
Committee Member

---

Li Xiong, Ph.D.
Committee Member

---

Wei Jin, Ph.D.
Committee Member

---

Huajie Shao, Ph.D.
Committee Member

Accepted:

---

Kimberly Jacob Arriola, Ph.D.

Dean of the James T. Laney School of Graduate Studies

_____

Date

# Representation Learning on Physical and Information Networks

By

Zheng Zhang
B.S., University of Science and Technology of China, Anhui, China 2017
M.S., College of William and Mary, VA, 2019

Advisor: Liang Zhao, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2024

Abstract

Representation Learning on Physical and Information Networks
By Zheng Zhang

Networks, encompassing both physical and information networks, are fundamental graph structures for modeling relationships among entities across diverse real-world applications. This thesis aims to advance general representation learning on network data by addressing several key challenges. Traditional graph representation learning methods primarily focus on topological structures, often neglecting the rich data modalities inherent in these networks and lacking theoretical guarantees on expressive power. Additionally, data quality challenges such as label scarcity, noisy data, and incompatibility between graph topology and other modalities hinder the development of robust and generalizable models.

For physical networks, we propose the spatial graph message passing neural network, a novel framework that seamlessly integrates spatial and topological information with theoretical guarantees on discriminative power. We enhance computational efficiency through an accelerated spanning tree sampling algorithm, reducing complexity from $O(N^3)$ to $O(N)$ while maintaining expressive capabilities. Furthermore, we extend the framework to accommodate networks embedded in irregular manifold spaces and generalize it to handle geometric trees, addressing the unique hierarchical structures in such data.

For information networks, we introduce a self-supervised learning framework called text-and-graph multi-view alignment. This framework unifies diverse data domains by leveraging text-attributed graphs, augmenting traditional graph structures with natural language descriptions. This framework incorporates a multi-view alignment module that preserves rich semantic information, topology, and their interplay. An accelerated algorithm reduces training time complexity from quadratic to linear, facilitating scalability to large datasets. We evaluate the framework's performance under label-scarce and transfer learning settings, demonstrating its effectiveness without reliance on extensive labeled data.

To enhance generalizability and robustness, we propose the relational curriculum learning method. This method improves representation learning on network data by addressing incompatibilities between graph topology and other data modalities. It introduces a novel edge selection criterion that quantifies the difficulty of understanding graph edges, incorporating them into the training process at appropriate times. Through extensive experiments on synthetic and real-world datasets, the proposed method demonstrates significant improvements in generalization ability and robustness.

In summary, this thesis presents novel frameworks and algorithms that advance representation learning on both physical and information networks. By providing theoretical guarantees, addressing data quality issues, and enhancing efficiency and scalability, these contributions hold significant implications for various downstream applications in chemistry, biomedicine, social sciences, and beyond.

# Representation Learning on Physical and Information Networks

By

Zheng Zhang
B.S., University of Science and Technology of China, Anhui, China 2017
M.S., College of William and Mary, VA, 2019

Advisor: Liang Zhao, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2024

Acknowledgments

I would like to express my deepest gratitude to my advisor, Dr. Liang Zhao. His meticulous and patient guidance enabled me to successfully complete my research, profoundly influencing my approach and way of thinking. I am also sincerely thankful to my committee members, Dr. Andreas Züfle, Dr. Li Xiong, Dr. Wei Jin, and Dr. Huajie Shao, for their support and insightful feedback. Their professional expertise significantly advanced my work.

My heartfelt appreciation goes to my collaborators and peers—particularly Chen Ling, Bo Pan, Yifei Zhang, Yuntong Hu, Shiyu Wang, Allen Zhang and Hossein Amiri. Their daily cooperation and discussions made my research process both supportive and collaborative. I am also grateful to the mentors and colleagues who greatly assisted me during my internships, especially Dr. Chen Zheng, Fan Yang, Ziyan Jiang, Dr. Zheng Chen, Dr. Zhengyang Zhao, Bowen Deng, Zach Fan, Dr. Hedi Xia, Dr. Jay Adams, and Raymond Hsu.

I owe an immeasurable debt to my mother, Qunkui Xia. Her selfless support over the years is then only foundation for me to achieve all of this. I also wish to thank my family members, especially my cousin Chang Wang, for their unwavering support. You have been my strongest foundation.

I want to express my deepest appreciation to my wife, Dr. Yawei Xiong. Her companionship and support during my doctoral studies helped me overcome numerous challenges and setbacks. Additionally, thank to my cat, Omega, for her companionship and, amusingly, her persistent efforts to distract me from studying.

I thank all those mentioned above and other friends and families for their care and support throughout my doctoral journey. This dissertation is not only a culmination of my own efforts but also a testament to the collective support I have received.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1

# Introduction

Networks, encompassing both physical and information systems, serve as essential data structures for modeling relationships among entities in a wide range of real-world applications [19, 73, 200]. Physical networks, such as transportation systems and biological networks, are embedded in the real world and consist of tangible entities and connections. Conversely, information networks represent abstract systems like information flow or social interactions, capturing intangible relationships that influence various phenomena. Combining these two types, networks provide a fundamental framework for capturing the complexities of interconnected data. In mathematical terms, networks are typically represented as graphs, where nodes signify entities and edges represent the connections or relationships between them [8, 149]. Additional attributes specific to the physical or informational context further enrich these graphs, leading to the term as modality-enriched graphs [58]. Discovering patterns and structures within these networks is essential for extracting insights and driving predictive, data-driven decision-making [194, 118].

While traditional studies on graph-structured data have a long history, they often rely on heuristic rules or domain-specific knowledge [42, 17]. These methods have limitations in handling diverse graph data and problems across various domains [16].

In recent years, the advancement of deep learning on graph-structured data has revolutionized the field of network analytics [114, 236]. Deep learning-based methods have demonstrated exceptional capabilities as a general strategy for learning powerful representations over diverse graph domain data [179], which is the core paradigm in the domain of learning the complex, non-linear relationships inherent in graph-structured data [210]. This has led to significant advancements in various graph problems, such as node classification [114], link prediction [78], graph classification [219], graph clustering [18], community detection [67], and graph generation [220].

Despite these advancements, existing methods often focus solely on studying the topological structures of networks while overlooking the importance of other rich data modalities inherent in physical and information networks [23, 14]. They typically treat additional modality information merely as plain node or edge attributes [74]. These approaches can be insufficient because other modalities can possess unique properties that require specialized handling and cannot be appropriately modeled as standard attributes [212]. For instance, physical networks are embedded in the real-world space, where nodes and edges are associated with spatial information such as coordinates. Proper handling of spatial information demands considerations of symmetry invariance or equivariance to accurately reflect geometric relationships inherent in the data [41, 170]. Moreover, other modalities may be intricately coupled with the graph's topology, necessitating the design of specific machine learning modules that can effectively model the interplay between the graph structure and modality-specific data properties [96, 223]. Addressing these challenges is crucial for developing more robust and generalizable graph representation learning techniques [210, 24].

Extracting powerful representations from modality-enriched physical and information network data is essential for understanding the underlying network mechanisms and performing a variety of downstream tasks such as biomedical property predictions [208], social network analysis [194, 68], recommendation systems [218], and

human mobility analysis [228, 229, 144, 145]. Achieving high-quality learned representations with deep learning models requires careful consideration of three main aspects: data, model, and tasks. Each aspect presents unique challenges, and effectively addressing these challenges is crucial for obtaining optimal representations. However, existing representation learning methods on graph data often struggle with these challenges in each perspective, highlighting the need for approaches that can integrate rich data modalities, advanced modeling techniques, and task-specific objectives to enhance performance.

First, there is a lack of considerations for interactions between graph topology and other data modalities. In real-world scenarios, both physical and information networks are enriched with additional modalities that describe nodes or edges, such as spatial coordinates, textual descriptions, or user demographic data. While graph structures capture the topological relationships among entities, these entities themselves and their connections are characterized by diverse forms of data. A significant challenge about learning representation over graph data is how to jointly handle the topological structure, the information from other modalities, and their crucial interplay. This integration is crucial because it allows for a more comprehensive understanding of the data, leading to more expressive and robust learned representations.

Secondly, there is an absence of theoretical guarantees about the expressive power of representations learned from modality-enriched physical and information network deep learning models. Considering the complexity and the exponentially large number of possible sub-structures within graphs, it is challenging to provide theoretical assurances on the models' ability to capture and distinguish these modular patterns effectively. The hierarchical and often overlapping nature of sub-structures adds further complexity to modeling efforts. Unfortunately, most existing graph deep learning models are treated as black boxes, with performance evaluations that are predominantly empirical and frequently overlook the influence of other modalities. Establish-

ing theoretical frameworks that explicitly account for modular structures in graphs is crucial to ensure a reliable and effective deep learning framework.

Third, the data quality issues introduced by modality-enriched physical and information networks are often overlooked. Most existing graph representation learning methods are developed under supervised learning settings, which assume the availability of high-quality, including large amounts of labeled data and straightforward node or edge attributes. However, in many real-world applications, modality-enriched physical and information networks involve diverse types of data that can be of relatively low quality or inconsistently structured. For instance, spatial coordinates may have varying levels of noise, or require specialized preprocessing to capture their unique properties. And textual descriptions can include significant amount of missing values. Additionally, the intricate interplay between the graph's topology and modality-specific data demands models that can effectively integrate and process interaction information. In these scenarios, ensuring the value of representations learned from complex and potentially lower-quality modality-enriched data remains a challenging yet promising area of research.

In addition to the main challenges mentioned above from the three core perspectives of an effective physical and information network representation learning framework, there are also minor, yet significant, research challenges to consider. These include the efficiency and scalability of the developed models, the interpretability of the designed models, as well as adapting models for specific real-world downstream applications. These factors are also crucial for the success of a representation learning model applied to modality-enriched physical and information network data.

Therefore, the primary objective of my research is to advance the general representation learning capabilities on network data, which spans both physical and information networks, by addressing several key issues. This includes to leverage the interplay between topological structures and other modality data forms in both

nodes and edges, developing models with theoretically guaranteed expressiveness, and tackling data quality issues such as label scarcity, noisy data and compatibility between graph topology and other data modalities. Additionally, my research explores methods to tailor learned representations for specific downstream applications and to improve the efficiency and scalability of training and inference processes. Detailed discussions of each of these issues are presented in the subsequent subsections.

## 1.1 Research Issues

This thesis focuses on developing general representation learning framework for physical and information networks. It also aims to develop training strategies for addressing label scarcity, noisy data and data incompatibility scenarios on graph data, with applications in the chemical, biomedical, and social science domains. The key research issues are outlined in the following sections.

### 1.1.1 Representation Learning on Physical Networks

Physical networks, also known as spatial networks, are networks for which the nodes and edges are constrained by geometry and embedded in real physical space, which has crucial effects on their topological properties. Although tremendous success has been achieved in spatial and network representation separately in recent years, there exist very little works on the representation of spatial networks. Existing graph representation learning research typically only focus on studying the connectivity topology information within graph data. However, in real-world application scenarios, the network structure of graph data are usually embedded in integrating the physical location of graph data. Extracting powerful representations from spatial networks requires the development of appropriate tools to uncover the pairing of both spatial and network information in the appearance of node permutation invariant, and rotation

and translation invariant. Hence it can not be modeled merely with either spatial or network models individually. Therefore, how to develop a generic framework for spatial network representation learning that can address these above challenges is crucial for advancing the research domain of physical networks, as well as many important downstream tasks such as molecule property predictions and protein structure analysis. Besides, how to have theoretical guarantees on the quality of learned representations is also important for a reliable learning framework, especially for diverse physical environments such as irregular non-Euclidean space. Finally, how to maintain low computational resource given incremental information remains a significant challenge in designing the learning framework.

### 1.1.2 Representation Learning on Information Networks

Information networks are networks that are usually abstract and not directly embedded in physical space. In such networks, nodes represent entities such as individuals, documents, or data sources, while edges signify the relationships or pathways through which information is shared, transmitted, or connected. A significant challenge in existing representation learning for information networks lies in the diversity of data domains such as social networks, citation networks and e-commerce networks. These domains introduce variations in node and edge features, as well as in predictive tasks, which complicates the development of a unified learning framework. Recent advancements in natural language processing, particularly the rise of pre-trained language models, have shown remarkable success in handling diverse data domains. In this research, we aim to unify various information network data domains through the use of text descriptions, resulting in a unified structure we refer to as Text-Attributed Graphs (TAGs). TAGs augment traditional graph structures with natural language descriptions, facilitating a richer and more detailed representation of data and their relationships across a wide range of real-world scenarios. However, current approaches

to TAG representation learning are predominantly supervised, relying heavily on labeled data, which limits their applicability in diverse contexts. Our research seeks to overcome this limitation by developing a fully unsupervised framework for TAG representation learning. This framework integrates the strengths of pre-trained language models, which excel at natural language understanding, with graph models that effectively capture structural information. By combining these approaches, we aim to produce high-quality representations from TAGs without the need for extensive labeled data, thereby broadening their applicability across various domains.

### 1.1.3 Enhancing Generalizability and Robustness of Learning Network Representations

Graph Neural Networks (GNNs) have achieved great success in representing network data by recursively propagating and aggregating messages along the edges. However, in real-world applications, the graph topology often does not compatible seamlessly with other data modalities due to data quality issues. Specifically, since edges typically represent dependency relationships between data entities, real-world graphs may contain edges of varying reliability, with some even introducing noise that can hinder performance on downstream tasks. This inconsistency poses challenges to the generalizability and robustness of graph representation learning when applied to diverse and noisy datasets. Unfortunately, existing GNNs may lead to suboptimal learned representations because they usually treat every edge in the graph equally. On the other hand, Curriculum Learning (CL), which mimics the human learning principle of learning data samples in a meaningful order, has been shown to be effective in improving the generalization ability and robustness of representation learners by providing learning order on data samples. Specifically, by gradually proceeding from easy to more difficult samples during training, CL can resolve the challenges associated with noisy or unreliable data samples, improving the overall learning process. Unfor-

tunately, existing CL strategies are typically designed for independent data samples and cannot trivially generalize to handle data dependencies in graphs. How to propose a novel CL strategy for dependent network data to alleviate the incompatibility issues between graph topology with other data modalities remains an open challenge.

## 1.2 Contribution

The major proposed research contributions that have been addressed up to now can be stated as follows:

### 1.2.1 Representation Learning on Physical Networks

1. **We propose a new Spatial Graph Message Passing neural network (SGMP) for learning the representations of generic spatial networks.** The new proposed method is equipped with a novel message passing neural network to organically aggregate the spatial and graph information with theoretical guarantee on discriminative power.

2. **We design a new accelerating algorithm for learning on graph-structured data to enhance efficiency.** The time complexity and memory complexity is reduced from $O(N^3)$ to $O(N)$ with respect to the average degree of nodes, while the accelerated algorithm can still maintain the theoretical guarantees of representation expressive power for spatial networks.

3. **We further propose an enhanced framework capable of generalizing to spatial networks embedded in irregular manifold spaces.** Many real-world networks are embedded in non-Euclidean spaces, such as manifolds. The updated framework is designed to consider the coupled graph topological information and its embedded spatial curve information.

4. **In addition, we further generalize the proposed framework to handle geometric trees.** While tree is one special format of graph, its unique hierarchical structure layout plays a crucial role in the formation mechanism. To address this, we propose a generalized framework that is tailored for spatial trees.

### 1.2.2 Representation Learning on Information Networks

1. **We propose a new self-supervised learning framework for text-attributed graphs, Text-And-Graph Multi-View Alignment (TAGA).** This proposal aims at seamlessly integrating TAGs' structural and semantic dimensions. The proposed framework is generic for all types of text-attributed graphs, with the potential to unify diverse information networks in one foundational learning framework.

2. **We propose a multi-view alignment module to preserve rich semantic information, topology information, and their interplay.** We propose to develop a new Graph2Text method that transforms the text-attributed graph into a natural hierarchical layout document. Existing Graph2Text methods often describe all edges within a graph in plain text, which is usually unnatural and tends to obscure higher-order structure information.

3. **We propose an accelerated algorithm to reduce training time complexity from quadratic to linear.** In order to support large-scale training, we present a random walk based algorithm to decrease the time complexity from quadratic to linear, and approximate the original algorithm without loss of information.

4. **We evaluate the framework performance on label-scarce and transfer learning settings.** Existing works typically evaluate methods in a supervised

learning setting, where obtaining training labels can be challenging in real-world applications. Our goal is to enhance performance in label-scarce scenarios, such as zero-shot and few-shot learning, and even in transfer learning across different graph domains.

### 1.2.3 Enhancing Generalizability and Robustness of Learning Network Representations

1. **We propose a novel CL algorithm named Relational Curriculum Learning (RCL).** The proposed method is aimed to improve the generalization ability and robustness of representation learners on network data by analyzing and resolving the incompatibility between graph topology and other modality information.

2. **We develop a novel graph edge selection criteria for automatically involving graph edges.** The proposed method can automatically quantify the difficulty of understanding graph edges and incorporate them into the training process at the appropriate time. This strategy aims at involving the proper topological structures that are compatible with other data modality information.

3. **We investigate the generalizability and robustness improvement of proposed curriculum learning strategy.** We compare RCL to state-of-the-art comparison methods through extensive experiments on both synthetic and real-world datasets.

## 1.3 Organization of Thesis

The remainder of this research thesis is organized as follows: Chapter 2 introduces the problem of graph representation learning on physical networks and proposes a message-passing-based model to address it. Additionally, this chapter discusses how to extend the method to non-Euclidean spaces and a special case involving spatial trees. Chapter 3 presents the work of representation learning on information networks that aims at designing graph foundation models based on text-attributed graphs, with a focus on exploring the boundaries of label-scarce graph representation learning and transfer learning settings. Chapter 4 describes the proposed curriculum learning strategy aimed at improving the generalizability and robustness of network representation learning models by alleviating the incompatibility issue between graph topology and other data modalities. Chapter 5 includes conclusions and future work plans.

# Chapter 2

# Representation Learning on Physical Networks

In this chapter, we first introduce the background of research problem of representation learning on physical networks in Section 2.1 and related works in Section 2.2. Then in Section 2.3, we propose the designed deep learning model architecture for handling general Euclidean spatial networks, where the experimental results are presented in Section 2.4. In addition, we generalize the framework to further consider non-Euclidean spatial networks in Section 2.5 by including the irregular manifold surface into model design. In Section 2.6, we present the experimental results on non-Euclidean spatial networks. Furthermore, the special cases of spatial networks, which are geometric trees with unique hierarchical properties, are introduced in Section 2.7. We propose unique self-supervised learning objectives that are designed for hierarchical trees and then present experimental analysis in Section 2.8. Finally, this chapter is ended with a conclusion section in Section 2.9.

This chapter includes three consecutive works. The first work of representation learning on Euclidean spatial networks [225] was published in *The 35th Conference on Neural Information Processing Systems* as a full research track paper, titled "Rep-

resentation Learning on Spatial Networks". The second work of representation learning on non-Euclidean spatial networks [231] was published in *SIAM Conference on Data Mining 2024* as a research paper, titled "Non-Euclidean Spatial Graph Neural Network". The third work of representation learning on geometric trees [233] was published in *30th SIGKDD Conference on Knowledge Discovery and Data Mining* as a full research paper, titled as "Representation Learning of Geometric Trees".

## 2.1 Introduction on Physical Networks

Spatial data and network data are both popular types of data in modern big data era. The study of spatial data focuses on the properties of *continuous* spatial entities under specific geometry, while analysis of network data investigates the properties of *discrete* objects and their pairwise relationship. Spanning these two data types, physical network, or know as spatial network, are a crucial type of data structure that nodes occupy positions in a real-world physical space, where spatial patterns and constraints may have a strong effect on their connectivity patterns [11]. Understanding the mechanism of organizing spatial networks has significant importance for a broad range of fields [55], ranging from micro-scale (e.g., molecule structure [208]), to middle-scale (e.g., biological neural network [57]), to macro-scale (e.g., mobility networks [37]). Effectively learning the representations of spatial networks is extremely challenging due to the close interactions between network and spatial topology, the incompatibility between the treatments for discrete and continuous data, and particular properties such as permutation invariant and rotation-translation invariant. Spatial networks have long been researched in the domains such as physics and mathematics, which usually extend complex networks and graph theory into spatial networks [173, 12]. They typically rely on network generation principles predefined by human heuristics and prior knowledge. Such methods usually characterize well on the aspects of the

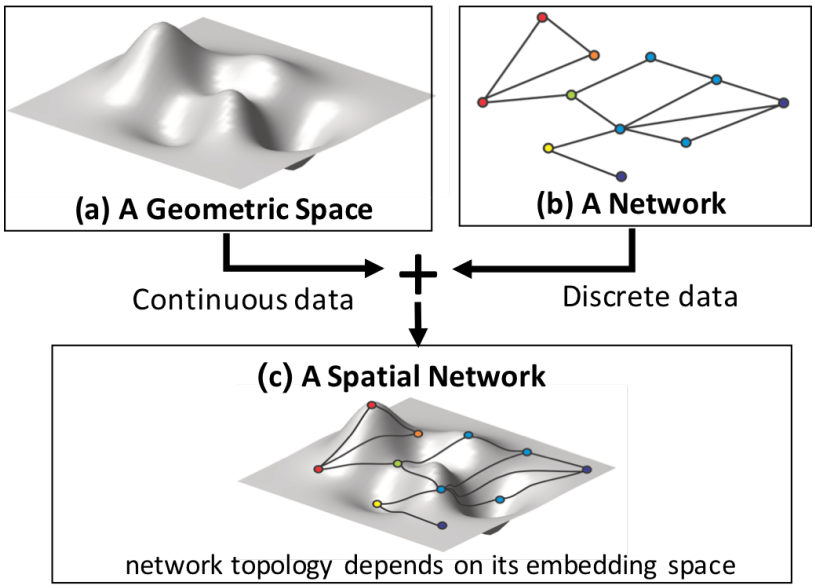


Figure 2.1: Spatial network contains not only the information of network topology and spatial topology but also their interaction.

data that have been covered by the predefined principles, but not on those have not been covered[11]. However, the underlying network process in complex networks is largely unknown and extremely difficult to be predefined in simple rules, especially in crucial and open domains such as brain network modeling [175], network catastrophic failure [158], and protein folding [53].

Remarkable progress has been made towards generalizing deep representation learning approaches in spatial data and network data [210, 28, 86, 82, 54], respectively, in recent years. For spatial data, deep learning achieved significant progress in different commonly used formats such as images [119, 163, 137, 44], point clouds [63, 157, 129], meshes [187, 198], and volumetric grids [209, 143]. On the other hand, deep learning has also boosted the research of encoding graph structure on network data [86, 114, 85], and downstream applications such as recommender systems [218, 135], drug discovery [75, 46, 79, 80], FinTech [204], customer care [205], and natural language processing [139, 13, 206].

Despite the respective progress in representation learning on spatial data and

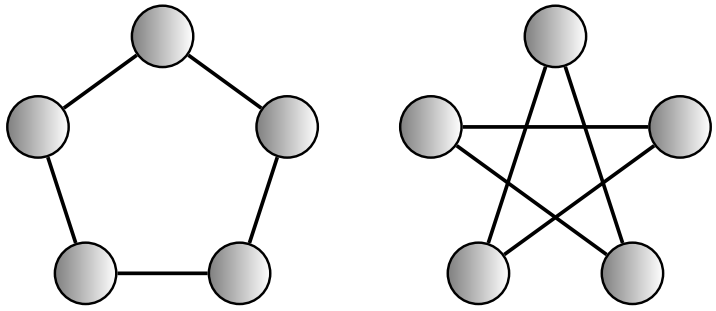


Figure 2.2: The left figure reflects closer nodes tend to connect with each other (known as the first law of geography [171]), while the right figure reflects a spatial teleconnecting pattern where faraway nodes tend to connect. Discriminating these two spatial networks requires new method that can jointly consider spatial and network properties.

network data in parallel, the representation learning for spatial networks have been largely underexplored and has just started to attract fast-increasing attention. Merely combining spatial and graph representations separately cannot handle that for spatial networks where spatial and network process are deeply coupled together [11, 147, 62]. For example, Fig. 2.2 shows a simple example with a pair of spatial networks, in which there are different formation rules on the edges relied on the spatial distance, that is non-distinguishable for spatial and network embedding methods, respectively.

Few recent attempts have been proposed to handle representation learning on spatial networks but still suffer from key challenges: Spatial network representation learning is a problem extremely difficult to address due to several unique challenges: **1) Difficult in distinguishing the patterns that require joint spatial and graph consideration.** Examples like Figure. 2.2 that share the same spatial and network topology, respectively, but with significantly different interaction mechanisms, are non-distinguishable to either spatial or graph methods. **2) Difficult in jointly maintaining that the learned representation is invariant to node permutation, and rotation and translation transformations.** Notice that spatial and

graph information confine each other which neutralizes conventional methods to have either of them. For example, although point clouds representation learning can easily preserve rotation- and translation-invariant by using spatial nearest neighbors, here in spatial networks the neighbor is confined also by graph neighbors. Such additional confinement largely harden our task. **3) High efficiency and scalability in the graph size.** The confinement between spatial and graph information inevitably leads to taking into account more entities simultaneously to maintain sufficient information. The requirement to handle incremental information increases the demand for model efficiency and scalability.

In order to address all the aforementioned challenges, we propose a new spatial graph message passing neural network (SGMP) for learning the representations of generic spatial networks, with theoretical guarantees on discriminative power and various spatial and network properties, and an accelerating algorithm which adjusts to our theoretical framework. Specifically, to capture and model the intrinsic coupled spatial and graph properties, we propose a novel message passing neural network to organically aggregate the spatial and graph information. To ensure the invariance of learned representation under rotation and translation transformations, a novel way to represent the node spatial information by characterizing geometric invariant features with lossless information is proposed. To alleviate the efficiency issue, we propose a new accelerating algorithm for learning on graph-structured data. The proposed accelerating algorithm effectively reduces the time and memory complexity from $O(N^3)$ to $O(N)$, and maintains the theoretical guarantees for spatial networks. Finally, we demonstrate the strength of our theoretical findings through extensive experiments on both synthetic datasets and real-world datasets.

## 2.2 Related Works

**Spatial Networks.** There has been a long time of research efforts on the subjects of spatial networks [11]. In the area of quantitative geography, Haggett and Chorley discussed the relevance of space in the formation and evolution of networks, and developed models to characterize spatial networks at least fourty years ago [84, 36]. New insights leading to modern quantitative solutions are gained due to the advance in complex networks [61, 195, 3, 9, 7, 40], and appears in more practical fields such as transportation networks [5, 121, 122], mobility networks [37, 48], biological networks [57, 164], and computational chemistry [74, 160, 70].

**Geometric Deep Learning.** This is a more recent domain which handles non-Euclidean structured data such as graphs and manifolds [28].

*Geometric Deep Learning on Manifolds.* There is a large body of research efforts of generalizing deep learning models to 3D shapes as manifolds in the computer graphics community. Many works have been conducted to find a better approach to generalize convolution-like operations to the non-Euclidean domain [142, 21, 162, 217, 140, 134]. J. Masci *et at.* proposed the framework of generalizing convolution neural network paradigm to manifolds by applying filters to extract local patches in polar coordinates [142]. Litany *et at.* [134] proposed FMNet to learn the dense correspondence between deformable 3D shapes.

*Geometric Deep Learning on Graphs.* The earliest attempts we are aware of to generalize neural networks to graphs are attributed to M. Gori *et at.* [77]. More recently, a number of approaches encouraged by the success of convolutional neural networks [119] have attempted to generalize the notion of convolution to graphs. One important stream of convolution graph neural networks is spectral-based, where emerges after the pioneering work of Bruna *et at.* [26] which based on the spectral graph theory. There have been many following works [92, 49, 114, 123]. Another stream of work define graph convolutions as extracting locally connected regions from

the graph [56, 131, 150, 85, 211, 146]. Many of these works were formulated in the family of message passing neural networks [74] which apply parametric functions to a node and its proximities, and use pooling operations to generate features for the node. Efficiency and scalability for deep graph learning is very important especially for large graphs and higher-order operations, which triggers research on accelerating GNNs [85, 31, 30]. Hamilton *et at.* [85] first introduced sampling scheme on neighborhood nodes to restrict the size. Chen *et at.* [31] proposed a method which samples vertices rather than neighbors. However, none of these works can guarantee the sampled graph is connected.

**Deep Learning on Spatial Data.** Deep learning has also boosted the study on spatial data. Significant progress has been achieved on deep learning on images since AlexNet [90, 163]. For 3D point clouds, PointNet [157] is a pioneering work which addressed the permutation invariance by a symmetric function. PointCNN [129] transforms the input points into a latent and potentially canonical order by a $\chi$-conv transformations. Volumetric-based methods usually apply a 3D Convolution Neural Network (CNN) to 3D grids [209, 143]. Wang *et at.* [187] first performed shape segmentation on 3D meshes by taking three low-level geometric features as its input.

Despite the success of generalizing deep learning to network and spatial data separately, there has been relatively little work that simultaneously characterize both of them and their interaction. Previous models such as [74, 160, 70] are domain-specific, [160, 70] treat spatial networks as point clouds which ignores the influence of network structure, and [189, 188] consider POI (Point of Interest) categories, hence such concept graphs are not physically embedded in a geometric space. In addition, existing works [193, 47] typically utilize the off-the-shelf deep neural networks with Cartesian coordinates as inputs and a large amount of rotation-and translation-augmented data, which is computationally expensive and lacks theoretical guarantee of rotation- and translation-invariant on the representation. To the best of our knowledge, our pro-

posed method is the first generic framework of spatial network representation learning that handles substantial properties of rotation- and translation-invariant and the interplay between spatial and graph patterns with a theoretical guarantee.

## 2.3 Representation Learning on Euclidean Spatial Networks

**Problem Definition.** Spatial graphs (also known as spatial networks [11]) are networks for which the nodes and edges are embedded in a geometric space. Spatial networks is ubiquitous in real world, such as molecular graphs [208], biological neural networks [57], and mobility networks [37], where the spatial and network properties are usually coupled together tightly. For example, chemical bonds are derived from spatially close atoms, and fiber nerves tend to connect neurons close to each other. A spatial network is typically defined as $S = (G, P)$, where a graph $G = (V, E)$ denotes the graph topology such that $V$ is the set of $N$ nodes and $E \subseteq V \times V$ is the set of $M$ edges. $e_{ij} \in E$ is an edge connecting nodes $v_i$ and $v_j \in V$. $P$ denotes the spatial information that is expressed as a set of points $P = \{(x_i, y_i, z_i)|x_i, y_i, z_i \in \mathbb{R}\}$ in Cartesian coordinate system, such that for a node $v_i \in V$, its coordinate is denoted as $(x_i, y_i, z_i) \in P$. Permutation invariance are crucial to graph structured data [210]. The collections of *permutation-invariant functions* on graph-structured data is defined so that $f(\pi^\dagger S\pi) = f(S)$, for all $\pi \in S_n$, where $S_n$ is the permutation group of $n$ elements. Rotation and translation invariance are in natural and common requirements for spatial data [69, 70]. The collections of *rotation- and translation-invariant functions* on spatial networks is defined so that $f(G, \mathcal{T}(P)) = f(G, P)$, for all $\mathcal{T} \in \mathrm{SE}(3)$, where SE(3) is the continuous Lie group of rotation and translation transformations in $\mathbb{R}^3$.

The main goal of this work is to learn the representation $f(S)$ of spatial network

$S = (G, P)$, with the simultaneous satisfaction of strong discriminative power and the aforementioned significant symmetry properties.

In order to achieve the novel spatial network representation learning by addressing the above-mentioned challenges, we propose a new method named spatial graph message passing neural network (SGMP) and a new accelerating algorithm which relies on sampling random spanning trees. Specifically, to discriminate spatial networks especially for the spatial-graph joint patterns, we propose a new message passing scenario which aggregates the node spatial information via higher-order edges as shown in Figure 2.3(a) and elaborated in Section 2.3.2. This scenario preserves graph and spatial information while aggregation with theoretical guarantees. To ensure that the representation is invariant to rotation and translation transformations, we propose to characterize several geometric properties in *length three path*, which is proved to represent node spatial information with guarantee on the properties of *rotation-invariant*, *translation-invariant*, and *information-lossless*. This is illustrated in Figure 2.3(b) and will be detailed in Section 2.3.1. To address the efficiency issue, an innovative sampling algorithm for accelerating training named Kirchhoff-normalized graph-sampled random spanning tree is proposed. The algorithm reduces the time and space complexity from $O(N^3)$ to $O(N)$ while still stay equivalent to original graph, which will be discussed in details in Section 2.3.3.

### 2.3.1 Node Spatial Information Representation

As mentioned above and in Figure 2.2, we need a novel way to represent the node spatial information that can preserve all the spatial structure information losslessly and also maintain rotation and translation invariance. We cannot directly use the Cartesian coordinates because they are not rotation- and translation-invariant. Although there are conventional node spatial information representation methods that maintained the rotation and translation invariance in the domain of spatial deep

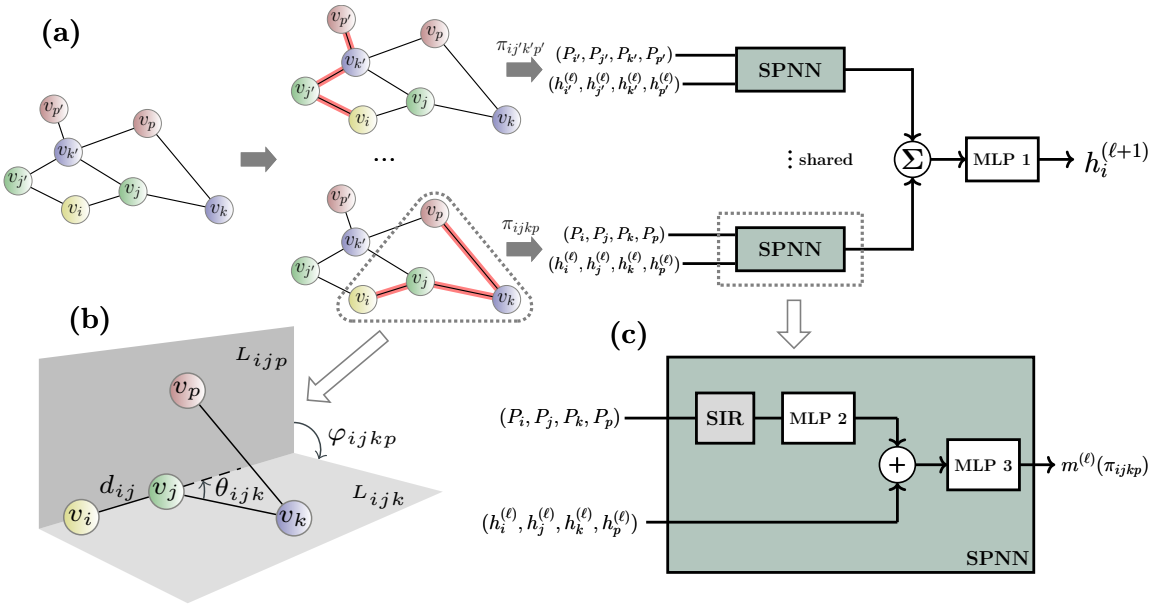


Figure 2.3: Illustration of the proposed spatial graph message passing neural network (SGMP). (a) The process of updating the hidden state embedding $h_i^{(\ell)}$ of node $v_i$ by aggregating the spatial-graph message information from *length three path*. (b) An example to illustrate each elements in our spatial information representation (Equation 2.1). Here $L_{ijp}$ is the plane defined by node $v_i, v_j$ and $v_p$ and $L_{ijk}$ is the plane defined by node $v_i, v_j$ and $v_k$. (c) This is the spatial path neural network block which is designed to learn the coupled spatial-graph property. This block also maintains the invariance to rotation and translation transformations by the spatial information representation (SIR).

learning [170, 69], we cannot simply use them to handle spatial networks because they cannot consider the confinement on neighborhood from graph perspective. Otherwise, the coupled spatial-graph properties cannot be captured. Therefore, we consider to leverage *length n path* to represent the node spatial information. The most simplest way is to just use the distance among nodes and we can have $n = 4$ to ensure the spatial information is preserved. However, we want to minimize the length of the path since the size of the neighborhood grows with a factor of $O(N)$ when one more length for the path is considered. To achieve this, we successfully reduce $n$ to 3 by proposing a new spatial information representation on path, where we use geometry features *distance*, *angle*, and *torsion* as detailed in the following equation and also illustrated in Figure 2.3(b).

The spatial information of a spatial network $S = (G, P)$ with $N$ nodes can be expressed as a set of Cartesian coordinates $P = \{(x_i, y_i, z_i)|x_i, y_i, z_i \in \mathbb{R}\}_{i=1}^{N}$. It can also be represented as $\mathbf{P} \in \mathbb{R}^{N\times 3}$ in a matrix form. The set of all *length n path* starts from node $v_i$ can be represented as $\Pi_n^i$. Particularly, a *length three path* $v_i \to v_j \to v_k \to v_p$ can be expressed as $\pi_{ijkp} \in \Pi_3^i$. Given a spatial network $S$ where its graph $G$ is strongly connected and the longest path $\zeta \geq 3$, the proposed spatial information representation can be expressed by one of its *length three path* $\pi_{ijkp} \in \Pi_3^i$ as

$$(d_{ij}, d_{jk}, d_{jp}, \theta_{ijk}, \theta_{ijp}, \varphi_{ijkp}), \tag{2.1}$$

where

$$\begin{aligned}
d_{ij} &= ||\mathbf{P}_{ij}||_2, d_{jk} = ||\mathbf{P}_{jk}||_2, d_{jp} = ||\mathbf{P}_{jp}||_2,\\
\theta_{ijk} &= \arccos(\langle \frac{\mathbf{P}_{ij}}{d_{ij}}, \frac{\mathbf{P}_{jk}}{d_{jk}} \rangle), \theta_{ijp} = \arccos(\langle \frac{\mathbf{P}_{ij}}{d_{ij}}, \frac{\mathbf{P}_{jp}}{d_{jp}} \rangle),\\
\varphi_{ijkp} &= \text{Parity} \cdot \bar{\varphi}_{ijkp},\\
\mathbf{n_{ijk}} &= \frac{\mathbf{P}_{ij} \times \mathbf{P}_{jk}}{||\mathbf{P}_{ij} \times \mathbf{P}_{jk}||_2}, \mathbf{n_{ijp}} = \frac{\mathbf{P}_{ij} \times \mathbf{P}_{jp}}{||\mathbf{P}_{ij} \times \mathbf{P}_{jp}||_2},\\
\bar{\varphi}_{ijkp} &= \arccos(\langle \mathbf{n_{ijk}}, \mathbf{n_{jkp}} \rangle),\\
\text{Parity} &= \langle \frac{\mathbf{n_{ijk}} \times \mathbf{n_{ijp}}}{||\mathbf{n_{ijk}} \times \mathbf{n_{ijp}}||_2}, \frac{\mathbf{P}_{ij}}{||\mathbf{P}_{ij}||_2} \rangle.
\end{aligned} \tag{2.2}$$

**Theorem 1.** *Here the distances $d_{ij} \in [0, \infty)$, angles $\theta_{ijk} \in [0, \pi)$ and torsions $\varphi_{ijkp} \in [-\pi, \pi)$ are rigorously invariant under all rotation and translation transformations $\mathcal{T} \in \text{SE}(3)$.*

The proof of this theorem is straightforward and can be found in Appendix A.1.1.

It is remarkable to mention that the proposed representation in Equation 2.1 not only satisfies the invariance under rotation and translation transformation but also retains the necessary information to reconstruct the original spatial networks under weak conditions, as described in the following theorem.

**Theorem 2.** *Given a spatial network $S = (G, P)$, if $G$ is a strongly connected graph with longest path $\zeta \geq 3$, then given Cartesian coordinates of three non-collinear connected nodes $(v_j, v_k, v_p)$ in a* length three path *$\pi_{ijkp}$ of one node $v_i$, the Cartesian coordinates $P$ can be determined by the representation defined in Equation 2.1.*

The proof to this theorem is a consequence of the following lemma.

**Lemma 1.** *Given Cartesian coordinates of three non-collinear connected nodes $(v_j, v_k, v_p)$ in a length three path $\pi_{ijkp}$ of one node $v_i$, the Cartesian coordinate $P_i$ of node $v_i$ can be determined by the representation defined in Equation 2.1.*

The proof of this lemma can be found in Appendix A.1.2.

Now we can prove Theorem 2. As stated in Lemma 1, the Cartesian coordinate of node $v_i$ can be determined by its connected neighbors $v_j, v_k, v_p$ in the path of $\pi_{ijkp}$. Due to the property of strong connectivity of graph $G = (V, E)$, we can repeatly solve the coordinate of a connected node to the set of nodes with known coordinates. Thus, start from an arbitrary *length three path* the Cartesian coordinates $P$ of whole spatial networks is determined. □

### 2.3.2 Spatial Graph Message Passing Neural Network

Spatial network representation learning requires us to do convolution that aggregates jointly the graph and spatial information from the graph neighborhood. The most important issue is to maintain the discriminative power without loss of graph and spatial information during the aggregation operation. In the meanwhile, we need to maintain permutation-invariant, rotation- and translation-invariant. To achieve this, we propose the following operation to update the hidden state embedding $h_i^{(\ell)}$ of node $v_i$ by aggregate the messages passing on all its *length three path* $\Pi_3^i$:

$$h_i^{(\ell+1)} = \sigma^{(\ell)}\Big(\mathrm{SUM}\big(\{m^{(\ell)}(\pi_{ijkp}) | \pi_{ijkp} \in \Pi_3^i\}\big)\Big), \tag{2.3}$$

where $\sigma^{(\ell)}$ is a multilayer perceptron (MLP) with ReLU as activation function and the spatial-graph interacted message $m^{(\ell)}(\pi_{ijkp})$ is generated by a spatial path neural network (SPNN) block:

$$\begin{aligned} m^{(\ell)}(\pi_{ijkp}) &= \phi^{(\ell)}\Big(\bar{m}^{(\ell)}(\pi_{ijkp}), \psi^{(\ell)}\big(\hat{m}(\pi_{ijkp})\big)\Big), \\ \bar{m}^{(\ell)}(\pi_{ijkp}) &= (h_i^{(\ell)}, h_j^{(\ell)}, h_k^{(\ell)}, h_p^{(\ell)}), \\ \hat{m}(\pi_{ijkp}) &= (d_{ij}, d_{jk}, d_{jp}, \theta_{ijk}, \theta_{ijp}, \varphi_{ijkp}), \end{aligned} \tag{2.4}$$

where $\phi^{(\ell)}$ and $\psi^{(\ell)}$ are two nonlinear functions to extract the complicated coupling relationship between spatial and graph information, in which we use the multilayer perceptron (MLP) with ReLU as the activation function in our settings.

Finally, the representation of spatial network $S$ can be achieved by applying a graph aggregation operation: $f(S) = \mathrm{AGG}(\{h_i^{(K)} | v_i \in G\})$, where AGG is a permutation invariant function such as SUM or MEAN, and $K$ is the number of our message passing operation layers.

Since the node spatial information is already rotation- and translation-invariant, these properties can be intrinsically preserved by the operation in Equation 2.3. Node permutation will also be preserved due to the usage of the permutation invariant function SUM. Moreover, the following theorem proves that the discriminative power is also preserved from the perspective of maintaining the necessary spatial information, when the dimensions of hidden state embedding are sufficiently large.

**Theorem 3.** Let $\mathcal{S}$ denote the collection of spatial networks with $N$ nodes given the graph $G = (V, E)$, and $\mathcal{F}$ denote the class of our SGMP functions while $\gamma$ is a continuous function. Suppose $g : S \to \mathbb{R}$ is a continuous set function. For all $\epsilon > 0$, there exists a function $f \in \mathcal{F}$, such that for any $S \in \mathcal{S}$,

$$|g(S) - \gamma(f(S))| < \epsilon. \tag{2.5}$$

The proof of this theorem can be found in Appendix A.1.3.

### 2.3.3 Accelerate Training through Sampling Random Spanning Trees

Note that our model is a high order message passing neural network whose time and memory consumption is cubic to the average number of node degree. To reduce the complexity of graph neural networks, a typical way is based on sampling [210]. Many graph-sampling methods have been proposed for accelerating graph neural network [85, 31], which typically focus on randomly extracting a subgraph from the original graph. However, they cannot guarantee the generated graph is a strongly connected graph, which is required by our node spatial information representation in order to maintain no information loss. To ensure that the sampled graphs are connected and sparse, we innovatively propose a Kirchhoff-normalized graph-sampled random spanning tree method for accelerating the training. The proposed method largely reduces the complexity and maintains the equivalence to the original graph. Specifically, a spanning tree $T = (V, E_T)$ of an undirected graph $G = (V, E)$ that is a tree which contains all vertices in $G$. The number of edges of spanning trees is $|E_T| = |V| - 1$, which implies that the time and space complexity during training will not be affected by the number of original edges $|E|$ in graph $G$. We modify our updating operation in Equation 2.1 as

$$h_i^{(\ell+1)} = \sigma^{(\ell)}\Big(\mathrm{SUM}\big(\big\{m^{(\ell)}(\pi_{ijkp})|\pi_{ijkp} \in \bar{\Pi}^i_{T,3}\big\}\big)\Big), \tag{2.6}$$

where we use $\bar{\Pi}^i_{T,3}$ denotes the set of all *length three path* starts from node $v_i$ in a sampled spanning tree $T = (V, E_T)$. It is noticed in Equation 2.6 that randomly sampling spanning trees $T$ from the original graph $G$ will introduce an uneven probability distribution for edges, which results in non-uniform weights for path messages in our

proposed message passing layer. Here we introduce the Kirchhoff-normalized method to remove the uneven distribution by pre-computing the sampling probability of a path $\pi_{ijkp}$ in a sampled random spanning tree $T$. We further modify the Equation 2.6 as

$$h_i^{(\ell+1)} = \sigma^{(\ell)}\Big(\mathrm{SUM}\big(\{\frac{m^{(\ell)}(\pi_{ijkp})}{q(\pi_{ijkp})}|\pi_{ijkp} \in \bar{\Pi}^i_{T,3}\}\big)\Big), \tag{2.7}$$

where $q(\pi_{ijkp})$ is the sampled probability of path $\pi_{ijkp}$ in a random spanning tree.

**Proposition 1.** *Let $T$ denote a uniformly random spanning tree of a graph $G$. Then for a length three path $\pi_{ijkp} = (e_{ij}, e_{jk}, e_{kp})$ we have that*

$$\Pr(\pi_{ijkp} \in T) = det[Y_{\pi_{ijkp}}], \tag{2.8}$$

where $Y$ is called the *transfer function matrix* [20]. The proof is achieved by applying graph theory theorems including Kirchhoff matrix tree theorem [29] and Burton-Pemantle theorem [27].

The proof of this proposition can be found in Appendix A.1.4.

The following result establishes that the approximated form in Equation 2.7 is consistent to original form.

**Proposition 2.** *If $\sigma^{(\ell)}$ is continuous, the expectation of the approximated form in Equation 2.7 converges surely to the original form in Equation 2.3 when the number of samples is sufficiently large.*

The proof is a consequence of the strong law of large numbers and the continuous mapping theorem, which can be found in Appendix A.1.5.

**Complexity analysis of a single layer.** Consider a spatial network with $N$ nodes and dense edge data, our full SGMP layer has $O(N^3)$ time and space complexity

according to the size of the neighborhood. Our accelerating algorithm based on sampling random spanning tree, however, has only $O(N)$ time and space complexity as only $N-1$ edges exist in the generated spanning trees.

## 2.4 Experiments on Euclidean Spatial Networks

In this section, the experimental settings are introduced first, then the performance of the proposed method is presented through a set of comprehensive experiments. All experiments are conducted on a 64-bit machine with an NVIDIA GPU (GTX 1080 Ti, 11016 MHz, 11 GB GDDR5). The proposed SGMP method is implemented with Pytorch deep learning framework [155]. The code for the proposed model is available at `https://github.com/rollingstonezz/SGMP_code`.

### 2.4.1 Experiment Setup

**Datasets.** *(i) Synthetic dataset.* The spatial growth graph model [11] is a spatial variant of the preferential attachment model proposed by Albert and Barabasi [3], which describes that spatial information concerns the formation of networks and long-range links are usually connecting the hubs (well-connected nodes). The process to generate such spatial networks starts from an initial connected network of $m_0$ nodes and introduces a new node $n$ at each time step. The new node is allowed to make $m \leq m_0$ connections towards existing nodes with a probability $\Pi_{n\to i} \sim k_i F[d_E(n,i)]$, where $k_i$ is the degree of node $i$ and $F$ is an exponential function $F(d) = e^{-d/r_c}$ of the euclidean distance $d_E(n,i)$ between the node $n$ and the node $i$ [10]. General characteristics of spatial networks [11] such as clustering coefficient $\mu$, spatial diameter $D$, spatial radius $r$ are set as the prediction targets. Besides, we also add the interaction range $r_c$, which is a significant coupled spatial-graph label that affects the formation of the spatial networks, as another prediction target. We vary the size

and other parameters of spatial networks to collect $3,200$ samples in our synthetic dataset. *(ii) Real-world molecular property datasets.* We experiment on 5 chemical molecule benchmark datasets from [208], including both classification (BACE, BBBP) and regression (ESOL, LIPO, QM9). Particularly, QM9 is a multi-task regression benchmark with 12 quantum mechanics properties. The data is obtained from the pytorch-geometric library [66]. *(iii) Real-world HCP brain network dataset.* We also conducted an experiment using the structural connectivity (SC) of the brain network to predict the age of the subjects, which is a significant task in understanding the aging process of the human brain [108]. In specific, SC is processed from the Magnetic Resonance Imaging (MRI) data obtained from the human connectome project (HCP) [176]. By following the preprocessing procedure in [182], the SC data is constructed by applying probabilistic tracking on the diffusion MRI data using the Probtrackx tool from FMRIB Software Library [99] with 68 predefined regions of interest (ROIs). Then a threshold is applied to SC data to construct the brain networks [165, 71]. The spatial coordinates of regions are expressed as the center point of each region.

**Comparison methods.** To the best of our knowledge, there has been little previous work to handle the generic spatial networks. Spatial graph convolutional networks (SGCN) is a recently proposed method to handle generic spatial networks by applying a convolution operation to learn the spatial-graph interacted information using the relative coordinates between nodes and their first-order neighbors. In addition, we compare with three strong graph neural networks (GIN, GAT and Gated GNN) methods and four spatial neural networks (PointNet, PPFNet, SchNet, and DimeNet) methods for comparisons. For methods in the class of GNNs, we feed the Cartesian coordinates as node attributes while we add the node attribute and graph connectivity information to the class of SNNs for a fair comparison. Besides the models

above, we also compare our model with a state-of-the-art higher-order graph neural networks PPGN [141]) in the QM9 benchmark, the results are provided from original authors. The following describes the details about our comparison models.

**Graph Neural Networks (GNNs).**

*(i) GIN.* Graph Isomorphism Networks (GIN) [211] is a variant of GNN, which has provably powerful discriminating power among the class of 1-order GNNs;

*(ii) GAT.* Graph Attention Networks (GAT) [179] uses multi-head attention layers to propagate information;

*(iii) Gated GNN.* Gated Graph Sequence Neural Networks (Gated GNN) [131] use gated recurrent units (GRU) [39] as a recurrent function, reducing the recurrence to a fixed number of steps.

**Spatial Neural Networks (SNNs).**

*(i) PointNet.* PointNet [157] learns pointwise features independently with several MLP layers and extracts global features with a max-pooling layer;

*(ii) PPFNet.* Point Pair Feature Network (PPFNet) [50] is a spatial deep learning framework to learn a globally aware 3D descriptor;

*(iii) SchNet.* SchNet [160] is a domain-specific model for predicting quantum chemistry. It utilizes a continuous filter function to the distances between nodes and their first-order neighbors.

*(iv) DimeNet.* DimeNet [70] is another domain-specific model for predicting quantum chemistry, which includes the directional information by aggregating the *length two path* messages based on a physical representation of distances and angles.

**Implementation Details.** The goal of the experiments is to validate the performance of our proposed model on spatial networks. We require all models follow the same architecture to utilize the same data for a fair comparison. Specifically, a single MLP layer $m_1$ is applied to the node attributes rather than the spatial information

| Taret | $\mu$ | $D$ | $r$ | $r_c$ |
|---|---|---|---|---|
| GIN | 0.136(.007) | 1.015(.047) | 0.659(.029) | 1.616(.075) |
| GAT | 0.129(.001) | 1.291(.049) | 0.888(.014) | 1.716(.017) |
| GatedGNN | 0.089(.013) | 0.753(.074) | 0.481(.066) | 1.411(.031) |
| PointNet | 0.129(.003) | 0.912(.030) | 0.615(.020) | 1.551(.066) |
| PPFNet | 0.106(.006) | 0.747(.037) | 0.527(.014) | 1.377(.057) |
| SGCN | 0.133(.003) | 1.269(.055) | 0.856(.044) | 1.736(.020) |
| SchNet | 0.128(.001) | 1.006(.058) | 0.686(.031) | 1.691(.039) |
| DimeNet | 0.103(.027) | 1.266(.147) | 0.556(.094) | 1.412(.059) |
| SGMP | **0.068(.005)** | 0.748(.168) | 0.450(.046) | 1.332(.031) |
| SGMP (with st) | 0.088(.001) | **0.291(.021)** | **0.252(.023)** | **1.266(.019)** |

Table 2.1: Root mean square error (RMSE) results on synthetic dataset. Here $\mu$ is clustering coefficient, $D$ is spatial diameter, $r$ is spatial radius and $r_c$ is the interaction radius in the formation of spatial growth graph.

before the convolution layers. Then another MLP layer $m_2$ with decreasing hidden unit sizes is applied after the convolution layers. Each dataset excluding QM9 is split randomly 5 times into 80% : 10% : 10% train, validation, and test. For the QM9 dataset we follow previous work's split [70]. For each split, we run each model 5 times to reduce the variance in particular data splits. Test results are according to the best validation results. For our accelerating method, we pre-sample and store the random spanning trees before the training phase. Note that even though we provide a novel Kirchhoff-normalized method to equivalent our sampled spanning trees to the original graph, the un-normalized version of our algorithm could also achieve competitive results in the experiments.

### 2.4.2 Experimental Performance

In this section, the performance of the proposed method and its accelerated algorithm with sampling random spanning tree (with st), as well as other methods on both synthetic and real-world datasets are presented first. Then we present the efficiency test on our sampling random spanning trees method. In addition, we measure the exactness of invariance of our proposed model under translation and rotation

| Task | Regression | | | Classification | |
|---|---|---|---|---|---|
| Dataset | ESOL | LIPO | HCP | BACE | BBBP |
| GIN | 0.776(.021) | 0.699(.047) | 0.792(.133) | 0.792(.025) | 0.864(.020) |
| GAT | 0.783(.053) | 0.757(.049) | 0.561(.037) | 0.780(.035) | 0.854(.025) |
| GatedGNN | 0.675(.050) | **0.630(.034)** | 0.566(.036) | 0.816(.023) | 0.858(.020) |
| PointNet | 0.716(.036) | 0.708(.030) | 0.720(.123) | 0.799(.023) | 0.843(.027) |
| PPFNet | 0.731(.054) | 0.720(.037) | 0.680(.065) | 0.805(.032) | 0.869(.023) |
| SGCN | 0.743(.056) | 0.726(.055) | 0.674(.059) | 0.778(.030) | 0.849(.021) |
| SchNet | 0.697(.051) | 0.691(.058) | 0.593(.037) | 0.803(.032) | 0.864(.036) |
| DimeNet | 0.730(.047) | 0.666(.047) | 0.818(.127) | 0.791(.031) | 0.864(.036) |
| SGMP | 0.646(.049) | 0.695(.027) | **0.524(.046)** | **0.830(.021)** | **0.880(.020)** |
| SGMP (with st) | **0.612(.054)** | 0.699(.021) | 0.555(.045) | 0.811(.024) | 0.873(.024) |

Table 2.2: Results for four molecule property datasets and the HCP brain network. We report accuracy score for BACE and BBBP datasets, root mean square error (RMSE) for ESOL and LIPO, and mean average error (MAE) for HCP brain network dataset.

transformations.

**Effectiveness Results.** *(i) Synthetic Dataset.* Table 2.1 summarizes the effectiveness comparison for the synthetic dataset, where our proposed SGMP model with sampling spanning tree outperforms the best benchmark model (GatedGNN) by 35.7% on average. Especially, our model achieves lower error on the target of interaction radius ($r_c$), which proves that our proposed model can better capture and exploit the significant coupled spatial-graph characteristics in spatial networks.

*(ii) Real-world Datasets.* Table 2.2 presents the results of four molecule property datasets and the HCP brain network dataset, where our proposed method achieves the best results in 4 out of 5 datasets. The results for the QM9 dataset are presented in Table 2.3, where our proposed method demonstrates its strength through outperforming the benchmark methods in 10 out of 12 targets, which is an improvement by over 14% on average. Particularly, we notice that the performance of the class of SNNs achieved significantly better results than the class of GNNs by a 38% improvement on average, which arguably implies that the quantum mechanics targets of the QM9 dataset are dominated by the spatial information. In addition, the group

| Target | GIN | GAT | Gated | PointNet | PPFNet | SGCN | PPGN | SchNet | DimeNet | SGMP | SGMP (with st) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\mu$ | 0.583 | 0.661 | 0.543 | 0.465 | 0.503 | 0.503 | **0.093** | 0.452 | 0.360 | 0.130 | 0.187 |
| $\alpha$ | 0.652 | 0.952 | 0.609 | 0.453 | 0.459 | 0.531 | 0.318 | 0.347 | 0.189 | **0.113** | 0.174 |
| $\epsilon_{\text{HOMO}}$ | 269.5 | 326.7 | 206.2 | 158.6 | 151.9 | 193.8 | 47.3 | 347.4 | 78.6 | 64.7 | **45.7** |
| $\epsilon_{\text{LOMO}}$ | 175.4 | 237.1 | 135.4 | 123.8 | 136.9 | 141.7 | 57.1 | 151.6 | 61.0 | **44.7** | 67.9 |
| $\delta_\epsilon$ | 361.4 | 510.3 | 314.4 | 245.5 | 221.9 | 275.5 | **78.9** | 120.6 | 103.7 | 83.7 | 98.8 |
| $\langle R^2 \rangle$ | 63.7 | 97.1 | 63.1 | 34.5 | 27.8 | 34.9 | 3.8 | 213.2 | 14.13 | 5.9 | **3.6** |
| ZPVE | 12.3 | 15.7 | 12.0 | 7.0 | 7.4 | 7.4 | 10.8 | 34.3 | 3.1 | 2.3 | **2.0** |
| $U_0$ | 260.1 | 335.9 | 222.5 | 112.7 | 153.5 | 201.3 | 36.8 | 101.7 | 26.8 | **26.1** | 31.9 |
| $U$ | 262.9 | 326.1 | 244.7 | 115.5 | 160.5 | 210.1 | 36.8 | 107.5 | 27.8 | **25.2** | 34.8 |
| $H$ | 269.0 | 329.7 | 239.2 | 123.1 | 157.6 | 199.2 | 36.8 | 107.0 | 27.9 | **27.5** | 31.3 |
| $G$ | 252.7 | 314.1 | 221.1 | 124.3 | 158.4 | 207.8 | 36.4 | 95.0 | 25.8 | **24.6** | 28.2 |
| $c_V$ | 0.344 | 0.430 | 0.283 | 0.196 | 0.221 | 0.277 | 0.055 | 0.452 | 0.064 | **0.043** | 0.064 |

Table 2.3: The mean average error (MAE) results for QM9 dataset. Here "with st" denotes with spanning tree sampling algorithm.

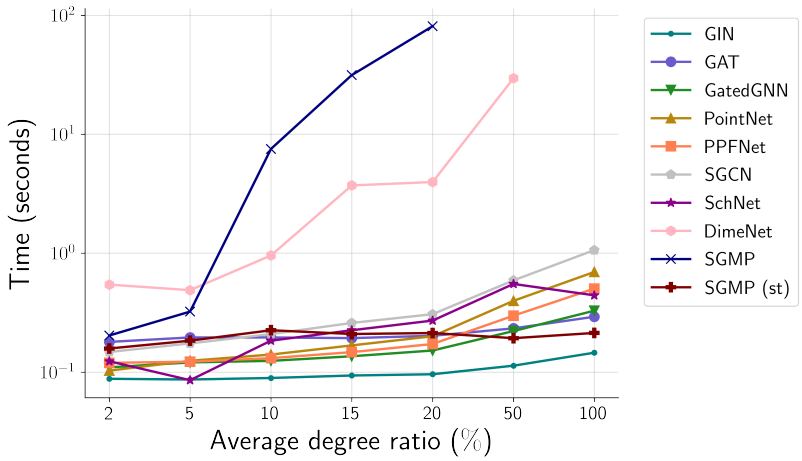


Figure 2.4: Efficiency analysis of our proposed models and all benchmark models. Note that our proposed algorithm with sampling random spanning tree significantly improves the scalability and efficiency.

of jointly-spatial-graph-based methods achieved a 68.9% improvement compared to the group of point-cloud-based methods. The twelve quantum mechanical properties in the QM9 dataset seems highly related to the spatial geometry properties between nodes. For example, the formation energy ($U$) is related to the distances, angles, and torsions among nodes. In this situation, we notice that the performance of the group of point-cloud-based methods is significantly better than the group of GNN based methods, and jointly-spatial-graph-based methods can better explore the coupled spatial-graph property.

| ADR (%) | wo st | w st | speed up |
|---|---|---|---|
| 2 | 0.203s | 0.158s | 1.3× |
| 5 | 0.323s | 0.184s | 1.7× |
| 10 | 7.52s | 0.225s | 33.4× |
| 15 | 31.43s | 0.209s | 150.3× |
| 20 | 80.96s | 0.213s | 379.7× |
| 50 | - | 0.193s | - |
| 100 | - | 0.214s | - |

Table 2.4: Training time per epoch for our full model without sampling spanning tree (wo st) and accelerating method with sampling spanning tree (w st). (-) indicates an out-of-memory error. The sampling algorithm is on average 113 times faster than our full method. ADR is short for the average degree ratio.

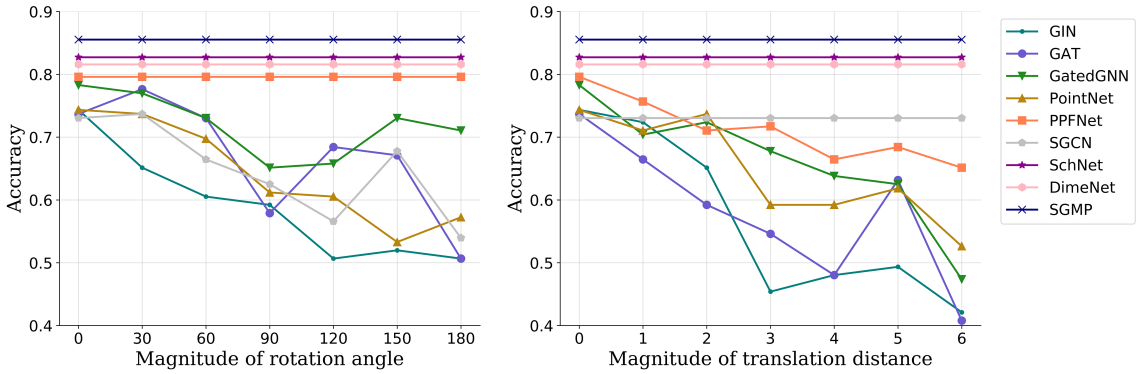


Figure 2.5: Robustness test of rotation and translation invariant: $x$-axis shows data augmentation on the test set. The $x$-value corresponds to the magnitude of rotation angle (left) or translation distance (right). The $y$-axis shows the accuracy score on the test set.

**Efficiency Analysis.** To validate the efficiency of the proposed sampling random spanning tree algorithm, we use our HCP brain network dataset with different thresholds on structural connectivity (SC) to obtain different average degrees for the nodes. The number of nodes is a fixed number (68) while we vary the average of degrees ratio (ADR= $\frac{E}{E_f}$, where $E$ is the number of edges and $E_f$ is the number of edges in complete graphs, e.g. ADR= 100% indicates a complete graph). We report the results of the average training time per epoch among all models for 20 epochs. As shown in Figure 2.4 and Table 2.4, our accelerating algorithm achieves significant improvements in training efficiency. Note that our method is even faster than most

of the first-order methods when the graph connections are dense (ADR over 50%). Notice that higher-order methods (e.g. our full method is third-order and DimeNet is second-order) are unable to handle complete graphs due to the limits of GPU memory. The scalability of our sampling method is remarkable, which can maintain a constant time and space complexity with the increasing number of connected edges.

**Rotation and translation invariant test.** Similar to previous work [69], we also measure the rotation and translation robustness by uniformly adding translation and rotation transformations to the input Cartesian coordinates. Here we only report the accuracy results of classification task on the molecular dataset BACE due to the space limit while the results are similar on all datasets. According to Figure 2.5, we can note that the performance of our proposed model stays invariant under both translation and rotation transformations. SchNet and DimeNet can also achieve invariance under transformations because they also only use the rotation- and translation-invariant spatial features in their models. PPFNet can stay invariant under rotation transformations but not translation transformations because it preserves the origin in the model. On the other hand, SGCN can stay invariant under translation transformations but not rotation invariant because it only utilizes relative coordinates. This experiment validates the importance of applying a rotation- and translation-invariant model since we can observe that the performance of models without a theoretical guarantee drop significantly under adding rotation and translation transformations.

## 2.5 Representation Learning on Non-Euclidean Spatial Networks

In this section, we further generalize the previous proposed representation learning framework on Euclidean spatial networks to non-Euclidean space.

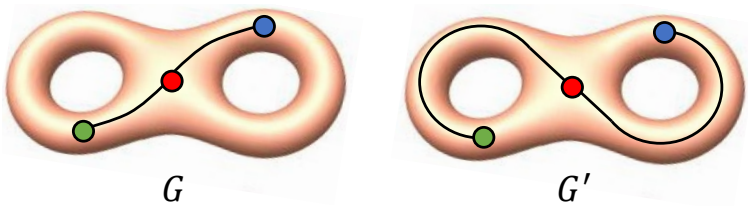


Figure 2.6: Two spatial networks with different connectivity mechanisms on a holomorphic manifold. The left figure reflects that nodes tend to be connected by the shortest distance (called the first law of geography [171]), while the right figure reflects the spatial pattern in which nodes tend to be connected by circuitous lines. Distinguishing these two spatial networks requires new approaches to jointly consider the spatial curves on the manifold and network topology.

### 2.5.1 Background on Non-Euclidean Spatial Networks

Although some efforts [61, 195, 3, 7] have been put toward understanding the mechanism of spatial networks in some traditional research domains such as physics or mathematics, they usually require predefined human heuristics and prior knowledge of the analytical formulation of embedded spatial manifolds, which is usually unavailable in many real-world cases. In the era of deep learning, existing representation learning works on spatial networks [160, 70, 225] can only consider networks that are embedded in Euclidean space, where edge connections between nodes are described as straight lines. However, many real-world networks are embedded in non-Euclidean spaces, such as manifolds. The oversimplified approximations in flat Euclidean space will inevitably lose the rich geometric information carried by the irregular manifolds. Examples in Figure 2.6 that share the same network topology and nodes' spatial coordinates, respectively, but with significantly different connecting curves between nodes, are non-distinguishable for existing representation learning methods on spatial networks. Therefore, jointly taking the irregularity of the embedded manifold with the network topology into account is crucial to extract powerful representations for spatial networks.

Unfortunately, there is no trivial way to simply combine previous representation learning methods on network data and spatial data together to accomplish the task

of representation learning on spatial networks due to several unique challenges: (1) **Difficulty in jointly considering discrete network and continuous spatial manifolds information, and their coupled interactions.** As shown in the example in Figure 2.6, some spatial networks may share the same spatial and network properties, respectively, but have significantly different interaction mechanisms. Simply combining spatial and graphical methods cannot distinguish these spatial networks. (2) **Difficulty in extracting the geometric information of nodes and edges embedded in the irregular manifold.** In real-world situations, the manifolds that networks embed in are often irregular and inhomogeneous in space, where an explicit analytical form is usually infeasible. Thus, how to represent the geometric information of nodes and edges that are embedded in the manifold is challenging.

**Problem Formulation.** Here we consider the connected smooth compact two-dimensional surface $M$, which is most commonly observed in our real-world 3D space. Locally around each point $x$ the manifold is homeomorphic to a two-dimensional Euclidean space referred to as the tangent plane and denoted by $T_xM$. Given the manifold $M$, a spatial network is typically defined as $G_M = (V, E, M_V, M_E)$ such that $V$ is the set of nodes and $E \subseteq V \times V$ is the set of edges. $e_{ij} \in E$ is an edge connecting nodes $v_i$ and $v_j \in V$. $M_V$ and $M_E$ denote the subset of the manifold $M$ that nodes and edges embed in, which is defined as $M_V \subseteq M, M_E \subseteq M$. Particularly, $M_V$ can be described as a set of 3D Cartesian coordinate points where we have $p_i \in M$ for each point $p_i$ representing the coordinates of node $v_i$. $M_E$ can be described as a set of curved lines that connect the nodes on the manifold, where for each $e_{ij} \in E$ we have its corresponding curved line as $l_{ij} \in M_E$. Specifically, a smooth curved line can be defined as a mapping function $l : [0, T] \to M_E$. The main goal is to learn the representation mapping function $f : G_M \to \mathbb{R}^{\mathcal{D}}$ to map an input spatial network to a high-dimensional vector, with the simultaneous satisfaction of strong discriminative power and significant symmetry properties.

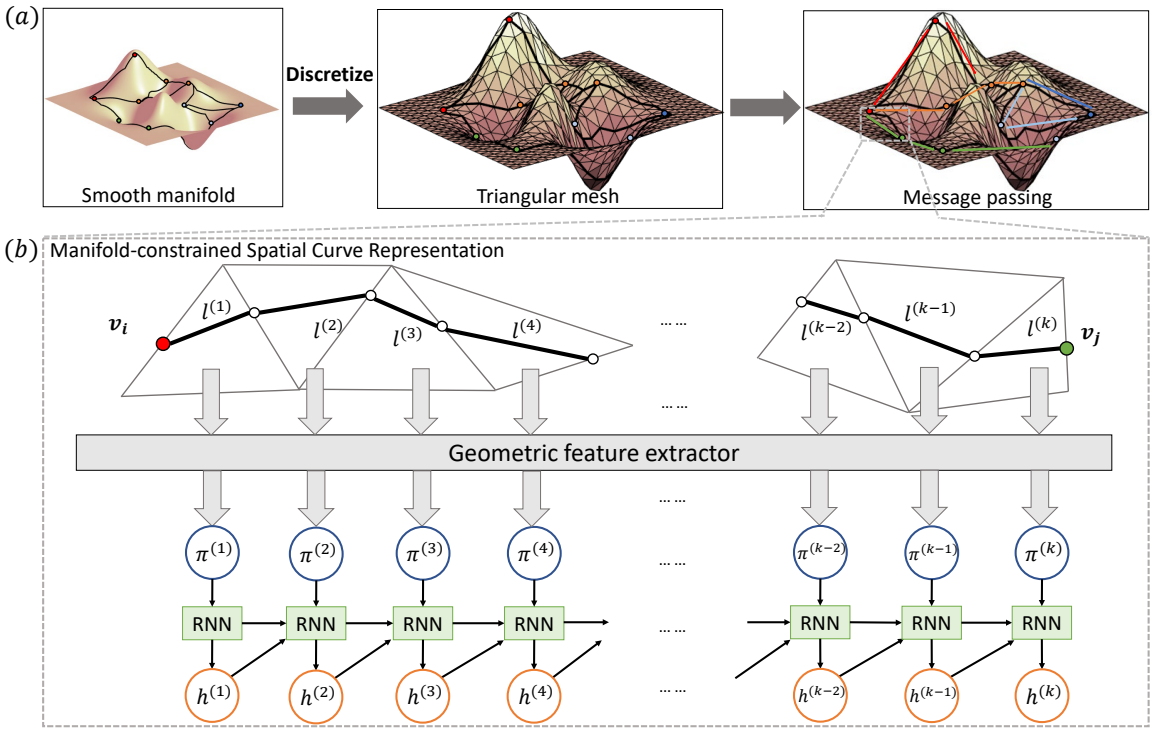


Figure 2.7: Illustration of the overall proposed framework. (a) The discretization process of the continuous manifold and convolutional neural networks for passing and aggregating the geometric information on spatial curves. (b) The RNN module extracts the geometric information along the irregular spatial curves between nodes.

### 2.5.2 Generalized Framework for Non-Euclidean Spatial Networks

In order to design an effective method for learning powerful representations on non-Euclidean spatial networks by addressing the above-mentioned challenges, we propose a novel method named **M**anifold **S**pace **G**raph **N**eural **N**etwork (**MSGNN**). To jointly learn network information and its embedded spatial manifold information, we propose a general learning message-passing framework. As shown in Figure 2.7(a), in order to represent a continuous curve on a manifold, we first discretize the manifold into a mesh tessellation, and then learn the representation of curves through a sequential model of mesh units. Curve representations are then treated as messages on edges, and coupled spatial graph information is learned by passing and aggregating

messages to nodes with graph convolutional layers. As shown in Figure 2.7(b), to deal with the irregularities of spatial curves and their embedded geometric manifolds, we propose to characterize several geometric features on each mesh unit of the curvilinear paths. Finally, we theoretically prove that the extracted spatial curve representations with a guarantee on the properties of *rotation-invariant*, *translation-invariant*, and *geometric information-lossless*. To demonstrate the strength of our theoretical findings, extensive experiments are performed on both synthetic and real-world datasets.

**Manifold Space Graph Neural Network**

Due to the incompatibility between discrete network data and continuous spatial data, the first question is how to combine these two data in one end-to-end framework. Besides, the explicit analytical format function of the spatial manifold is usually extremely difficult to obtain because of the irregularity and non-uniformity of the real-world manifold surfaces. To address this problem, we first propose to discretize the continuous manifold space into discrete mesh data. Triangular mesh, which preserves shape surfaces and topology, is a popular format for efficiently approximating manifold shapes. Specifically, a triangular mesh can be defined as a collection of $C$ triangle faces $\mathcal{F} = \{f^{(1)}, f^{(2)}, \ldots, f^{(C)}\}$, where the vertices of each $f^{(c)} \in \mathcal{F}$ are located on the surface of manifold $M$.

Given the discretized mesh to describe the embedded spatial surface, we can describe the spatial curved lines between nodes as a sequence of discrete units. As shown in Figure 2.7(b), the sequence consists of triangular faces that the curve line passes. Therefore, the spatial information of each edge can be represented as a sequence of mesh units and line segments embedded on them. More concretely, for each edge $e_{ij} \in E$ exists in the given spatial graph, it corresponds to a curved spatial line $l_{ij}$ embedded on manifold $M$. Given the discretized triangular mesh, the embedded line $l_{ij}$ can be represented as a set of $K$ line segments $l_{ij} = \{l_{ij}^{(1)}, l_{ij}^{(2)}, \ldots, l_{ij}^{(K)}\}$ with

their corresponding embedded triangle faces $F_{ij} = \{f_{ij}^{(1)}, f_{ij}^{(2)}, \dots f_{ij}^{(K)}\}$, where $K$ is the number of faces on this spatial path and each line segment $l_{ij}^{(k)}$ is embedded in its corresponding face $f_{ij}^{(k)}$. In summary, we can represent the spatial information on edge $e_{ij}$ as a sequence of pairs $((l_{ij}^{(1)}, f_{ij}^{(1)}), (l_{ij}^{(2)}, f_{ij}^{(2)}), \dots, (l_{ij}^{(K)}, f_{ij}^{(K)}))$.

Given the sequences of pairs of faces and their embedded line segments, the next question is to incorporate them with the graph topology in a general model to learn the coupled spatial-graph representations. Since the length of the sequence may vary on different edges, some common methods such as MLP or CNN cannot be easily generalized to extract representations here. To address this issue, we propose a novel approach that mimics natural language processing approaches by analogizing each pair of spatial units as a token in a sentence, which is shown in Figure 2.7(b). Therefore, a recurrent graph neural network (RNN) model such as GRU or LSTM is a natural choice to extract latent embeddings from these sequences. Formally, the extracted latent embedding $h(e_{ij})$ on edge $e_{ij}$ can be denoted as $\mathbf{RNN}(\pi(l_{ij}^{(1)}, f_{ij}^{(1)}), \pi(l_{ij}^{(2)}, f_{ij}^{(2)}), \dots, \pi(l_{ij}^{(K)}, f_{ij}^{(K)}))$, where $\pi(\cdot)$ denotes the geometric information of the unit and will be introduced in Section 2.5.2. The extracted spatial information can then be treated as the message on graph edges. A graph message passing neural network model is then performed to jointly combine node information and all incoming messages on edges into updated node embeddings, where the update function is as follows:

$$\hat{h}(v_i) = \mathbf{AGGREGATE}\{\xi(h(v_j), h(e_{ij}))|j \in \mathcal{N}(i)\},$$
$$h(e_{ij}) = \mathbf{RNN}(\pi(l_{ij}^{(1)}, f_{ij}^{(1)}), \dots, \pi(l_{ij}^{(K)}, f_{ij}^{(K)})),$$

where $h$ represents latent embeddings and $\xi$ denotes a nonlinear function such as multiple layer perceptron (MLP). **AGGREGATE** denotes any feasible set aggregate function.

**Manifold-Constrained Spatial Curve Representation**

Given the above framework, a key question is how to define the spatial information extractor $\pi(\cdot)$ on each unit of a line segment and its embedded triangular mesh. As mentioned previously, we need a novel way to represent the spatial path information of the network edge that can preserve all the geometric shape information, and simultaneously maintain rotation- and translation-invariance. Obviously, simply feeding the Cartesian coordinates of each line segment in units can not guarantee invariance to the important symmetries such as rotation-and translation-transformations. Although there exist a few spatial information representation methods in the domain of spatial deep learning that can guarantee rotation and translation invariant features, we can not directly use them because they either can not capture the coupled spatial-graph properties because they are purely spatial-based methods, or can not guarantee all the geometric structure information is preserved because the information they extracted is not lossless. Here, the term lossless information means given the extracted geometric features, the information is sufficient to recover the input geometric structure. To handle this issue, for each sequence of spatial path line segments and their embedded meshes, we propose to extract a combination of geometric features on each mesh unit and the relative spatial relationship with their neighboring units. We theoretically guarantee the extracted information is sufficient to recover the full original geometries and also stay invariant to rotation and translation transformations.

Without loss of generality, we consider the spatial path between node $v_i$ and node $v_j$ in the given node set $V$ of graph that there exists an edge $e_{ij}$ between them. Formally, for each unit $(l_{ij}^{(k)}, f_{ij}^{(k)})$ belongs to the sequence of edge $e_{ij}$, where $k \in [1, K]$, we use $\pi(l_{ij}^{(k)}, f_{ij}^{(k)})$ to represent the extracted geometric features on this unit. The sequence of spatial information on edge $e_{ij}$ is denoted as $((l_{ij}^{(1)}, f_{ij}^{(1)}), (l_{ij}^{(2)}, f_{ij}^{(2)}), \ldots, (l_{ij}^{(K)}, f_{ij}^{(K)}))$. For the purpose of simplicity, we omit the subscript symbol $_{ij}$ in the rest of this section.

To extract the necessary spatial information, we consider both the spatial information along the line segment and its embedded triangle mesh. The spatial information on the line segment, denoted as $\pi(l^{(k)})$, and the embedded mesh face, denoted as $\pi(f^{(k)})$, capture important geometric structure and relative orientation directions. For the line segment $l^{(k)}$, we extract its length $d^{(k)}$ and angle $\theta^{(k)}$ with respect to the connecting line $L$ between nodes $v_i$ and $v_j$. These features confine the relative position to a sphere in 3D space, allowing rotation around $L$. To fix the relative position, we further extract the torsion angles $\phi^{(k,k-1)}$ and $\phi^{(k,k+1)}$ between the current line segment $l^{(k)}$ and its neighboring segments $l^{(k-1)}$ and $l^{(k+1)}$.

For the embedded mesh faces $\pi(f^{(k)})$, we consider the curvature vector direction to understand the spatial information of the surface environment. However, simply calculating the orientation $\mathbf{n}^{(k)}$ of the curvature vector does not guarantee rotation and translation invariance. To address this, we calculate the relative angles $\varphi^{(k,k-1)}$ and $\varphi^{(k,k+1)}$ between the curvature vectors of the given face and its neighboring faces. Additionally, we compute the angle $\varphi^{(k-1,k+1)}$ between the two neighboring curvature vectors. These angles form a triangle, ensuring fixed relative orientation. Mathematically, the representation of spatial information for the $k$-th mesh unit on the spatial path sequence that forms the edge $e_{ij}$ between nodes $v_i$ and $v_j$ can be denoted as:

$$\begin{aligned}\pi(l^{(k)}, f^{(k)}) =&(d^{(k)}, \theta^{(k)}, \phi^{(k,k-1)}, \phi^{(k,k+1)},\\ &\varphi^{(k,k-1)}, \varphi^{(k,k+1)}, \varphi^{(k-1,k+1)}),\end{aligned} \tag{2.9}$$

where

$$
\begin{aligned}
d^{(k)} &= ||\mathbf{l}^{(k)}||_2, \theta^{(k)} = \arccos\langle\frac{\mathbf{l}^{(k)}}{d^{(k)}}, \frac{\mathbf{L}_{i,j}}{d_{i,j}}\rangle, \\
\mathbf{L}_{i,j} &= \mathbf{p}_j - \mathbf{p}_i, d_{i,j} = ||\mathbf{L}_{i,j}||_2 \\
\phi^{(k,k-1)} &= \langle\frac{\mathbf{c}^{(k-1)} \times \mathbf{c}^{(k)}}{||\mathbf{c}^{(k-1)} \times \mathbf{c}^{(k)}||_2}, \frac{\mathbf{L}_{ij}}{||\mathbf{L}_{ij}||_2}\rangle \cdot \bar{\phi}^{(k,k-1)}, \\
\phi^{(k,k+1)} &= \langle\frac{\mathbf{c}^{(k)} \times \mathbf{c}^{(k+1)}}{||\mathbf{c}^{(k)} \times \mathbf{c}^{(k+1)}||_2}, \frac{\mathbf{L}_{ij}}{||\mathbf{L}_{ij}||_2}\rangle \cdot \bar{\phi}^{(k,k+1)}, \\
\bar{\phi}^{(k,k-1)} &= \arccos\langle\mathbf{c}^{(k)}, \mathbf{c}^{(k-1)}\rangle, \\
\bar{\phi}^{(k,k+1)} &= \arccos\langle\mathbf{c}^{(k)}, \mathbf{c}^{(k+1)}\rangle, \\
\mathbf{c}^{(k)} &= \frac{\mathbf{L}_{ij} \times \mathbf{l}^{(k)}}{||\mathbf{L}_{ij} \times \mathbf{l}^{(k)}||_2}, \varphi^{(k-1,k+1)} = \arccos\langle\mathbf{n}^{(k-1)}, \mathbf{n}^{(k+1)}\rangle, \\
\mathbf{c}^{(k-1)} &= \frac{\mathbf{L}_{ij} \times \mathbf{l}^{(k-1)}}{||\mathbf{L}_{ij} \times \mathbf{l}^{(k-1)}||_2}, \varphi^{(k-1,k)} = \arccos\langle\mathbf{n}^{(k-1)}, \mathbf{n}^{(k)}\rangle, \\
\mathbf{c}^{(k+1)} &= \frac{\mathbf{L}_{ij} \times \mathbf{l}^{(k+1)}}{||\mathbf{L}_{ij} \times \mathbf{l}^{(k+1)}||_2}, \varphi^{(k,k+1)} = \arccos\langle\mathbf{n}^{(k)}, \mathbf{n}^{(k+1)}\rangle
\end{aligned}
$$

**Theorem 4.** Here the distances $d \in [0, \infty)$, angle $\theta \in [0, \pi)$, torsions $\phi \in [-\pi, \pi)$, and relative orientation angle $\varphi \in [0, \pi)$ are rigorously invariant under all rotation and translation transformations $\mathcal{T} \in \mathrm{SE}(3)$.

The proof is straightforward and can be found in Appendix A.2.1. Intuitively, distance, angle, torsion, and orientation angle are invariant to translation and rotation transformations, since only relative coordinates are used in the formula. It is remarkable to mention that the proposed representation in Equation 2.9 not only satisfies the invariance under rotation and translation transformation but also retains the necessary information to recover the entire geometric structure of original spatial networks under weak conditions, as described in the following theorem.

**Theorem 5.** Given a spatial network $G_M = (V, E, M_V, M_E)$, if $G_M$ is a connected graph, then for any edge $e_{ij}$ with spatial curve sequence that has length of sequence $\tau \geq 3$, given Cartesian coordinates of two endpoints and one arbitrary point, the

whole Cartesian coordinates of the given spatial networks can be determined by the spatial representation defined in Equation 2.9.

The proof can be found in Appendix A.2.2.

**Complexity analysis.** The time complexity of an $L$-layer GNN is $O(L\left|\mathcal{E}\right|b + L\left|\mathcal{V}\right|b^2)$, where $b$ is the number of latent dimensions. Second, the time complexity of extracting geometric features from edge trajectory by LSTM is $O(L\left|\mathcal{E}\right|(Kb^2 + Kbd))$ where $K$ is the average length of spatial trajectories and $d$ is the number of computed geometric features. Therefore, the total time complexity of our algorithm is $O(L\left|\mathcal{E}\right|(Kb^2 + Kbd + b) + L\left|\mathcal{V}\right|b^2)$.

## 2.6 Experimental Results on Non-Euclidean Spatial Networks

In this section, we first introduce the experimental settings, then the effectiveness of our proposed framework on both synthetic and real-world datasets is presented. The link to our code is at the GitHub repository `https://github.com/rollingstonezz/SDM24_Manifold_spatial_networks`.

### 2.6.1 Experimental Settings.

**Synthetic datasets.** In order to examine the effectiveness of our proposed MSGNN method in learning the coupled network and spatial manifold information, we follow previous works [11] to generate a set of synthetic datasets. We generalize the preferential attachment model [3] to a spatial variant that all nodes and edges are embedded in a defined spatial manifold surface. Specifically, we first randomly generate a manifold surface in 3D space from a designed candidate pool of geometric shapes such as sphere or paraboloid. The process to generate such spatial networks

starts from an initial connected network of $m_0$ nodes that are randomly sampled on the manifold surface. Then we introduce a new node $v_j$ to connect to the existing network at each iteration step. The new node is allowed to make $m \leq m_0$ connections towards existing nodes with a probability $\Pi_{j \to i} \sim k_i F[d_g(i,j)]$, where $k_i$ is the degree of node $v_i$ and $F$ is an exponential function $F(d_g) = e^{-d_g/r_c}$ of the geodesic distance $d_g(i,j)$ between the newly added node $v_j$ and the node $v_i$ on the manifold. Therefore, the formation mechanism of generated spatial networks is jointly determined by the spatial and network information. General characteristics of spatial networks [11] such as spatial diameter $D$, and spatial radius $r$ are set as the prediction targets. Besides, we also add the interaction range $r_c$, which is a significant coupled spatial-graph label that affects the formation of the spatial networks, as another prediction target. We vary the type of embedded manifolds and other parameters of spatial networks to collect $5,000$ samples.

**Real-world datasets.** To further evaluate the performance of our proposed MSGNN and comparison methods in real-world scenarios, five public benchmark real-world spatial network datasets with different application domains are utilized as benchmark datasets in our experiments. Specifically, we include one brain network dataset and two 3D shapes datasets for graph classification task, and two airline transportation networks for link prediction task. We provide a brief description of these datasets as follow.

(1) **HCP brain networks.** Classify the activity states of subjects based on processed functional connectivity (FC) networks derived from the human brain manifold environment. Each data sample is associated with an activity state (e.g., rest, gamble) as the target for prediction. To construct brain networks, a threshold is applied to the FC values to filter out highly correlated edges. The spatial trajectories of interest are defined as geodesic paths between the centers of the ROIs.

(2) **Air transportation networks.** We adopt two publicly available flight networks

| Target | GCN | GIN | PointNet | PPFNet | MeshCNN | CurvaNet | SchNet | SGMP | MSGNN |
|---|---|---|---|---|---|---|---|---|---|
| $r_c$ | 3.35±0.14 | 2.55±0.18 | 2.45±0.08 | 2.68±0.11 | 2.06±0.13 | 1.54±0.10 | <u>0.97±0.06</u> | 1.08±0.05 | **0.83±0.04** |
| $D$ | 2.20±0.15 | 2.73±0.21 | 1.82±0.06 | 1.98±0.12 | 1.87±0.06 | 1.93±0.11 | 1.94±0.10 | <u>1.86±0.05</u> | **1.67±0.05** |
| $r$ | 2.63±0.14 | 2.60±0.26 | 1.95±0.09 | 2.07±0.09 | 1.60±0.07 | 1.74±0.06 | 1.98±0.07 | <u>1.88±0.05</u> | **1.47±0.11** |

Table 2.5: The RMSE results of the synthetic dataset. The best performance for each predictive target is shown in bold, while we also underline the second-best performing models.

*Flight-NA* and *Flight-GL*, where *Flight-NA* contains 456 airports and 71,959 airlines in the North America and *Flight-GL* contains 3,214 airports and 66,771 airlines spanning the globe. The earth surface is considered as the manifold to include the curved airline trajectory.

(3) **3D shapes classification.** We further conduct experiments on classifying 3D shapes in two datasets *SHREC* and *FAUST*. We follow previous studies [89] to sample a lower resolution (∼500 faces) from a higher solution. The vertices of the lower-resolution triangle tessellation are then treated as the nodes and their geodesic trajectories are treated as the spatial curves.

**Comparison models.** We compare our proposed MSGNN against several categories of competitive methods, spanning two graph neural networks methods **GCN** [114] and **GIN** [211], two spatial deep learning methods on point clouds **PointNet** [157] and **PPFNet** [50], two spatial deep learning methods on mesh **MeshCNN** [89] and **CurvaNet** [91]. We also include two state-of-the-art deep learning methods on Euclidean spatial networks **SchNet** [160] and **SGMP** [225]. To ensure a fair comparison, we provided Cartesian coordinates as node attributes for GNN-based methods. For spatial deep learning methods on point clouds and mesh, we augmented the node attributes with graph connectivity information obtained from a trained Node2Vec model [78]. Additionally, we established a consistent search range for model hyperparameters, such as the number of convolutional layers or the dimensionality of hidden embeddings, to maintain fairness across all models.

### 2.6.2 Effectiveness Results

**Synthetic datasets results.** Here we report the root mean squared error (RMSE) results of our proposed MSGNN with comparison methods on the synthetic dataset in Table 2.5. We summarize our observations on the model effectiveness below:

(1) The results demonstrate the strength of our proposed MSGNN by consistently achieving the best results in predicting all three coupled spatial-graph targets. Specifically, our model outperformed all the benchmark models by over 38.0% on average, as well as outperformed the second-best model by 17.6% on average.

(2) Our proposed MSGNN method consistently achieves superior performance with respect to all predictive targets, which proves the robustness of MSGNN. In comparison, the spatial neural network methods on point clouds and mesh have significantly different performances on different tasks. For example, they shows competitive performance to spatial network methods on targets spatial diameter $D$, and spatial radius $r$. While their performance on predicting interaction range $r_c$ is significantly worse than spatial network methods by over 53.0% on average, which may indicate that simply combining graph and spatial representation can not capture the interactions between these two data sources.

(3) It is also worth noting that the category of methods on spatial networks (SchNet, SGMP, and MSGNN) show a more competitive performance than methods in other categories, by over 34.2% on average, which indicates that either graph neural network or spatial neural network methods have limited capability to effectively learn coupled spatial-graph properties. MSGNN shows a stronger performance compared to other methods on spatial networks by 18.2% on average, which demonstrates our method takes advantage of the irregular manifold information within the context of the spatial paths to acquire a more competitive performance.

**Real-world datasets results.** Here we report the accuracy results of our proposed MSGNN with comparison methods on the real-world dataset in Table 2.6.

| | GCN | GIN | PointNet | PPFNet | MeshCNN | CurvaNet | SchNet | SGMP | MSGNN |
|---|---|---|---|---|---|---|---|---|---|
| HCP | 0.835±0.014 | 0.920±0.007 | 0.845±0.027 | 0.876±0.008 | 0.784±0.031 | 0.759±0.025 | 0.896±0.012 | 0.927±0.004 | **0.951±0.005** |
| Flight-NA | 0.674±0.015 | 0.706±0.006 | 0.694±0.011 | 0.698±0.004 | 0.521±0.023 | 0.597±0.019 | 0.710±0.008 | 0.719±0.004 | **0.730±0.005** |
| Flight-GL | 0.722±0.003 | 0.756±0.014 | 0.737±0.010 | 0.715±0.012 | 0.556±0.032 | 0.628±0.025 | 0.750±0.008 | 0.761±0.009 | **0.785±0.005** |
| SHREC | 0.525±0.042 | 0.533±0.034 | 0.567±0.007 | 0.887±0.010 | 0.910±0.003 | 0.902±0.004 | 0.575±0.012 | 0.896±0.005 | **0.918±0.004** |
| FAUST | 0.535±0.010 | 0.783±0.013 | 0.905±0.010 | 0.918±0.005 | 0.903±0.008 | 0.923±0.004 | 0.865±0.023 | 0.840±0.035 | **0.925±0.005** |

Table 2.6: The accuracy results of the real-world datasets. The best performance for each predictive target is shown in bold, while we also underline the second-best performing models.

(1) Our proposed MSGNN method consistently achieved the best results among all methods in all five real-world datasets. Specifically, our results outperformed all the benchmark models by over 14.1% on average and outperformed the second-best model by 4.2% on average. The superior performance demonstrates the effectiveness of MSGNN for learning powerful representations in complex real-world scenarios.

(2) In two air transportation networks, our method achieves more considerable performance gains on the global network (Flight-GL) than on the North American network (Flight-NA). One possible reason is that North America is relatively small compared to the globe, and the curved effect on the surface is not significant. This may indicate that our method can exploit the curvature of the embedded surface to further improve the representation ability.

(3) Different classes of methods perform significantly differently on different datasets. For example, the class of spatial neural networks on mesh (MeshCNN and CurvaNet) have achieved competitive results in 3D shapes classification tasks (SHREC and FAUST) by outperforming other benchmark models by 13.1% on average. However, they also performed poorly on air transportation and brain datasets by achieving the worst performance among all classes of methods. Such behavior indicates that these methods can not well handle generic spatial networks.

(4) It is also worth noting that on the SHREC dataset, only methods that consider orientation information achieved competitive results, which may arguably indicate that orientation information is important in the prediction task on this dataset.

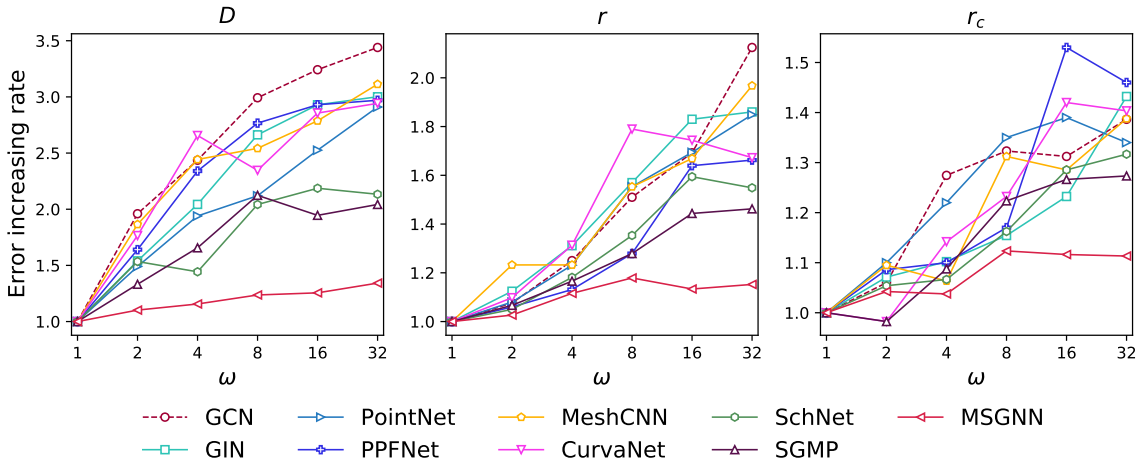


Figure 2.8: Accuracy trend results for our proposed MSGNN model and all competing models against varying degree of manifold irregularity. The performance of models at the lowest degree of irregularity ($\omega = 1$) is set as the base value.

### 2.6.3 Effect of Manifold Irregularity Analysis

Compared to existing representation learning methods on spatial networks, a contribution of our work is that our method can handle irregular geometric manifolds, rather than simply using Euclidean space approximations. To investigate the impact of manifold irregularity on model performance, we further introduce a series of experiments to vary the degree of irregularity of the manifold embedded by the network. Specifically, in our synthetic dataset setting, we choose a sinusoidal surface as the manifold for embedding the network, and we use a frequency parameter $\omega$ to control the irregularity of the generated manifold surface. The mathematical formulation of the sinusoidal surface can be written as $z = \sin(\omega\sqrt{x^2 + y^2})$. Larger values of $\omega$ here indicate that the resulting manifold surface will have a larger degree of irregularity. We vary the value of $\omega$ from 1 to 32 to generate a total of 6 datasets, and we compare the performance of predicting three targets on our method and competing methods. Particularly, we set the RMSE performance of all models at $\omega = 1$ as a benchmark, and then calculate the increasing rate of RMSE when $\omega$ increases. The results are shown in Figure 2.8. According to the figure, compared to all baseline models, our

| Number of faces | 1,000 | 2,000 | 4,000 | 8,000 | 16,000 | 32,000 |
|---|---|---|---|---|---|---|
| HCP | 0.928 | 0.939 | 0.944 | 0.947 | 0.950 | 0.951 |
| Flight-GL | 0.773 | 0.780 | 0.782 | 0.784 | 0.785 | 0.785 |

Table 2.7: Sensitivity analysis of model performance against the number of discretized mesh units.

model consistently shows a significantly slower increasing trend of the RMSE error as the degree of manifold irregularity increases. Such model behavior demonstrates that our model can effectively handle the irregularity in geometric manifolds. More interestingly, our model also shows a convergence trend in predicting the spatial radius $r$ and the interaction range $r_c$, which arguably further demonstrates the robustness of our model to extremely irregular manifold environments.

### 2.6.4 Sensitivity Analysis

We investigate the impact of the number of triangle mesh tessellations on our method to test the sensitivity of our model. We vary the number of mesh faces from high-resolution $32,000$ to low-resolution $1,000$ on two real-world datasets as shown in Table 2.7 (1) According to Table 2.7, with the increase of triangular mesh subdivision number, the accuracy scores on all datasets show a upward and converging trend. The convergent performance trends demonstrate the effectiveness of using mesh tessellations to approximate spatial manifold surfaces. Specifically, as the resolution of the mesh tessellations increases, the approximate discrete surface is approaching the underlying continuous manifold space. (2) It is also worth noting that as the number of mesh tessellations decreases, the performance of our proposed MSGNN on all methods gradually approaches and converges to the spatial network method on Euclidean space. The reason is that as the resolution of the mesh subdivision decreases, the approximate spatial path between nodes eventually converges to a Euclidean approximation.

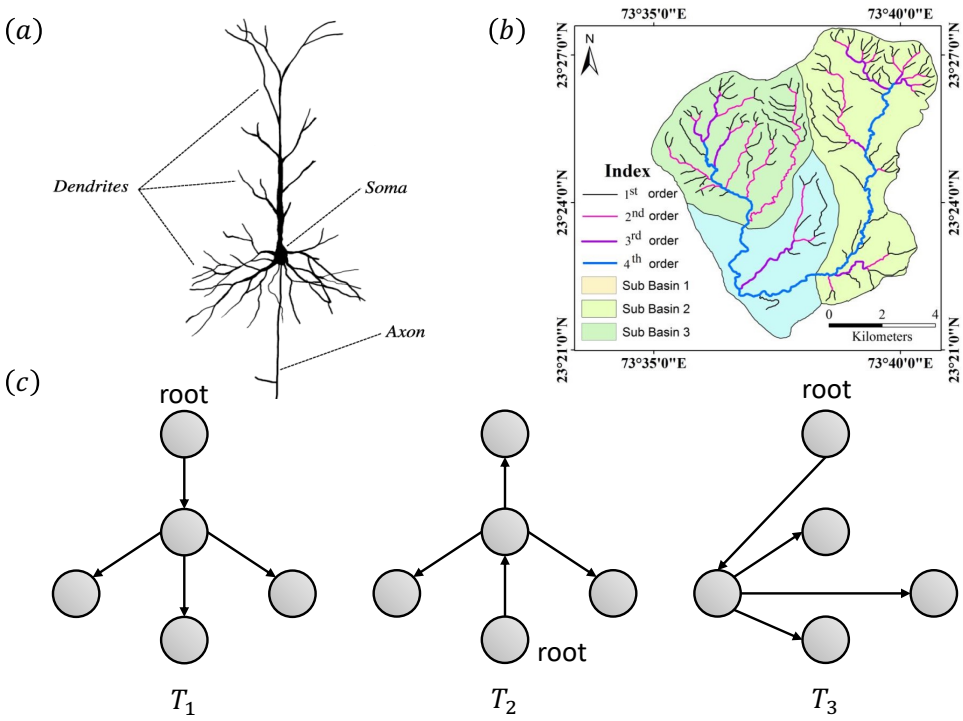


Figure 2.9: (a) Illustration of a neuron's geometric tree-like structure; (b) Representation of a river network exhibiting a tree structure embedded within a geometric landscape; (c) Three different geometric trees with isomorphic network connectivity and identical spatial coordinates. Distinguishing these geometric trees requires jointly considering spatial, topology, and hierarchical layout information.

## 2.7 Representation Learning on Spatial Trees

In this section, we further consider representation learning on spatial trees (or geometric trees), which is a special case of spatial networks but with significantly unique hierarchical layout properties.

### 2.7.1 Background on Spatial Trees

A geometric tree is a hierarchically arranged, tree-structured graph with nodes and edges that are spatially constrained, influencing their connectivity patterns. It is a particularly important data structure that is ubiquitous in different domains such as river geomorphology [22, 202], neuron morphology [4, 154], and vascular vessels [57]. Geometric trees are highly complex, non-linear structures and thus cannot be pro-

cessed directly by common math and statistical tools. This makes representation learning on geometric trees a fundamental necessity in order to further apply them to downstream tasks such as classification, clustering, and generation. Although tree-structure representation learning has been extensively researched by techniques including sequence-based models, such as Tree-LSTM [167], and graph neural networks [28, 85, 86, 114], they are not able to jointly consider the geometric information that is coupled with the hierarchy and topology that are core properties of geometric trees. As shown in Figure 2.9(a), for a pyramidal neuronal cell, the closer to the cell body a branch is, the more curvature it exhibits. Similarly, as shown in Figure 2.9(b), the node degree within a watershed's tree structure is indicative of the breadth of its corresponding subtree's expansion. Furthermore, on a theoretical level, as shown in Figure 2.9(c), the three geometric trees are isomorphic if geometric information and hierarchy information about the levels from the root are not jointly considered.

Although recently some progress has been made with spatial graphs [160, 70, 225], they cannot be used to directly handle geometric trees as they overlook the hierarchical ordering of nodes and edges, which, as mentioned above, is crucial. Therefore, this paper focuses on developing a method that can learn the representations of geometric trees by preserving their geometric, topological, and hierarchical ordering as well as their interplay. To achieve this, three aspects need to be addressed: **1. How to reflect the hierarchical patterns of geometric trees in the learned representations?** Existing methods tend to be either fully permutable or non-permutable; however, the hierarchical nature of geometric tree structures requires a partially permutable pattern. This pattern requires the preservation of parent-child ordering, whereas the ordering among siblings, for instance, does not need to be as rigidly preserved. To tackle this issue, we introduce a partial ordering constraint module that enforces a strict directional relationship in the embedding space for parent-child pairs to reflect their hierarchical order. **2. How to learn the representation**

**with a scarcity of labels?** Unsupervised or self-supervised learning is often used nowadays for geometric data (e.g., graphs, images, spatial, etc.) representation learning. However, the inductive bias for self-supervised learning needs to be customized to specific geometric data types, and those pertaining to geometric trees are significantly underexplored. To address this issue, a novel Geometric Tree Self-Supervised Learning (GT-SSL) framework is introduced through an innovative subtree-growing-guided objective, which aims to align the observed subtree and the expected subtree by its root.

### Problem Formulation

A geometric tree can be formally represented as $S = (T, P)$, where $T = (V, E)$ symbolizes the tree-structured graph. In this representation, $V$ is the set of $N$ nodes, and $E$, a subset of $V \times V$ with $|E| = |V| - 1$, represents the $N - 1$ edges. Each edge $e_{ij} \in E$ connects a *parent* node $v_i$ to a *child* node $v_j$ in $V$, establishing a hierarchical relationship where $v_i$ is the parent of $v_j$. In this structure, starting from any node $v_i \in V$, it is impossible to traverse a *path* (e.g., $v_i \to v_{i1} \to v_{i2} \to v_i$) that forms a loop, ensuring the acyclic nature of the tree. A rooted tree, denoted as $T_i$, originates from any node $v_i$. If a node $v_j$ is a *descendant* of node $v_i$, then its rooted tree $T_j$ forms a *subtree* within the larger rooted tree $T_i$. In this hierarchical arrangement, node $v_i$ is recognized as the *ancestor* of node $v_j$. The set $P$ represents the spatial coordinates, defined as $P = \{(x_i, y_i, z_i) | x_i, y_i, z_i \in \mathbb{R}\}$, within the Cartesian coordinate system. For each node $v_i \in V$, its spatial position is denoted by the coordinate tuple $(x_i, y_i, z_i) \in P$.

The primary objective is to learn the representation $f(S)$ for geometric trees $S = (T, P)$, aiming to achieve a strong discriminative capability for unique geometric tree structures and to capture significant symmetry properties.

### 2.7.2 Self-Supervised Geometric Tree Representation Learning

In order to develop a novel geometric tree representation learning method by addressing the challenges outlined above, we propose a new representation learning model named Geometric Tree Branch Message Passing (GTMP) to fully exploit the interplay between geometric and tree-topological structures. In addition, a novel Geometric Tree Self-Supervised Learning (GT-SSL) framework is introduced to extract the customized geometric tree properties without any supervision labels. Specifically, to discriminate geometric trees, especially for the spatial tree joint patterns, we generalize the proposed spatial networks message passing method SGMP to aggregate the geometric information via tree branches, which is elaborated on in Section 2.7.3. This scenario preserves the geometric structure of tree information with theoretical guarantees on the invariance to SE(3)-symmetric transformations and *spatial-information-lossless*. To address the issue of insufficient labels, we developed two self-supervised learning objectives that are tailored for intrinsic geometric tree structures. Specifically, to incorporate the underlying hierarchical relationships, a partial ordering constraint over the parent-child pair embeddings is introduced in Section 2.7.4. This implies that a node's embedding should maintain a clear directional relationship with its subtree nodes to accurately represent the hierarchical structure. To introduce geometric tree-specific inductive bias as self-supervised learning target, we further propose a top-down subtree growth learning process. As discussed in Section 2.7.5, our goal is to align the observed geometric structure of the subtree with the structure anticipated by its root.

### 2.7.3 Tree Branch Geometric-Topology Information Representation Learning

To effectively tackle the complexities of geometric tree representation learning, we generalize SGMP to Geometric Tree Branch Message Passing (GTMP). As illustrated in Figure 2.10, this approach first harnesses the interplay between geometric properties and topological structures, enabling the computation of a comprehensive geometric-topology information representation for all branches originating from a tree node. Subsequently, we employ a neural network designed in a message-passing fashion, tailored specifically to account for the hierarchical ordering inherent in the tree branch structure. Our method systematically aggregates spatial information along an ordered tree branch. This strategy not only maintains the integrity of the tree's geometric structure but also assures robustness against SE(3)-symmetric transformations, thereby enhancing the discriminative power of the process.

Formally, the spatial information of a geometric tree with $N$ nodes can be expressed as a set of Cartesian coordinates $P = \{(x_i, y_i, z_i)|x_i, y_i, z_i \in \mathbb{R}\}_{i=1}^{N}$. It can also be represented as $\mathbf{P} \in \mathbb{R}^{N\times 3}$ in a matrix form. The set of all *length n tree paths* starting from node $v_i$ to its descendant nodes can be represented as $\pi_n^i$. In particular, a *length three branch* $v_i \to v_j \to v_k \to v_p$ can be expressed as $\pi_{ijkp} \in \Pi_3^i$, where $v_i$ is the parent node of $v_j$, $v_j$ is the parent node of $v_k$, and $v_k$ is the parent node of $v_p$. Given a *length three branch* $\pi_{ijkp} \in \Pi_3^i$, the proposed spatial information representation can be expressed in the format of Equation 2.1:

$$(d_{ij}, d_{jk}, d_{jp}, \theta_{ijk}, \theta_{ijp}, \varphi_{ijkp}).$$

Upon extracting geometric features as outlined in Equation 2.1, the next step involves creating a convolutional strategy. This strategy aims to merge tree-topology and spatial information derived from both the geometric representation of branches

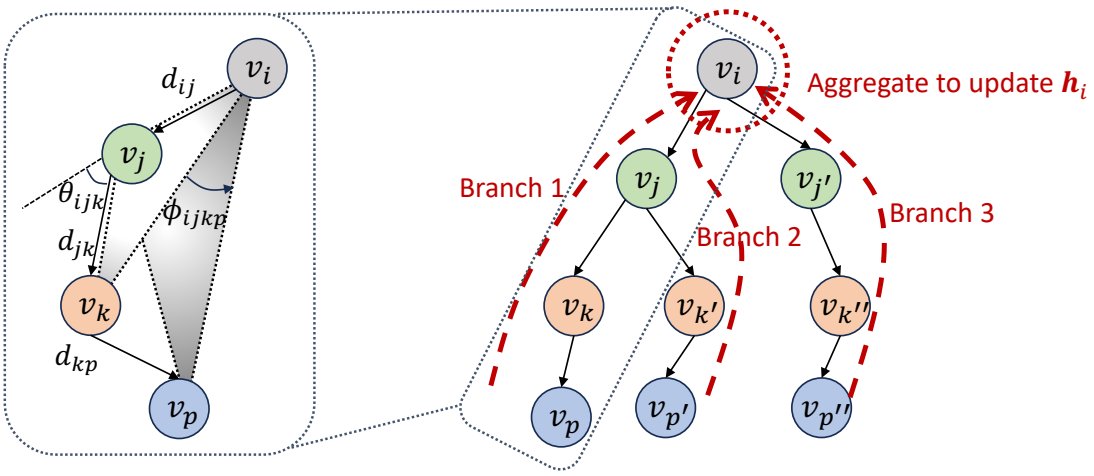


Figure 2.10: Illustration of the GTMP model. The geometric information is first extracted on each length-three branch starting from node $v_i$, then they are aggregated with other node information to update the node embedding $\mathbf{h}_i$.

and their structural layouts into a comprehensive tree node representation. The essential challenge lies in preserving the model's ability to discriminate, while also maintaining the integrity of the tree's structure and its geometric details during the aggregation process.

To achieve this, we propose the following operation to update the hidden state embeddings $h_i^\ell$ by aggregating the message passing along all length three branches $\Pi_3^i$ originating from node $v_i$:

$$h_i^{(\ell+1)} = \sigma^{(\ell)}\Big(\mathrm{AGG}\big(\{m^{(\ell)}(\pi_{ijkp})|\pi_{ijkp} \in \Pi_3^i\}\big)\Big), \tag{2.10}$$

where $\sigma^{(\ell)}$ is an arbitrary nonlinear transformation function (e.g. multilayer perceptron) and AGG denotes a set aggregation function.

In our model, the representation of a branch $\pi_{ijkp}$ at layer $\ell$ integrates both topological and geometric information to produce a comprehensive message. The foundation of this approach lies in the aggregation of node features and the geometric configuration of the branch.

The integration of node features with geometric data is accomplished through a

function $\phi^{(\ell)}$, which combines the aggregated node features $\bar{m}^{(\ell)}$ with a transformation of the geometric information $\psi^{(\ell)}(\hat{m}(\pi_{ijkp}))$. The final message for the branch, $m^{(\ell)}(\pi_{ijkp})$, is thus given by:

$$m^{(\ell)}(\pi_{ijkp}) = \phi^{(\ell)}\Big(\bar{m}^{(\ell)}(\pi_{ijkp}), \psi^{(\ell)}\big(\hat{m}(\pi_{ijkp})\big)\Big),$$

$$\begin{aligned} \bar{m}^{(\ell)}(\pi_{ijkp}) &= (h_i^{(\ell)}, \alpha_1 h_j^{(\ell)}, \alpha_2 h_k^{(\ell)}, \alpha_3 h_p^{(\ell)}), \\ \hat{m}(\pi_{ijkp}) &= (d_{ij}, d_{jk}, d_{jp}, \theta_{ijk}, \theta_{ijp}, \varphi_{ijkp}), \end{aligned} \tag{2.11}$$

where $\phi^{(\ell)}$ and $\psi^{(\ell)}$ are two nonlinear functions to extract the complicated coupling relationship between geometric and tree topology information. The aggregated node features, $\bar{m}^{(\ell)}(\pi_{ijkp})$, are computed as a weighted combination of the features from node $v_i$ and its descendants in order: $v_j$, $v_k$, and $v_p$. Here $\alpha_1$, $\alpha_2$, and $\alpha_3$ adjust the influence of each ancestor's features at layer $\ell$.

### 2.7.4 Hierarchical Relationship Modeling through Partial Ordering Objective Function

To accurately represent the inherent hierarchical relationships among nodes in tree structures, we introduce a partial ordering constraint module. The fundamental concept behind this function is to constrain embeddings in such a way that the embedding of a node in the tree not only represents itself but also maintains a structured relationship over its subtree nodes in the embedding space.

To formally define the ordering constraint between node embeddings, we introduce the concept of partial ordering in the embedding space, ensuring that hierarchical relationships are accurately represented: as shown in Fig. 2.11, if $T_j$ is a subtree of $T_i$, then the embedding $h_j$ of node $j$ has to be within the "lower-left" region of node $i$' embedding $h_i$:

$$h_j[b] \leq h_i[b], \forall_{b=1}^{D} \quad \text{iff} \quad T_j \subseteq T_i, \tag{2.12}$$

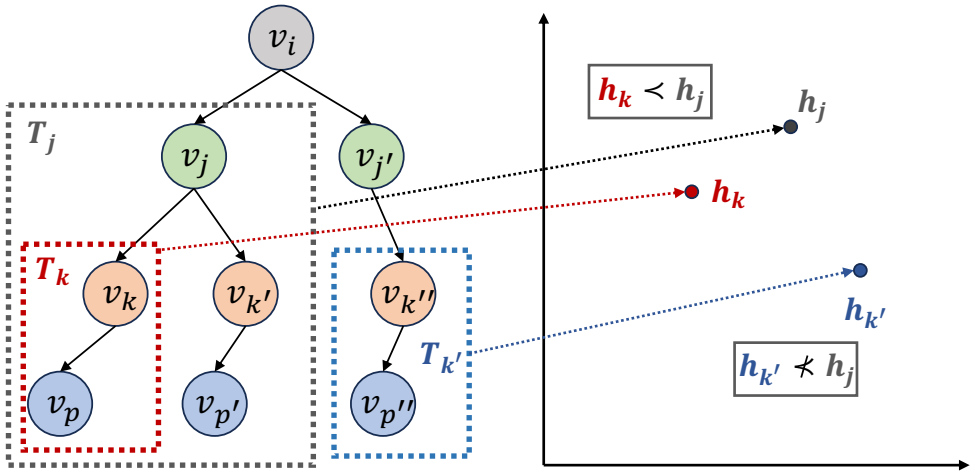


Figure 2.11: Illustration of the hierarchical relationship between tree nodes through partial ordering. In this scenario, $T_k$ is a subtree of $T_j$, establishing a partial ordering relationship between their respective subtree embeddings. Conversely, since $T_{k'}$ does not constitute a subtree of $T_j$, there is no requirement for a partial ordering relationship between the embeddings of these two subtrees.

where $D$ is the dimension of hidden embeddings and $[b]$ denotes the $b$-th dimension of hidden embeddings.

To operationalize the above constraint into a function that can be optimized, we accordingly define the objective function for generating embeddings to utilize the max margin loss:

$$\mathcal{L}_{order} = \sum_{(h_i,h_j)\in\mathcal{P}} \max(0, h_j - h_i) + \sum_{(h_i,h_j)\in\mathcal{N}} \max(0, \delta - \|h_i - h_j\|^2), \tag{2.13}$$

where $\mathcal{P}$ and $\mathcal{N}$ denote the set of positive pairs and negative pairs in the minibatch where tree $T_j$ is a subtree of tree $T_i$. The term $\delta$ represents a margin that enforces a minimum distance between the embeddings of negative pairs compared to positive pairs, ensuring that $h_j$ (the embedding of the lower hierarchical node) is within the lower-left space to $h_i$ (the embedding of the higher hierarchical node), and by at least a margin distance $\delta$ apart for negative pairs.

### 2.7.5 Self-Supervised Learning via Subtree Growth Learning

To tackle the challenge of insufficient training labels in real-world scenarios, we introduce an innovative Geometric Tree Self-Supervised Learning (GT-SSL) framework. While there are existing self-supervised learning objectives aimed at general graph data, they are insufficient when applied to geometric trees. The primary limitation is their inability to incorporate both intrinsic hierarchical relationships and coupled geometric-tree topology information into the self-supervised learning objectives. Our GT-SSL framework, by contrast, is specifically tailored to address these complexities, ensuring that the resulting representations fully reflect the unique structural and spatial characteristics of geometric trees.

Our approach focuses on growing the geometric tree information from its root node. We introduce a unique subtree-growth learning goal, which generates the entire geometric structure from top to bottom. Specifically, this process unfolds iteratively, with each step predicting the geometric configuration of a node's subtree based on the geometric structure of its ancestors. This approach is inspired by the observed natural growth patterns in geometric trees, where evolution occurs in a hierarchical manner, cascading from higher-level nodes down to lower-level ones. An illustrative example can be found in river systems, where the configuration of a tributary is largely influenced by the main river's structure and the characteristics of the surrounding geometric environment.

To formalize our approach, for a node $v_i$ within a geometric tree $T$, we denote $\mathcal{C}(v_i)$ as the set of its child nodes and $\mathcal{A}(v_i)$ as the set of its ancestor nodes within the tree's hierarchy. The objective of our subtree growing process is to accurately predict the geometric structure of the child nodes, $\tilde{\mathcal{G}}(\mathcal{C}(v_i))$, given both the geometric structures of the node itself and that of its ancestors. This can be mathematically expressed as:

$$\tilde{\mathcal{G}}(\mathcal{C}(v_i)) = g\left(\mathcal{G}\left(v_i \cup \mathcal{A}(v_i)\right)\right), \tag{2.14}$$

where $g$ represents a learnable function designed to synthesize the geometric details of the child nodes based on the aggregated geometric information of $v_i$ and its ancestors. In our study, we prioritize predicting essential geometric features such as the distances between a node and its child nodes as well as the angles between the parent node, the current node, and its child nodes.

Unfortunately, to feasibly represent the geometric structure information as $\mathcal{G}$, a significant challenge arises from the variable number of child nodes associated with any given node in a tree-structured graph, complicating the prediction of these geometric features. Although predicting simple aggregated indicators, such as averaging the distances, could be employed, but they risk obscuring the comprehensive geometric structure of the child nodes.

To overcome this limitation, we introduce an approach that involves converting the geometric features into the frequency domain. This transformation enables us to focus on predicting the frequency distribution of the geometric features across the child nodes, rather than attempting to directly predict specific values of the geometric features, thus avoiding the issue posed by the uncertain number of child nodes.

Formally, to represent the geometric information in a form that is amenable to pattern recognition and prediction, we expand these geometric features into the frequency space with radial basis functions. Mathematically, for a given node $v_i$ with child nodes $\mathcal{C}(v_i) = \{v_{i1}, v_{i2}, \ldots, v_{in}\}$, the distances $\{d_{i1}, d_{i2}, \ldots, d_{in}\}$ are expanded as follows:

$$e_k(v_i) = \sum_{v_{ij} \in \mathcal{C}(v_i)} \exp(-\gamma \|d_{ij} - \mu_k\|^2), \tag{2.15}$$

where $e_k$ denotes the $k$-th radial basis and $\mu_k$ is the corresponding distances. To obtain the distribution of geometric features over the radial basis, the ground truth distribution can be denoted as:

$$\mathcal{G}(\mathcal{C}(v_i)) = \left[ \sum_{v_{ij} \in \mathcal{C}(v_i)} e_k(v_i) \right]_{k=1}^{K}, \tag{2.16}$$

where $K$ is the total number of radial basis functions employed. Thus, the estimated distribution can be written as:

$$\hat{\mathcal{G}}(\mathcal{C}(v_i)) = g\left( \left[ \sum_{v_{ij} \in \mathcal{A}(v_i) \cup v_i} e_k(v_i) \right]_{k=1}^{K} \right). \tag{2.17}$$

Therefore, we formulate the objective function using the Earth Mover's Distance (EMD) to measure the discrepancy between the estimated distribution $\tilde{\mathcal{G}}(\mathcal{C}(v_i))$ and the ground truth distribution $\mathcal{G}(\mathcal{C}(v_i))$:

$$\mathcal{L}_{generative} = \sum_{v_i \in V} \text{EMD}\left( \tilde{\mathcal{G}}(\mathcal{C}(v_i)), \mathcal{G}(\mathcal{C}(v_i)) \right), \tag{2.18}$$

where EMD denotes the Earth Mover's Distance to quantify the cost of transforming the estimated distribution into the ground truth distribution.

Finally, the overall self-supervised learning objective function can be written as the combination of the subtree generative objective and the partial ordering function:

$$\mathcal{L}_{\text{GT}-\text{SSL}} = \mathcal{L}_{generative} + \mathcal{L}_{order} \tag{2.19}$$

## 2.8 Experiments on Spatial Trees

All experiments are conducted on a 64-bit machine with four NVIDIA A4000 GPUs (16 GB GDDR5). The proposed method is implemented with PyTorch [155] and the PyTorch-Geometric [65] deep learning framework. The code to our work can be found in the Github repository: `https://github.com/rollingstonezz/KDD24_geometric_trees`.

### 2.8.1 Experimental Settings

**Datasets.** To evaluate the performance of our proposed GTMP and comparison methods in real-world scenarios, eight geometric tree datasets are used in our experiments, which includes five neuron morphology datasets and three river flow network datasets across multiple tasks.

1) Neuron morphology: We conducted classification experiments using neuronal morphology data from NeuroMorpho.org, the largest online collection of 3D neural cell reconstructions contributed by hundreds of laboratories around the world [2]. Specifically, we constructed binary classification tasks between control cells and cells from two experimental conditions, 5xFAD and lipopolysaccharide injection (lps), where experimental condition was the target for prediction. All cells were mouse neural cells but tasks were split across three cell types: glia, interneurons (inter), and principal cells (pc). In total, we constructed five tasks: mouse 5xFAD glia versus mouse control glia; 5xFAD primary cells versus control primary cells; lps glia versus control glia; lps interneurons versus control interneurons; and lps primary cells versus control primary cells. A statistical description of the datasets is shown in Table 2.9.

2) River flow networks: We also conducted regression experiments using publicly available river flow network data from the United States Geological Survey's (USGS) National Hydrography Dataset (NHD) [174, 72]. We incorporated 2,231 river flow network tree samples with an average node count of 11,141 per tree. To facilitate prediction, we established three key geometric topology-coupled metrics as targets: the clustering coefficient, spatial diameter, and spatial radius.

**Comparison Methods.** To the best of our knowledge, there has been little previous work to directly handle geometric trees. Several advanced spatial graph networks have been developed to address generic spatial networks; among these, SchNet [160], DimeNet [70], and SGMP [225] are some of the closest related works to our approach and selected as comparison methods. Additionally, we benchmark our ap-

proach against three prominent graph neural network (GNN) methods— GCN [114], GAT [179], and GIN [211]— and two spatial neural network (SNN) techniques, PointNet [157] and SpatialNet [47]. For GNN methods, Cartesian coordinates are provided as node attributes, while for SNN methods, both node attribute and graph connectivity information are incorporated to ensure an equitable comparison.

**Implementation Details.** In the supervised learning configuration, all models utilize an identical architecture comprising three convolutional layers, with the hidden dimension size set to 64. During the self-supervised representation learning pretraining phase, this architecture is maintained with three convolutional layers leading to final embeddings of 64 dimensions. For subsequent fine-tuning on specific tasks, a three-layer Multilayer Perceptron (MLP) is appended to the convolutional base, utilizing ReLU activation functions to enhance non-linear processing capabilities. To ensure a balanced evaluation across our proposed message passing mechanism and other GNN methods under comparison, we standardized the hyperparameter selection process.

We executed each experiment five times, subsequently averaging the results and computing the standard deviation. We adopted AUC score as the evaluation metric for the classification tasks on the neuron datasets due to the imbalanced distributions of all classes. On all runs in all tasks, the datasets were randomly divided into training, validation, and test sets with an 80:10:10 ratio, and identical hyperparameters were employed across all tasks for each dataset, except for the random seed that was responsible for the data split.

### 2.8.2 Effectiveness Results

In this section, we first assess the performance of our GTMP method against competing approaches across the real-world datasets within a supervised learning manner. Additionally, we explore the efficacy of our GT-SSL framework to evaluate the quality

| | | Neuron - Classification (↑) | | | | | River - Regression (↓) | | |
|---|---|---|---|---|---|---|---|---|---|
| | | lps-glia | lps-inter | lps-pc | 5xfad-glia | 5xfad-pc | $\mu$ | $D$ | $r$ |
| GCN | Supervised | 0.6063 ± 0.0235 | 0.5000 ± 0.0000 | 0.5000 ± 0.0000 | 0.7934 ± 0.0218 | 0.7543 ± 0.0611 | 0.1134 ± 0.0244 | 172.1041 ± 2.8023 | 87.6032 ± 1.2830 |
| | GT-SSL | 0.9798 ± 0.0197 | 0.9568 ± 0.0134 | 0.9546 ± 0.0122 | 0.8131 ± 0.0111 | 0.8938 ± 0.0134 | 0.0941 ± 0.0321 | 95.5785 ± 2.3104 | 76.4080 ± 1.5892 |
| | diff (+/−) | 37.35% | 45.68% | 45.46% | 1.97% | 13.95% | 17.01% | 44.46% | 12.77% |
| GIN | Supervised | 0.6060 ± 0.0326 | 0.5173 ± 0.0388 | 0.5351 ± 0.0355 | 0.8204 ± 0.0083 | 0.7903 ± 0.0407 | 0.1872 ± 0.0150 | 172.5895 ± 3.6765 | 87.4989 ± 1.0998 |
| | GT-SSL | 0.9784 ± 0.0147 | 0.9181 ± 0.0237 | 0.7943 ± 0.1235 | 0.8046 ± 0.0258 | 0.9343 ± 0.0109 | 0.1135 ± 0.0819 | 102.9059 ± 2.5516 | 73.3257 ± 1.3291 |
| | diff (+/−) | 37.24% | 40.08% | 25.92% | -1.58% | 14.4% | 39.37% | 40.38% | 16.20% |
| GAT | Supervised | 0.5878 ± 0.0411 | 0.5221 ± 0.0231 | 0.5000 ± 0.0000 | 0.8101 ± 0.0232 | 0.6576 ± 0.0981 | 0.2335 ± 0.0731 | 171.1645 ± 2.2507 | 89.0507 ± 2.5466 |
| | GT-SSL | 0.5032 ± 0.0048 | 0.5000 ± 0.0000 | 0.5000 ± 0.0000 | 0.8318 ± 0.0177 | 0.8726 ± 0.0390 | 0.2752 ± 0.1201 | 141.8387 ± 5.1076 | 76.5700 ± 1.9842 |
| | diff (+/−) | -8.46% | -2.21% | 0% | 2.17% | 21.5% | -17.86% | 17.13% | 14.02% |
| PointNet | Supervised | 0.6308 ± 0.0835 | 0.7295 ± 0.0310 | 0.5508 ± 0.0431 | 0.8108 ± 0.0078 | 0.8960 ± 0.0140 | 0.1244 ± 0.0102 | 169.0530 ± 2.0981 | 85.9079 ± 2.4306 |
| | GT-SSL | 0.9733 ± 0.0102 | 0.9543 ± 0.0140 | 0.6432 ± 0.0741 | 0.7707 ± 0.0316 | 0.9771 ± 0.0111 | 0.0754 ± 0.0107 | 94.3706 ± 2.7810 | 71.4937 ± 1.8721 |
| | diff (+/−) | 34.25% | 22.48% | 9.24% | -4.01% | 8.11% | 39.39% | 44.18% | 16.78% |
| SpatialNet | Supervised | 0.7300 ± 0.0432 | 0.7445 ± 0.0419 | 0.5354 ± 0.0796 | 0.8614 ± 0.0113 | 0.8243 ± 0.0353 | 0.2335 ± 0.0418 | 151.0569 ± 2.5466 | 86.2461 ± 1.2461 |
| | GT-SSL | 0.5010 ± 0.0025 | 0.5003 ± 0.0007 | 0.9232 ± 0.0460 | 0.8972 ± 0.0150 | 0.9764 ± 0.0164 | 0.0353 ± 0.0381 | 91.6590 ± 3.2311 | 66.9319 ± 3.1098 |
| | diff (+/−) | -22.9% | -24.42% | 38.78% | 3.58% | 15.21% | 84.88% | 39.32% | 22.39% |
| SchNet | Supervised | 0.7561 ± 0.0319 | 0.7597 ± 0.0286 | 0.6178 ± 0.0509 | 0.8994 ± 0.0105 | 0.9502 ± 0.0162 | 0.0342 ± 0.0072 | 154.4101 ± 3.2536 | 82.6100 ± 1.2830 |
| | GT-SSL | 0.9698 ± 0.0120 | 0.9393 ± 0.0231 | 0.9612 ± 0.0202 | 0.9046 ± 0.0092 | 0.9893 ± 0.0083 | 0.0112 ± 0.0050 | 85.0109 ± 2.3832 | 37.8552 ± 1.4879 |
| | diff (+/−) | 21.37% | 17.96% | 34.34% | 0.52% | 3.91% | 67.25% | 44.94% | 54.18% |
| DimeNet | Supervised | 0.7049 ± 0.0620 | 0.8338 ± 0.0226 | 0.5601 ± 0.1020 | 0.9123 ± 0.0089 | 0.9544 ± 0.0120 | 0.0196 ± 0.0053 | 134.4048 ± 2.4952 | 80.9102 ± 1.2827 |
| | GT-SSL | 0.9351 ± 0.0194 | 0.9627 ± 0.0176 | 0.9345 ± 0.0131 | **0.9540 ± 0.0059** | 0.9902 ± 0.0015 | 0.0077 ± 0.0033 | 80.2417 ± 2.0114 | 35.3140 ± 0.9910 |
| | diff (+/−) | 23.02% | 12.89% | 37.44% | 4.17% | 3.58% | 60.71% | 40.30% | 56.35% |
| SGMP | Supervised | 0.7599 ± 0.0442 | 0.8078 ± 0.0382 | 0.6231 ± 0.0512 | 0.8917 ± 0.0123 | 0.9839 ± 0.0034 | 0.0087 ± 0.0031 | **123.3789 ± 3.8385** | 68.0974 ± 2.1300 |
| | GT-SSL | 0.9568 ± 0.0170 | 0.9709 ± 0.0142 | **0.9952 ± 0.0009** | 0.9333 ± 0.0085 | 0.9814 ± 0.0072 | **0.0033 ± 0.0012** | 76.7912 ± 1.4728 | 36.0287 ± 1.3890 |
| | diff (+/−) | 19.69% | 16.31% | 37.21% | 4.16% | -0.25% | 62.07% | 37.76% | 47.09% |
| GTMP | Supervised | **0.7996 ± 0.0392** | **0.8529 ± 0.0370** | **0.6560 ± 0.0621** | **0.9417 ± 0.0123** | **0.9887 ± 0.0063** | **0.0052 ± 0.0024** | 125.4951 ± 2.8295 | **61.2353 ± 1.3177** |
| | GT-SSL | **0.9836 ± 0.0096** | **0.9996 ± 0.0106** | 0.9872 ± 0.0013 | 0.9011 ± 0.0192 | **0.9992 ± 0.0004** | 0.0041 ± 0.0019 | **76.7699 ± 1.8740** | **33.0287 ± 0.7680** |
| | diff (+/−) | 18.4% | 11.87% | 33.12% | -4.06% | 1.05% | 21.15% | 38.83% | 46.06% |

Table 2.8: The main experimental results on neuron morphology and river flow network datasets. Here we present the performance of our GTMP method alongside other comparative methods within both supervised learning and our GT-SSL training approach. For each dataset, we highlight in bold both the best performance in the supervised learning context across all methods, and also the top performer within our GT-SSL training framework. Additionally, we show the percentage improvement in performance of all methods when leveraging our GT-SSL over traditional supervised learning settings. Specifically for the river flow network dataset, we use $\mu$ to represent the clustering coefficient, $D$ for spatial diameter, and $r$ for spatial radius.

of the generated representations in a pretrain-finetune manner over the same dataset. Given that the GT-SSL framework is designed to be a general approach applicable to various representation learning methods, we present the outcomes for both our GTMP approach and all other methods being compared. In this section, we pretrain and finetune the same datasets for implementing the GT-SSL framework, ensuring a fair comparison with the supervised learning results.

The comparison of AUC scores for the neuron morphology datasets and MAE results for the river datasets is provided in Table 2.8. We summarize our observations on the effectiveness of the GTMP model and the GT-SSL training framework below: (1) **Strength of GTMP model in learning effective geometric tree representations.** The supervised learning results demonstrate the strength of our proposed method, which consistently achieved the best results in seven out of eight datasets and

securing the second-best result in the only dataset where the best performance was not attained. Specifically, our results outperformed the other comparative models by over 12.7% on average for the neuron morphology datasets and 22.1% on average for the river flow network datasets. The outcomes demonstrate that our GTMP model successfully leverages the specially designed branch message passing mechanism to generate representations, which significantly enhances performance on downstream supervised learning tasks.

(2) **Benefits of utilizing GT-SSL framework to enhance the quality of geometric tree representations.** Table 2.8 reveals that the GT-SSL's pretrain-finetune approach consistently enhances the performance of all representation learning models on 62 of the 72 total prediction tasks when compared to traditional supervised learning settings. Notably, the GT-SSL method surpasses supervised learning by an average margin of over 23.02% across all tasks. This substantial improvement underscores the effectiveness of the GT-SSL pretraining framework in significantly enhancing the quality of learned representations via self-supervised learning objectives tailored to geometric tree structures.

(3) **Integrating the GTMP model with the GT-SSL framework results in superior overall performance.** It is worth noting that the combination of our proposed GTMP model and GT-SSL framework shows a more competitive performance than any other combination of methods by achieving the best performance in six out of eight datasets. Specifically, our results outperformed the other comparative models by over 10.0% on average for the neuron morphology datasets, and 29.3% on average for the river flow network datasets. The results demonstrate that integrating the GTMP model with the GT-SSL framework can successfully lead to state-of-the-art representation learning performance on geometric tree datasets.

(4) **Advantage of specialized spatial network representation learning methods over conventional graph and spatial neural networks.** It is also worth

| | | Source | | | | |
|---|---|---|---|---|---|---|
| | | lps-glia | lps-inter | lps-pc | 5xfad-glia | 5xfad-pc |
| | # of Trees | 28,687 | 8,092 | 28,224 | 33,757 | 28,086 |
| Average | # of Nodes | 2,025 | 4,488 | 3,146 | 1,994 | 3,152 |
| Target | lps-glia | 0.9836 | 0.9983 | 0.9987 | 0.9231 | **0.9999** |
| | lps-inter | 0.9111 | 0.9996 | 0.9936 | 0.8627 | **0.9999** |
| | lps-pc | **0.9995** | 0.9964 | 0.9872 | 0.9705 | 0.9969 |
| | 5xfad-glia | 0.9357 | **0.9981** | 0.9399 | 0.9011 | 0.9740 |
| | 5xfad-pc | 0.9798 | 0.9969 | 0.9959 | 0.8993 | **0.9992** |

Table 2.9: Transfer learning results. We underline the results where the source and target datasets are the same. Additionally, we highlight the best results for each target dataset.

noting that the specialized spatial network representation learning methods (SchNet, DimeNet, SGMP, and GTMP) show a more competitive performance than both the vanilla graph neural network-based methods (GCN, GIN, and GAT) and the spatial network-based methods (PointNet and SpatialNet). Specifically, these specialized methods surpass traditional graph neural networks by an average of over 19.3% in supervised learning contexts and 22.3% when integrated with the GT-SSL framework; against spatial neural network approaches, they demonstrate an average improvement of 11.1% in supervised settings and 23.7% with GT-SSL. These results indicate that standard graph neural network and spatial neural network methods have limited capability to effectively discriminate patterns that require joint consideration of geometric information and tree topological information.

### 2.8.3 Transfer Ability Analysis

We further investigate the transferability of our GT-SSL framework. In practical settings, the ability to deploy a model trained on one dataset to a new, unseen dataset without requiring retraining is highly beneficial. This strategy aims to address two main goals: (1) overcoming the obstacle posed by insufficient data in the new dataset, which might hinder effective model training, and (2) saving computational resources,

as developing a model from the ground up demands significant time and resources. To assess how well our model adapts to new datasets, we initially pretrain the GTMP model using self-supervised learning objectives on source datasets. Subsequently, we finetune this pretrained model on target datasets to evaluate its performance in downstream task predictions.

The experimental results and sizes of datasets are shown in Table 2.9. (1) **Strong transfer ability across different source and target datasets.** It is evident that the transfer model, when applied from the source to the target dataset, can achieve performance on par with, or in some instances even surpassing, the model directly trained on the source dataset. Specifically, the discrepancy in average performance between scenarios where the source and target datasets are identical and those where they differ is a mere 0.91%. (2) **Correlation between tree sizes and transfer performance.** More importantly, our findings reveal that pretraining on datasets with larger average tree sizes can significantly enhance transfer performance on target datasets. In particular, the average performance when pretraining on datasets characterized by relatively larger tree sizes (such as lps-inter, lps-pc, and 5xfad-pc) surpassed that of datasets with smaller tree sizes (such as lps-glia and 5xfad-glia) by an average of 5.51%. Notably, the dataset with the largest average tree size, lps-inter, achieved an impressive average AUC score of 0.9979 across all datasets. These results underscore the ability of our proposed model to leverage larger dataset sizes for improving performance on unseen, relatively smaller-sized datasets. This capability presents a strategic advantage in addressing challenges related to data scarcity and computational constraints.

### 2.8.4 Ablation Studies

This paper primarily concentrates on exploring the fundamental question of how effectively representation learning can leverage uniquely designed properties of geometric

| | Neuron (↑) | | | River (↓) | |
|---|---|---|---|---|---|
| | lps-glia | lps-inter | 5xfad-pc | Diameter | Radius |
| Supervised | 0.7996 | 0.8529 | 0.9887 | 125.4951 | 61.2353 |
| GT-SSL | **0.9836** | **0.9996** | **0.9992** | **76.7699** | **33.0287** |
| No Ordering | 0.9769 | 0.9922 | 0.9981 | 78.8122 | 33.8345 |
| No Generative | 0.8235 | 0.8834 | 0.9847 | 108.5359 | 60.5332 |

Table 2.10: Ablation study results. NO Ordering refers to a variant that removes the partial ordering constraint module. NO Generative refers to another variant that removes the subtree growth learning module. The best result of each dataset is highlighted in bold.

trees. Here, we investigate the impact of the proposed two self-supervised learning components of GT-SSL framework. We first consider a variant **No Ordering** that removes the *Partial Ordering* module. To study the effectiveness of the proposed *subtree growth learning* module, we further construct a variant **No Generative** that removes the corresponding module. Due to length constraints, we only present the results of five real-world datasets in Table 2.10.

(1) Our full GT-SSL framework achieved the best performance on all five datasets. Specifically, the full model outperforms the variants **No Ordering** and **No Generative** by 10.3% on average. In turn, these two variants exceeded the performance of the supervised learning model by an average of 13.3%. These outcomes confirm that incorporating *partial ordering* and *subtree growth learning* modules significantly enhances geometric tree representation learning tasks.

(2) The performance drops significantly when we remove the *subtree growth learning* module, in comparison to removing the partial ordering module, which may indicate that this module plays a more critical role in understanding the joint geometric and tree topological properties towards learning powerful representations.

## 2.9 Conclusion

We first focus on the crucial problem of learning powerful representations from physical networks, which has tightly coupled spatial and graph information that can not be addressed by applying spatial and network methods separately. The proposed **S**patial **G**raph **M**essage **P**assing neural network (**SGMP**) effectively addresses the unique challenges in spatial networks by jointly considering the spatial and graph properties, and still maintain the invariance to node permutations, as well as rotation and translation transformations. In addition, our proposed accelerating algorithm largely alleviates the efficiency issue in solving spatial network issues. Experimental results on synthetic and real-world datasets demonstrate the outstanding discriminative power of our model, and the efficiency test shows a remarkable improvement in training time and scalability of our proposed accelerating method.

Besides, we generalize the problem to learning representations from spatial networks embedded in non-Euclidean manifolds, which is an underexplored area and can not be well handled by existing works. The proposed framework **M**anifold **S**pace **G**raph **N**eural **N**etwork (**MSGNN**) effectively addresses the unique challenges of representing irregular spatial networks by first converting the manifold space into a discrete mesh tessellation, and then converting the geometric information of the curves between nodes into messages on edges. Theoretical guarantees are given to prove that our learned representations are invariant to important symmetries such as rotation and translation, and simultaneously maintain strong distinguish power in geometric structures. Extensive experimental results on both synthetic and real-world datasets demonstrate the strength of our theoretical findings.

Finally, we propose the **G**eometric **T**ree **M**essage **P**assing (**GTMP**) model, designed to efficiently learn coupled spatial-topology representations from geometric tree-structured data. Theoretical guarantees are given to assure its ability to preserve essential geometric structure information. To overcome the challenge of insufficient

labeled data and to enhance transferability, we also introduce the Self-Supervised Learning Framework for Geometric Trees (GT-SSL). This framework significantly improves geometric tree representations by leveraging their inherent hierarchies and tree-oriented geometric structures. The integration of the GTMP model with the GT-SSL framework further accentuates its effectiveness, leading to state-of-the-art performance on various datasets.

# Chapter 3

# Representation Learning on Information Networks

In this chapter, we introduce our proposed deep learning framework designed to unify various information network domains through text-attributed graphs. We begin with the background in Section 3.1 and review related works in Section 3.2. The proposed self-supervised learning framework is detailed in Section 3.3, followed by extensive experimental results in Section 3.4. We conclude the chapter in Section 3.5.

This chapter features a work currently under submission [230], titled "TAGA: Text-Attributed Graph Self-Supervised Learning by Synergizing Graph and Text Mutual Transformations."

## 3.1 Introduction on Information Networks

Information networks [6], which represent abstract relationships between entities such as individuals, documents, or data sources, play a crucial role in a wide range of applications, including social networks [153, 148], citation networks [136], and recommendation systems [207, 98]. These networks are characterized by complex and diverse data domains, each introducing unique node and edge features, as well as dis-

tinct predictive tasks. This diversity presents significant challenges for developing a unified representation learning framework capable of generalizing across multiple domains. Existing approaches often struggle to accommodate such variations, limiting their effectiveness in diverse network types.

Recent advances in natural language processing (NLP), particularly with the advent of pre-trained language models [177, 113, 25], have demonstrated remarkable success in handling data from various domains, offering potential solutions to these challenges. In this research, we seek to unify the representation of information networks across diverse domains by leveraging textual descriptions of network elements. We introduce a novel structure, Text-Attributed Graphs (TAGs), that integrates textual attributes into graph representations, thereby providing a unified framework capable of learning from and generalizing across a wide range of information networks. This approach represents a significant step towards more robust and adaptable representation learning for complex, multi-domain information networks.

Text-Attributed Graphs are text documents that are connected in graph structures, allowing for deeper analysis and interpretation of complex relationships [224, 105, 106]. TAGs are prevalently used in numerous real-world applications, such as social networks [153], citation networks [136], and recommendation systems [207]. TAGs encompass textual content in both nodes and edges that elucidate the meaning of individual documents and who they are semantically correlated with. For instance, a scientific article network is a type of TAG that stores the texts of research papers and details about how they cite, criticize, and summarize each other within paragraphs. As shown in Figure 2.1(a), extracting knowledge like "*the first law proposed in Paper A is a special case of Paper B's Theorem 1 when under macro scale and low velocity*" from this scientific article network requires jointly considering semantics, topology, and their entanglement in the TAG.

Representation learning on TAGs is a promising, yet open research area that starts

to attract fast-increasing attention [216, 181, 35, 95, 64, 168, 130]. Existing TAG representation learning methods typically treat each text document as an independent node embedding and then rely entirely on message passing mechanisms to model the interaction between different texts. These approaches ignore the semantic-level textual connections between different nodes. Additionally, existing works are typically only applicable for supervised learning, which require extensively labeled data that is often unavailable in real-world scenarios. Moreover, the reliance on supervised tasks means that models are usually optimized for specific tasks and domains reflected in the training dataset, which significantly constrains their applicability to new domains or broader tasks. This limitation undermines the unique advantage of TAGs to leverage their universal linguistic attributes effectively. Although there are some graph pre-training models [93, 178, 221, 130] operate in an unsupervised manner, they often focus on either graph topology or node features independently, neglecting the crucial interplay between textual semantics and structural information inherent in TAGs.

Therefore, there is a pressing need for a method that comprehensively addresses the unique nature of TAGs, seamlessly integrating both their structural and semantic dimensions within a unified unsupervised framework. This presents a significant research challenge with several substantial hurdles to overcome. Primarily, developing a representation that can simultaneously leverage the textual semantic content, the graph structure, and their complex interplay presents significant difficulties. The scarcity of labeled training data further exacerbates this issue, making traditional supervised approaches impractical and necessitating innovative unsupervised strategies. Furthermore, the computational demands of such representation learning are substantial. The integration of large PLMs for textual corpus processing to be considered in TAGs creates a significant computational burden.

In order to address the aforementioned challenges, this paper proposes a new self-supervised learning framework named **T**ext-**A**nd-**G**raph Multi-View **A**lignment

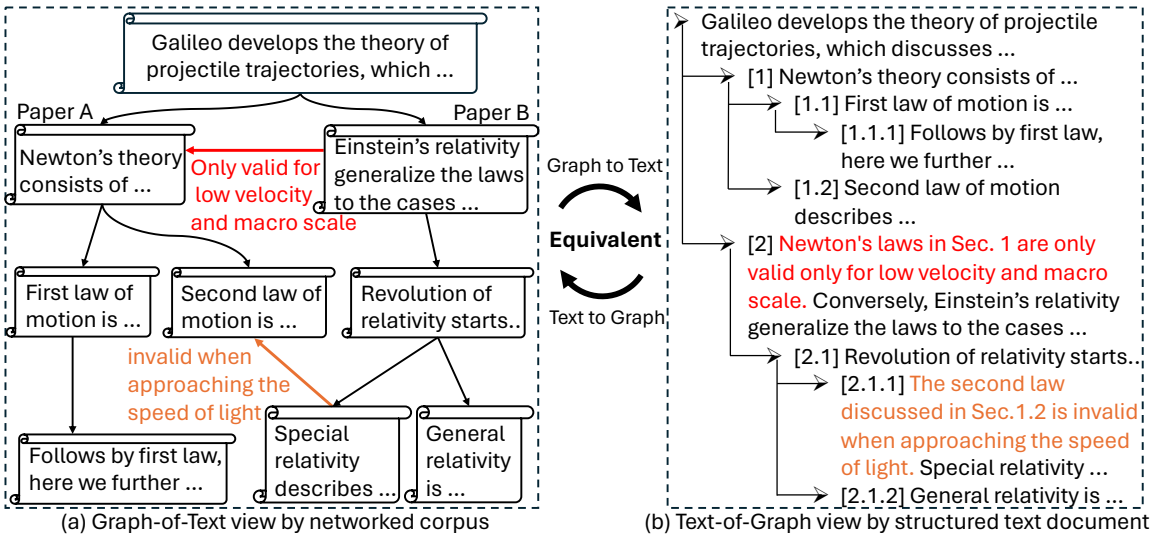


Figure 3.1: Illustration of the two distinct views of TAGs: (left) Graph-of-Text and (right) Text-of-Graph. Graph-of-Text view constructs a graph-structured data over the individual text corpora, while Text-of-Graph view organizes the text node and their connection description in a hierarchical layout document. These two views can be mutually transformed to each other.

(**TAGA**). TAGA jointly preserves rich semantic information, topology information, and their interplay by aligning representations of TAGs from two complementary views: the *Text-of-Graph* view and the *Graph-of-Text* view. As illustrated in Figure 3.1, these two views offer different representation formats of a TAG yet contain equivalent information. Specifically, the *Text-of-Graph* view organizes node texts into a structured textual document according to the TAG's topology. As exemplified in Figure 3.1(b), structured textual documents are universal ways to represent the relations among different text pieces in large corpus, especially in books, long articles, web files, etc. Here we propose a novel Graph2Text encoding module to automatically transfer a TAG to a structured textual document, which is readily to be processed by language models. Conversely, the *Graph-of-Text* view transforms textual nodes and topology into graph-structured data, which is then processed by a graph representation learning module (e.g. graph neural network). By aligning the representations learned from these two views, we encourage the learned representation to capture both textual and structural information, resulting in a unified, comprehensive representa-

tion of the TAG. Furthermore, to accelerate the training process, we propose a novel structure-preserving random walk algorithm. Finally, we demonstrate the strength of our proposed representation learning framework through extensive experiments on eight real-world datasets in zero-shot and few-shot prediction scenarios.

## 3.2 Related Works

### 3.2.1 Text-Attributed Graphs Representation Learning

Existing methods typically focus on supervised learning. GraphFormers [214] introduce GNN-nested Transformers to simultaneously capture graph topology and textual semantics, enhancing interactions between textual content and graph structure. Learning on Large-scale Text-attributed Graphs via Variational Inference [234] presents a variational inference framework that efficiently learns node representations on large-scale TAGs. Patton [104] pretrains language models on text-rich networks to capture semantic relationships. Recent developments have also seen efforts [199, 168, 130] in aligning graph representations with textual representations. For instance, G2P2 [199] employs contrastive learning to align GNN representations with text encoder outputs by averaging individual node text embeddings across various neighborhood hops during its pre-training phase. However, these methods often simplify the treatment of textual encoder embeddings for neighborhoods by averaging the embeddings of individual nodes. Similarly, GRENADE [130] implements a dual-level alignment strategy. This approach overlooks the underlying interactions within neighborhoods, leading to a loss of information that could be crucial for the contrastive objectives of alignment models.

### 3.2.2 Unsupervised Graph Pre-Train Methods

Existing unsupervised graph pre-training methods can be categorized into several categories based on their objectives and architectures. Graph autoencoder methods, graph autoencoder methods [115, 93] convert node and edge features into low-dimensional embeddings, which are then used to reconstruct the original graph data. Contrastive learning approaches, like DGI [178], GraphCL [221], GRACE [238], and $S^3$-CL [52], generate perturbed graph pairs by altering structural features, such as adding or removing nodes and edges or masking features, aiming to align the embeddings of these modified graphs closer in the embedding space. However, these methods often produce domain-specific embeddings with limited generalization ability across different domains, reducing their effectiveness in data-scarce or label-limited scenarios.

Recent developments have also seen efforts [199, 168, 130] in aligning graph representations with textual representations. For instance, G2P2 [199] employs contrastive learning to align GNN representations with text encoder outputs by averaging individual node text embeddings across various neighborhood hops during its pre-training phase. However, these methods often simplify the treatment of textual encoder embeddings for neighborhoods by averaging the embeddings of individual nodes. Similarly, GRENADE [130] implements a dual-level alignment strategy. This approach not only aligns GNN and text encoder embeddings but also encourages embeddings of connected node pairs to exhibit similarity. This approach overlooks the underlying interactions within neighborhoods, leading to a loss of information that could be crucial for the contrastive objectives of alignment models.

### 3.2.3 Graph2Text Encoding Methods

Recently, research include approaches [216, 181, 35, 95, 97, 64] that first transform the text-attributed graph into text sequence and then directly utilize LLMs as the

predictor given the transformed text and corresponding question as input prompt. These methods typically designs text templates to explicitly describe local graph structure by stating nodes and how they are connected in plain text. For example, *"The first node is .... The second node is .... .... First node connects to third node. Second node connects to ..."*. However, these methods do not present the structure in a natural language-speaking manner, which fails to fully leverage the pretrained capabilities of language models. This is due to the distributional shift between the transformed text from the graph and the original pretrained corpus, resulting in lower quality embeddings and high variance of performance [64].

### 3.2.4 Efficient and Scalable Methods for Large-Size Graph Neighborhoods

Efficiency and scalability are crucial for deep graph learning, particularly when dealing with large graphs or high-order interactions. Traditional graph sampling techniques, such as node sampling [31], edge sampling [85], or subgraph sampling [222], aim to reduce neighborhood size. However, these methods may not be suitable for TAGs, as they can result in the loss of important hierarchical interactive connection during the random sampling process. Meanwhile, in the NLP domain, some efforts [156, 88, 32, 100, 51] have been made to address the long context issue of PLMs. These approaches typically involve compressing input tokens into latent vectors [100] or modifying the attention mask [34, 88, 51] to reduce significant interactions. However, these methods often fail to preserve the original structure of the input corpus and might alter the hierarchical layout.

## 3.3 Self-Supervised Learning Framework on TAGs

To effectively address the substantial challenges of unsupervised representation learning on TAGs, we propose a novel self-supervised learning framework called **T**ext-**A**nd-**G**raph Multi-View **A**lignment (**TAGA**). Specifically, to jointly preserve both rich semantic information, topology information, and their interplay, we propose to learn and align the representations of TAG in two complementary views, namely text view and graph view. In particular, the text view is a *Text-of-Graph*, where the TAG's node texts are organized according to the TAG's topology into a collective textual hierarchical document, which inherently has the power to encompass logic and relational information among different node texts. The graph view is a *Graph-of-Text*, where the TAG's nodes and topology are turned into a graph structured data. These two views contain equivalent information but in different formats, allowing them to mutually supervise each other. Then the text view can be transformed by PLMs, which are adept at preserving textual information, while the graph view can be transformed by GNN, which are designed to guarantee preserving graph information. Therefore, by aligning the representations learned from these two views, we encourage the graph view's representation to also capture textual information and the text view's representation to also capture graph information. The above new idea is shown in Figure 2.3, where Figure 2.3(a) illustrates construction of *Graph-of-Text* view while Figure 2.3(b) illustrates *Text-of-Graph* view, as detailed in Section 3.3.1. In Section 3.3.2, we propose the Graph2Text module that can information loselessly transform the *Graph-of-Text* view to *Text-of-Graph* view. Their respectively transformed embeddings are aligned by our proposed TAG-hierarchical self-supervised learning framework, which is elaborated in Section 3.3.3. Finally, a novel acceleration algorithm of our learning process to reduce computational complexity to near linear is detailed in Section 3.3.4.

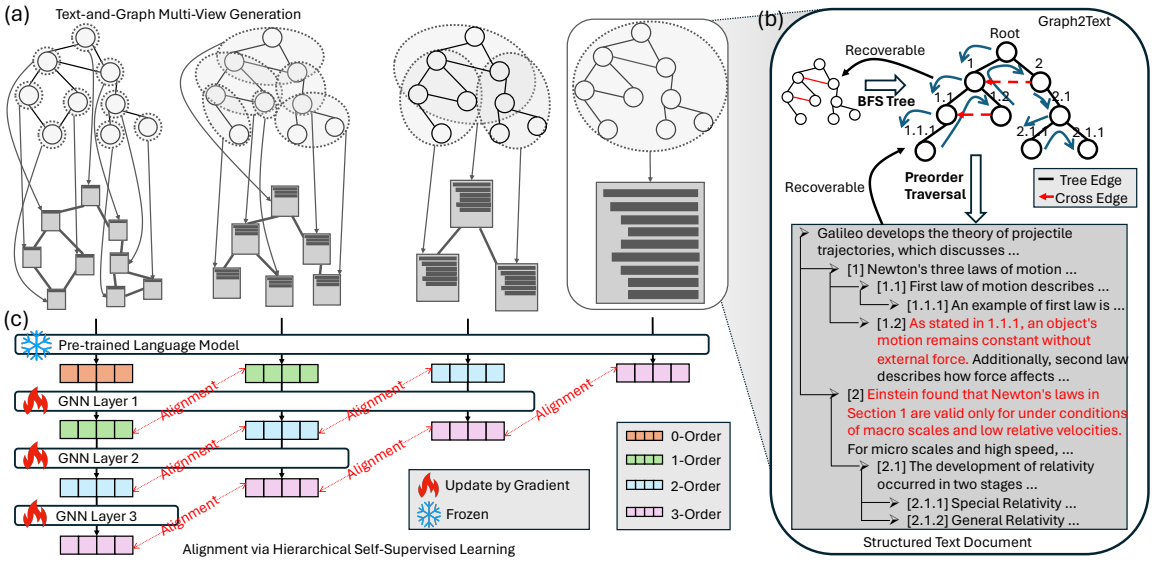


Figure 3.2: Illustration of the proposed self-supervised learning framework. (a) Generation of different orders of *Graph-of-Text* views; (b) The Graph2Text module that transforms a *Graph-of-Text* view into a *Graph-of-Text* view; (c) The alignment module via hierarchical self-supervised learning.

### 3.3.1 Text-and-Graph Multi-View Construction

Existing methods for learning representations on TAGs typically simply use GNNs to aggregate individual node embeddings generated from node texts. These methods lack the ability to consider the textual semantic relationship between different node texts in a joint document, and usually require supervised labels for training. Moreover, the resulting embeddings often lack generalization capabilities beyond the specific domain and task of their training data. To address these, our proposed framework **TAGA** first leverages two views of a TAG: *Text-of-Graph* (*TofG*) and *Graph-of-Text* (*GofT*). Each view can be defined at different neighborhood orders, allowing for a multi-order hierarchical representation. Specifically, a $k$-order *TofG* view represents a node's $k$-hop neighborhood as a single textual corpus that encompasses all nodes and their connections within that neighborhood. This corpus is then processed by a PLM to extract semantic embeddings that capture the combined content and structure within that $k$-hop neighborhood. In contrast, the corresponding $k$-order *GofT* view

is constructed as a graph structure, where nodes represent lower order *TofG*s within the $k$-hop neighborhood. A GNN model is then applied to aggregate information from these connected lower order *TofG*s, capturing the overall neighborhood context. This ensures that both *TofG* and *GofT* views at the same order encode equivalent information about the neighborhood.

To illustrate, consider a node with a 3-hop neighborhood, as shown in Figure 2.3(a). Its 3-order *TofG* is constructed by transforming the entire 3-hop neighborhood as a single text corpus. Three distinct 3-order *GofT* views can then be created using *TofG*s of orders 0, 1, and 2 as nodes in the graph structure. To maintain information consistency, the number of GNN aggregation layers decreases with increasing *TofG* order: 3 layers for 0-order *TofG*s, 2 for 1-order *TofG*s, and 1 for 2-order *TofG*s. This ensures that each 3-order *GofT* view captures the same 3-hop neighborhood information as the 3-order *TofG* view, facilitating information equivalent views to enable further self-supervised learning alignment.

### 3.3.2 Represent Text Neighborhood Information via Hierarchical Document Layout

The key to our proposed self-supervised learning framework is ensuring that the two distinct graph views (*TofG* and *GofT*) contain equivalent information. This necessitates constructing a *TofG* view through the Graph2Text module that preserves all connectivity information present in the original TAG. Existing methods [64, 97, 199, 168] often struggle to effectively represent the structural information of graphs in a way that is both comprehensive and natural to language model understanding. These methods typically designs text templates to explicitly describe local graph structure by stating nodes and how they are connected in plain text. For example, *"The first node is .... The second node is .... First node connects to third node. Second node connects ..."*. However, these methods usually fails to fully leverage the pretrained

capabilities of language models because they do not present the structure in a natural language-speaking manner. This discrepancy between the transformed graph text and the original pre-training corpus leads to a distributional shift, hindering the PLM's ability to generate high-quality embeddings that accurately reflect both the semantic and structural aspects of the TAG.

To address this issue, we introduce a novel Graph2Text approach that transforms a graph neighborhood into a hierarchical text document. This hierarchical structure mirrors the original graph's topology, ensuring that the document's latent structure is equivalent to the graph itself. Crucially, the resulting document resembles a natural document, aligning with the distribution of majority text data used to pre-train PLMs. This alignment mitigates the distributional shift issue, allowing PLMs to generate embeddings that accurately reflect both the semantic and structural aspects of the graph.

Specifically, the structure of a node and its k-hop neighborhood can be represented as an ego graph, with the node itself as the root. This ego graph can be decomposed into a hierarchical tree backbone and a set of cross-edges, as illustrated in Figure 2.3(b). The reading order is established for the *TofG* document through a pre-order traversal of this tree structure (first visit the root, then the left subtree, then the right subtree), capturing the hierarchical relationships between nodes. To fully represent the neighborhood's structure, we then incorporate cross-edges into the document. These cross-edges indicate connections from later sections of the document back to earlier ones, effectively mirroring the original graph's topology within the text format.

As shown in Algorithm 1, the $k$-hop neighborhood of a target node $v$ in graph $G$ is represented as an ego-graph $\mathcal{G}(v, k)$. A breadth-first search (BFS) tree $\hat{\mathcal{T}}(v, k)$, rooted at $v$, provides a hierarchical structure for the document, while cross-edges (edges outside the BFS tree) are identified. A pre-order traversal of $\hat{\mathcal{T}}(v, k)$ establishes the

document's hierarchical layout, assigning each node a section number. Cross-edges are then integrated by adding references at source nodes to the sections containing their respective destination nodes, if the destination node appears earlier in the traversal. This approach ensures that the document faithfully reflects the graph's structure.

### 3.3.3 Multi-View Alignment via TAG Hierarchical Self-Supervised Learning

Upon construction of both views at different orders, a hierarchical self-supervised learning module is proposed to align the embeddings from both views. Given a TAG $\mathcal{G}$ with at most $K$-hop neighborhood size, for each node $v_i \in \mathcal{V}$, its $k$-hop neighborhood can be denoted as $\mathcal{N}_k(v_i)$ and its corresponding $k$-order *TofG* view embedding can be represented as:

$$\begin{aligned} \mathbf{h}_k(v_i) &= \text{PLM}\left(\mathit{TofG}(v_i; k)\right), \\ \mathit{TofG}(v_i; k) &= \text{Graph2Text}\left(v_i \cup \mathcal{N}(v_i, k)\right), \end{aligned} \tag{3.1}$$

where PLM is a pre-trained language model (e.g. BERT [112] or LlaMA [172]). Graph2Text is an encoding template function that can transform individual nodes and edges text into a textual corpus. Meanwhile, its corresponding $k$-order *GofT* views embeddings can be denoted as GNN aggregated representations of lower order *TofG*s:

$$\mathbf{b}_k^l(v_i) = f^{(k-l)}\left(\{\mathbf{h}_l(v_b) | v_b \in v_i \cup \mathcal{N}(v_i, k-l)\}\right), \tag{3.2}$$

where $l$ covers from 0 to $k-1$ and $f^{(k-l)}$ denotes the GNN model with $k-l$ layers.

By aggregating $k-l$ layers of information over the connected $l$-order *TofG*s, the obtained $k$-order *GofT* embeddings cover equivalent information with the $k$-order *TofG* view embedding. Therefore, given all the embeddings from level 1 to $K$, the

**Algorithm 1** Hierarchical Document Layout (HDL) for Graph2Text

**Input:** Graph $G$, target node $v$, hop count $k$
**Output:** Hierarchical text document $D$

$\hat{\mathcal{G}}(v,k) \leftarrow$ Construct ego-graph of $v$ up to $k$ hops in $G$
$\hat{\mathcal{T}}(v,k) \leftarrow$ BFS tree of $\hat{\mathcal{G}}(v,k)$ rooted at $v$
$\hat{\mathcal{E}}^{\text{cross}}(v,k) \leftarrow$ Cross-edges in $\hat{\mathcal{G}}(v,k)$
$D \leftarrow$ Assign document sections to nodes following pre-order traversal
**for** each cross-edge $e = (u, w)$ **do**
    **if** $w$ precedes $u$ **then**
        Add reference at $u$ to section containing $w$ in $D$
    **end if**
**end for**
**return** $D$

**Algorithm 2** Structure-Preserving Random Walk Traversal

**Input:** Root node $v$, cross-edge probability $p$, maximum length $L$
**Output:** Traversal path $P$

$P \leftarrow [v]$
**while** $|P| < L$ and $v$ has children **do**
    **if** random() ¡ $p$ and $v$ has cross-edges **then**
        $v \leftarrow$ Random neighbor by cross-edge
    **else**
        $v \leftarrow$ Random child of $v$
    **end if**
    $P \leftarrow P + [v]$
**end while**
**return** $P$

supervision objective function can be written as:

$$\mathcal{L}_{\text{positive}} = -\frac{1}{K|\mathcal{B}|} \sum_{v_i \in \mathcal{B}} \sum_{k \in [1,K]} \sum_{l \in [0,k-1]} \rho\left(\mathbf{b}_k^l(v_i), \mathbf{h}_k(v_i)\right), \tag{3.3}$$

where $\mathcal{B}$ represents the minibatch and $\rho$ denotes a similarity function, such as cosine similarity. Additionally, we include the negative samples that chosen from other nodes within the minibatch:

$$\mathcal{L}_{\text{negative}} = \frac{1}{K|\mathcal{B}|} \sum_{v_i, v_j \in \mathcal{B}, v_1 \neq v_2} \sum_{k \in [1,K]} \sum_{l \in [0,k-1]} \rho\left(\mathbf{b}_k^l(v_i), \mathbf{h}_k(v_j)\right), \tag{3.4}$$

Thus, the overall objective function can be denoted as:

$$\mathcal{L} = \mathcal{L}_{\text{positive}} + \mathcal{L}_{\text{negative}} \tag{3.5}$$

**Time Complexity Analysis.** Consider a TAG with a maximum $K$-hop neighborhood size, where each node has an average degree $d$ and text attribute length

$L$. Assume the feature dimensionality is $F$. In the case of transformer-based PLMs, the time complexity for processing the *TofG* view of a node would be $O((dL)^2K^2)$, due to the quadratic complexity of self-attention mechanisms with respect to input sequence length. In contrast, our method employs a GNN to aggregate information from lower-order *TofG*s, each of length $dL$. Assuming a GNN with constant complexity per layer, the time complexity for aggregating information from all $K$ levels of the *GofT* view would be $O(L^2dK)$. Our method achieves significantly higher efficiency than directly using PLMs for *TofG* views, with details available in the Appendix B.2.

### 3.3.4 Accelerating Training on Large TAGs with Structure-Preserving Random Walk

While TAGA significantly improves efficiency during inference by transferring knowledge from the PLM to a GNN model, the pre-training stage still encounters computational bottlenecks due to the quadratic complexity of transformers with respect to context length when generating *TofG* view embeddings. Existing graph sampling methods (e.g., node or edge dropping) can partially alleviate this issue, but at the cost of sacrificing valuable structure information, which is crucial for capturing the intricate relationships within TAGs.

To address this issue while preserving the structure of corpus, we propose a novel approach inspired by human reading patterns. Our method segments the hierarchical corpus into multiple related sub-corpora, mirroring how humans naturally engage with complex documents: starting with a general overview (top of the hierarchy) and delving into specific sections (sub-corpora). By navigating the corpus multiple times, focusing on different sub-corpora each time, the combined insights gained can effectively approximate the understanding achieved from processing the entire corpus.

To facilitate this behavior, we introduce a random walk-based neighborhood traversal algorithm. It simulates a reader starting at the root node and progressing towards

leaf nodes in the BFS tree, transitioning from general to specific information. Additionally, at each step, there is a probability $p$ of jumping to another node via cross-edges, imitating the non-linear navigation often observed in human reading (e.g., jumping to related topics or backtracking). By averaging multiple random walk traversals, the generated paths can approximate the complete corpus. As detailed in Algorithm 2, each traversal begins at the root node $v$ and iteratively samples child nodes to form a path down the hierarchy. At each step, a jump to another node via cross-edges is possible with probability $p$. This traversal continues until reaching a predefined length or a leaf node.

## 3.4 Experiments

In this section, the experimental settings are introduced first in Section 3.4.1, then the zero-shot and few-shot node classification performances are presented in Section 3.4.2, and link prediction performance is presented in Appendix B.1.2. We further present the effectiveness under transfer learning settings in Section 3.4.3. We measure model efficiency in Section 3.4.6. The effectiveness of framework components through ablation studies is in Section 3.4.4. The parameter sensitivity experiments are present in Section 3.4.5.

### 3.4.1 Experimental Settings

**Datasets.** We evaluate on eight real-world text-attributed graph datasets across different domains. Specifically, three citation networks Cora [215], Pubmed [215] and Arxiv [94], two book networks Children [125] and History [125], and three E-commerce networks Computers [125], Photo [125], and Sports [213] are chosen as our evaluation datasets. Datasets statistics can be found in Table 3.1.

**Comparison Methods.** We choose the textual embedding of the text corpus as the

baseline, which is denoted as "PLM" in our experimental results tables. Additionally, we compare our proposed framework with six state-of-the-art graph pre-train methods. Specifically, GraphMAE [115] — utilizes masked autoencoder technique to predict of graph structure and node features. GraphCL [221] and GRACE [238] applies various graph augmentations to generate contrastive pairs. GraphFormers [214] and Patton [104] insert GNN layer into transformers architecture. G2P2 [199] aligns GNN embeddings and text encoder embeddings through contrastive learning.

**Implementation Details.** We choose two different pre-trained language models (OpenAI's `text-embedding-3-small` [151] and `UAE-Large-V1` [127]) to generate text embeddings for robust results. Commonly used GNN models (GCN [116], GIN [85], GraphSAGE [211]) are chosen as the backbone model as the backbone model for both our method and all comparison methods. For a fair comparison, all models are required to adhere to the same GNN architecture, including the number of convolution layers and hidden dimensions. More details about hyperparameters can be found in Appendix B.1.1. Further technical details can be found in Appendix B.2. Our code can be found at anonymous link `https://anonymous.4open.science/r/TAGA-32B7/`.

### 3.4.2 Effectiveness Results

In this section, we assess the effectiveness of our proposed unsupervised representation learning framework compared to other methods under conditions of label scarcity. Our representation learning models are initially pre-trained on each TAG dataset without any supervised labels. After the pre-training phase, we evaluate the quality of the obtained node embeddings under zero-shot conditions by measuring the similarity between these embeddings and the corresponding text label embeddings. To further gauge performance in scenarios with limited labeled data, we conduct evaluations using 1, 3, 5, 10, 20, 50, and 100-shot settings. Due to space limitation, the results with

| $k$-Shot | Model | Arxiv | Children | Computers | Cora | History | Photo | Pubmed | Sports |
|---|---|---|---|---|---|---|---|---|---|
| | # Nodes | 169,343 | 76,875 | 87,229 | 2,708 | 41,551 | 48,362 | 19,717 | 173,055 |
| | # Edges | 1,166,243 | 1,554,578 | 721,107 | 10,556 | 358,574 | 500,939 | 44,338 | 1,773,594 |
| | Avg # Words | 220.7 | 199.3 | 90.7 | 148.2 | 218.7 | 144.5 | 50.1 | 9.8 |
| 0 | PLM | 0.500 ± 0.001 | 0.094 ± 0.003 | 0.427 ± 0.001 | 0.624 ± 0.005 | 0.169 ± 0.001 | 0.387 ± 0.009 | 0.475 ± 0.008 | 0.316 ± 0.002 |
| | GraphMAE | 0.104 ± 0.001 | 0.021 ± 0.001 | 0.049 ± 0.001 | 0.194 ± 0.006 | 0.019 ± 0.001 | 0.152 ± 0.001 | 0.438 ± 0.001 | 0.112 ± 0.001 |
| | GraphCL | 0.089 ± 0.001 | 0.037 ± 0.001 | 0.173 ± 0.001 | 0.176 ± 0.003 | 0.191 ± 0.001 | 0.174 ± 0.001 | 0.368 ± 0.001 | 0.140 ± 0.001 |
| | GRACE | 0.045 ± 0.001 | 0.034 ± 0.001 | 0.169 ± 0.001 | 0.146 ± 0.004 | 0.079 ± 0.001 | 0.025 ± 0.001 | 0.335 ± 0.001 | 0.057 ± 0.001 |
| | GraphFormers | 0.465 ± 0.003 | 0.076 ± 0.001 | 0.147 ± 0.001 | 0.641 ± 0.004 | 0.185 ± 0.005 | 0.192 ± 0.003 | 0.441 ± 0.005 | 0.368 ± 0.002 |
| | PATTON | 0.496 ± 0.005 | 0.027 ± 0.001 | 0.106 ± 0.003 | 0.579 ± 0.003 | 0.096 ± 0.003 | 0.118 ± 0.002 | 0.329 ± 0.005 | 0.421 ± 0.005 |
| | G2P2 | 0.453 ± 0.002 | 0.201 ± 0.001 | 0.453 ± 0.001 | 0.644 ± 0.004 | 0.322 ± 0.003 | **0.452 ± 0.001** | 0.576 ± 0.006 | 0.436 ± 0.001 |
| | TAGA | **0.537 ± 0.003** | **0.224 ± 0.001** | **0.498 ± 0.004** | **0.682 ± 0.005** | **0.351 ± 0.009** | 0.419 ± 0.001 | **0.616 ± 0.009** | **0.448 ± 0.003** |
| | TAGA-rw | 0.530 ± 0.001 | 0.221 ± 0.001 | 0.494 ± 0.001 | 0.680 ± 0.002 | 0.301 ± 0.003 | 0.394 ± 0.001 | 0.599 ± 0.002 | 0.434 ± 0.002 |
| 1 | PLM | 0.280 ± 0.044 | 0.122 ± 0.042 | 0.238 ± 0.039 | 0.412 ± 0.080 | 0.284 ± 0.078 | 0.230 ± 0.051 | 0.503 ± 0.067 | 0.282 ± 0.068 |
| | GraphMAE | 0.255 ± 0.041 | 0.128 ± 0.028 | 0.300 ± 0.052 | 0.474 ± 0.058 | 0.231 ± 0.052 | 0.304 ± 0.066 | 0.492 ± 0.076 | 0.270 ± 0.042 |
| | GraphCL | 0.123 ± 0.031 | 0.157 ± 0.066 | 0.256 ± 0.039 | 0.402 ± 0.059 | 0.371 ± 0.124 | 0.325 ± 0.079 | 0.414 ± 0.040 | 0.347 ± 0.079 |
| | GRACE | 0.263 ± 0.034 | 0.138 ± 0.035 | 0.336 ± 0.051 | 0.435 ± 0.071 | 0.266 ± 0.085 | 0.295 ± 0.053 | 0.514 ± 0.095 | 0.282 ± 0.045 |
| | GraphFormers | 0.233 ± 0.042 | 0.131 ± 0.038 | 0.247 ± 0.052 | 0.463 ± 0.069 | 0.231 ± 0.055 | 0.284 ± 0.043 | 0.471 ± 0.054 | 0.284 ± 0.057 |
| | PATTON | 0.217 ± 0.059 | 0.127 ± 0.042 | 0.305 ± 0.048 | 0.487 ± 0.057 | 0.286 ± 0.078 | 0.318 ± 0.053 | 0.523 ± 0.051 | 0.243 ± 0.068 |
| | G2P2 | 0.308 ± 0.052 | 0.145 ± 0.029 | 0.359 ± 0.044 | 0.477 ± 0.082 | 0.361 ± 0.092 | 0.372 ± 0.066 | 0.522 ± 0.085 | 0.356 ± 0.042 |
| | TAGA | **0.323 ± 0.040** | **0.180 ± 0.073** | **0.380 ± 0.062** | 0.509 ± 0.089 | **0.413 ± 0.114** | **0.417 ± 0.077** | **0.563 ± 0.062** | 0.440 ± 0.070 |
| | TAGA-rw | 0.307 ± 0.050 | 0.171 ± 0.013 | 0.365 ± 0.042 | **0.561 ± 0.063** | 0.383 ± 0.078 | 0.380 ± 0.037 | 0.548 ± 0.073 | **0.498 ± 0.084** |
| 5 | PLM | 0.500 ± 0.019 | 0.210 ± 0.025 | 0.377 ± 0.027 | 0.641 ± 0.031 | 0.557 ± 0.040 | 0.420 ± 0.037 | 0.632 ± 0.040 | 0.478 ± 0.056 |
| | GraphMAE | 0.425 ± 0.028 | 0.212 ± 0.029 | 0.434 ± 0.036 | 0.704 ± 0.038 | 0.459 ± 0.038 | 0.489 ± 0.038 | 0.625 ± 0.049 | 0.452 ± 0.037 |
| | GraphCL | 0.231 ± 0.015 | 0.201 ± 0.040 | 0.397 ± 0.040 | 0.641 ± 0.044 | 0.531 ± 0.047 | 0.462 ± 0.041 | 0.584 ± 0.037 | 0.477 ± 0.048 |
| | GRACE | 0.445 ± 0.028 | 0.227 ± 0.031 | 0.472 ± 0.040 | 0.685 ± 0.027 | 0.481 ± 0.061 | 0.515 ± 0.042 | 0.628 ± 0.047 | 0.482 ± 0.040 |
| | GraphFormers | 0.461 ± 0.022 | 0.230 ± 0.031 | 0.374 ± 0.031 | 0.731 ± 0.029 | 0.458 ± 0.045 | 0.498 ± 0.032 | 0.619 ± 0.039 | 0.568 ± 0.053 |
| | PATTON | 0.471 ± 0.039 | 0.227 ± 0.040 | 0.405 ± 0.032 | 0.699 ± 0.025 | 0.466 ± 0.038 | 0.518 ± 0.030 | 0.605 ± 0.042 | 0.532 ± 0.048 |
| | G2P2 | 0.466 ± 0.025 | 0.240 ± 0.034 | 0.510 ± 0.039 | 0.703 ± 0.032 | 0.617 ± 0.053 | 0.583 ± 0.051 | 0.640 ± 0.051 | 0.565 ± 0.055 |
| | TAGA | **0.483 ± 0.022** | 0.263 ± 0.031 | **0.543 ± 0.038** | 0.752 ± 0.028 | **0.636 ± 0.046** | 0.602 ± 0.041 | 0.649 ± 0.044 | 0.664 ± 0.061 |
| | TAGA-rw | 0.471 ± 0.031 | **0.276 ± 0.053** | 0.508 ± 0.019 | **0.764 ± 0.027** | 0.621 ± 0.076 | **0.594 ± 0.025** | **0.684 ± 0.027** | **0.675 ± 0.070** |
| 10 | PLM | 0.526 ± 0.013 | 0.240 ± 0.018 | 0.463 ± 0.029 | 0.690 ± 0.017 | 0.639 ± 0.038 | 0.491 ± 0.028 | 0.679 ± 0.023 | 0.535 ± 0.038 |
| | GraphMAE | 0.461 ± 0.017 | 0.234 ± 0.014 | 0.511 ± 0.028 | 0.761 ± 0.023 | 0.535 ± 0.042 | 0.543 ± 0.035 | 0.659 ± 0.028 | 0.508 ± 0.028 |
| | GraphCL | 0.301 ± 0.018 | 0.233 ± 0.029 | 0.488 ± 0.031 | 0.702 ± 0.025 | 0.566 ± 0.043 | 0.523 ± 0.044 | 0.632 ± 0.025 | 0.531 ± 0.035 |
| | GRACE | 0.488 ± 0.018 | 0.251 ± 0.015 | 0.552 ± 0.028 | 0.754 ± 0.018 | 0.567 ± 0.054 | 0.567 ± 0.031 | 0.670 ± 0.025 | 0.529 ± 0.033 |
| | GraphFormers | 0.482 ± 0.019 | 0.248 ± 0.030 | 0.447 ± 0.028 | 0.778 ± 0.022 | 0.498 ± 0.035 | 0.538 ± 0.026 | 0.633 ± 0.034 | 0.601 ± 0.040 |
| | PATTON | 0.501 ± 0.028 | 0.247 ± 0.024 | 0.451 ± 0.026 | 0.738 ± 0.020 | 0.533 ± 0.029 | 0.539 ± 0.028 | 0.643 ± 0.028 | 0.564 ± 0.041 |
| | G2P2 | **0.527 ± 0.014** | 0.269 ± 0.018 | 0.598 ± 0.031 | 0.753 ± 0.020 | 0.649 ± 0.046 | 0.632 ± 0.037 | 0.691 ± 0.029 | 0.618 ± 0.037 |
| | TAGA | 0.521 ± 0.017 | 0.288 ± 0.025 | **0.622 ± 0.025** | 0.788 ± 0.021 | **0.679 ± 0.041** | **0.651 ± 0.048** | **0.714 ± 0.024** | **0.705 ± 0.045** |
| | TAGA-rw | 0.518 ± 0.010 | **0.288 ± 0.040** | 0.595 ± 0.024 | **0.806 ± 0.011** | 0.652 ± 0.046 | 0.626 ± 0.020 | 0.679 ± 0.013 | 0.662 ± 0.056 |

Table 3.1: Performance in zero-shot and few-shot node classification for each dataset and setting. The best-performing model is highlighted in **bold**, and the second-best performing model is underlined.

text encoder `UAE-Large-V1` under zero-shot and 1, 5, 10-shot settings is reported in Table 3.1. Our acceleration method with random walk is denoted as "TAGA-rw". The results with `text-embedding-3-small` and other few-shot settings can be found in Appendix B.1.3. We also present zero-shot link prediction performance in Appendix B.1.2.

**Zero-shot performance.** Table 3.1 presents node classification accuracy under zero-shot conditions, where our method consistently outperforms all comparison methods in seven out of eight datasets. On average, our method surpasses other graph pre-training methods by 47.84% and exceeds the second-best model by 6.78%. These findings demonstrate the enhanced ability of our pre-trained model to effectively learn representations that enable zero-shot predictions. Furthermore, compared to direct

textual embeddings from the PLM, our method improves zero-shot performance by an average of 20.76%. This demonstrates our method's capacity in integrating structural and textual information from neighborhoods over directly using the PLM. Interestingly, our method exhibits a stronger performance advantage when dealing with data rich in textual information. Specifically, for the two citation networks (Arxiv and Cora), which possess significantly longer text attributes compared to other datasets, our method surpasses the second-best performing graph pretrained model by an average of 10.33%. This proves our method can effectively leverage the rich textual information.

**Few-shot performance.** For few-shot experiments, our method consistently outperforms all comparison methods, achieving a 15.55% average improvement and surpassing the second-best model by 6.28% on average. Notably, our method exhibits a more pronounced advantage in scenarios with limited labeled data (¡=5 shots), where it outperforms all other methods by an average of 19.79% and exceeds the second-best model by 7.91% on average. This underscores the effectiveness of our method, particularly in settings where few-shot learning is essential due to data labels constraints.

**Remarks.** It is worth noting that for some datasets, the zero-shot performance of our method can match or even exceed few-shot predictive results, particularly when the number of training samples for few-shot learning is limited. For example, on five datasets (Arxiv, Children, Computers, Cora, and Pubmed), the zero-shot performance surpasses 1-shot performance by an average of 23.54%. Remarkably, the zero-shot performance can even be comparable to that of 5-shot. This demonstrates the strong potential of our method in scenarios where labeled data is scarce or unreachable.

| | Source<br><br>Target | Cora<br>↓<br>Arxiv | Arxiv<br>↓<br>Cora | Cora<br>↓<br>Pubmed | Pubmed<br>↓<br>Cora | Children<br>↓<br>History | History<br>↓<br>Children | Computers<br>↓<br>Photo | Photo<br>↓<br>Computers |
|---|---|---|---|---|---|---|---|---|---|
| 0-shot | GRACE | 0.021 | 0.173 | 0.360 | 0.302 | 0.073 | 0.065 | 0.099 | 0.070 |
| | GraphMAE | 0.012 | 0.153 | 0.434 | 0.239 | 0.009 | 0.030 | 0.082 | 0.004 |
| | GraphCL | 0.015 | 0.232 | 0.368 | 0.178 | 0.045 | 0.024 | 0.094 | 0.135 |
| | G2P2 | 0.241 | 0.647 | 0.421 | 0.533 | **0.204** | 0.100 | 0.297 | 0.340 |
| | TAGA | **0.406** | **0.679** | **0.484** | **0.559** | 0.184 | 0.200 | 0.452 | **0.372** |
| | TAGA-rw | 0.398 | 0.624 | 0.408 | 0.526 | 0.176 | **0.203** | **0.455** | 0.348 |
| 5-shot | GRACE | 0.426 | 0.721 | 0.591 | 0.657 | 0.609 | 0.219 | 0.483 | 0.382 |
| | GraphMAE | 0.426 | 0.645 | 0.578 | 0.515 | 0.527 | 0.160 | 0.367 | 0.294 |
| | GraphCL | 0.107 | 0.678 | 0.436 | 0.416 | 0.598 | 0.178 | 0.395 | 0.345 |
| | G2P2 | 0.395 | 0.749 | 0.633 | 0.708 | 0.623 | 0.239 | 0.509 | 0.429 |
| | TAGA | **0.475** | 0.754 | **0.655** | **0.734** | **0.651** | **0.257** | **0.528** | **0.448** |
| | TAGA-rw | 0.443 | **0.764** | 0.644 | 0.674 | 0.617 | 0.250 | 0.482 | 0.436 |

Table 3.2: Transfer learning results for node classification. The best-performing model is highlighted in **bold**.

### 3.4.3 Transfer Ability Analysis

In real-world applications, not only labels are difficult to obtain, but the data itself is also scarce. This necessitates the generalization of a pre-trained model to a data domain distinct from the pre-training data. Here we evaluate the zero-shot and few-shot performance under transfer learning settings. Specifically, the model is unsupervisedly pre-trained on the source data domain and then transferred to the target data domain. No further fine-tuning is performed for zero-shot prediction, and is fine-tuned using the limited training samples for few-shot prediction.

In Table 3.2, we present the performance of zero-shot and five-shot predictions across eight pairs of source and target datasets. The results demonstrate a clear advantage for our method in the zero-shot setting, where it consistently outperforms all other methods across all dataset pairs. Notably, our method achieves an average improvement of 26.5% over the second-best performing method. In the five-shot setting, our method continues outperforming the second-best performing method by 4.53% on average. Particularly when transferring from Cora to Arxiv and Pubmed, and Children to History, our method achieves significant performance gain by 6.30% on average, demonstrating its ability to effectively leverage limited labeled data in

| | Method | arxiv | children | computers | cora | history | photo | pubmed | sports |
|---|---|---|---|---|---|---|---|---|---|
| 0-shot | Full | **0.537** | **0.224** | 0.498 | **0.682** | **0.351** | **0.419** | **0.616** | **0.448** |
| | TofG-0 | 0.500 | 0.099 | 0.423 | 0.575 | 0.318 | 0.392 | 0.471 | 0.318 |
| | TofG-1 | 0.521 | 0.102 | 0.544 | 0.601 | 0.349 | 0.336 | 0.512 | 0.444 |
| | TofG-2 | 0.519 | 0.098 | **0.556** | 0.606 | 0.348 | 0.327 | 0.532 | **0.448** |
| | Glo-GofT | 0.533 | 0.205 | 0.482 | 0.657 | 0.329 | 0.407 | 0.522 | 0.417 |
| 5-shot | Full | 0.483 | **0.263** | 0.543 | **0.752** | **0.636** | **0.602** | **0.649** | **0.664** |
| | TofG-0 | **0.500** | 0.210 | 0.377 | 0.641 | 0.557 | 0.420 | 0.632 | 0.478 |
| | TofG-1 | 0.496 | 0.234 | 0.549 | 0.709 | 0.598 | 0.582 | 0.631 | 0.615 |
| | TofG-2 | 0.490 | 0.234 | **0.558** | 0.706 | 0.589 | 0.590 | 0.631 | 0.654 |
| | Glo-GofT | 0.479 | 0.257 | 0.512 | 0.726 | 0.623 | 0.592 | 0.635 | 0.629 |

Figure 3.3: Ablation studies results of zero- and five-shot settings. Here "Full" denotes our full model.

the target domain.

### 3.4.4 Ablation Study

To investigate the effectiveness of our proposed model compared to simpler heuristics, we conducted a series of ablation analyses. We began by considering textual embeddings obtained directly by applying the PLM to the Text of Graph views' corpus at different orders. This allowed us to assess the impact of our training procedure compared to a simpler approach that relies solely on Text-of-Graph view representations. In addition, we compare our full model with a variant, *Glo-GofT*, which only aligns the GNN embeddings that aggregate individual node's text embeddings but removes all higher-order Graph-of-Text embeddings. The results of these ablation studies are presented in Table 3.3, which reveals that removing components of our full model generally leads to a decrease in performance. In the zero-shot setting, the full model outperforms the variant models by 2.79% to 8.49% on average, and ranges from 1.74% to 9.71% in the five-shot setting. These results underscore the contribution of each component to TAGA's overall effectiveness. In Appendix B.1.4, we have shown additional ablation studies that evaluate how will aligning on different orders of hierarchies will influence the representation due to space limitation.

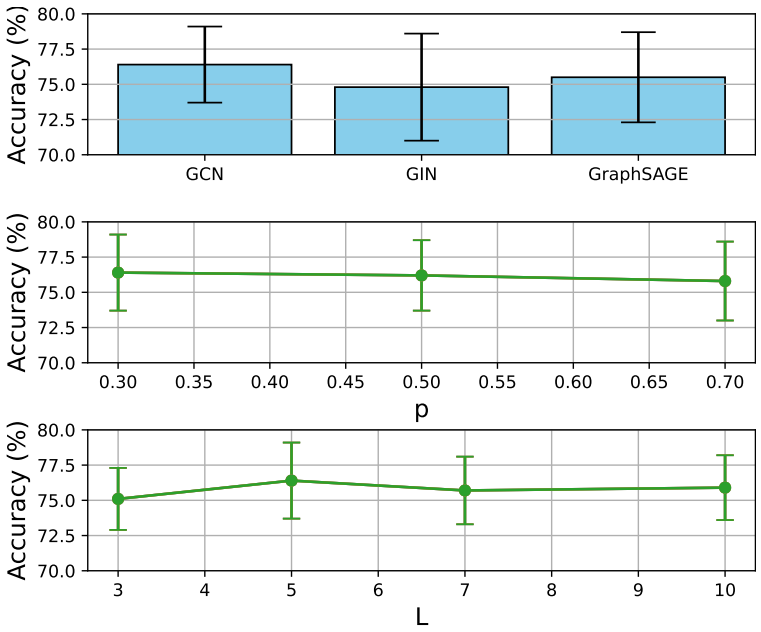


Figure 3.4: Comparison of five-shot performance between (top) different GNN encoder choices, and (middle) varying jumping ratio, and (bottom) maximum walk length of random walks.

### 3.4.5 Sensitivity Analysis

In this section, we investigate the sensitivity of the key hyperparameters and their impact on TAGA's performance. Specifically, we first evaluate how different GNN backbones (GCN, GIN, and GraphSAGE) affect performance. Then we evaluate how jumping ratio ($p$) and maximum walk length ($L$) would affect random walk's performance. The results are presented in Figure 3.4. The sensitivity analysis conducted on TAGA's performance demonstrates that the method is robust across a range of hyperparameters. Specifically, the variance in performance across different GNN backbones is 0.84%, indicating a stable behavior regardless of the backbone employed. Similarly, adjustments in the jumping ratio ($p$) and maximum walk length ($L$) exhibit 0.33% and 0.76% variance on average, which underscores that our method is not sensitive to the hyperparameters chosen.

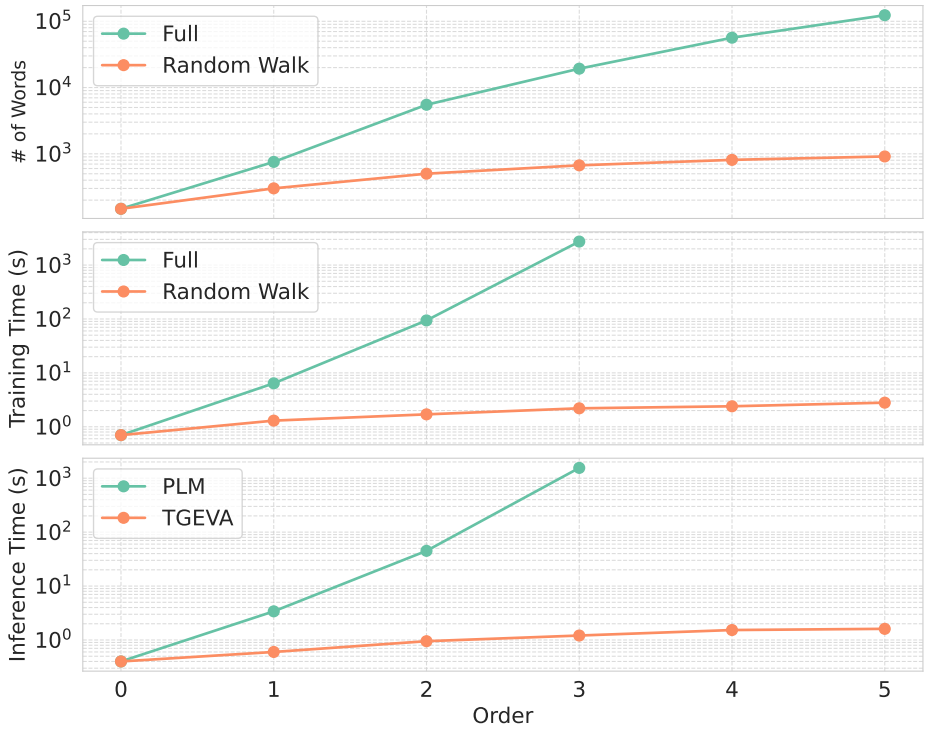


Figure 3.5: (top) Comparison of the full method and the random walk algorithm in terms of the number of words, and (middle) training time, and (bottom) inference time comparison between PLM and TAGA in terms of the number of hops.

### 3.4.6 Efficiency Analysis

To validate the efficiency and scalability of our proposed full method and random walk algorithm during both training and inference phases, we conduct experiments on the Cora dataset. We vary the number of hops from 0 to 5 and record the number of words in the input corpus, training time, and inference time. The results are presented in Figure 3.5. As depicted in top figure, the exponential growth in input size for the full method compared to the near-linear growth of the random walk method demonstrates the our's superior scalability in managing larger graph neighborhoods. The middle figure further demonstrates the efficiency advantage of the random walk algorithm, as its training time increases linearly with the number of hops, whereas the full method experiences a much steeper increase, becoming infeasible beyond 3 hops due to out-of-memory (OOM) errors. Finally, the bottom figure highlights the speedup

achieved by our proposed method during inference compared to directly using a PLM. The inference time for our method remains linear growth trend across different hops, while the PLM-based approach suffers from rapidly increasing inference time with the hops number.

## 3.5 Conclusions

In this work, we study the problem of representation learning on information networks. We propose to use text description to unify diverse information network domains to construct text-attributed graph structures. We introduce TAGA, a novel self-supervised learning framework designed to address the challenges of unsupervised representation learning on TAGs. TAGA integrates both textual and structural information within TAGs by aligning representations from two complementary views: *Text-of-Graph* and *Graph-of-Text*. To enhance the preservation of structural information in the *Text-of-Graph* view, we propose a natural hierarchical document layout that mirrors the graph's topology. Additionally, we introduce a structure-preserving random walk algorithm to accelerate the training process on large TAGs. Extensive experiments on eight real-world datasets demonstrate TAGA's superior performance in zero-shot and few-shot learning scenarios, showcasing its strong generalization capabilities across diverse domains.

# Chapter 4

# Enhancing Generalizability and Robustness of Learning Network Representations

In this chapter, we introduce our proposed curriculum learning strategy to handle the incompatibility between graph topological structure and other including data modalities. Specifically, we introduce the background in Section 4.1 and related works in Section 4.2. The proposed curriculum learning framework is presented in Section 4.3. Extensive experiments are shown in Section 4.4. The conclusions are presented in Section 4.5.

This chapter includes one published work [232], which was published in *The 37th Conference on Neural Information Processing Systems* as a full research track paper, titled "Curriculum learning for graph neural networks: Which edges should we learn first".

## 4.1 Introduction

In real-world applications, the incompatibility between graph topology and other data modalities often arises due to discrepancies and imperfections in how different types of data represent relationships among entities [107, 111]. While the graph topology captures structural connections through edges, other data modalities from physical and information networks—such as spatial information or textual descriptions—may reflect additional or alternative relationships not encoded in the graph [179]. This misalignment can occur because of data quality issues like noise, missing values, or outdated information, leading to edges that are unreliable or not representative of the true underlying interactions. For example, in a citation network, the co-authorship graph may not fully align with the topical similarities derived from the content of the papers [169]. Such incompatibility poses significant challenges for graph representation learning, as traditional Graph Neural Networks (GNNs) assume that all edges are equally informative and reliable [212]. Studying this incompatibility is crucial for obtaining high quality representations that can intelligently reconcile differences between the graph structure and other data modalities [210]. By addressing these challenges, we can enhance the generalizability and robustness of GNNs, leading to more accurate and meaningful representations in physical and information networks. This, in turn, improves performance on downstream tasks like node classification, link prediction, and community detection, ultimately enabling better insights and decision-making in complex networked systems.

Inspired by cognitive science studies [59, 159] that humans can benefit from the sequence of learning basic (easy) concepts first and advanced (hard) concepts later, curriculum learning (CL) [15] suggests training a machine learning model with easy data samples first and then gradually introducing more hard samples into the model according to a designed pace, where the difficulty of samples can usually be measured by their training loss [120]. Many previous studies have shown that this easy-to-hard

learning strategy can effectively improve the generalization ability of the model [15, 103, 87, 76, 161, 197], and some studies [103, 87, 76] have shown that CL strategies can also increase the robustness of the learned model against noisy training samples. An intuitive explanation is that in CL settings noisy data samples correspond to harder samples, and CL learner spends less time with the harder (noisy) samples to achieve better generalization performance and robustness.

Although CL strategies have achieved great success in many fields such as computer vision and natural language processing, existing methods are designed for independent data (such as images) while designing effective CL methods for data with dependencies has been largely underexplored. For example, in a citation network, two researchers with highly related research topics (e.g. machine learning and data mining) are more likely to collaborate with each other, while the reason behind a collaboration of two researchers with less related research topics (e.g. computer architecture and social science) might be more difficult to understand. Prediction on one sample impacts that of another, forming a graph structure that encompasses all samples connected by their dependencies. There are many machine learning techniques for such graph-structured data, ranging from traditional models like conditional random field [166], graph kernels [180], to modern deep models like GNNs [132, 133, 225, 184, 210, 81, 226, 190]. However, traditional CL strategies are not designed to handle the curriculum of the dependencies between nodes in graph data, which are insufficient. Handling graph-structured data require not only considering the difficulty in individual samples, but also the difficulty of their dependencies to determine how to gradually composite correlated samples for learning.

As previous CL strategies indicated that an easy-to-hard learning sequence on data samples can improve the generalization and robustness performance, an intuitive question is whether a similar strategy on data dependencies that iteratively involves easy-to-hard edges in learning can also benefit. Unfortunately, there exists no trivial

way to directly generalize existing CL strategies on independent data to handle data dependencies due to several unique challenges: (1) **Difficulty in quantifying edge selection criteria**. Existing CL studies on independent data often use supervised computable metrics (e.g. training loss) to quantify sample difficulty, but how to quantify the difficulties of understanding the dependencies between data samples which has no supervision is challenging. (2) **Difficulty in designing an appropriate curriculum to gradually involve edges**. Similar to the human learning process, the model should ideally retain a certain degree of freedom to adjust the pacing of including edges according to its own learning status. As existing CL methods for graph data typically use fixed pacing function to involve samples, they can not provide this flexibility. Designing an adaptive pacing function for handling graph data is difficult since it requires joint optimization of both supervised learning tasks on nodes and the number of chosen edges. (3) **Difficulty in ensuring convergence and a numerical steady process for CL in graphs**. Discrete changes in the number of edges can cause drift in the optimal model parameters between training iterations. How to guarantee a numerically stable learning process for CL on edges is challenging.

In order to address the aforementioned challenges, in this paper, we propose a novel CL algorithm named **R**elational **C**urriculum **L**earning (**RCL**) to improve the generalization ability and robustness of representation learners on data with dependencies. To address the first challenge, we propose an approach to select the edges by quantifying their corresponding difficulties in a self-supervised learning manner. Specifically, for each training iteration, we choose $K$ easiest edges whose corresponding relations are most well-expected by the current model. Second, to design an appropriate learning pace for gradually involving more edges in training, we present the learning process as a concise optimization model, which automatically lets the model gradually increase the number $K$ to involve more edges in training according

to its own status. Third, to ensure convergence of optimizing the model, we propose an alternative optimization algorithm with a theoretical convergence guarantee and an edge reweighting scheme to smooth the graph structure transition. Finally, we demonstrate the superior performance of RCL compared to state-of-the-art methods through extensive experiments on both synthetic and real-world datasets.

## 4.2 Related Works

**Curriculum Learning (CL).** Bengio et al.[15] pioneered the concept of Curriculum Learning (CL) within the machine learning domain, aiming to improve model performance by gradually including easy to hard samples in training the model. Self-paced learning [120] measures the difficulty of samples by their training loss, which addressed the issue in previous works that difficulties of samples are generated by prior heuristic rules. Therefore, the model can adjust the curriculum of samples according to its own training status. Following works [102, 101, 237] further proposed many supervised measurement metrics for determining curriculums, for example, the diversity of samples [101] or the consistency of model predictions [237]. Meanwhile, many empirical and theoretical studies were proposed to explain why CL could lead to generalization improvement from different perspectives. For example, studies such as MentorNet [103] and Co-teaching [87] empirically found that utilizing CL strategy can achieve better generalization performance when the given training data is noisy. [76] provided theoretical explanations on the denoising mechanism that CL learners waste less time with the noisy samples as they are considered harder samples. Some studies [15, 161, 197, 83, 117] also realized that CL can help accelerate the optimization process of non-convex objectives and improve the speed of convergence in the early stages of training.

Despite great success, most of the existing designed CL strategies are for inde-

pendent data such as images, and there is little work on generalizing CL strategies to handle samples with dependencies. Few existing attempts on graph-structured data [124, 109, 128], such as [192, 38, 196, 128], simply treat nodes as independent samples and then apply CL strategies on independent data, which ignore the fundamental and unique dependency information carried by the structure in data, and thus can not well handle the correlation between data samples. Furthermore, these models are mostly based on heuristic-based sample selection strategies [38, 196, 128], which largely limit the generalizability of these methods.

**Graph structure learning.** Another stream of existing studies that are related to our work is *graph structure learning*. Recent studies have shown that GNN models are vulnerable to adversarial attacks on graph structure [45, 203]. In order to address this issue, studies in *graph structure learning* usually aim to jointly learn an optimized graph structure and corresponding graph representations. Existing works [60, 33, 107, 235, 138] typically consider the hypothesis that the intrinsic graph structure should be sparse or low rank from the original input graph by pruning "irrelevant" edges. Thus, they typically use pre-deterministic methods [45, 239, 60] to preprocess graph structure such as singular value decomposition (SVD), or dynamically remove "redundant" edges according to the downstream task performance on the current sparsified structure [33, 107, 138]. However, modifying the graph topology will inevitably lose potentially useful information lying in the removed edges. More importantly, the modified graph structure is usually optimized for maximizing the performance on the training set, which can easily lead to overfitting issues.

## 4.3 Relational Curriculum Learning

### 4.3.1 Preliminaries

Graph neural networks (GNNs) are a class of methods that have shown promising progress in representing structured data in which data samples are correlated with each other. Typically, the data samples are treated as nodes while their dependencies are treated as edges in the constructed graph. Formally, we denote a graph as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is a set of nodes that $N = |\mathcal{V}|$ denotes the number of nodes in the graph and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. We also let $\mathbf{X} \in \mathbb{R}^{N \times b}$ denote the node attribute matrix and let $\mathbf{A} \in \mathbb{R}^{N \times N}$ represent the adjacency matrix. Specifically, $A_{ij} = 1$ denotes that there is an edge connecting nodes $v_i$ and $v_j \in \mathcal{V}$, otherwise $A_{ij} = 0$. A GNN model $f$ maps the node feature matrix $\mathbf{X}$ associated with the adjacency matrix $\mathbf{A}$ to the model predictions $\hat{\mathbf{y}} = f(\mathbf{X}, \mathbf{A})$, and get the loss $L_{\text{GNN}} = L(\hat{\mathbf{y}}, \mathbf{y})$, where $L$ is the objective function and $\mathbf{y}$ is the ground-truth label of nodes. The loss on one node $v_i$ is denoted as $l_i = L(\hat{y}_i, y_i)$.

As previous CL methods have shown that an easy-to-hard learning sequence of independent data samples can improve the generalization ability and robustness of the representation learner, the goal of this paper is to develop an effective CL method on data with dependencies, which is extremely difficult due to several unique challenges: (1) Difficulty in designing a feasible principle to select edges by properly quantifying their difficulties. (2) Difficulty in designing an appropriate pace of curriculum to gradually involve more edges in training based on model status. (3) Difficulty in ensuring convergence and a numerical steady process for optimizing the CL model.

In order to address the above challenges, we propose a novel CL method named **R**elational **C**urriculum **L**earning (**RCL**). The sequence, which gradually includes edges from easy to hard, is called *curriculum* and learned in different grown-up stages of training. In order to address the first challenge, we propose a self-supervised mod-

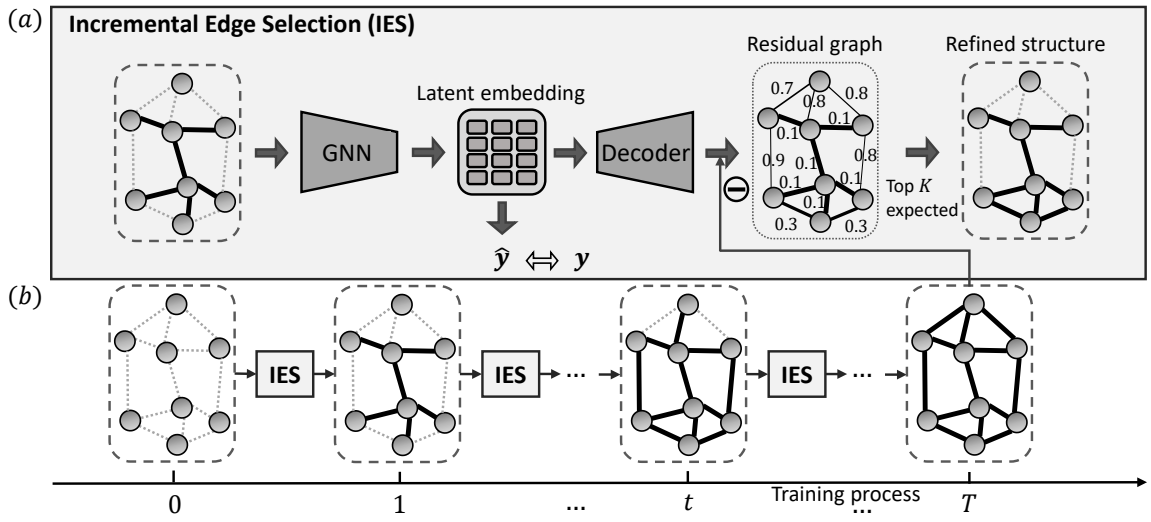


Figure 4.1: The overall framework of RCL. (a) The *Incremental Edge Selection* module first extracts the latent node embedding by the GNN model given the current training structure, then jointly learns the node prediction label $\mathbf{y}$ and reconstructs the input structure by a decoder. A small residual error on an edge indicates the corresponding dependency is well expected and thus can be added to the refined structure for the next iteration. (b) The iterative learning process of RCL. The model starts with an empty structure and gradually includes more edges until the training structure converges to the input structure.

ule *Incremental Edge Selection (IES)*, which is shown in Figure 4.1(a), to select the $K$ easiest edges at each training iteration that are mostly expected by the current model. The details are elaborated in Section 4.3.2. To address the second challenge, we present a joint optimization framework to automatically increase the number of selected edges $K$ given its own training status. The framework is elaborated in Figure 4.1(b) and details can be found in Section 4.3.3. Finally, to ensure convergence of optimization and steady the numerical process, we propose an EM-style alternative optimization algorithm with a theoretical convergence guarantee in Section 4.3.3 Algorithm 1 and an edge reweighting scheme to smooth the discrete edge incrementing process in Section 4.3.4.

### 4.3.2 Incremental Edge Selection by Quantifying Difficulties of Sample Dependencies

Here we propose a novel way to select edges by first quantifying their difficulty levels. Existing works on independent data typically use supervised metrics such as training loss of samples to quantify their difficulty level, but there exists no supervised metrics on edges. To address this issue, we propose a self-supervised module *Incremental Edge Selection (IES)*. We first quantify the difficulty of edges by measuring how well the edges are expected from the currently learned embeddings of their connected nodes. Then the most well-expected edges are selected as the easiest edges for the next iteration of training. As shown in Figure 4.1(a), given the currently selected edges at iteration $t$, we first feed them to the GNN model to extract the latent node embeddings. Then we restore the latent node embeddings to the original graph structure through a decoder, which is called the reconstruction of the original graph structure. The residual graph $\mathbf{R}$, which is defined as the degree of mismatch between the original adjacency matrix $\mathbf{A}$ and the reconstructed adjacency matrix $\tilde{\mathbf{A}}^{(t)}$, can be considered a strong indicator for describing how well the edges are expected by the current model. Specifically, a smaller residual error indicates a higher probability of being a well-expected edge.

With the developed self-supervised method to measure the difficulties of edges, here we formulate the key learning paradigm of selecting the top $K$ easiest edges. To obtain the training adjacency matrix $\mathbf{A}^{(t)}$ that will be fed into the GNN model $f^{(t)}$, we introduce a learnable binary mask matrix $\mathbf{S}$ with each element $\mathbf{S}_{ij} \in \{0, 1\}$. Thus, the training adjacency matrix at iteration $t$ can be represented as $\mathbf{A}^{(t)} = \mathbf{S}^{(t)} \odot \mathbf{A}$. To filter out the edges with $K$ smallest residual error, we penalize the summarized residual errors over the selected edges, which can be represented as $\sum_{i,j} \mathbf{S}_{ij}\mathbf{R}_{ij}$. Therefore, the learning objective can be presented as follows:

$$\begin{aligned} &\min_{\mathbf{w}} L_{\text{GNN}} + \beta \sum_{i,j} \mathbf{S}_{ij}\mathbf{R}_{ij}, \\ &s.t. \left\|\mathbf{S}\right\|_1 \geq K, \end{aligned} \tag{4.1}$$

where the first term $L_{\text{GNN}} = L(f(\mathbf{X}, \mathbf{A}^{(t)}; \mathbf{w}), \mathbf{y})$ is the node-level predictive loss, e.g. cross-entropy loss for the node classification task. The second term $\sum_{i,j} \mathbf{S}_{ij}\mathbf{R}_{ij}$ aims at penalizing the residual errors over the edges selected by the mask matrix $\mathbf{S}$. $\beta$ is a hyperparameter to tune the balance between terms. The constraint is to guarantee only the most $K$ well-expected edges are selected.

More concretely, the value of a residual edge $\tilde{\mathbf{A}}_{ij}^{(t)} \in [0, 1]$ can be computed by a non-parametric kernel function $\kappa(\mathbf{z}_i^{(t)}, \mathbf{z}_j^{(t)})$, e.g. the inner product kernel. Then the residual error $\mathbf{R}_{ij}$ between the input structure and the reconstructed structure can be defined as $\left\|\tilde{\mathbf{A}}_{ij}^{(t)} - \mathbf{A}_{ij}\right\|$, where $\|\cdot\|$ is commonly chosen to be the squared $\ell_2$-norm.

### 4.3.3 Automatically Control the Pace of Increasing Edges

In order to dynamically include more edges into training, an intuitive way is to iteratively increase the value of $K$ in Equation 4.1 to allow more edges to be selected. However, it is difficult to determine an appropriate value of $K$ with respect to the training status of the model. Besides, directly solving Equation 4.1 is difficult since $\mathbf{S}$ is a binary matrix where each element $\mathbf{S}_{ij} \in \{0, 1\}$, optimizing $\mathbf{S}$ would require solving a discrete constraint program at each iteration. To address this issue, we first relax the problem into continuous optimization so that each $\mathbf{S}_{ij}$ can be allowed to take any value in the interval $[0, 1]$. Note that the inequality $||\mathbf{S}||_1 \geq K$ in Eqn. 4.1 is equivalent to the equality $||\mathbf{S}||_1 = K$. This is because the second term in the loss function would always encourage fewer selected edges by the mask matrix $\mathbf{S}$, as all values in the residual error matrix $\mathbf{R}$ and mask matrix $\mathbf{S}$ are nonnegative. Given

**Algorithm 3** Alternating Minimization Algorithm for Equation 4.2

**Input** Node features $\mathbf{X}$, adjacency matrix $\mathbf{A}$, a stepsize $\mu$ and hyperparameter $\gamma$
**Output** The model parameter $\mathbf{w}$ of GNN model $f$.

Initialize $\mathbf{w}^{(0)}$, $\mathbf{S}^{(0)}$, $\lambda$
**while** Not Converged **do**
  $\mathbf{w}^{(t)} = \arg\min_{\mathbf{w}} L(f(\mathbf{X}, \mathbf{A}^{(t-1)}; \mathbf{w}), \mathbf{y}) + \beta \sum_{i,j} \mathbf{S}_{ij} \left\| \tilde{\mathbf{A}}_{ij}^{(t-1)} - \mathbf{A}_{ij} \right\| + \frac{\gamma}{2} \left\| \mathbf{w} - \mathbf{w}^{(t-1)} \right\|$
  Given $\mathbf{w}^{(t)}$, extract latent nodes embedding $\mathbf{Z}^{(t)}$ from GNN model $f$
  Calculate reconstructed structure $\tilde{\mathbf{A}}_{ij}^{(t)} = \kappa(\mathbf{z}_i^{(t)}, \mathbf{z}_j^{(t)})$ for all pairs of $i, j$
  $\mathbf{S}^{(t)} = \arg\min_{\mathbf{S}} \beta \sum_{i,j} \mathbf{S}_{ij} \left\| \mathbf{A}_{ij} - \tilde{\mathbf{A}}_{ij}^{(t)} \right\| + g(\mathbf{S}; \lambda) + \frac{\gamma}{2} \left\| \mathbf{S} - \mathbf{S}^{(t-1)} \right\|$
  Compute $\mathbf{A}^{(t)} = \mathbf{S}^{(t)} \odot \mathbf{A}$
  **if** $\mathbf{A}^{(t)} \neq \mathbf{A}$ **then** increase $\lambda$ by stepsize $\mu$
**end while**
**return w**

this, we can incorporate the equality constraint as a Lagrange multiplier and rewrite the loss function as $\mathcal{L} = L_{GNN} + \beta \sum_{i,j} \mathbf{S}_{ij} \mathbf{R}_{ij} - \lambda(||\mathbf{S}||_1 - K)$. Considering that $K$ remains constant, the optimization of the loss function can be equivalently framed by substituting the given constraint with a regularization term denoted as $g(\mathbf{S}; \lambda)$. As such, the overall loss function can be reformulated as:

$$\min_{\mathbf{w}, \mathbf{S}} L_{\text{GNN}} + \beta \sum_{i,j} \mathbf{S}_{ij} \mathbf{R}_{ij} + g(\mathbf{S}; \lambda), \tag{4.2}$$

where $g(\mathbf{S}; \lambda) = \lambda \left\| \mathbf{S} - \mathbf{A} \right\|$ and $\|\cdot\|$ is commonly chosen to be the squared $\ell_2$-norm. Since the training adjacency matrix $\mathbf{A}^{(t)} = \mathbf{S}^{(t)} \odot \mathbf{A}$, as $\lambda \to \infty$, more edges in the input structure are included until the training adjacency matrix $\mathbf{A}^{(t)}$ converges to the input adjacency matrix $\mathbf{A}$. Specifically, the regularization term $g(\mathbf{S}; \lambda)$ controls the learning scheme by the *age parameter* $\lambda$, where $\lambda = \lambda(t)$ grows with the number of iterations. By monotonously increasing the value of $\lambda$, the regularization term $g(\mathbf{S}; \lambda)$ will push the mask matrix gradually converge to the input adjacency matrix $\mathbf{A}$, resulting in more edges automatically involved in the training structure.

**Optimization of learning objective.** In optimizing the objective function

in Equation 4.2, we need to jointly optimize parameter $\mathbf{w}$ for GNN model $f$ and the mask matrix $\mathbf{S}$. To tackle this, we introduce an EM-style optimization scheme (detailed in Algorithm 3) that iteratively updates both. The algorithm uses the node feature matrix $\mathbf{X}$, the original adjacency matrix $\mathbf{A}$, a step size $\mu$ to control the age parameter $\lambda$ increase rate, and a hyperparameter $\gamma$ for regularization adjustments. Post initialization of $\mathbf{w}$ and $\mathbf{S}$, it alternates between: optimizing GNN model $f$ (Step 3), extracting latent node embeddings and reconstructing the adjacency matrix (Steps 4 & 5), refining the mask matrix using the reconstructed matrix and regularization, and results in more edges are gradually involved (Step 6), updating the training adjacency matrix (Step 7), and incrementing $\lambda$ when the training matrix $\mathbf{A}^{(t)}$ differs from input matrix $\mathbf{A}$, incorporating more edges in the next iteration.

**Theorem 6.** We have the following convergence guarantees for Algorithm 3:

• **Avoidance of Saddle Points.** If the second derivatives of $L(f(\mathbf{X}, \mathbf{A}^{(t)}; \mathbf{w}), \mathbf{y})$ and $g(\mathbf{S}; \lambda)$ are continuous, then for sufficiently large $\gamma$, any bounded sequence $(\mathbf{w}^{(t)}, \mathbf{S}^{(t)})$ generated by Algorithm 3 with random initializations will not converge to a strict saddle point of $F$ almost surely.

• **Second Order Convergence.** If the second derivatives of $L(f(\mathbf{X}, \mathbf{A}^{(t)}; \mathbf{w}), \mathbf{y})$ and $g(\mathbf{S}; \lambda)$ are continuous, and $L(f(\mathbf{X}, \mathbf{A}^{(t)}; \mathbf{w}), \mathbf{y})$ and $g(\mathbf{S}; \lambda)$ satisfy the Kurdyka-Łojasiewicz (KL) property [183], then for sufficiently large $\gamma$, any bounded sequence $(\mathbf{w}^{(t)}, \mathbf{S}^{(t)})$ generated by Algorithm 3 with random initialization will almost surely converge to a second-order stationary point of $F$.

The proof of this theorem can be found in Appendix C.1.

### 4.3.4 Smooth Structure Transition by Edge Reweighting

Note that in the Algorithm 1, the optimization process requires iteratively updating the parameters $\mathbf{w}$ of the GNN model $f$ and current adjacency matrix $\mathbf{A}^{(t)}$, where

$\mathbf{A}^{(t)}$ varies discretely between iterations. However, GNN models mostly work in a message-passing fashion, which computes node representations by iteratively aggregating information along edges from neighboring nodes. Discretely modifying the number of edges will result in a great drift of the optimal model parameters between iterations. In Appendix Figure , we demonstrate that a shift in the optimal parameters of the GNN results in a spike in the training loss. Therefore, it can increase the difficulty of finding optimal parameters and even hurt the generalization ability of the model in some cases. Besides the numerical problem caused by discretely increasing the number of edges, another issue raised by the CL strategy in Section 4.3.2 is the trustworthiness of the estimated edge difficulty, which is inferred by the residual error on the edges. Although the residual error can reflect how well edges are expected in the ideal case, the quality of the learned latent node embeddings may affect the validity of this metric and compromise the quality of the designed curriculum by the CL strategy.

To address both issues, we propose a novel edge reweighting scheme to (1) smooth the transition of the training structure between iterations, and (2) reduce the weight of edges that connect nodes with low-confidence latent embeddings. Formally, we use a smoothed version of structure $\bar{\mathbf{A}}^{(t)}$ to substitute $\mathbf{A}^{(t)}$ for training the GNN model $f$ in step 3 of Algorithm 3, where the mapping from $\mathbf{A}^{(t)}$ to $\bar{\mathbf{A}}^{(t)}$ can be represented as:

$$\bar{\mathbf{A}}_{ij}^{(t)} = \pi_{ij}^{(t)} \mathbf{A}_{ij}^{(t)}, \tag{4.3}$$

where $\pi_{ij}^{(t)}$ is the weight imposed on edge $e_{ij}$ at iteration $t$. $\pi_{ij}^{(t)}$ is calculated by considering the counted occurrences of edge $e_{ij}$ until the iteration $t$ and the confidence of the latent embedding for the connected pair of nodes $v_i$ and $v_j$:

$$\pi_{ij}^{(t)} = \psi(e_{ij})\rho(v_i)\rho(v_j), \tag{4.4}$$

where $\psi$ is a function that reflects the number of edge occurrences and $\rho$ is a function to reflect the degree of confidence for the learned latent node embedding. The details of these two functions are described as follow.

**Smooth the transition of the training structure between iterations.** In order to obtain a smooth transition of the training structure between iterations, we take the learned curriculum of selected edges into consideration. Formally, we model $\psi$ by a smooth function of the edge selected occurrences compared to the model iteration occurrences before the current iteration:

$$\psi(e_{ij}) = t(e_{ij})/t, \tag{4.5}$$

where $t$ is the number of current iterations and $t(e_{ij})$ represents the counting number of selecting edge $e_{ij}$. Therefore, we transform the original discretely changing training structure into a smoothly changing one by taking the historical edge selection curriculum into consideration.

**Reduce the influence of nodes with low confidence latent embeddings.** As introduced in our Algorithm 1 line 6, the estimated structure $\tilde{A}$ is inferred from the latent embedding $\mathbf{Z}$, which is extracted from the trained GNN model $f$. Such estimated latent embedding may possibly differ from the true underlying embedding, which results in the inaccurately reconstructed structure around the node. In order to alleviate this issue, we model the function $\rho$ by the training loss on nodes, which indicates the confidence of their learned latent embeddings. This idea is similar to previous CL strategies on inferring the difficulty of data samples by their supervised training loss. Specifically, a larger training loss indicates a low confident latent node embedding. Mathematically, the weights $\rho(v_i)$ on node $v_i$ can be represented as a distribution of their training loss:

$$\rho(v_i) \sim e^{-l_i} \tag{4.6}$$

where $l_i$ is the training loss on node $v_i$. Therefore, a node with a larger training loss will result in a smaller value of $\rho(v_i)$, which reduces the weight of its connecting edges.

## 4.4 Experimental Results of RCL

In this section, the experimental settings are introduced first in Section 4.4.1, then the performance of the proposed method on both synthetic and real-world datasets are presented in Section 4.4.2. We further present the robustness test on our CL method against topological structure noise in Section 4.4.3. We verify the effectiveness of framework components through ablation studies in Section 4.4.4. Intuitive visualizations of the edge selection curriculum are shown in Section 4.4.5. In addition, we measure the parameter sensitivity in Section 4.4.6 and running time analysis in Section 4.4.7.

### 4.4.1 Experimental Settings

**Synthetic datasets.** To evaluate the effectiveness of our proposed method on datasets with ground-truth difficulty labels on structure, we first follow previous studies [110, 1] to generate a set of synthetic datasets, where the difficulty of edges in generated graphs are indicated by their formation probability. Specifically, as shown in Figure 4.2, each generated graph is with 5,000 nodes, which are divided into 10 equally sized node classes $1, 2, \ldots, 10$. The node features are sampled from overlapping multi-Gaussian distributions. Each generated graph is associated with a *homophily coefficient (homo)* which indicates the likelihood of a node forming a connection to another node with the same label (same color in Figure 4.2). For example, a generated graph with $homo = 0.5$ will have on average half of the edges formed between nodes with the same label. For the rest edges that are formed between nodes with different labels (different colors in Figure 4.2), the probability of forming an edge

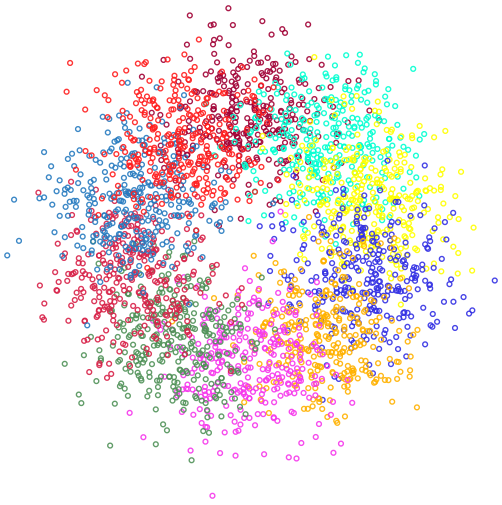

Figure 4.2: Visualization of synthetic datasets. Each color represents a class of nodes. Node attributes are sampled from overlapping multi-Gaussian distributions, where the attributes of nodes with close labels are likely to have short distances. Homogeneous edges represent edges that connect nodes of the same class (with the same color). The probability of connecting two nodes of different classes decreases with the distance between the center points of their class distribution. Therefore, the formation probability of a node denotes the edge difficulty, since edges between nodes with close classes are more likely to positively contribute to the prediction under the homogeneous assumption.

is inversely proportional to the distances between their labels. Mathematically, the probability of forming an edge between node $u$ and node $v$ follows $p_{u\to v} \propto e^{-|c_u - c_v|}$, where the distances between labels $|c_u - c_v|$ means shortest distance of two classes on a circle. Therefore, the probability of forming an edge in the synthetic graph can reflect how well this edge is expected. Specifically, edges with a higher formation probability, e.g. connecting nodes with the same label or close labels, meaning that there is a higher chance that this connection will positively contribute to the prediction (less chance to be a noisy edge). Conversely, edges with a lower formation probability, e.g., connecting nodes with faraway labels, mean that there is a higher chance that this connection will negatively contribute to the prediction (higher chance to be a noisy edge). We vary the value of *homo* from $0.1, 0.2, \ldots, 0.9$ to generate nine graphs in

total. Similar to previous works [110, 1], we randomly partition each synthetic graph into equal-sized train, validation, and test node splits.

**Real-world datasets.** To further evaluate the performance of our proposed method in real-world scenarios, nine benchmark real-world attributed network datasets, including four citation network datasets Cora, Citeseer, Pubmed [215] and ogbn-arxiv [94], two coauthor network datasets CS and Physics [125], two Amazon co-purchase network datasets Photo and Computers [125], and one protein interation network ogbn-proteins [94]. We follow the data splits from [31] on citation networks and use a 5-fold cross-validation setting on coauthor and Amazon co-purchase networks. All datasets are publicly available from Pytorch-geometric library [66] and Open Graph Benchmark (OGB) [94], where basic statistics are reported in Table 4.2.

**Comparison methods.** We incorporate three commonly used GNN models, including GCN [116], GraphSAGE [85], and GIN [211], as the baseline model and also the backbone model for RCL. In addition to evaluating our proposed method against the baseline GNNs, we further leverage two categories of state-of-the-art comparison methods in the experiments: (1) We incorporate four graph structure learning methods GNNSVD [60], ProGNN [107], NeuralSparse [235], and PTDNet [138]; (2) We further compare with a curriculum learning method named CLNode [196] which gradually select nodes in the order of the difficulties defined by a heuristic-based strategy. The following describes the details of our comparison models.

**Graph Neural Networks (GNNs).** We first introduce three baseline GNN models as follows.

**(i) GCN.** Graph Convolutional Networks (GCN) [116] is a commonly used GNN, which introduces a first-order approximation architecture of the Chebyshev spectral convolution operator;

**(ii) GIN.** Graph Isomorphism Networks (GIN) [211] is a variant of GNN, which

has provably powerful discriminating power among the class of 1-order GNNs;

**(iii) GraphSage.** GraphSage [85] is a GNN method that computes the hidden representation of the root node by aggregating the hidden node representations hierarchically from bottom to top.

**Graph structure learning.** We then introduce four state-of-the-art methods for jointly learning the optimal graph structure and downstream tasks.

**(i) GNNSVD.** GNNSVD [60] first apply singular value decomposition (SVD) on the graph adjacency matrix to obtain a low-rank graph structure and apply GNN on the obtained low-rank structure;

**(ii) ProGNN.** ProGNN [107] is a method to defend against graph adversarial attacks by obtaining a sparse and low-rank graph structure from the input structure;

**(iii) NeuralSparse.** NeuralSparse [235] is a method to learn robust graph representations by iteratively sampling $k$-neighbor subgraphs for each node and sparsing the graph according to the performance on the node classification;

**(iv) PTDNet.** PTDNet [138] learns a sparsified graph by pruning task-irrelevant edges, where sparsity is controlled by regulating the number of edges.

**Curriculum learning on graph data.** We introduce a recent curriculum learning work on node classification as follows.

**(i) CLNode.** CLNode [196] regards nodes as data samples and gradually incorporates more nodes into training according to their difficulty. They apply a heuristic-based strategy to measure the difficulty of nodes, where the nodes that connect neighboring nodes with different classes are considered difficult.

**Initializing graph structure by a pre-trained model.** It is worth noting that the model needs an initial training graph structure $\mathbf{A}^{(0)}$ in the initial stage of training. An intuitive way is that we can initialize the model to work in a purely data-driven scenario that starts only with isolated nodes where no edges exist. However, an

instructive initial structure can greatly reduce the search cost and computational burden. Inspired by many previous CL works [197, 83, 103, 237] that incorporate prior knowledge of a pre-trained model into designing curriculum for the current model, we initialize the training structure $\mathbf{A}^{(0)}$ by a pre-trained vanilla GNN model $f^*$. Specifically, we follow the same steps from line 4 to line 7 in the algorithm 1 to obtain the initial training structure $\mathbf{A}^{(0)}$ but the latent node embedding is extracted from the pre-trained model $f^*$.

**Implementation Details.** We use the baseline model (GCN, GIN, GraphSage) as the backbone model for both our RCL method and all comparison methods. For a fair comparison, we require all models follow the same GNN architecture with two convolution layers. For each split, we run each model 10 times to reduce the variance in particular data splits. Test results are according to the best validation results. General training hyperparameters (such as learning rate or the number of training epochs) are equal for all models. For the pre-trained model to initialize the training structure, we utilize the same model as the backbone model utilized by our method. For example, if we use GCN as the backbone model for RCL, the pre-trained model to initialize is also GCN. All experiments are conducted on a 64-bit machine with four NVIDIA Quadro RTX 8000 GPUs. The proposed method is implemented with Pytorch deep learning framework [155].

### 4.4.2 Effectiveness Results

Table 4.1 presents the node classification results of the synthetic datasets. We report the average accuracy and standard deviation for each model against the *homo* of generated graphs. From the table, we observe that our proposed method RCL consistently achieves the best or most competitive performance to all the comparison methods over three backbone GNN architectures. Specifically, RCL outperforms

| Homo ratio | **0.1** | **0.2** | **0.3** | **0.4** | **0.5** | **0.6** | **0.7** | **0.8** | **0.9** |
|---|---|---|---|---|---|---|---|---|---|
| GCN | 50.84±1.03 | 56.50±0.50 | 65.17±0.48 | 77.94±0.54 | 87.15±0.44 | 93.27±0.24 | 97.48±0.25 | 99.10±0.17 | 99.93±0.03 |
| GNNSVD | 54.96±0.76 | 58.45±0.56 | 63.06±0.63 | 70.23±0.61 | 80.51±0.41 | 85.02±0.46 | 90.31±0.27 | 94.23±0.22 | 96.74±0.23 |
| ProGNN | 47.87±0.87 | 54.59±0.55 | 65.39±0.44 | 76.96±0.49 | 87.76±0.51 | 93.16±0.34 | 97.60±0.31 | 99.04±0.19 | **99.94±0.03** |
| NeuralSparse | 51.42±1.35 | 57.99±0.69 | 65.10±0.43 | 75.37±0.34 | 87.40±0.29 | 93.54±0.28 | 97.16±0.15 | 99.01±0.22 | 99.83±0.07 |
| PTDNet | 48.21±1.98 | 55.52±2.82 | 65.82±0.94 | 79.37±0.45 | 89.17±0.39 | 94.19±0.18 | 98.61±0.12 | 99.51±0.09 | 99.81±0.05 |
| CLNodes | 50.37±0.73 | 56.64±0.56 | 65.04±0.66 | 77.52±0.48 | 86.85±0.44 | 93.10±0.47 | 97.34±0.25 | 99.02±0.18 | 99.88±0.04 |
| RCL | **57.57±0.43** | **62.06±0.28** | **73.98±0.55** | **84.54±0.75** | **92.69±0.09** | **97.42±0.17** | **99.62±0.05** | **99.89±0.02** | 99.93±0.06 |
| GIN | 48.33±1.89 | 53.62±1.39 | 64.08±0.99 | 77.55±1.10 | 85.31±0.75 | 90.57±0.36 | 97.82±0.18 | 99.59±0.11 | 99.91±0.02 |
| GNNSVD | 43.21±1.60 | 45.68±1.66 | 54.90±1.16 | 68.29±0.79 | 79.76±0.52 | 85.63±0.44 | 93.65±0.39 | 97.22±0.17 | 98.94±0.17 |
| ProGNN | 45.76±1.40 | 52.96±1.01 | 64.12±1.07 | 76.95±0.87 | 85.13±0.71 | 89.96±0.55 | 96.54±0.48 | 99.51±0.12 | 99.78±0.05 |
| NeuralSparse | 50.23±2.05 | 54.12±1.52 | 62.81±0.75 | 76.98±1.17 | 85.14±0.94 | 92.57±0.44 | 98.02±0.20 | 99.61±0.12 | 99.91±0.05 |
| PTDNet | 53.23±2.76 | 56.12±2.03 | 65.81±1.38 | 77.81±1.02 | 86.14±0.65 | 93.21±0.74 | 97.08±0.41 | 99.51±0.18 | 99.91±0.03 |
| CLNodes | 45.36±1.42 | 51.10±1.15 | 62.53±0.88 | 75.83±1.07 | 87.76±0.90 | 94.25±0.44 | **98.30±0.26** | **99.60±0.09** | **99.92±0.03** |
| RCL | **57.63±0.66** | **62.08±1.17** | **71.02±0.61** | **80.61±0.69** | **88.62±0.43** | **94.88±0.36** | 98.19±0.19 | 99.32±0.08 | 99.89±0.04 |
| GraphSAGE | 62.57±0.55 | 67.33±0.64 | 71.06±0.74 | 80.88±0.54 | 85.88±0.51 | 91.42±0.37 | 95.26±0.33 | 97.78±0.16 | 99.52±0.13 |
| GNNSVD | 64.42±0.80 | 65.71±0.39 | 67.12±0.58 | 68.47±0.50 | 77.70±0.65 | 82.86±0.50 | 87.81±0.71 | 91.61±0.55 | 95.01±0.50 |
| ProGNN | 58.57±2.09 | 66.75±0.91 | 72.14±0.64 | 81.27±0.44 | 86.89±0.47 | 92.10±0.39 | 95.21±0.30 | 97.51±0.23 | 99.50±0.11 |
| NeuralSparse | 61.70±0.77 | 66.65±0.66 | 70.60±0.79 | 79.65±0.45 | 84.19±0.91 | 91.31±0.54 | 94.86±0.53 | 97.16±0.23 | 99.55±0.19 |
| PTDNet | 65.72±1.08 | 69.25±0.92 | 72.60±0.77 | 79.65±0.45 | 86.54±0.56 | 91.79±0.53 | 96.10±0.58 | 97.98±0.13 | **99.78±0.08** |
| CLNodes | **69.41±0.66** | 70.83±0.58 | 75.51±0.36 | 82.65±0.43 | 87.08±0.56 | 91.58±0.41 | 95.91±0.38 | 98.33±0.26 | 99.57±0.14 |
| RCL | 68.03±0.37 | **71.39±0.51** | **76.99±0.99** | **83.76±0.55** | **88.24±0.30** | **93.34±0.56** | **97.66±0.52** | **98.86±0.28** | 99.64±0.08 |

Table 4.1: Node classification accuracy on synthetic datasets (%). The best-performing method on each backbone GNN model is highlighted in bold, while the second-best method is underlined. In situations where RCL's performance is not strictly the best among all methods, we can see that almost all methods can achieve a near-perfect performance and RCL is still close to the best methods.

the second best method on average by 4.17%, 2.60%, and 1.06% on GCN, GIN, and GraphSAGE backbones, respectively. More importantly, the proposed RCL method performs significantly better than the second best model when the *homo* of generated graphs is low ($\leq 0.5$), on average by 6.55% on GCN, 4.17% on GIN, and 2.93% on GraphSAGE backbones. These demonstrate that our proposed RCL method significantly improves the model's capability of learning an effective representation to downstream tasks especially when the edge difficulties vary largely in the data.

We report the experimental results of the real-world datasets in Table 4.2. The results demonstrate the strength of our proposed method by consistently achieving the best results in all 9 datasets by GCN backbone architecture, all 9 datasets by GraphSAGE backbone architecture, and 8 out of 9 datasets by GIN backbone architecture. Specifically, our proposed method improved the performance of baseline models on average by 1.86%, 2.83%, and 1.62% over GCN, GIN, and GraphSAGE, and outperformed the second best models model on average by 1.37%, 2.49%, and 1.22% over the three backbone models, respectively. The results demonstrate that

| | **Cora** | **Citeseer** | **Pubmed** | **CS** | **Physics** | **Photo** | **Computers** | **ogbn-arxiv** | **ogbn-proteins** |
|---|---|---|---|---|---|---|---|---|---|
| # nodes | 2,708 | 3,327 | 19,717 | 18,333 | 34,493 | 7,650 | 13,752 | 169,343 | 132,534 |
| # edges | 10,556 | 9,104 | 88,648 | 163,788 | 495,924 | 238,162 | 491,722 | 1,166,243 | 39,561,252 |
| # features | 1,433 | 3,703 | 500 | 6,805 | 8,415 | 745 | 767 | 100 | 8 |
| GCN | 85.74±0.42 | 78.93±0.32 | 87.91±0.09 | 93.03±0.32 | 96.55±0.15 | 93.25±0.70 | 88.09±0.40 | 71.74±0.29 | 72.51±0.35 |
| GNNSVD | 83.24±1.03 | 74.80±0.87 | 88.81±0.38 | 93.79±0.11 | 96.11±0.13 | 89.63±0.73 | 86.49±0.77 | 67.44±0.51 | 66.92±0.64 |
| ProGNN | 85.66±0.61 | 74.78±0.55 | 87.22±0.33 | 94.04±0.19 | 96.75±0.26 | 92.07±0.67 | 88.72±0.59 | (OOM) | (OOM) |
| NeuralSparse | 85.95±0.98 | 76.24±0.48 | 86.83±0.40 | 92.31±0.47 | 95.56±0.30 | 90.57±0.90 | 88.62±0.83 | (OOM) | (OOM) |
| PTDNet | 83.84±0.95 | 77.54±0.42 | 87.89±0.08 | 93.60±0.43 | 96.56±0.09 | 88.92±0.87 | 87.52±0.70 | (OOM) | (OOM) |
| CLNode | 85.67±0.33 | 78.99±0.57 | 89.50±0.28 | 93.83±0.24 | 95.76±0.16 | 93.39±0.83 | 89.28±0.38 | 70.95±0.18 | 71.40±0.32 |
| RCL | **87.15±0.44** | **79.79±0.55** | **89.79±0.12** | **94.66±0.32** | **97.02±0.23** | **94.41±0.76** | **90.23±0.23** | **74.08±0.33** | **75.19±0.26** |
| GIN | 84.43±0.65 | 74.87±0.20 | 85.72±0.40 | 91.48±0.36 | 95.62±0.30 | 93.02±0.91 | 86.94±1.58 | 69.26±0.34 | 74.51±0.32 |
| GNNSVD | 82.23±0.65 | 72.11±0.70 | 88.31±0.15 | 91.40±0.87 | 95.30±0.29 | 89.49±1.11 | 82.66±2.26 | 67.79±0.41 | 70.65±0.53 |
| ProGNN | 85.02±0.41 | **78.12±0.93** | 87.82±0.51 | (OOM) | (OOM) | 92.23±0.67 | 83.54±1.48 | (OOM) | (OOM) |
| NeuralSparse | 84.92±0.58 | 75.44±0.87 | 86.11±0.49 | 89.66±0.82 | 95.05±0.57 | 93.28±0.83 | 87.22±0.54 | (OOM) | (OOM) |
| PTDNet | 83.02±1.01 | 75.00±0.74 | 88.04±0.29 | 91.01±0.21 | 95.57±0.40 | 90.70±0.76 | 87.08±0.65 | (OOM) | (OOM) |
| CLNode | 83.52±0.77 | 75.82±0.58 | 86.92±0.61 | 91.71±0.41 | 95.75±0.46 | 92.78±0.90 | 85.93±1.53 | 70.58±0.17 | 73.97±0.31 |
| RCL | **86.64±0.39** | 77.60±0.18 | **89.17±0.29** | **93.92±0.27** | **96.75±0.17** | **93.88±0.51** | **89.76±0.19** | **72.55±0.15** | **78.76±0.22** |
| GraphSAGE | 86.22±0.27 | 77.27±0.23 | 88.50±0.16 | 94.22±0.18 | 96.26±0.34 | 93.82±0.51 | 88.62±0.21 | 71.49±0.27 | 77.68±0.20 |
| GNNSVD | 83.11±0.82 | 73.19±0.49 | 88.42±0.38 | 93.86±0.36 | 95.96±0.12 | 89.31±0.53 | 81.46±1.15 | 69.82±0.34 | 71.82±0.39 |
| ProGNN | 86.23±0.42 | 74.45±0.83 | 88.52±0.45 | (OOM) | (OOM) | 90.89±0.69 | 89.34±0.54 | (OOM) | (OOM) |
| NeuralSparse | 84.60±0.52 | 76.32±0.55 | 89.02±0.39 | 93.89±0.58 | 96.67±0.20 | 90.78±1.06 | 88.37±0.37 | (OOM) | (OOM) |
| PTDNet | 86.03±0.60 | 76.07±0.58 | 86.78±0.45 | 93.78±0.43 | 95.32±0.31 | 92.96±0.87 | 84.89±1.47 | (OOM) | (OOM) |
| CLNode | 86.60±0.64 | 77.23±0.54 | 88.76±0.57 | 94.13±0.34 | 96.87±0.45 | 93.90±0.42 | 89.57±0.62 | 71.54±0.20 | 78.40±0.41 |
| RCL | **86.90±0.39** | **78.95±0.18** | **90.14±0.43** | **95.05±0.23** | **96.88±0.19** | **95.06±0.52** | **90.47±0.38** | **73.13±0.14** | **79.89±0.35** |

Table 4.2: Node classification results on real-world datasets (%). The best-performing method on each backbone is highlighted in bold and second-best is underlined. (OOM) shorts for out-of-memory.

the proposed RCL method consistently improves the performance of GNN models in real-world scenarios.

Our experimental results are statically sound. In 43 out of 48 tasks our method outperforms the second-best performing model with strong statistical significance. Specifically, we have in 30 out of 43 cases with a significance $p < 0.001$, in 8 out of 43 cases with a significance $p < 0.01$, and in 5 out of 43 cases with a significance $p < 0.05$. Such statistical significance results can demonstrate that our proposed method can consistently perform better than the baseline models in both scenarios.

### 4.4.3 Robustness Analysis Against Topological Noise

To further examine the robustness of the RCL method on extracting powerful representation from correlated data samples, we follow previous works [107, 138] to randomly inject fake edges into real-world graphs. This adversarial attack can be viewed as adding random noise to the topological structure of graphs. Specifically, we randomly connect $M$ pairs of previously unlinked nodes in the real-world datasets, where

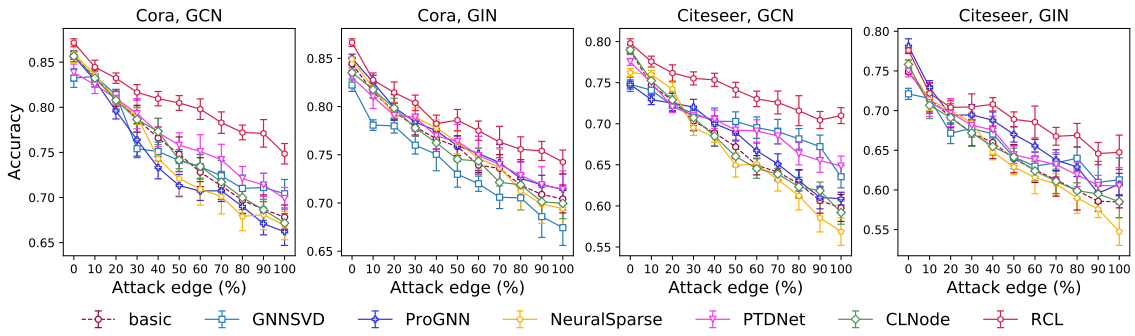


Figure 4.3: Node classification accuracy (%) on Cora and Citeseer under random structure attack. The attack edge ratio is computed versus the original number of edges, where 100% means that the number of inserted edges is equal to the number of original edges.

the value of $M$ varies from 10% to 100% of the original edges. We then train RCL and all the comparison methods on the attacked graph and evaluate the node classification performance. The results are shown in Figure 4.3, we can observe that RCL shows strong robustness to adversarial structural attacks by consistently outperforming all compared methods on all datasets. Especially, when the proportion of added noisy edges is large ($> 50\%$), the improvement becomes more significant. For instance, under the extremely noisy ratio at 100%, RCL outperforms the second best model by 4.43% and 2.83% on Cora dataset, and by 6.13%, 3.47% on Citeseer dataset, with GCN and GIN backbone models, respectively.

To investigate the effectiveness of our proposed model with some simpler heuristics, we deploy a series of abalation analysis. We first train the model with node classification task purely and select the top K expected edges as suggested by the reviewer. Specifically, we follow previous works [191, 196] using two classical selection pacing functions as follows:

$$\text{Linear}: K_{\text{linear}}(t) = \frac{t}{T}|E|; \quad \text{Root}: K_{\text{root}}(t) = \sqrt{\frac{t}{T}}|E|,$$

where $t$ is the number of current iterations and $T$ is the number of total iterations,

| | **Synthetic1** | **Synthetic2** | **Citeseer** | **CS** | **Computers** |
|---|---|---|---|---|---|
| Full | **73.98±0.55** | **97.42±0.17** | **79.79±0.55** | **94.66±0.22** | **90.23±0.23** |
| Curriculum-linear | 70.93±0.54 | 95.19±0.19 | 79.04±0.38 | 94.14±0.26 | 89.28±0.21 |
| Curriculum-root | 70.13±0.72 | 95.50±0.18 | 78.27±0.54 | 94.47±0.34 | 89.27±0.15 |
| Random-linear | 58.76±0.46 | 89.78±0.11 | 77.43±0.49 | 92.76±0.14 | 88.76±0.18 |
| Random-root | 61.04±0.20 | 91.04±0.09 | 76.81±0.35 | 92.92±0.15 | 88.81±0.28 |
| w/o edge appearance | 70.70±0.43 | 95.77±0.16 | 77.77±0.65 | 94.39±0.21 | 89.56±0.30 |
| w/o node confidence | 72.38±0.41 | 96.86±0.17 | 78.72±0.72 | 94.34±0.13 | 90.03±0.62 |
| w/o pre-trained model | 72.56±0.69 | 93.89±0.14 | 78.28±0.77 | 94.50±0.14 | 89.80±0.55 |

Table 4.3: Ablation study. Here "Full" represents the original method without removing any component. The best-performing method on each dataset is highlighted in bold.

and $|E|$ is the number of total edges. We name these two variants Curriculum-linear and Curriculum-root, respectively. In addition, we also remove the edge difficulty measurement module and use random selection instead. Specifically, we gradually incorporate more edges into training in random order to verify the effectiveness of the learned curriculum. We name two variants as Random-linear and Random-root with the above two mentioned pacing functions, respectively.

In order to further investigate the impact of the proposed components of RCL. We also first consider variants of removing the edge smoothing components mentioned in Section 4.3.4. Specifically, we consider two variants *w/o EC* and *w/o NC*, which remove the smoothing function of the edge occurrence ratio and the component to reflect the degree of confidence for the latent node embedding in RCL, respectively. In addition to examining the effectiveness of edge smoothing components, we further consider a variant *w/o pre-trained model* that avoids using a pre-trained model to initialize model, which is mentioned in Section 4.4.1, to initialize the training structure by a pre-trained model and instead starts with inferred structure from isolated nodes with no connections.

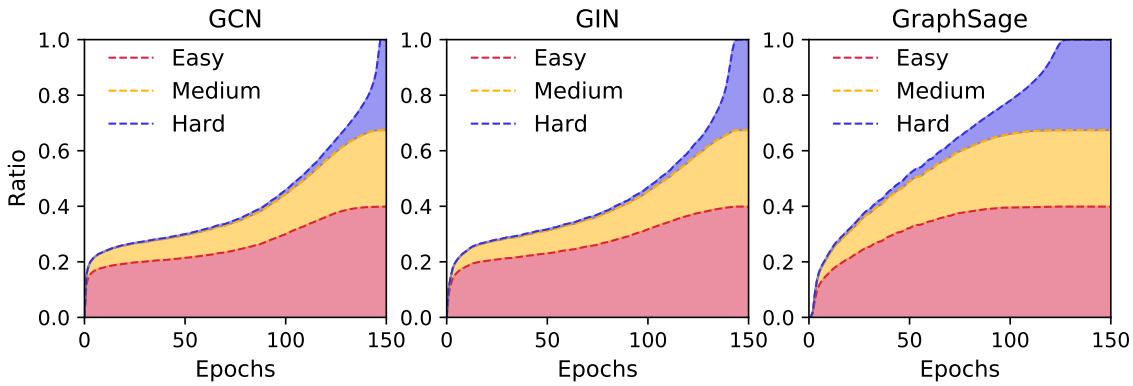


Figure 4.4: Visualization of edge selection process during training.

### 4.4.4 Ablation Study

We present the results of two synthetic datasets (*homophily coefficient*= $0.3, 0.6$) and three real-world datasets in Table 4.3. We summarize our findings from the above table as below: (i) Our full model consistently outperforms the two variants Curriculum-linear and Curriculum-root by an average of 1.59% on all datasets, suggesting that our pacing module can benefit model training. It is worth noting that these two variants also outperform the baseline vanilla GNN model Vanilla by an average of 1.92%, which supports the assumption that even a simple curriculum learning strategy can still improve model performance. (ii) We observe that the performance of the two variants Random-linear and Random-root on all datasets drops by 3.86% on average compared to the variants Curriculum-linear and Curriculum-root. Such behavior demonstrates the effectiveness of our proposed edge difficulty quantification module by showing that randomly involving edges into training cannot benefit model performance. (iii) We can observe a significant performance drop consistently for all variants that remove the structural smoothing techniques and initialization components. The results validate that all structural smoothing and initialization components can benefit the performance of RCL on the downstream tasks.

### 4.4.5 Visualization of Learned Edge Selection Curriculum

Besides the effectiveness and robustness of the RCL method on downstream classification results, it is also interesting to verify whether the learned edge selection curriculum satisfies the rule from easy to hard. Since real-world datasets do not have ground-truth labels of difficulty on edges, we conduct visualization experiments on synthetic datasets, where the difficulty of each edge can be indicated by its formation probability. Specifically, we classify edges into three balanced categories according to their difficulty: easy, medium, and hard. Here, we define all homogenous edges that connect nodes with the same class as easy, edges connecting nodes with adjacent classes as medium, and the remaining edges connecting nodes with far away classes as hard. We report the proportion of edges selected for each category during training in Figure 4.4. We can observe that RCL can effectively select most of the easy edges at the early stage of training, then more easy edges and most medium edges are gradually included during training, and most hard edges are left unselected until the end stage of training. Such edge selection behavior is highly consistent with the core idea of designing a curriculum for edge selection, which verifies that our proposed method can effectively design curriculums to select edges according to their difficulty from easy to hard.

### 4.4.6 Effectiveness Experiments on Heterophilic Datasets

In order to further verify the effectiveness of our proposed strategy on heterophilic graph datasets, we have included new experiments on six real-world heterophilic datasets. As shown in Table 4.4, our method consistently improve performance of backbone GNN models on these heterophilic datasets. Secifically, RCL outperforms the second best method on average by 5.04%, and 4.55%, on GCN and GIN backbones, respectively. The results can demonstrate our method is not limited to homophily

| Dataset | Edge homo ratio | GCN | GCN-RCL | GIN | GIN-RCL |
|---|---|---|---|---|---|
| Texas | 0.11 | 0.5645 | **0.6006** | 0.5885 | **0.6156** |
| Cornell | 0.30 | 0.4084 | **0.5045** | 0.4234 | **0.4925** |
| Wisconsin | 0.21 | 0.4923 | **0.5294** | 0.5141 | **0.5599** |
| Actor | 0.22 | 0.2868 | **0.3186** | 0.2678 | **0.3006** |
| Squirrel | 0.22 | 0.2743 | **0.2999** | 0.2347 | **0.2519** |
| Chameleon | 0.23 | 0.3625 | **0.4385** | 0.3233 | **0.4033** |

Table 4.4: Node classfication results for six real-world heterophilic datasets, where the best performance of each model category in one dataset is highlighted.

| | Synthetic | Citeseer | Computers | ogbn-arxiv | ogbn-proteins |
|---|---|---|---|---|---|
| Vanilla | 7.32s | 3.90s | 16.88s | 55.22s | 1438.23s |
| GNNSVD | 11.49s | 3.82s | 35.96s | 135.72s | 2632.42s |
| CLNode | 6.29s | 3.96s | 17.02s | 58.53s | 1545.53s |
| ProGNN | 220.25s | 72.42s | 1953.23s | (-) | (-) |
| NeuralSparse | 310.02s | 88.91s | 6553.34s | (-) | (-) |
| PTDNet | 153.43s | 48.42s | 2942.02s | (-) | (-) |
| Ours | 4.07s | 2.42s | 14.62s | 71.49s | 2239.05s |

Table 4.5: Running time of our method and comparison methods. Here (-) denotes an out-of-memory error and Vanilla denotes the standard GNN model.

graphs.

Although the inner product decoder utilized in experiments might imply an underlying homophily assumption, our method can still benefit from leveraging the edge curriculum present within the input datasets. A reasonable explanation is that standard GNN models are usually struggled with the heterophily edges, while our methodology designs a curriculum allowing more focus on homophily edges, which potentially leads to the observed performance boost.

### 4.4.7 Time Complexity Analysis

Here we consider GCN as the backbone. First, the time complexity of an $L$-layer GCN is $O(L|\mathcal{E}|b + L|\mathcal{V}|b^2)$ , where $b$ is the number of node attributes. Second, the time complexity of measuring the difficulty levels of edges by reconstruction is $O(|\mathcal{E}|d)$ where $d$ is the number of latent embedding dimensions. Third, the time complexity

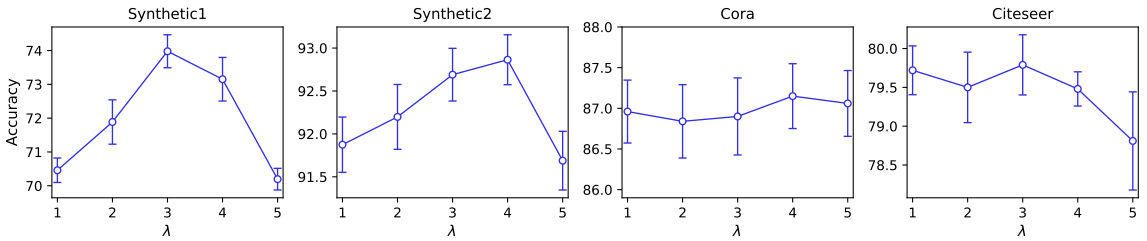


Figure 4.5: Parameter sensitivity analysis on four datasets. Here a larger value of $\lambda$ means the training structure will converge to the original structure at an earlier training stage.

of selecting the edges to add is $O(|\mathcal{E}|)$ . Therefore, the total time complexity of our algorithm is $O(|\mathcal{E}|(Lb+d)+L|\mathcal{V}|b^2)$ .

In addition, we compare the total running time of our method and all comparison methods in the Table 4.5. We can observe that the running time of our proposed method is comparable to that of standard GNN models in all datasets. Notably, our method is even faster than standard GNN models in some datasets. One possible reason is that at the beginning of training, the graphs in our model have much fewer edges than those in standard GNN models. Therefore, the computational cost of the GNN model is also reduced.

### 4.4.8 Parameter Sensitivity Analysis

Recall that RCL learns a curriculum to gradually add edges in a given input graph structure to the training process until all edges are included. An interesting question is how the speed of adding edges will affect the performance of the model. Here we conduct experiments to explore the impact of age parameter $\lambda$ which controls the speed of adding edges to the model performance. Here a larger value of $\lambda$ means that the training structure will converge to the input structure earlier. For example, $\lambda = 1$ means that the training structure will probably not converge to the input structure until the last iteration, and $\lambda = 5$ means that the training structure will converge to the input structure around half of the iterations are complete, and then the model will

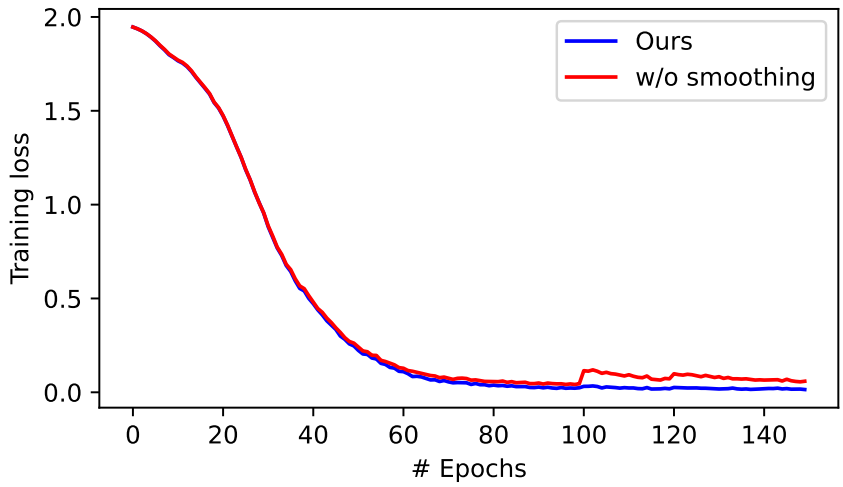


Figure 4.6: The comparison between our full model and the version without smoothing technique on the training loss trend.

be trained with the full input structure for the remaining iterations. We present the results on two synthetic datasets (*homophily coefficient*$= 0.3, 0.6$) and two real-world datasets in Figure 4.5. As can be seen from the figure, the classification results are steady that the average standard deviation is only 0.41%. It is also worth noting that the peak values for all datasets consistently appear around $\lambda = 3$, which indicates that the best performance is when the training structure converges to the full input structure around two-thirds of the iterations are completed.

### 4.4.9 Visualization of Importance on Smoothing Component

Our experimental results demonstrated the importance of applying our smoothing component in stablizing the optimization process of training. Figure 4.6 shows that without the smoothing technique, the training loss spiked that reflects the GNN parameter shifts, which was caused by the number of edges discretely changed. However, after adding the smoothing technique, the training loss can smoothly converge, hence, the smoothing technique plays an important role in stabilizing the training process.

| Dataset | PNA | PNA-RCL | PNA-linear | PNA-root | GCN | GCN-RCL | GCN-linear | GCN-root |
|---|---|---|---|---|---|---|---|---|
| Synthetic-0.3 | 0.6982 | **0.7667** | 0.7463 | 0.7445 | 0.6517 | **0.7398** | 0.6641 | 0.6533 |
| Synthetic-0.5 | 0.8742 | **0.9016** | 0.8476 | 0.8704 | 0.8715 | **0.9269** | 0.8494 | 0.8854 |
| Synthetic-0.7 | 0.9658 | **0.9821** | 0.9514 | 0.9766 | 0.9748 | **0.9962** | 0.9712 | 0.9796 |
| Cora | 0.8310 | **0.8521** | 0.8145 | 0.8254 | 0.8574 | **0.8715** | 0.8327 | 0.8553 |
| Citeseer | 0.7478 | **0.7652** | 0.7482 | 0.7505 | 0.7893 | **0.7979** | 0.7723 | 0.7814 |
| Computers | 0.8989 | **0.9096** | 0.8866 | 0.8975 | 0.8809 | **0.9023** | 0.8713 | 0.8985 |
| ogbn-arxiv | 0.7175 | **0.7441** | 0.6980 | 0.7242 | 0.7174 | **0.7408** | 0.7288 | 0.7359 |

Table 4.6: Node classfication results for our method and traditional CL methods using PNA and GCN as backbone. Here '-RCL' denotes our method, while '-linear' and '-root' denotes two traditional CL methods with different pacing functions.

### 4.4.10 Effectiveness Experiments on PNA Backbone Model

In Table 4.6, new experiments that adopt modern GNN architecture - PNA model [43] have been added. From the table we can observe that our proposed method improves the performance of PNA backbone by 2.54% on average, which further verified the effectiveness of our method under different choices of backbone GNN model.

In addition, in Table 4.6 we further include two traditional CL methods for independent data as additional baselines, following classical works [15, 120]. We employed the supervised training loss of a pretrained GNN model as the difficulty metric, and selected two well-established pacing functions for curriculum design: linear and root pacing, defined as follows:

$$\text{Linear}: K_{\text{linear}}(t) = \frac{t}{T}|V|;$$

$$\text{Root}: K_{\text{root}}(t) = \sqrt{\frac{t}{T}}|V|,$$

where $t$ is the number of current iterations and $T$ is the number of total iterations, and $|V|$ is the number of nodes.

We utilized GCN and PNA as backbone architectures, identified by the suffixes '-linear' and '-root'. Across all datasets, the results consistently demonstrate that our proposed method outperforms traditional CL approaches.

| Dataset | Method | 0% | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Cora | PNA | 0.8310 | 0.7911 | 0.7621 | 0.7402 | 0.7331 | 0.7210 | 0.6894 | 0.7042 | 0.6792 | 0.6617 |
| Cora | PNA-RCL | 0.8521 | 0.8315 | 0.8162 | 0.7969 | 0.7992 | 0.7951 | 0.7571 | 0.7642 | 0.7457 | 0.7371 |
| Citeseer | PNA | 0.7478 | 0.7195 | 0.7184 | 0.6934 | 0.6952 | 0.6920 | 0.6852 | 0.6552 | 0.6481 | 0.6327 |
| Citeseer | PNA-RCL | 0.7652 | 0.7422 | 0.7222 | 0.7254 | 0.7041 | 0.7012 | 0.6953 | 0.6921 | 0.6884 | 0.6794 |

Table 4.7: Further robustness test using PNA as backbone model. Here the percentage denotes the ratio of number of added random edges to the original edges.

### 4.4.11 Robustness Experiments on PNA Backbone Model

We present further robustness test against random noisy edges by using the PNA backbone model. The results are shown in Table 4.7, which further proves that our curriculum learning approach improves the robustness against edge noise with the advanced PNA model as the backbone.

## 4.5 Conclusion

We focus on developing a novel CL method to improve the generalization ability and robustness of GNN models on learning representations of data samples with dependencies. The proposed method **R**elational **C**urriculum **L**earning (**RCL**) effectively addresses the unique challenges in designing CL strategy for handling dependencies. First, a self-supervised learning module is developed to select appropriate edges that are expected by the model. Then an optimization model is presented to iteratively increment the edges according to the model training status and a theoretical guarantee of the convergence on the optimization algorithm is given. Finally, an edge reweighting scheme is proposed to steady the numerical process by smoothing the training structure transition. Extensive experiments on synthetic and real-world datasets demonstrate the strength of RCL in improving the generalization ability and robustness.

# Chapter 5

# Conclusions

The main goal of this dissertation research is to advance representation learning methods for both physical and information networks by jointly incorporating additional physical or abstract information, the graph's topological structure, and their critical interplay. Our aim is to develop a general foundational model that can be applied across diverse data domains. Additionally, we seek to address significant real-world challenges in foundational models, including issues related to data quality, such as label scarcity, noisy data, and the incompatibility between topological structures and other modalities. To achieve this primary goal, there are three sub-goals focused on different aspects of representation learning for graph-structured data: Data, Model and Task. As illustrated in Figure 5.1, these three key components form the general working pipeline for extract high quality representations from graph structured data.

First, from the data perspective, we aim to combine graph data with other key data modalities to enrich the semantic information of the learned representations. This will be achieved by leveraging the incremental information provided by the graph structure, other data modalities, and their crucial interplay. In my proposed research, we focus on two fundamental data modalities to equipped with graph data: spatial data and textual data.

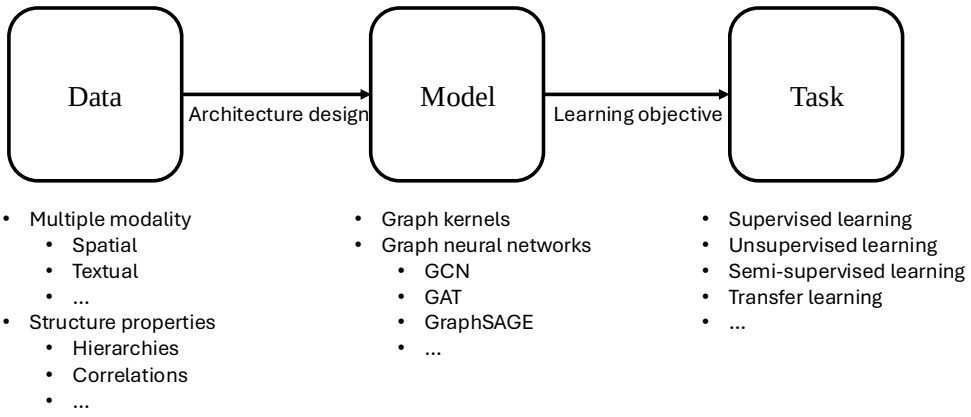


Figure 5.1: Illustration of general work pipeline of representation learning on graphs structured data.

1. For physical networks that combine graph and spatial data, our proposed SGMP method has demonstrated strong discriminative power on Euclidean spatial networks, achieving high predictive performance on extensive biomedical and chemical benchmark datasets. Additionally, we have extended the framework to handle non-Euclidean spatial networks in manifold spaces. The method can also be generalized to special cases such as spatial trees, significantly improving predictive performance in neuron cell prediction and river network prediction.

2. Text data serves as a fundamental modality with the potential to unify different graph data domains into a general format, paving the way for foundational graph representation models. For text-attributed graphs that combine graph and textual data, we propose the TAGA framework, which leverages the unique properties of both structural and textual semantic information to ensure strong expressive power.

Second, from the model perspective, our goal is to design corresponding model architecture that preserves the maximum amount of information from the input data

while maintaining strong expressive power with certain theoretical guarantees. For the proposed SGMP and subsequent frameworks on physical networks, we can theoretically ensure that the model architecture preserves all input geometric structures without information loss. Similarly, the TAGA framework for information networks is designed to transform information losslessly between the two data modalities, thereby ensuring the strong expressive power of the learned representations. Additionally, we have developed accelerated algorithms for each framework to reduce time and memory complexity to linear without compromising the theoretical guarantees of information preservation.

Third, from the perspective of training strategies, our primary goal is to enhance the generalizability and robustness of the learned representations. We have developed a curriculum learning strategy that aims at resolving the incompatibility between noisy graph topology and features from other data modalities. The proposed method gradually incorporates edges based on their difficulty and noise level, demonstrating superior performance compared to state-of-the-art methods in tasks like node classification. Furthermore, our ongoing objective is to advance the field of graph representation learning to address more practical and challenging application scenarios, including label-scarce environments, noisy data, and transfer learning. Future work will focus on extending the proposed framework to tackle complex tasks on graph data, such as zero-shot and few-shot node classification, link prediction, graph clustering, and transfer learning.

## 5.1 Research Contributions

The major research contributions are described as follows.

### 5.1.1 Representation Learning on Physical Networks

1. **A novel generic framework for learning expressive representations on physical networks.** We propose generic framework SGMP with theoretical guarantees on discriminative power and various spatial and network properties. The proposed framework can capture and model the intrinsic coupled spatial and graph properties and ensure the invariance of learned representation under rotation and translation transformations.

2. **An accelerating algorithm for efficiency.** The proposed accelerating algorithm effectively reduces the time and memory complexity from $O(N^3)$ to $O(N)$, and maintains the theoretical guarantees for spatial networks.

3. **A novel generalized framework to handle spatial networks in non-Euclidean space.** The generalized framework effectively addresses the unique challenges of representing irregular spatial networks by first converting the manifold space into a discrete mesh tessellation, and then converting the geometric information of the curves between nodes into messages on edges.

4. **A novel specialized framework to handle geometric trees.** This framework significantly improves geometric tree representations by leveraging their inherent hierarchies and tree-oriented geometric structures.

5. **Extensive experiments to evaluate the performance on synthetic and real-world datasets.** The strength of our theoretical findings through extensive experiments should be demonstrated on both synthetic datasets and real-world datasets across biomedical, chemical, neuroscience and river network domains.

### 5.1.2 Representation Learning on Information Networks

1. **The proposal of a novel generic self-supervised learning framework for representation learning on information networks through text-attributed graphs representing format.** The proposed framework TAGA jointly preserves rich semantic information, topology information, and their interplay by aligning representations of TAGs from both graph and text data modalities.

2. **The proposal of a novel graph-to-text transformation module.** This transformation module requires to maintain the information lossless transformation between graph and text domains, which ensures the equivalent information of alignment process.

3. **Extensive experiments on label scarce scenarios.** We have demonstrated the performance of the proposed framework in label-scarce application settings, such as zero-shot and few-shot predictions. These are challenging tasks where few existing works have demonstrated significant results.

4. **Extension to diverse graph tasks.** We have extended the experiments to several significantly important graph tasks, such as node classification and link prediction.

5. **Extension to transfer learning settings.** Transferring learned knowledge from one graph domain to a new graph domain without strong supervision is a crucial yet extremely challenging task in the field of graph deep learning. This is also a key step toward developing a graph foundation model. We have validated the proposed framework under these challenging settings under zero-shot, few-shot and transfer learning scenarios.

### 5.1.3 Enhancing Generalizability and Robustness of Learning Network Representations

1. **The development of a novel generic curriculum learning framework for representation learning on graph structured data.** The framework is aimed at improving the generalization ability and robustness of representation learners on data with dependencies.

2. **The proposal of a novel graph edge selection criteria based on their difficulty level.** The proposed method select the edges by quantifying their corresponding difficulties in a self-supervised learning manner, thus without the need of extra labels or external human knowledge.

3. **A novel automatic curriculum pacing function.** We present the learning process as a concise optimization model, which automatically lets the model gradually increase the number of including edges to involve more edges in training according to its own status.

4. **A novel edge reweighting scheme.** In order to guarantee a numerical steady process for curriculum learning in graphs, a novel edge reweighting scheme is proposed to smooth the graph structure transition process.

5. **Extensive experiments to evaluate the generalizability and robustness on synthetic and real-world datasets.** The performance of the proposed relational curriculum learning strategy needs to be compared to state-of-the-art comparison methods through extensive experiments on both synthetic and real-world datasets.

## 5.2 Publications and In-Preparation Submissions

### 5.2.1 Published Works

1. Zheng Zhang, and Liang Zhao. "Representation learning on spatial networks." Advances in Neural Information Processing Systems 34 (2021): 2303-2318. [225]

2. Zheng Zhang, and Liang Zhao. "Unsupervised deep subgraph anomaly detection." 2022 IEEE International Conference on Data Mining (ICDM). IEEE, 2022. (**Best Paper Award**) [226]

3. Zheng Zhang, Junxiang Wang, and Liang Zhao. "Curriculum learning for graph neural networks: Which edges should we learn first." Advances in Neural Information Processing Systems 36 (2024). [232]

4. Zheng Zhang, Sirui Li, Jingcheng Zhou, Junxiang Wang, Abhinav Angirekula, Allen Zhang, and Liang Zhao. "Non-Euclidean Spatial Graph Neural Network." In Proceedings of the 2024 SIAM International Conference on Data Mining (SDM), pp. 154-162. Society for Industrial and Applied Mathematics, 2024. [231]

5. Zheng Zhang, and Liang Zhao. "Self-Similar Graph Neural Network for Hierarchical Graph Learning." In Proceedings of the 2024 SIAM International Conference on Data Mining (SDM), pp. 28-36. Society for Industrial and Applied Mathematics, 2024. [227]

6. Zheng Zhang, Allen Zhang, Ruth Nelson, Giorgio Ascoli, Liang Zhao. "Representation Learning of Geometric Trees", 30th SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2024), Research Track, accepted, 2024. [233]

7. Bo Pan, Zheng Zhang, Yifei Zhang, Yuntong Hu and Liang Zhao. "Distilling Large Language Models for Text-Attributed Graph Learning", 33rd ACM

International Conference on Information and Knowledge Management (CIKM 2024), Full Research Track, accepted, 2024. [152]

8. Zheng Zhang, Hossein Amiri, Dazhou Yu, Yuntong Hu, Liang Zhao, Andreas Zufle. Transferable Unsupervised Outlier Detection Framework for Human Semantic Trajectories, ACM Special Interest Group on Spatial Information (ACM SIGSPATIAL 2024), accepted, 2024. (**Best Paper Award Candidates**) [229]

### 5.2.2 Submitted and In-preparation Papers

1. Zheng Zhang, and Liang Zhao. "Transferable Deep Clustering Model." arXiv preprint arXiv:2310.04946 (2023).

2. Zheng Zhang, Fan Yang, Ziyan Jiang, Zheng Chen, Zhengyang Zhao, Chengyuan Ma, Liang Zhao, and Yang Liu. "Position-Aware Parameter Efficient Fine-Tuning Approach for Reducing Positional Bias in LLMs." arXiv preprint arXiv: 2404.01430 (2024).

3. Zheng Zhang, Hossein Amiri, Zhenke Liu, Andreas Züfle, and Liang Zhao. "Large language models for spatial trajectory patterns mining." arXiv preprint arXiv:2310.04942 (2023).

4. Hu, Yuntong, Zheng Zhang, and Liang Zhao. "Beyond Text: A Deep Dive into Large Language Models' Ability on Understanding Graph Data." arXiv preprint arXiv:2310.04944 (2023).

5. Zheng Zhang, Chen Zheng, Da Tang, Ke Sun, Yukun Ma, Yingtong Bu, Xun Zhou, and Liang Zhao. "Balancing specialized and general skills in llms: The impact of modern tuning and data strategy." arXiv preprint arXiv:2310.04945 (2023).

# Appendix A

# Representation Learning on Physical Networks

## A.1 Representation Learning on Euclidean Spatial Networks

### A.1.1 Proof of Theorem 1

It is obvious to show that distances, angles, and torsions are invariant to translation transformations since only relative coordinates are using in the formulation.

For rotation transformations $R \in SO(3)$ (the rotation group in 3D space), we show two identity equations first:

$$\begin{aligned} \langle R\mathbf{x}, R\mathbf{y} \rangle &= \langle \mathbf{x}, \mathbf{y} \rangle \\ (R\mathbf{x}) \times (R\mathbf{y}) &= R(\mathbf{x} \times \mathbf{y}) \end{aligned} \tag{A.1}$$

Thus we have

$$
\begin{aligned}
d_{ij} &= ||\mathbf{P}_{ij}||_2 = \langle \mathbf{P}_{ij}, \mathbf{P}_{ij} \rangle = \langle R\mathbf{P}_{ij}, R\mathbf{P}_{ij} \rangle, \\
\theta_{ijk} &= \arccos(\langle \frac{\mathbf{P}_{ij}}{d_{ij}}, \frac{\mathbf{P}_{jk}}{d_{jk}} \rangle) = \arccos(\langle \frac{R\mathbf{P}_{ij}}{d_{ij}}, \frac{R\mathbf{P}_{jk}}{d_{jk}} \rangle), \\
\bar{\varphi}_{ijkp} &= \arccos(\langle \mathbf{n_{ijk}}, \mathbf{n_{jkp}} \rangle) = \arccos(\langle R\mathbf{n_{ijk}}, R\mathbf{n_{jkp}} \rangle), \\
\text{Parity} &= \langle \frac{\mathbf{n_{ijk}} \times \mathbf{n_{ijp}}}{||\mathbf{n_{ijk}} \times \mathbf{n_{ijp}}||_2}, \frac{\mathbf{P}_{ij}}{||\mathbf{P}_{ij}||_2} \rangle \\
&= \langle \frac{R(\mathbf{n_{ijk}} \times \mathbf{n_{ijp}})}{||R(\mathbf{n_{ijk}} \times \mathbf{n_{ijp}})||_2}, \frac{R\mathbf{P}_{ij}}{||R\mathbf{P}_{ij}||_2} \rangle \\
&= \langle \frac{(R\mathbf{n_{ijk}}) \times (R\mathbf{n_{ijp}})}{||(R\mathbf{n_{ijk}}) \times (R\mathbf{n_{ijp}})||_2}, \frac{R\mathbf{P}_{ij}}{||R\mathbf{P}_{ij}||_2} \rangle.
\end{aligned}
$$

All elements are invariant under rotation and translation transformations. □

### A.1.2 Proof of Lemma 1

Note that from Equation 2.2 we have

$$
\begin{aligned}
||\mathbf{P}_{ij} \times \mathbf{P}_{jk}||_2 &= d_{ij} d_{jk} \sin\theta_{ijk}, \\
\mathbf{n_{ijk}} \times \mathbf{n_{ijp}} &= \frac{1}{d_{ij}^2 d_{jk} d_{jp} \sin\theta_{ijk} \sin\theta_{ijp}} (\mathbf{P}_{ij} \times \mathbf{P}_{jk}) \times (\mathbf{P}_{ij} \times \mathbf{P}_{jp}) \\
&= \frac{1}{d_{ij}^2 d_{jk} d_{jp} \sin\theta_{ijk} \sin\theta_{ijp}} (\mathbf{P}_{ij} \cdot (\mathbf{P}_{jk} \times \mathbf{P}_{jp})) \mathbf{P}_{ij}, \\
\langle \mathbf{n_{ijk}}, \mathbf{n_{ijp}} \rangle &= \cos\bar{\varphi}_{ijkp} = \cos\varphi_{ijkp}, \\
\text{Parity} &= \frac{\langle \mathbf{n_{ijk}} \times \mathbf{n_{ijp}}, \mathbf{P}_{ij} \rangle}{d_{ij} \sin\bar{\varphi}_{ijkp}} = \frac{\mathbf{P}_{ij} \cdot (\mathbf{P}_{jk} \times \mathbf{P}_{jp})}{d_{ij} d_{jk} d_{jp} \sin\theta_{ijk} \sin\theta_{ijp} \sin\bar{\varphi}_{ijkp}}.
\end{aligned}
$$

Suppose that the solution set of the Equation 2.1 contains two different solutions $v_i$ and $v_i'$, then the Equation 2.2 implies that due to the same representation, we have

$$
\begin{aligned}
\text{Parity}(i) &= \text{Parity}(i') \\
\frac{\mathbf{P}_{ij} \cdot (\mathbf{P}_{jk} \times \mathbf{P}_{jp})}{d_{ij} d_{jk} d_{jp} \sin\theta_{ijk} \sin\theta_{ijp} \sin\bar{\varphi}_{ijkp}} &= \frac{\mathbf{P}_{i'j} \cdot (\mathbf{P}_{jk} \times \mathbf{P}_{jp})}{d_{i'j} d_{jk} d_{jp} \sin\theta_{i'jk} \sin\theta_{i'jp} \sin\bar{\varphi}_{i'jkp}} \\
0 &= \mathbf{P}_{ii'} \cdot (\mathbf{P}_{jk} \times \mathbf{P}_{jp}).
\end{aligned}
$$

Since $v_j, v_k, v_p$ is non-colinear, $\mathbf{P}_{jk} \times \mathbf{P}_{jp}$ is nonzero, the equation causes a contradiction. Thus, the Cartesian coordinate of node $v_i$ can be uniquely determined by the Equation 2.1. □

### A.1.3 Proof of Theorem 3

In this theorem we denote $S$ as the representation of spatial network in Equation 2.1. Due to the continuity of $g$, we take $\delta_\epsilon$ so that $|g(S)-g(S')| < \epsilon$ for any $S, S' \in \mathcal{S}$ if the Hausdorff distance between spatial information $d_H(S, S') < \delta_\epsilon$. Define the $\mathcal{K}_d, \mathcal{K}_\theta, \mathcal{K}_\varphi$ as the resolution of geometric features $d$, $\theta$, $\varphi$, and without lossing generality, we can suppose $\mathcal{K} = \mathcal{K}_d = \mathcal{K}_\theta = \mathcal{K}_\varphi = \lceil \frac{1}{\delta_\epsilon} \rceil$. Let the mapping function $\Lambda$ as $\Lambda(S) = \frac{\lfloor \mathcal{K} S \rfloor}{\mathcal{K}}$ which maps all the elements in an interval to the left end of the interval, such that we have $|g(S) - g(\Lambda(S))| < \epsilon$ for any $S \in \mathcal{S}$.

Let $f(S) \in \mathbb{R}^I$, where $I$ is the number of dimension of embedding vectors for $S$. Then consider $f_\iota(S) = e^{||S-\hat{S}_\iota||_2}, \iota \in [1, ..., I]$, where $\hat{S}_\iota$ is one unit space in the transformed space of the mapping function $\Lambda$. Intuitively, we can consider each $f_\iota(S)$ measures if $S$ is located in a unit $\hat{S}_\iota$ of the discretized space by a smooth indicator. Similarly, we can define another mapping $\tilde{\xi}$ with discretize value as $\tilde{\xi}(S) = [\tilde{\xi}_1(S); \ldots; \tilde{\xi}_I(S)]$ with each $\tilde{\xi}_\iota(S) = 1$ indicating $S$ is located in the $\iota$-th unit, otherwise 0. The mapping from $f_\iota(S)$ to $\tilde{\xi}_\iota(S)$ can be easily learned by a linear function with ReLU as the activation function, which is exactly the setting in our framework. We denote this function as $\omega$. We finally have $\tilde{\xi}(S) = \omega(f(S))$.

It is obvious that $\tilde{\xi}(S)$ is equivalent to $\Lambda(S)$. Let $\bar{\gamma}$ be a continuous function from $\mathbb{R}^I$ to $\mathbb{R}$ such that $\bar{\gamma}(\tilde{\xi}(S)) = g(\tilde{\xi}(S))$ and we can rewrite $\gamma = \bar{\gamma} \circ \omega$ Then

$$
\begin{aligned}
&|\bar{\gamma}(\tilde{\xi}(S)) - g(S)| \\
=&|\bar{\gamma}(\omega(f(S))) - g(S)| \\
=&|\gamma(f(S)) - g(S)| < \epsilon
\end{aligned}
$$

The proof is complete here. □

### A.1.4 Proof of Proposition 1

The number of spanning trees in a graph $G$ is given by Kirchhoff's matrix tree theorem [29], showing that the number can be computed as $\mathrm{KMT}(G) = det\big[L[u]\big]$, where $L[u]$ is the graph Laplacian matrix $L$ with its $u^{th}$ row and column removed, and $u$ denotes a randomly chosen vertice. Obviously, the probability $\Pr(e_{ij} \in T)$ of sampling one edge $e_{ij}$ in a random spanning tree $T$ can be computed as

$$\Pr(e_{ij} \in T) = \frac{KMT(G) - KMT(\tilde{G}_{e_{ij}})}{KMT(G)},$$

where $\tilde{G}_{e_{ij}}$ is the graph $G$ removed edge $e_{ij}$. Then we write

$$\begin{aligned}\Pr(\pi_{ijkp} \in T) &= \Pr(e_{ij} \in T | e_{jk}, e_{kp} \in T)\Pr(e_{jk}, e_{kp} \in T) \\ &= \Pr(e_{ij} \in T | e_{jk}, e_{kp} \in T)\Pr(e_{jk} | e_{kp} \in T)\Pr(e_{kp} \in T),\end{aligned}$$

which can be formulated as

$$\Pr(\pi_{ijkp} \in T) = det[Y_{\pi_{ijkp}}],$$

given by Burton-Pemantle theorem [27]. We do not give the proof of Burton-Pemantle theorem here since it is not the focus of this paper. □

### A.1.5 Proof of Proposition 2

Because the random spanning trees are sampled i.i.d. from a uniform distribution, by the strong law of large number,

$$\tilde{h}_i^{(\ell+1)} = \mathrm{SUM}\big(\big\{\frac{m^{(\ell)}(\pi_{ijkp})}{q(\pi_{ijkp})} | \pi_{ijkp} \in \bar{\Pi}_{T,3}^{i}\big\}\big),$$

converges almost surely to the expected value $\mathrm{SUM}(m^{(\ell)}(\pi_{ijkp}))$. Then since $\sigma^{(\ell)}$ is continuous, by the continuous mapping theorem, the limits are preserved such that Equation 2.1

$$h_i^{(\ell+1)} = \sigma^{(\ell)}\Big(\mathrm{SUM}\big(\{\frac{m^{(\ell)}(\pi_{ijkp})}{q(\pi_{ijkp})}|\pi_{ijkp} \in \bar{\Pi}^i_{T,3}\}\big)\Big),$$

converges almost surely to Equation 2.3. □

## A.2 Representation Learning on Non-Euclidean Spatial Networks

### A.2.1 Proof of Theorem 4

*Proof.* Intuitively, distance, angle, torsion, and orientation angle are invariant to translation and rotation transformations, since only relative coordinates are used in the formula. Formally, for translation transformations $\mathcal{T} \in SE(3)$ and rotation transformations $\mathcal{R} \in SE(3)$, the following identity equations hold:

$$\begin{aligned} \mathcal{T}(\mathbf{x}-\mathbf{y}) &= \mathbf{x}-\mathbf{y}, \\ \langle \mathcal{R}(\mathbf{x}), \mathcal{R}(\mathbf{y}) \rangle &= \langle \mathbf{x}, \mathbf{y} \rangle \\ \mathcal{R}(\mathbf{x}) \times \mathcal{R}(\mathbf{y}) &= \mathcal{R}(\mathbf{x} \times \mathbf{y}) \end{aligned} \tag{A.2}$$

Thus we have

$$
\begin{aligned}
d &= ||\mathbf{l}||_2 = ||\mathbf{p} - \mathbf{p}'||_2 = ||\mathcal{T}(\mathbf{p}) - \mathcal{T}(\mathbf{p}')||_2, \\
d &= ||\mathbf{l}||_2 = \langle \mathbf{l}, \mathbf{l} \rangle = \langle \mathcal{R}(\mathbf{l}), \mathcal{R}(\mathbf{l}) \rangle, \\
\theta &= \arccos(\langle \frac{\mathbf{l}}{d}, \frac{\mathbf{L_{ij}}}{d} \rangle) \\
&= \arccos(\langle \frac{\mathcal{R}(\mathbf{l})}{d}, \frac{\mathcal{R}(\mathbf{L_{ij}})}{d} \rangle), \\
\phi^{(k,k+1)} &= \langle \frac{\mathbf{c}^{(k)} \times \mathbf{c}^{(k+1)}}{||\mathbf{c}^{(k)} \times \mathbf{c}^{(k+1)}||_2}, \frac{\mathbf{L}_{ij}}{||\mathbf{L}_{ij}||_2} \rangle \cdot \bar{\phi}^{(k,k+1)} \\
\text{where} \quad \mathbf{c} &= \frac{\mathbf{L}_{ij} \times \mathbf{l}}{||\mathbf{L}_{ij} \times \mathbf{l}||_2}, \\
\mathbf{c}^{(k)} \times \mathbf{c}^{(k+1)} &= \frac{(\mathbf{L}_{ij} \times \mathbf{l}^{(k)}) \times (\mathbf{L}_{ij} \times \mathbf{l}^{(k+1)})}{||\mathbf{L}_{ij} \times \mathbf{l}^{(k)}||_2 ||\mathbf{L}_{ij} \times \mathbf{l}^{(k+1)}||_2}, \\
&= \frac{(\mathbf{L}_{ij} \cdot (\mathbf{l}^{(k)} \times \mathbf{l}^{(k+1)}))\mathbf{L}_{ij}}{||\mathbf{L}_{ij} \times \mathbf{l}^{(k)}||_2 ||\mathbf{L}_{ij} \times \mathbf{l}^{(k+1)}||_2}, \\
\text{thus } &\langle \frac{\mathbf{c}^{(k)} \times \mathbf{c}^{(k+1)}}{||\mathbf{c}^{(k)} \times \mathbf{c}^{(k+1)}||_2}, \frac{\mathbf{L}_{ij}}{||\mathbf{L}_{ij}||_2} \rangle \\
&= \langle \frac{(\mathbf{L}_{ij} \cdot (\mathbf{l}^{(k)} \times \mathbf{l}^{(k+1)}))\mathbf{L}_{ij}}{||\mathbf{c}^{(k)} \times \mathbf{c}^{(k+1)}||_2 ||\mathbf{L}_{ij} \times \mathbf{l}^{(k)}||_2 ||\mathbf{L}_{ij} \times \mathbf{l}^{(k+1)}||_2}, \frac{\mathbf{L}_{ij}}{||\mathbf{L}_{ij}||_2} \rangle \\
&= \frac{\mathbf{L}_{ij} \cdot (\mathbf{l}^{(k)} \times \mathbf{l}^{(k+1)})}{||\mathbf{c}^{(k)} \times \mathbf{c}^{(k+1)}||_2 ||\mathbf{L}_{ij} \times \mathbf{l}^{(k)}||_2 ||\mathbf{L}_{ij} \times \mathbf{l}^{(k+1)}||_2} \\
&= \frac{\mathcal{R}(\mathbf{L}_{ij}) \cdot (\mathcal{R}(\mathbf{l}^{(k)} \times \mathbf{l}^{(k+1)}))}{||\mathbf{c}^{(k)} \times \mathbf{c}^{(k+1)}||_2 ||\mathbf{L}_{ij} \times \mathbf{l}^{(k)}||_2 ||\mathbf{L}_{ij} \times \mathbf{l}^{(k+1)}||_2} \\
\varphi^{(k-1,k)} &= \arccos(\langle \mathbf{n}^{(k-1)}, \mathbf{n}^{(k)} \rangle), \\
&= = \arccos(\mathcal{R}(\langle \mathbf{n}^{(k-1)}, \mathbf{n}^{(k)} \rangle)).
\end{aligned}
$$

All extracted geometric features are invariant under rotation and translation transformations. □

### A.2.2 Proof of Theorem 5

The proof of Theorem 5 is a consequence of the Lemma 1.

Then we provide the proof for Theorem 5.

*Proof.* As stated in Lemma 1, the Cartesian coordinates of a point $p^{(k)}$ can be derived from its two endpoints and their connected neighbor points $p^{(k-1)}$. Leveraging the connectivity of the spatial graph, we can iteratively compute the coordinates of a connected point based on the set of points with known coordinates. By initiating the process from any arbitrary point, we can determine the Cartesian coordinates for the entire spatial network. □

*Proof.* Note that from Equation 2.1 we have followings:

$$
\begin{aligned}
||\mathbf{L}_{i,j} \times \mathbf{l}^{(k)}||_2 &= ||(\mathbf{p}_i - \mathbf{p}_j) \times (\mathbf{p}^{(k)} - \mathbf{p}^{(k-1)})||_2 = d^k d_{i,j} \sin\theta^{(k)}, \\
\mathbf{c}^{(k)} \times \mathbf{c}^{(k-1)} &= \frac{(\mathbf{L}_{i,j} \times (\mathbf{p}^{(k)} - \mathbf{p}_j)) \times (\mathbf{L}_{i,j} \times (\mathbf{p}^{(k-1)} - \mathbf{p}_j))}{d_{i,j}^2 d^{(k)} d^{(k-1)} \sin\theta^{(k)} \sin\theta^{(k-1)}} \\
&= \frac{(\mathbf{L}_{i,j} \cdot ((\mathbf{p}^{(k)} - \mathbf{p}_j) \times (\mathbf{p}^{(k-1)} - \mathbf{p}_j)))\mathbf{L}_{i,j}}{d_{i,j}^2 d^{(k)} d^{(k-1)} \sin\theta^{(k)} \sin\theta^{(k-1)}}
\end{aligned}
$$

$$
\begin{aligned}
\langle \mathbf{c}^{(k)}, \mathbf{c}^{(k-1)} \rangle &= \cos\bar{\phi}^{(k,k-1)} = \cos\phi^{(k,k-1)}, \\
\frac{\langle \mathbf{c}^{(k)} \times \mathbf{c}^{(k-1)}, \mathbf{L}_{i,j} \rangle}{d_{i,j} \sin\bar{\phi}^{(k,k-1)}} &= \frac{\mathbf{L}_{i,j} \cdot ((\mathbf{p}_j - \mathbf{p}^{(k)}) \times (\mathbf{p}_j - \mathbf{p}^{(k-1)}))}{d_{i,j} d^{(k)} d^{(k-1)} \sin\theta^{(k)} \sin\theta^{(k-1)} \sin\bar{\phi}^{(k,k-1)}}.
\end{aligned}
$$

Suppose there exists two different positions of point $p^{(k)}$ and $p^{(k)\prime}$ that satisfy Equation 2.1, then it implies that

$$
\begin{aligned}
&\frac{\mathbf{L}_{i,j} \cdot ((\mathbf{p}_j - \mathbf{p}^{(k)}) \times (\mathbf{p}_j - \mathbf{p}^{(k-1)}))}{d_{i,j} d^{(k)} d^{(k-1)} \sin\theta^{(k)} \sin\theta^{(k-1)} \sin\bar{\phi}^{(k,k-1)}} \\
&= \frac{\mathbf{L}_{i,j} \cdot ((\mathbf{p}_j - \mathbf{p}^{(k)\prime}) \times (\mathbf{p}_j - \mathbf{p}^{(k-1)}))}{d_{i,j} d^{(k)} d^{(k-1)} \sin\theta^{(k)} \sin\theta^{(k-1)} \sin\bar{\phi}^{(k,k-1)}} \\
0 &= (\mathbf{p}^{(k)} - \mathbf{p}^{(k)\prime}) \cdot ((\mathbf{p}_j - \mathbf{p}^{(k)}) \times (\mathbf{p}_j - \mathbf{p}^{(k-1)})).
\end{aligned}
$$

Here $((\mathbf{p}_j - \mathbf{p}^{(k)}) \times (\mathbf{p}_j - \mathbf{p}^{(k-1)}))$ is nonzero as long as $\mathbf{p}_j$, $\mathbf{p}^{(k)}$, $\mathbf{p}^{(k-1)}$ is non-colinear. Therefore, the above equation causes a contradiction which proves that $p^{(k)}$ can not have two different positions given the representation in Equation 2.1. □

# Appendix B

# Representation Learning on Information Networks

## B.1 Additional Experimental Results and Settings

In this section, we present additional experimental settings and results due to the space limitation of the main paper.

### B.1.1 Additional Implementation Settings

All experiments are conducted on a 64-bit machine with four 16GB NVIDIA GPUs. Each experiment involves running the models 20 times with different random seeds to minimize variance due to specific data splits. Accuracy is adopted as the evaluation metric for node classification tasks. Specifically, for smaller datasets such as Cora and PubMed, we employ 3 convolution layers, while for larger datasets, we utilize 2 layers. Latent dimension is aligned with the PLM embedding dimension. During the pre-train stage, the model is trained with 40,000 steps on each dataset with minibatch size 8. The learning rate is initialized as $1e^{-3}$ and with decay rate 0.999 each 10 steps. For zero-shot predictions, we utilize the entire dataset as the test set. In the case of

$k$-shot predictions, we randomly select $k$ samples from each class to form the training set, dividing the remaining data into validation and test sets at a ratio of 1:9. All models undergo finetune for 100 epochs, and testing is based on the best validation results.

### B.1.2 Additional Link Prediction Experiments

In order to verify the generalizability of our method, the transfer learning setting is adopted. The representation learning method is pre-trained on source dataset, and then directly perform link prediction task on target dataset without any finetune process. The ratio of positive and negative edges is 1:1 and we use cosine similarity to measure the scores. From the Table B.1 we can observe that our proposed method outperforms all the comparison methods in 15 out of 16 tasks on ROC-AUC metric, which further verified the effectiveness and generalizability of our proposed representation learning method.

| **Source** | **Target** | **GRACE** | **G2P2** | **TAGA** |
|---|---|---|---|---|
| Pubmed | Cora | 0.6007 ± 0.0019 | 0.9964 ± 0.0001 | **0.9971 ± 0.0005** |
| | Pubmed | 0.8240 ± 0.0008 | 0.9564 ± 0.0003 | **0.9683 ± 0.0002** |
| | Sports | 0.6094 ± 0.0002 | **0.9864 ± 0.0000** | 0.9844 ± 0.0000 |
| | Arxiv | 0.5318 ± 0.0002 | 0.9847 ± 0.0000 | **0.9865 ± 0.0001** |
| Arxiv | Cora | 0.9170 ± 0.0008 | 0.9928 ± 0.0002 | **0.9947 ± 0.0003** |
| | Pubmed | 0.8047 ± 0.0006 | 0.9563 ± 0.0003 | **0.9662 ± 0.0004** |
| | Sports | 0.7636 ± 0.0001 | 0.9907 ± 0.0000 | **0.9940 ± 0.0000** |
| | Arxiv | 0.9386 ± 0.0001 | 0.9857 ± 0.0000 | **0.9886 ± 0.0000** |
| Cora | Cora | 0.9646 ± 0.0005 | 0.9886 ± 0.0004 | **0.9959 ± 0.0002** |
| | Pubmed | 0.9363 ± 0.0006 | 0.9508 ± 0.0005 | **0.9634 ± 0.0002** |
| | Sports | 0.9727 ± 0.0000 | 0.9816 ± 0.0000 | **0.9913 ± 0.0000** |
| | Arxiv | 0.9735 ± 0.0001 | 0.9620 ± 0.0001 | **0.9901 ± 0.0000** |
| Sports | Cora | 0.7847 ± 0.0010 | 0.9911 ± 0.0002 | **0.9955 ± 0.0002** |
| | Pubmed | 0.8718 ± 0.0005 | 0.9611 ± 0.0003 | **0.9667 ± 0.0005** |
| | Sports | 0.9353 ± 0.0000 | 0.9906 ± 0.0000 | **0.9942 ± 0.0000** |
| | Arxiv | 0.8990 ± 0.0001 | 0.9780 ± 0.0000 | **0.9842 ± 0.0000** |

Table B.1: The ROC-AUC experimental results of zero-shot link prediction tasks by transferring from the source dataset to target dataset.

| $k$-Shot | Model | Arxiv | Children | Computers | Cora | History | Photo | Pubmed | Sports |
|---|---|---|---|---|---|---|---|---|---|
| | # Nodes | 169,343 | 76,875 | 87,229 | 2,708 | 41,551 | 48,362 | 19,717 | 173,055 |
| | # Edges | 1,166,243 | 1,554,578 | 721,107 | 10,556 | 358,574 | 500,939 | 44,338 | 1,773,594 |
| | Avg # Words | 220.7 | 199.3 | 90.7 | 148.2 | 218.7 | 144.5 | 50.1 | 9.8 |
| 0 | PLM | 0.500 ± 0.001 | 0.094 ± 0.003 | 0.427 ± 0.001 | 0.624 ± 0.005 | 0.169 ± 0.001 | 0.387 ± 0.009 | 0.475 ± 0.008 | 0.316 ± 0.002 |
| | GraphMAE | 0.104 ± 0.001 | 0.021 ± 0.001 | 0.049 ± 0.001 | 0.194 ± 0.006 | 0.019 ± 0.001 | 0.152 ± 0.001 | 0.438 ± 0.001 | 0.112 ± 0.001 |
| | GraphCL | 0.089 ± 0.001 | 0.037 ± 0.001 | 0.173 ± 0.001 | 0.176 ± 0.003 | 0.191 ± 0.001 | 0.174 ± 0.001 | 0.368 ± 0.001 | 0.140 ± 0.001 |
| | GRACE | 0.045 ± 0.001 | 0.034 ± 0.001 | 0.169 ± 0.001 | 0.146 ± 0.004 | 0.079 ± 0.001 | 0.025 ± 0.001 | 0.335 ± 0.001 | 0.057 ± 0.001 |
| | G2P2 | 0.453 ± 0.002 | 0.201 ± 0.001 | 0.453 ± 0.001 | 0.644 ± 0.004 | 0.322 ± 0.003 | **0.452 ± 0.001** | 0.576 ± 0.006 | 0.436 ± 0.001 |
| | TAGA | **0.537 ± 0.003** | **0.224 ± 0.001** | **0.498 ± 0.004** | **0.682 ± 0.005** | **0.351 ± 0.009** | 0.419 ± 0.001 | **0.616 ± 0.009** | **0.448 ± 0.003** |
| | TAGA-rw | 0.530 ± 0.001 | 0.221 ± 0.001 | 0.494 ± 0.001 | 0.680 ± 0.002 | 0.301 ± 0.003 | 0.394 ± 0.001 | 0.599 ± 0.002 | 0.434 ± 0.002 |
| 1 | PLM | 0.280 ± 0.044 | 0.122 ± 0.042 | 0.238 ± 0.039 | 0.412 ± 0.080 | 0.284 ± 0.078 | 0.230 ± 0.051 | 0.503 ± 0.067 | 0.282 ± 0.068 |
| | GraphMAE | 0.255 ± 0.041 | 0.128 ± 0.028 | 0.300 ± 0.052 | 0.474 ± 0.058 | 0.231 ± 0.052 | 0.304 ± 0.066 | 0.492 ± 0.076 | 0.270 ± 0.042 |
| | GraphCL | 0.123 ± 0.031 | 0.157 ± 0.066 | 0.256 ± 0.039 | 0.402 ± 0.059 | 0.371 ± 0.124 | 0.325 ± 0.079 | 0.414 ± 0.040 | 0.347 ± 0.079 |
| | GRACE | 0.263 ± 0.034 | 0.138 ± 0.035 | 0.336 ± 0.051 | 0.435 ± 0.071 | 0.266 ± 0.085 | 0.295 ± 0.053 | 0.514 ± 0.095 | 0.282 ± 0.045 |
| | G2P2 | 0.308 ± 0.052 | 0.145 ± 0.029 | 0.359 ± 0.044 | 0.477 ± 0.082 | 0.361 ± 0.092 | 0.372 ± 0.066 | 0.522 ± 0.085 | 0.356 ± 0.042 |
| | TAGA | **0.323 ± 0.040** | **0.180 ± 0.073** | **0.380 ± 0.062** | 0.509 ± 0.089 | **0.413 ± 0.114** | **0.417 ± 0.077** | **0.563 ± 0.062** | 0.440 ± 0.070 |
| | TAGA-rw | 0.307 ± 0.050 | 0.171 ± 0.013 | 0.365 ± 0.042 | **0.561 ± 0.063** | 0.383 ± 0.078 | 0.380 ± 0.037 | 0.548 ± 0.073 | **0.498 ± 0.084** |
| 3 | PLM | 0.436 ± 0.036 | 0.194 ± 0.029 | 0.318 ± 0.038 | 0.588 ± 0.036 | 0.448 ± 0.071 | 0.352 ± 0.044 | 0.611 ± 0.051 | 0.392 ± 0.041 |
| | GraphMAE | 0.379 ± 0.039 | 0.182 ± 0.025 | 0.389 ± 0.035 | 0.634 ± 0.044 | 0.362 ± 0.050 | 0.432 ± 0.051 | 0.597 ± 0.061 | 0.363 ± 0.050 |
| | GraphCL | 0.192 ± 0.029 | 0.186 ± 0.039 | 0.343 ± 0.046 | 0.563 ± 0.044 | 0.484 ± 0.071 | 0.382 ± 0.052 | 0.476 ± 0.038 | 0.373 ± 0.071 |
| | GRACE | 0.398 ± 0.031 | 0.200 ± 0.038 | 0.442 ± 0.045 | 0.622 ± 0.043 | 0.404 ± 0.057 | 0.447 ± 0.053 | 0.620 ± 0.055 | 0.398 ± 0.045 |
| | G2P2 | 0.430 ± 0.027 | 0.207 ± 0.038 | 0.469 ± 0.042 | 0.623 ± 0.033 | **0.508 ± 0.073** | 0.528 ± 0.049 | 0.641 ± 0.064 | 0.464 ± 0.050 |
| | TAGA | **0.445 ± 0.035** | 0.241 ± 0.062 | **0.497 ± 0.035** | 0.695 ± 0.050 | 0.551 ± 0.094 | **0.551 ± 0.045** | 0.659 ± 0.058 | **0.586 ± 0.057** |
| | TAGA-rw | 0.442 ± 0.040 | **0.222 ± 0.060** | 0.467 ± 0.025 | **0.705 ± 0.021** | 0.558 ± 0.072 | 0.513 ± 0.070 | **0.632 ± 0.043** | 0.569 ± 0.105 |
| 5 | PLM | 0.500 ± 0.019 | 0.210 ± 0.025 | 0.377 ± 0.027 | 0.641 ± 0.031 | 0.557 ± 0.040 | 0.420 ± 0.037 | 0.632 ± 0.040 | 0.478 ± 0.056 |
| | GraphMAE | 0.425 ± 0.028 | 0.212 ± 0.029 | 0.434 ± 0.036 | 0.704 ± 0.038 | 0.459 ± 0.038 | 0.489 ± 0.038 | 0.625 ± 0.049 | 0.452 ± 0.037 |
| | GraphCL | 0.231 ± 0.015 | 0.201 ± 0.040 | 0.397 ± 0.040 | 0.641 ± 0.044 | 0.531 ± 0.047 | 0.462 ± 0.041 | 0.584 ± 0.037 | 0.477 ± 0.048 |
| | GRACE | 0.445 ± 0.028 | 0.227 ± 0.031 | 0.472 ± 0.040 | 0.685 ± 0.027 | 0.481 ± 0.061 | 0.515 ± 0.042 | 0.628 ± 0.047 | 0.482 ± 0.040 |
| | G2P2 | 0.466 ± 0.025 | 0.240 ± 0.034 | 0.510 ± 0.039 | 0.703 ± 0.032 | 0.617 ± 0.053 | 0.583 ± 0.051 | 0.640 ± 0.051 | 0.565 ± 0.055 |
| | TAGA | **0.483 ± 0.022** | 0.263 ± 0.031 | **0.543 ± 0.038** | 0.752 ± 0.028 | **0.636 ± 0.046** | 0.602 ± 0.041 | 0.649 ± 0.044 | 0.664 ± 0.061 |
| | TAGA-rw | 0.471 ± 0.031 | **0.276 ± 0.053** | 0.508 ± 0.019 | **0.764 ± 0.027** | 0.621 ± 0.076 | **0.594 ± 0.025** | **0.684 ± 0.027** | **0.675 ± 0.070** |
| 10 | PLM | 0.526 ± 0.013 | 0.240 ± 0.018 | 0.463 ± 0.029 | 0.690 ± 0.017 | 0.639 ± 0.038 | 0.491 ± 0.028 | 0.679 ± 0.023 | 0.535 ± 0.038 |
| | GraphMAE | 0.461 ± 0.017 | 0.234 ± 0.014 | 0.511 ± 0.028 | 0.761 ± 0.023 | 0.535 ± 0.042 | 0.543 ± 0.035 | 0.659 ± 0.028 | 0.508 ± 0.028 |
| | GraphCL | 0.301 ± 0.018 | 0.233 ± 0.029 | 0.488 ± 0.031 | 0.702 ± 0.025 | 0.566 ± 0.043 | 0.523 ± 0.044 | 0.632 ± 0.025 | 0.531 ± 0.035 |
| | GRACE | 0.488 ± 0.018 | 0.251 ± 0.015 | 0.552 ± 0.028 | 0.754 ± 0.018 | 0.567 ± 0.054 | 0.567 ± 0.031 | 0.670 ± 0.025 | 0.529 ± 0.033 |
| | G2P2 | **0.527 ± 0.014** | 0.269 ± 0.018 | 0.598 ± 0.031 | 0.753 ± 0.020 | 0.649 ± 0.046 | 0.632 ± 0.037 | 0.691 ± 0.029 | 0.618 ± 0.037 |
| | TAGA | 0.521 ± 0.017 | 0.288 ± 0.025 | **0.622 ± 0.025** | 0.788 ± 0.021 | **0.679 ± 0.041** | **0.651 ± 0.048** | **0.714 ± 0.024** | **0.705 ± 0.045** |
| | TAGA-rw | 0.518 ± 0.010 | **0.288 ± 0.040** | 0.595 ± 0.024 | **0.806 ± 0.011** | 0.652 ± 0.046 | 0.626 ± 0.020 | 0.679 ± 0.013 | 0.662 ± 0.056 |
| 100 | PLM | 0.592 ± 0.005 | 0.337 ± 0.013 | 0.610 ± 0.008 | 0.753 ± 0.014 | 0.753 ± 0.008 | 0.634 ± 0.015 | 0.771 ± 0.005 | 0.690 ± 0.013 |
| | GraphMAE | 0.573 ± 0.005 | 0.319 ± 0.008 | 0.650 ± 0.008 | 0.835 ± 0.007 | 0.684 ± 0.011 | 0.655 ± 0.012 | 0.744 ± 0.010 | 0.677 ± 0.009 |
| | GraphCL | 0.435 ± 0.005 | 0.313 ± 0.024 | 0.629 ± 0.006 | 0.804 ± 0.014 | 0.675 ± 0.026 | 0.653 ± 0.012 | 0.737 ± 0.007 | 0.703 ± 0.016 |
| | GRACE | 0.579 ± 0.007 | 0.339 ± 0.009 | 0.681 ± 0.006 | 0.838 ± 0.008 | 0.725 ± 0.014 | 0.678 ± 0.010 | 0.753 ± 0.010 | 0.712 ± 0.014 |
| | G2P2 | 0.578 ± 0.007 | 0.360 ± 0.009 | 0.711 ± 0.007 | 0.838 ± 0.010 | 0.748 ± 0.009 | 0.710 ± 0.008 | 0.758 ± 0.009 | 0.725 ± 0.010 |
| | TAGA | **0.631 ± 0.008** | 0.375 ± 0.021 | **0.731 ± 0.006** | 0.849 ± 0.008 | **0.754 ± 0.022** | **0.738 ± 0.015** | **0.787 ± 0.007** | **0.802 ± 0.014** |
| | TAGA-rw | 0.595 ± 0.010 | **0.385 ± 0.016** | 0.704 ± 0.010 | **0.853 ± 0.005** | 0.749 ± 0.023 | 0.716 ± 0.010 | 0.776 ± 0.011 | 0.767 ± 0.021 |

Table B.2: Full table of performance in zero-shot and few-shot node classification for each dataset and setting. The best-performing model is highlighted in **bold**, and the second-best performing model is underlined.

### B.1.3 Additional Node Classification Analysis

We present additional zero-shot and few-shot performance under two different text encoders `UAE-Large-V1` and `Text-embedding-3-small`. The zero-shot results are present in Table B.3. The few-shot results with text encoder `UAE-Large-V1` is present in Table B.4, and few-shot results with text encoder`Text-embedding-3-small` is present in Table B.5. From the table, we can observe that our method TAGA consistently achieve the best performance on two different choices of text encoder models. This demonstrates the effectiveness and robustness of our proposed method.

| Text Encoder | Model | arxiv | children | computers | cora | history | photo | pubmed | sports |
|---|---|---|---|---|---|---|---|---|---|
| UAE-Large-V1 | PLM | 0.500 ± 0.001 | 0.094 ± 0.003 | 0.427 ± 0.001 | 0.624 ± 0.005 | 0.169 ± 0.001 | 0.387 ± 0.009 | 0.475 ± 0.008 | 0.316 ± 0.002 |
| | GraphMAE | 0.104 ± 0.001 | 0.021 ± 0.001 | 0.049 ± 0.001 | 0.194 ± 0.006 | 0.019 ± 0.001 | 0.152 ± 0.001 | 0.438 ± 0.001 | 0.112 ± 0.001 |
| | GraphCL | 0.089 ± 0.001 | 0.037 ± 0.001 | 0.173 ± 0.001 | 0.176 ± 0.003 | 0.191 ± 0.001 | 0.174 ± 0.001 | 0.368 ± 0.001 | 0.140 ± 0.001 |
| | GRACE | 0.045 ± 0.001 | 0.034 ± 0.001 | 0.169 ± 0.001 | 0.146 ± 0.004 | 0.079 ± 0.001 | 0.025 ± 0.001 | 0.335 ± 0.001 | 0.057 ± 0.001 |
| | G2P2 | 0.453 ± 0.002 | 0.201 ± 0.001 | 0.453 ± 0.001 | 0.644 ± 0.004 | 0.322 ± 0.003 | 0.452 ± 0.001 | 0.576 ± 0.006 | 0.436 ± 0.001 |
| | TAGA | 0.537 ± 0.003 | 0.224 ± 0.001 | 0.498 ± 0.004 | 0.682 ± 0.005 | 0.351 ± 0.009 | 0.419 ± 0.001 | 0.616 ± 0.009 | 0.448 ± 0.003 |
| Text-embedding-3-small | PLM | 0.351 ± 0.001 | 0.098 ± 0.002 | 0.434 ± 0.005 | 0.561 ± 0.006 | 0.125 ± 0.001 | 0.321 ± 0.001 | 0.306 ± 0.001 | 0.424 ± 0.002 |
| | GraphMAE | 0.101 ± 0.001 | 0.025 ± 0.001 | 0.108 ± 0.001 | 0.162 ± 0.003 | 0.158 ± 0.001 | 0.033 ± 0.001 | 0.205 ± 0.001 | 0.364 ± 0.001 |
| | GraphCL | 0.127 ± 0.001 | 0.045 ± 0.001 | 0.282 ± 0.001 | 0.197 ± 0.004 | 0.106 ± 0.001 | 0.163 ± 0.001 | 0.383 ± 0.001 | 0.240 ± 0.003 |
| | GRACE | 0.023 ± 0.001 | 0.022 ± 0.001 | 0.117 ± 0.001 | 0.085 ± 0.004 | 0.039 ± 0.001 | 0.037 ± 0.001 | 0.319 ± 0.001 | 0.088 ± 0.001 |
| | G2P2 | 0.332 ± 0.001 | 0.092 ± 0.001 | 0.449 ± 0.001 | 0.637 ± 0.006 | 0.168 ± 0.001 | 0.298 ± 0.001 | 0.569 ± 0.001 | 0.511 ± 0.003 |
| | TAGA | 0.369 ± 0.001 | 0.084 ± 0.001 | 0.615 ± 0.001 | 0.668 ± 0.005 | 0.264 ± 0.001 | 0.423 ± 0.001 | 0.639 ± 0.001 | 0.548 ± 0.003 |

Table B.3: Zero-shot node classification performance.

### B.1.4 Additional Ablation Studies

Here we have included an ablation analysis to verify the effectiveness of neighborhood size. The results in Table B.6 demonstrate that our method achieves stable performance when using a neighborhood size of 2 or more orders.

## B.2 Additional Technical Details

**Efficiency Comparison with Directly Using PLM Embeddings.** It is worth noting that the textual embeddings of *TofG* views $\mathbf{h}(v_i)$ can directly represent the entire TAG. However, it may cause significant scalability and efficiency issue during the inference phase. Existing PLMs typically adopts transformer architecture and it has a quadratic complexity with the input number of text tokens, this is especially important to TAGs since the number of input size grows exponentially with the number of neighborhood hops. By aligning the knowledge from PLM with GNN model through our framework, we can simultaneously maintain generalization ability of TAG embeddings and high efficiency and scalability to large-sized graphs.

**Enabling Zero-Shot and Few-Shot Predictions.** Our pretrained strategy ensures that the embeddings obtained from the GNN models at each layer remain aligned within the textual embedding space. This alignment enables direct zero-shot predictions using the self-supervised trained embeddings without requiring any additional fine-tuning.

| $k$-Shot | Model | Arxiv | Children | Computers | Cora | History | Photo | Pubmed | Sports |
|---|---|---|---|---|---|---|---|---|---|
| 1 | PLM | 0.280 ± 0.044 | 0.122 ± 0.042 | 0.238 ± 0.039 | 0.412 ± 0.080 | 0.284 ± 0.078 | 0.230 ± 0.051 | 0.503 ± 0.067 | 0.282 ± 0.068 |
| | GraphMAE | 0.255 ± 0.041 | 0.128 ± 0.028 | 0.300 ± 0.052 | 0.474 ± 0.058 | 0.231 ± 0.052 | 0.304 ± 0.066 | 0.492 ± 0.076 | 0.270 ± 0.042 |
| | GRACE | 0.263 ± 0.034 | 0.138 ± 0.035 | 0.336 ± 0.051 | 0.435 ± 0.071 | 0.266 ± 0.085 | 0.295 ± 0.053 | 0.514 ± 0.095 | 0.282 ± 0.045 |
| | G2P2 | 0.308 ± 0.052 | 0.145 ± 0.029 | 0.359 ± 0.044 | 0.477 ± 0.082 | 0.361 ± 0.092 | 0.372 ± 0.066 | 0.522 ± 0.085 | 0.356 ± 0.042 |
| | TAGA | 0.323 ± 0.040 | 0.180 ± 0.073 | 0.380 ± 0.062 | 0.509 ± 0.089 | 0.413 ± 0.114 | 0.417 ± 0.077 | 0.563 ± 0.062 | 0.440 ± 0.070 |
| 3 | PLM | 0.436 ± 0.036 | 0.194 ± 0.029 | 0.318 ± 0.038 | 0.588 ± 0.036 | 0.448 ± 0.071 | 0.352 ± 0.044 | 0.611 ± 0.051 | 0.392 ± 0.041 |
| | GraphMAE | 0.379 ± 0.039 | 0.182 ± 0.025 | 0.389 ± 0.035 | 0.634 ± 0.044 | 0.362 ± 0.050 | 0.432 ± 0.051 | 0.597 ± 0.061 | 0.363 ± 0.050 |
| | GRACE | 0.398 ± 0.031 | 0.200 ± 0.038 | 0.442 ± 0.045 | 0.622 ± 0.043 | 0.404 ± 0.057 | 0.447 ± 0.053 | 0.620 ± 0.055 | 0.398 ± 0.045 |
| | G2P2 | 0.430 ± 0.027 | 0.207 ± 0.038 | 0.469 ± 0.042 | 0.623 ± 0.033 | 0.508 ± 0.073 | 0.528 ± 0.049 | 0.641 ± 0.064 | 0.464 ± 0.050 |
| | TAGA | 0.445 ± 0.035 | 0.241 ± 0.062 | 0.497 ± 0.035 | 0.695 ± 0.050 | 0.551 ± 0.094 | 0.551 ± 0.045 | 0.659 ± 0.058 | 0.586 ± 0.057 |
| 5 | PLM | 0.500 ± 0.019 | 0.210 ± 0.025 | 0.377 ± 0.027 | 0.641 ± 0.031 | 0.557 ± 0.040 | 0.420 ± 0.037 | 0.632 ± 0.040 | 0.478 ± 0.056 |
| | GraphMAE | 0.425 ± 0.028 | 0.212 ± 0.029 | 0.434 ± 0.036 | 0.704 ± 0.038 | 0.459 ± 0.038 | 0.489 ± 0.038 | 0.625 ± 0.049 | 0.452 ± 0.037 |
| | GRACE | 0.445 ± 0.028 | 0.227 ± 0.031 | 0.472 ± 0.040 | 0.685 ± 0.027 | 0.481 ± 0.061 | 0.515 ± 0.042 | 0.628 ± 0.047 | 0.482 ± 0.040 |
| | G2P2 | 0.466 ± 0.025 | 0.240 ± 0.034 | 0.510 ± 0.039 | 0.703 ± 0.032 | 0.617 ± 0.053 | 0.583 ± 0.051 | 0.640 ± 0.051 | 0.565 ± 0.055 |
| | TAGA | 0.483 ± 0.022 | 0.263 ± 0.031 | 0.543 ± 0.038 | 0.752 ± 0.028 | 0.636 ± 0.046 | 0.602 ± 0.041 | 0.649 ± 0.044 | 0.664 ± 0.061 |
| 10 | PLM | 0.526 ± 0.013 | 0.240 ± 0.018 | 0.463 ± 0.029 | 0.690 ± 0.017 | 0.639 ± 0.038 | 0.491 ± 0.028 | 0.679 ± 0.023 | 0.535 ± 0.038 |
| | GraphMAE | 0.461 ± 0.017 | 0.234 ± 0.014 | 0.511 ± 0.028 | 0.761 ± 0.023 | 0.535 ± 0.042 | 0.543 ± 0.035 | 0.659 ± 0.028 | 0.508 ± 0.028 |
| | GRACE | 0.488 ± 0.018 | 0.251 ± 0.015 | 0.552 ± 0.028 | 0.754 ± 0.018 | 0.567 ± 0.054 | 0.567 ± 0.031 | 0.670 ± 0.025 | 0.529 ± 0.033 |
| | G2P2 | 0.527 ± 0.014 | 0.269 ± 0.018 | 0.598 ± 0.031 | 0.753 ± 0.020 | 0.649 ± 0.046 | 0.632 ± 0.037 | 0.691 ± 0.029 | 0.618 ± 0.037 |
| | TAGA | 0.521 ± 0.017 | 0.288 ± 0.025 | 0.622 ± 0.025 | 0.788 ± 0.021 | 0.679 ± 0.041 | 0.651 ± 0.048 | 0.714 ± 0.024 | 0.705 ± 0.045 |
| 20 | PLM | 0.526 ± 0.013 | 0.240 ± 0.018 | 0.463 ± 0.029 | 0.690 ± 0.017 | 0.639 ± 0.038 | 0.491 ± 0.028 | 0.679 ± 0.023 | 0.535 ± 0.038 |
| | GraphMAE | 0.501 ± 0.009 | 0.264 ± 0.013 | 0.558 ± 0.015 | 0.801 ± 0.014 | 0.597 ± 0.033 | 0.596 ± 0.016 | 0.689 ± 0.021 | 0.572 ± 0.025 |
| | GRACE | 0.521 ± 0.011 | 0.277 ± 0.013 | 0.605 ± 0.017 | 0.791 ± 0.017 | 0.640± 0.037 | 0.615 ± 0.02 | 0.704 ± 0.029 | 0.607 ± 0.027 |
| | G2P2 | 0.556 ± 0.010 | 0.301 ± 0.015 | 0.649 ± 0.015 | 0.813 ± 0.012 | 0.716 ± 0.025 | 0.672 ± 0.015 | 0.726 ± 0.025 | 0.690 ± 0.025 |
| | TAGA | 0.561 ± 0.010 | 0.319 ± 0.023 | 0.673 ± 0.014 | 0.814 ± 0.012 | 0.721 ± 0.035 | 0.694 ± 0.021 | 0.745 ± 0.022 | 0.759 ± 0.026 |
| 50 | PLM | 0.526 ± 0.013 | 0.240 ± 0.018 | 0.463 ± 0.029 | 0.690 ± 0.017 | 0.639 ± 0.038 | 0.491 ± 0.028 | 0.679 ± 0.023 | 0.535 ± 0.038 |
| | GraphMAE | 0.541 ± 0.007 | 0.300± 0.010 | 0.612 ± 0.015 | 0.815 ± 0.008 | 0.657 ± 0.012 | 0.631 ± 0.010 | 0.729 ± 0.011 | 0.631 ± 0.018 |
| | GRACE | 0.553 ± 0.007 | 0.314 ± 0.012 | 0.649 ± 0.012 | 0.818 ± 0.012 | 0.706 ± 0.017 | 0.661 ± 0.019 | 0.732 ± 0.014 | 0.678 ± 0.022 |
| | G2P2 | 0.578 ± 0.009 | 0.340 ± 0.011 | 0.692 ± 0.012 | 0.827 ± 0.013 | 0.738 ± 0.009 | 0.700 ± 0.014 | 0.758 ± 0.009 | 0.725 ± 0.014 |
| | TAGA | 0.586 ± 0.010 | 0.348 ± 0.015 | 0.712 ± 0.012 | 0.836 ± 0.010 | 0.743 ± 0.022 | 0.715 ± 0.016 | 0.771 ± 0.011 | 0.784 ± 0.016 |
| 100 | PLM | 0.592 ± 0.005 | 0.337 ± 0.013 | 0.610 ± 0.008 | 0.753 ± 0.014 | 0.753 ± 0.008 | 0.634 ± 0.015 | 0.771 ± 0.005 | 0.690 ± 0.013 |
| | GraphMAE | 0.573 ± 0.005 | 0.319 ± 0.008 | 0.650 ± 0.008 | 0.835 ± 0.007 | 0.684 ± 0.011 | 0.655 ± 0.012 | 0.744 ± 0.010 | 0.677 ± 0.009 |
| | GRACE | 0.579 ± 0.007 | 0.339 ± 0.009 | 0.681 ± 0.006 | 0.838 ± 0.008 | 0.725 ± 0.014 | 0.678 ± 0.010 | 0.753 ± 0.010 | 0.712 ± 0.014 |
| | G2P2 | 0.578 ± 0.007 | 0.360 ± 0.009 | 0.711 ± 0.007 | 0.838 ± 0.010 | 0.748 ± 0.009 | 0.710 ± 0.008 | 0.758 ± 0.009 | 0.725 ± 0.010 |
| | TAGA | 0.631 ± 0.008 | 0.375 ± 0.021 | 0.731 ± 0.006 | 0.849 ± 0.008 | 0.754 ± 0.022 | 0.738 ± 0.015 | 0.787 ± 0.007 | 0.802 ± 0.014 |

Table B.4: Performance of all few-shot node classification for each dataset. The text encoder choice is `UAE-Large-V1`.

Specifically, suppose there are $L$ prediction labels $\{l_1, l_2, \ldots, l_L\}$. Their textual embeddings are obtained through the pretrained language model (PLM) as follows:

$$h^{(l)}(l_i) = \text{PLM}(l_i) \quad \text{for } i \in \{1, \ldots, L\} \tag{B.1}$$

The probability that node $v_i$ belongs to class $l_j$ is computed in an unsupervised manner by measuring the cosine similarity (or another appropriate similarity measure) between the learned GNN embeddings $h^{(g)}(v_i)$ and the label textual embeddings $h^{(l)}(l_j)$:

$$p(v_i \to l_j) = \frac{e^{\rho(h^{(g)}(v_i), h^{(l)}(l_j))}}{\sum_{k=1}^{L} e^{\rho(h^{(g)}(v_i), h^{(l)}(l_k))}} \tag{B.2}$$

| $k$-Shot | Model | Arxiv | Children | Computers | Cora | History | Photo | Pubmed | Sports |
|---|---|---|---|---|---|---|---|---|---|
| 1 | PLM | 0.199 ± 0.044 | 0.106 ± 0.025 | 0.347 ± 0.084 | 0.486 ± 0.095 | 0.285 ± 0.108 | 0.339 ± 0.055 | 0.491 ± 0.066 | 0.443 ± 0.098 |
| | GraphMAE | 0.167 ± 0.041 | 0.112 ± 0.052 | 0.257 ± 0.037 | 0.447 ± 0.095 | 0.268 ± 0.063 | 0.263 ± 0.080 | 0.456 ± 0.069 | 0.331 ± 0.090 |
| | GRACE | 0.224 ± 0.038 | 0.136 ± 0.034 | 0.329 ± 0.046 | 0.403 ± 0.067 | 0.304 ± 0.096 | 0.312 ± 0.049 | 0.513 ± 0.086 | 0.287 ± 0.039 |
| | G2P2 | 0.308 ± 0.052 | 0.145 ± 0.029 | 0.359 ± 0.044 | 0.477 ± 0.082 | 0.361 ± 0.092 | 0.372 ± 0.066 | 0.522 ± 0.085 | 0.356 ± 0.042 |
| | TAGA | 0.306 ± 0.057 | 0.173 ± 0.072 | 0.430 ± 0.067 | 0.523 ± 0.101 | 0.395 ± 0.101 | 0.431 ± 0.083 | 0.581 ± 0.073 | 0.510 ± 0.099 |
| 3 | PLM | 0.322 ± 0.046 | 0.148 ± 0.024 | 0.495 ± 0.061 | 0.66 ± 0.037 | 0.422 ± 0.075 | 0.438 ± 0.044 | 0.608 ± 0.033 | 0.577 ± 0.082 |
| | GraphMAE | 0.276 ± 0.033 | 0.169 ± 0.051 | 0.339 ± 0.038 | 0.657 ± 0.038 | 0.425 ± 0.097 | 0.347 ± 0.048 | 0.553 ± 0.060 | 0.398 ± 0.064 |
| | GRACE | 0.360 ± 0.030 | 0.191 ± 0.037 | 0.455 ± 0.045 | 0.580 ± 0.041 | 0.448 ± 0.067 | 0.461 ± 0.045 | 0.623 ± 0.064 | 0.426 ± 0.045 |
| | G2P2 | 0.430 ± 0.027 | 0.207 ± 0.038 | 0.469 ± 0.042 | 0.623 ± 0.033 | 0.508 ± 0.073 | 0.528 ± 0.049 | 0.641 ± 0.064 | 0.464 ± 0.050 |
| | TAGA | 0.442 ± 0.023 | 0.248 ± 0.052 | 0.548 ± 0.058 | 0.702 ± 0.032 | 0.523 ± 0.08 | 0.575 ± 0.047 | 0.683 ± 0.056 | 0.67 ± 0.062 |
| 5 | PLM | 0.365 ± 0.037 | 0.174 ± 0.039 | 0.55 ± 0.036 | 0.705 ± 0.02 | 0.522 ± 0.094 | 0.502 ± 0.039 | 0.601 ± 0.032 | 0.67 ± 0.05 |
| | GraphMAE | 0.308 ± 0.030 | 0.196 ± 0.059 | 0.384 ± 0.026 | 0.711 ± 0.030 | 0.511 ± 0.058 | 0.412 ± 0.032 | 0.563 ± 0.068 | 0.484 ± 0.038 |
| | GRACE | 0.399 ± 0.026 | 0.223 ± 0.028 | 0.501 ± 0.043 | 0.635 ± 0.028 | 0.513 ± 0.051 | 0.527 ± 0.040 | 0.640 ± 0.052 | 0.521 ± 0.049 |
| | G2P2 | 0.466 ± 0.025 | 0.240 ± 0.034 | 0.510 ± 0.039 | 0.703 ± 0.032 | 0.617 ± 0.053 | 0.583 ± 0.051 | 0.640 ± 0.051 | 0.565 ± 0.055 |
| | TAGA | 0.468 ± 0.023 | 0.299 ± 0.034 | 0.584 ± 0.04 | 0.74 ± 0.031 | 0.618 ± 0.067 | 0.6 ± 0.041 | 0.676 ± 0.048 | 0.735 ± 0.063 |
| 10 | PLM | 0.398 ± 0.024 | 0.189 ± 0.026 | 0.627 ± 0.025 | 0.741 ± 0.018 | 0.586 ± 0.056 | 0.541 ± 0.022 | 0.667 ± 0.025 | 0.708 ± 0.039 |
| | GraphMAE | 0.375 ± 0.017 | 0.208 ± 0.011 | 0.469 ± 0.029 | 0.763 ± 0.027 | 0.564 ± 0.047 | 0.491 ± 0.034 | 0.613 ± 0.034 | 0.539 ± 0.028 |
| | GRACE | 0.449 ± 0.018 | 0.249 ± 0.019 | 0.577 ± 0.027 | 0.714 ± 0.023 | 0.601 ± 0.047 | 0.578 ± 0.030 | 0.682 ± 0.025 | 0.569 ± 0.039 |
| | G2P2 | 0.527 ± 0.014 | 0.269 ± 0.018 | 0.598 ± 0.031 | 0.753 ± 0.020 | 0.649 ± 0.046 | 0.632 ± 0.037 | 0.691 ± 0.029 | 0.618 ± 0.037 |
| | TAGA | 0.509 ± 0.020 | 0.315 ± 0.028 | 0.661 ± 0.028 | 0.781 ± 0.018 | 0.67 ± 0.049 | 0.646 ± 0.033 | 0.724 ± 0.022 | 0.756 ± 0.032 |
| 20 | PLM | 0.434 ± 0.016 | 0.223 ± 0.032 | 0.659 ± 0.014 | 0.767 ± 0.015 | 0.641 ± 0.04 | 0.581 ± 0.015 | 0.712 ± 0.021 | 0.761 ± 0.026 |
| | GraphMAE | 0.429 ± 0.011 | 0.236 ± 0.020 | 0.535 ± 0.023 | 0.799 ± 0.014 | 0.625 ± 0.024 | 0.559 ± 0.017 | 0.655 ± 0.030 | 0.602 ± 0.028 |
| | GRACE | 0.486 ± 0.014 | 0.282 ± 0.015 | 0.613 ± 0.019 | 0.770 ± 0.017 | 0.654 ± 0.027 | 0.629 ± 0.016 | 0.697 ± 0.022 | 0.657 ± 0.025 |
| | G2P2 | 0.556 ± 0.010 | 0.301 ± 0.015 | 0.649 ± 0.015 | 0.813 ± 0.012 | 0.716 ± 0.025 | 0.672 ± 0.015 | 0.726 ± 0.025 | 0.690 ± 0.025 |
| | TAGA | 0.547 ± 0.010 | 0.332 ± 0.023 | 0.691 ± 0.017 | 0.805 ± 0.011 | 0.708 ± 0.039 | 0.682 ± 0.015 | 0.745 ± 0.027 | 0.808 ± 0.022 |
| 50 | PLM | 0.480 ± 0.007 | 0.252 ± 0.022 | 0.695 ± 0.010 | 0.785 ± 0.009 | 0.702 ± 0.02 | 0.609 ± 0.013 | 0.749 ± 0.011 | 0.784 ± 0.014 |
| | GraphMAE | 0.477 ± 0.010 | 0.278 ± 0.012 | 0.603 ± 0.012 | 0.819 ± 0.011 | 0.675 ± 0.019 | 0.630 ± 0.015 | 0.692 ± 0.016 | 0.673 ± 0.021 |
| | GRACE | 0.520 ± 0.006 | 0.324 ± 0.012 | 0.664 ± 0.013 | 0.806 ± 0.014 | 0.694 ± 0.022 | 0.668 ± 0.020 | 0.727 ± 0.015 | 0.712 ± 0.020 |
| | G2P2 | 0.578 ± 0.009 | 0.340 ± 0.011 | 0.692 ± 0.012 | 0.827 ± 0.013 | 0.738 ± 0.009 | 0.700 ± 0.014 | 0.758 ± 0.009 | 0.725 ± 0.014 |
| | TAGA | 0.576 ± 0.009 | 0.368 ± 0.014 | 0.734 ± 0.007 | 0.826 ± 0.009 | 0.738 ± 0.021 | 0.717 ± 0.016 | 0.773 ± 0.009 | 0.828 ± 0.014 |
| 100 | PLM | 0.508 ± 0.005 | 0.272 ± 0.010 | 0.722 ± 0.007 | 0.800 ± 0.014 | 0.73 ± 0.015 | 0.629 ± 0.009 | 0.772 ± 0.008 | 0.802 ± 0.006 |
| | GraphMAE | 0.499 ± 0.008 | 0.298 ± 0.014 | 0.634 ± 0.008 | 0.844 ± 0.010 | 0.704 ± 0.015 | 0.652 ± 0.017 | 0.721 ± 0.007 | 0.709 ± 0.011 |
| | GRACE | 0.546 ± 0.007 | 0.344 ± 0.008 | 0.693 ± 0.006 | 0.823 ± 0.013 | 0.714 ± 0.011 | 0.688 ± 0.011 | 0.745 ± 0.006 | 0.753 ± 0.010 |
| | G2P2 | 0.578 ± 0.007 | 0.360 ± 0.009 | 0.711 ± 0.007 | 0.838 ± 0.010 | 0.748 ± 0.009 | 0.710 ± 0.008 | 0.758 ± 0.009 | 0.725 ± 0.010 |
| | TAGA | 0.602 ± 0.007 | 0.400 ± 0.017 | 0.747 ± 0.009 | 0.838 ± 0.009 | 0.755 ± 0.017 | 0.738 ± 0.010 | 0.786 ± 0.006 | 0.846 ± 0.013 |

Table B.5: Performance of all few-shot node classification for each dataset. The text encoder choice is `Text-embedding-3-small`.

The final predicted class of node $v_i$ is determined as follows:

$$l(v_i) = \text{argmax}_j \, p(v_i \to l_j) \tag{B.3}$$

where $l(v_i)$ is the predicted class label for node $v_i$, determined by selecting the class $l$ that maximizes the similarity measure $\rho$ between the GNN embedding of the node $h^{(g)}(v_i)$ and each of the label embeddings $h^{(l)}(l_j)$.

Additionally, to further refine the learned embeddings, we introduce a learnable transformation function for few-shot learning adaptation:

$$h^{(g)}_{\text{adapted}}(v_i) = g(h^{(g)}(v_i), \mathcal{D}_{\text{support}}) \tag{B.4}$$

| Method | arxiv | children | computers | cora | history | photo | pubmed | sports |
|---|---|---|---|---|---|---|---|---|
| 3-order | 0.532 | 0.223 | 0.493 | 0.678 | 0.351 | 0.415 | 0.622 | 0.387 |
| 2-order | 0.537 | 0.224 | 0.498 | 0.682 | 0.344 | 0.419 | 0.616 | 0.408 |
| 1-order | 0.500 | 0.197 | 0.463 | 0.635 | 0.318 | 0.392 | 0.566 | 0.448 |
| Glo-GofT | 0.533 | 0.205 | 0.482 | 0.657 | 0.329 | 0.407 | 0.522 | 0.417 |

Table B.6: Additional ablation studies results of zero-shot settings. Here we show the results with different orders of alignment at 1, 2 and 3 order. We also show the results of a variant, *Glo-GofT*, which only aligns the GNN embeddings that aggregate individual node's text embeddings but removes all higher-order Graph-of-Text embeddings.

where $g$ represents a transformation function with learnable parameters (e.g., a multi-layer perceptron), and $\mathcal{D}_{\text{support}}$ denotes a set of support examples for few-shot learning. This adapted embedding $h^{(g)}_{\text{adapted}}$ is then utilized to compute the updated predictive probabilities:

$$p(v_i \to l_j) = \frac{e^{\rho(h^{(g)}_{\text{adapted}}(v_i), h^{(l)}(l_j))}}{\sum_{k=1}^{L} e^{\rho(h^{(g)}_{\text{adapted}}(v_i), h^{(l)}(l_k))}} \tag{B.5}$$

## B.3 Limitations

This work aims to pioneer unsupervised representation learning in the text-attributed graph research domain. Our approach demonstrates significant performance improvements over existing state-of-the-art methods in zero-shot and few-shot prediction tasks. However, we acknowledge certain limitations. While our work pushes the boundaries of graph foundation models, the model's transfer capabilities may be limited when training and inference domains are vastly different (e.g., from social networks to chemical networks). We consider the development of a universal graph foundation model, capable of generalizing across diverse domains, to be an important direction for future research.

# Appendix C

# Enhancing Generalizability and Robustness of Learning Network Representations

## C.1 Mathematical Proof for Theorem 6

*Proof.* We prove this theorem by Theorem 10 and Corollary 3 from [126].

[**Avoidance of Saddle Points**] Because the sequence $(\mathbf{w}^{(t)}, \mathbf{S}^{(t)})$ is bounded, and the second derivatives of $L$ and $g$ are continuous, then they are bounded. In other words, we have $\max\{\left\|\nabla^2_{\mathbf{w}} L(f(\mathbf{X}, \mathbf{A}^{(t)}; \mathbf{w}^{(t)}), \mathbf{y})\right\|, \left\|\nabla^2_{\mathbf{S}} g(S^{(t)}; \lambda)\right\|\} \leq p$, where $p > 0$ is a constant. Similarly, it is easy to check that the second derivative of the term $\sum_{i,j} \mathbf{S}_{ij} \left\|\tilde{\mathbf{A}}^{(t)}_{ij} - \mathbf{A}_{ij}\right\|_2^2$ is bounded, i.e.,

$$\max\{\left\|\nabla^2_{\mathbf{w}} \sum_{i,j} \mathbf{S}_{ij} \left\|\tilde{\mathbf{A}}^{(t)}_{ij} - \mathbf{A}_{ij}\right\|_2^2\right\|, \left\|\nabla^2_{\mathbf{S}} \sum_{i,j} \mathbf{S}_{ij} \left\|\tilde{\mathbf{A}}^{(t)}_{ij} - \mathbf{A}_{ij}\right\|_2^2\right\|\} \leq q,$$

where $q > 0$ is constant and $\tilde{\mathbf{A}}$ is a function of $\mathbf{w}$. Therefore, it means that the objective $F$ is bi-smooth, i.e. $\max\{\|\nabla^2_{\mathbf{w}} F\|\}, \|\nabla^2_{\mathbf{S}} F\|\} \leq p + q$. In other words, $F$

satisfies Assumption 4 from [126]. Moreover, the second derivative of $F$ is continuous. For any $\gamma > p + q$, any bounded sequence $(\mathbf{w}^{(t)}, \mathbf{S}^{(t)})$ generated by Algorithm 3 will not converge to a strict saddle of $F$ almost surely by Theorem 10 from [126].

**[Second Order Convergence]** From the above proof of avoidance of saddle points, we know that $F$ satisfies Assumption 4 from [126]. Moreover, because $L$ and $g$ satisfy the KL property, and the term $\sum_{i,j} \mathbf{S}_{ij} \left\| \tilde{\mathbf{A}}_{ij}^{(t)} - \mathbf{A}_{ij} \right\|_2^2$ satisfies the KL property, we conclude that $F$ satisfy the KL property as well. From the proof above, we also know that the second derivative of $F$ is continuous. Because continuous differentiability implies Lipschitz continuity [201], it infers that the first derivative of $F$ is Lipschitz continuous. As a result, $F$ satisfies Assumption 1 from [126]. Because $F$ satisfies Assumptions 1 and 4, then for any $\gamma > p + q$, any bounded sequence $(\mathbf{w}^{(t)}, \mathbf{S}^{(t)})$ generated by Algorithm 3 will almost surely converges to a second-order stationary point of $F$ by Corollary 3 from [126]. □

While the convergence of Algorithm 3 entails the second-order optimality conditions of $f$ and $g$, some commonly used $f$ such as the GNN with sigmoid or tanh activations and some commonly used $g$ such as the squared $\ell_2$ norm satisfy the KL property [185, 186], and Algorithm 3 is guaranteed to avoid a strict saddle point and converges to a second-order stationary point.

# Bibliography

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

[2] Masood A Akram, Sumit Nanda, Patricia Maraver, Rubén Armañanzas, and Giorgio A Ascoli. An open repository for single-cell reconstructions of the brain forest. *Scientific data*, 5(1):1–12, 2018.

[3] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[4] Giorgio A Ascoli, Duncan E Donohue, and Maryam Halavi. Neuromorpho. org: a central resource for neuronal morphologies. *Journal of Neuroscience*, 27(35): 9247–9251, 2007.

[5] Jayanth R Banavar, Amos Maritan, and Andrea Rinaldo. Size and form in efficient transportation networks. *Nature*, 399(6732):130–132, 1999.

[6] Albert-László Barabási. Network science. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1987): 20120375, 2013.

[7] Albert-László Barabási and Eric Bonabeau. Scale-free networks. *Scientific american*, 288(5):60–69, 2003.

[8] Albert-László Barabási. *Network Science*. Cambridge University Press, 2016.

[9] Alain Barrat, Marc Barthelemy, and Alessandro Vespignani. *Dynamical processes on complex networks*. Cambridge university press, 2008.

[10] Marc Barthélemy. Crossover from scale-free to spatial networks. *EPL (Europhysics Letters)*, 63(6):915, 2003.

[11] Marc Barthélemy. Spatial networks. *Physics Reports*, 499(1-3):1–101, 2011.

[12] Marc Barthelemy. *Morphogenesis of spatial networks*. Springer, 2018.

[13] Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2017.

[14] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[15] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[16] Kenneth A. Berman and Jerome L. Paul. *Graph Theory and Its Applications*. CRC Press, 1999.

[17] Dimitri P. Bertsekas. *Linear Network Optimization: Algorithms and Codes*. MIT Press, 1991.

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

[19] John Adrian Bondy and U.S.R. Murty. *Graph Theory*. Springer, 2008.

[20] Stephen P Borgatti and Daniel S Halgin. On network theory. *Organization science*, 22(5):1168–1181, 2011.

[21] Davide Boscaini, Jonathan Masci, Emanuele Rodoià, and Michael Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 3197–3205, 2016.

[22] Gary J Brierley and Kirstie A Fryirs. *Geomorphology and river management: applications of the river styles framework*. John Wiley & Sons, 2013.

[23] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[24] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[25] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Rad-

ford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020. URL `https://api.semanticscholar.org/CorpusID:218971783`.

[26] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and deep locally connected networks on graphs. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

[27] Robert Burton and Robin Pemantle. Local characteristics, entropy and limit theorems for spanning trees and domino tilings via transfer-impedances. *The Annals of Probability*, pages 1329–1371, 1993.

[28] Wenming Cao, Zhiyue Yan, Zhiquan He, and Zhihai He. A comprehensive survey on geometric deep learning. *IEEE Access*, 8:35929–35949, 2020.

[29] Seth Chaiken and Daniel J Kleitman. Matrix tree theorems. *Journal of combinatorial theory, Series A*, 24(3):377–381, 1978.

[30] Jianfei Chen, Jun Zhu, and Le Song. Stochastic training of graph convolutional networks with variance reduction. In *International Conference on Machine Learning*, pages 942–950. PMLR, 2018.

[31] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolu-tional networks via importance sampling. In *International Conference on Learning Representations*. International Conference on Learning Representations, ICLR, 2018.

[32] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

[33] Yu Chen, Lingfei Wu, and Mohammed Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in neural information processing systems*, 33:19314–19326, 2020.

[34] Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Longlora: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations*.

[35] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2):42–61, 2024.

[36] Richard J Chorley and Petercoed Haggett. Models in geography. Technical report, 1967.

[37] Gerardo Chowell, James M Hyman, Stephen Eubank, and Carlos Castillo-Chavez. Scaling laws for the movement of people between locations in a large city. *Physical Review E*, 68(6):066102, 2003.

[38] Guanyi Chu, Xiao Wang, Chuan Shi, and Xunqiang Jiang. Cuco: Graph representation with curriculum contrastive learning. In *IJCAI*, pages 2300–2306, 2021.

[39] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[40] Reuven Cohen and Shlomo Havlin. *Complex networks: structure, robustness and function*. Cambridge university press, 2010.

[41] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[42] Stephen A. Cook. An algorithm for testing planarity of a graph. *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 335–353, 1971.

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

[44] Camille Couprie, Clément Farabet, Laurent Najman, and Yann Lecun. Indoor semantic segmentation using depth information. In *First International Conference on Learning Representations (ICLR 2013)*, pages 1–8, 2013.

[45] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International conference on machine learning*, pages 1115–1124. PMLR, 2018.

[46] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for molecule generation. In *Proceedings of the International Conference on Learning Representations*, 2018.

[47] Tomasz Danel, Przemysław Spurek, Jacek Tabor, Marek Śmieja, Łukasz Struski, Agnieszka Słowik, and Łukasz Maziarka. Spatial graph convolutional networks. In *International Conference on Neural Information Processing*, pages 668–675. Springer, 2020.

[48] Andrea De Montis, Marc Barthélemy, Alessandro Chessa, and Alessandro Vespignani. The structure of interurban traffic: a weighted network analysis. *Environment and Planning B: Planning and Design*, 34(5):905–924, 2007.

[49] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv preprint arXiv:1606.09375*, 2016.

[50] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 195–205, 2018.

[51] Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*, 2023.

[52] Kaize Ding, Yancheng Wang, Yingzhen Yang, and Huan Liu. Eliciting structural and semantic global knowledge in unsupervised graph contrastive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7378–7386, 2023.

[53] Christopher M Dobson. Protein folding and misfolding. *Nature*, 426(6968): 884–890, 2003.

[54] Yuanqi Du, Shiyu Wang, Xiaojie Guo, Hengning Cao, Shujie Hu, Junji Jiang, Aishwarya Varala, Abhinav Angirekula, and Liang Zhao. Graphgt: Machine learning datasets for graph generation and transformation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[55] César Ducruet and Laurent Beauguitte. Spatial science and network science: review and outcomes of a complex relationship. *Networks and Spatial Economics*, 14(3):297–316, 2014.

[56] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional net-

works on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

[57] Victor M Eguiluz, Dante R Chialvo, Guillermo A Cecchi, Marwan Baliki, and A Vania Apkarian. Scale-free brain functional networks. *Physical review letters*, 94(1):018102, 2005.

[58] Yasha Ektefaie, George Dasoulas, Ayush Noori, Maha Farhat, and Marinka Zitnik. Multimodal learning with graphs. *Nature Machine Intelligence*, 5(4): 340–350, 2023.

[59] Jeffrey L Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.

[60] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 169–177, 2020.

[61] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[62] Paul Expert, Tim S Evans, Vincent D Blondel, and Renaud Lambiotte. Uncovering space-independent communities in spatial networks. *Proceedings of the National Academy of Sciences*, 108(19):7663–7668, 2011.

[63] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[64] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: En-

coding graphs for large language models. In *The Twelfth International Conference on Learning Representations*, 2023.

[65] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[66] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[67] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5): 75–174, 2010.

[68] Maximilian Franzke, Tobias Emrich, Andreas Züfle, and Matthias Renz. Pattern search in temporal social networks. In *Proceedings of the 21st International Conference on Extending Database Technology*, 2018.

[69] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in neural information processing systems*, 33:1970–1981, 2020.

[70] Johannes Gasteiger, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations*, 2020.

[71] Linda Geerligs, Remco J Renken, Emi Saliasi, Natasha M Maurits, and Monicque M Lorist. A brain-wide study of age-related changes in functional connectivity. *Cerebral cortex*, 25(7):1987–1999, 2015.

[72] Geological Survey (U.S.). *National Hydrography Dataset*. U.S. Dept. of the Interior, U.S. Geological Survey, Reston, Va., 2004.

[73] Alan Gibbons. *Algorithmic Graph Theory*. Cambridge University Press, 1985.

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

[75] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[76] Tieliang Gong, Qian Zhao, Deyu Meng, and Zongben Xu. Why curriculum learning & self-paced learning work in big/noisy data: A theoretical perspective. *Big Data & Information Analytics*, 1(1):111, 2016.

[77] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

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

[79] Xiaojie Guo, Yuanqi Du, and Liang Zhao. Property controllable variational autoencoder via invertible mutual dependence. In *International Conference on Learning Representations*, 2020.

[80] Xiaojie Guo, Yuanqi Du, and Liang Zhao. Deep generative models for spatial networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 505–515, 2021.

[81] Xiaojie Guo, Shiyu Wang, and Liang Zhao. Graph neural networks: Graph

transformation. *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 251–275, 2022.

[82] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[83] Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *International Conference on Machine Learning*, pages 2535–2544. PMLR, 2019.

[84] Peter Haggett and Richard J Chorley. *Network analysis in geography*, volume 1. Hodder Education, 1969.

[85] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[86] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[87] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems*, 31, 2018.

[88] Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. Lm-infinite: Simple on-the-fly length generalization for large language models. *arXiv preprint arXiv:2308.16137*, 2023.

[89] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and

Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[90] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[91] Wenchong He, Zhe Jiang, Chengming Zhang, and Arpan Man Sainju. Curvanet: Geometric deep learning based on directional curvature for 3d shape analysis. In *Proceedings of the 26th ACM SIGKDD*, pages 2214–2224, 2020.

[92] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[93] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 594–604, 2022.

[94] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[95] Yuntong Hu, Zheng Zhang, and Liang Zhao. Beyond text: A deep dive into large language models' ability on understanding graph data. In *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*.

[96] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, pages 2704–2710, 2020.

[97] Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. Can llms effectively leverage graph structural information: When and why. In *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*.

[98] Zan Huang, Wingyan Chung, and Hsinchun Chen. A graph model for e-commerce recommender systems. *Journal of the American Society for information science and technology*, 55(3):259–274, 2004.

[99] Mark Jenkinson, Christian F Beckmann, Timothy EJ Behrens, Mark W Woolrich, and Stephen M Smith. Fsl. *Neuroimage*, 62(2):782–790, 2012.

[100] Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *arXiv preprint arXiv:2310.06839*, 2023.

[101] Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. Self-paced learning with diversity. *Advances in neural information processing systems*, 27, 2014.

[102] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. Self-paced curriculum learning. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

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

[104] Bowen Jin, Wentao Zhang, Yu Zhang, Yu Meng, Xinyang Zhang, Qi Zhu, and Jiawei Han. Patton: Language model pretraining on text-rich networks. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

[105] Bowen Jin, Yu Zhang, Yu Meng, and Jiawei Han. Edgeformers: Graph-empowered transformers for representation learning on textual-edge networks. In *The Eleventh International Conference on Learning Representations,{ICLR} 2023*. OpenReview. net, 2023.

[106] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. Large language models on graphs: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[107] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 66–74, 2020.

[108] Benedikt Atli Jónsson, Gyda Bjornsdottir, TE Thorgeirsson, Lotta María Ellingsen, G Bragi Walters, DF Gudbjartsson, Hreinn Stefansson, Kari Stefansson, and MO Ulfarsson. Brain age prediction using deep learning uncovers associated sequence variants. *Nature communications*, 10(1):1–10, 2019.

[109] Wei Ju, Zheng Fang, Yiyang Gu, Zequn Liu, Qingqing Long, Ziyue Qiao, Yifang Qin, Jianhao Shen, Fang Sun, Zhiping Xiao, et al. A comprehensive survey on deep graph representation learning. *Neural Networks*, page 106207, 2024.

[110] Fariba Karimi, Mathieu Génois, Claudia Wagner, Philipp Singer, and Markus Strohmaier. Homophily influences ranking of minorities in social networks. *Scientific reports*, 8(1):1–12, 2018.

[111] Anees Kazi, Luca Cosmo, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael M Bronstein. Differentiable graph module (dgm) for graph convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1606–1617, 2022.

[112] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2. Minneapolis, Minnesota, 2019.

[113] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2. Minneapolis, Minnesota, 2019.

[114] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

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

[116] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[117] Yajing Kong, Liu Liu, Jun Wang, and Dacheng Tao. Adaptive curriculum learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5067–5076, 2021.

[118] Dirk Koschützki, Katharina A. Lehmann, Lars Peeters, Stephan Richter, Dieter Tenfelde-Podehl, and Oliver Zlotowski. Centrality indices. *Network Analysis: Methodological Foundations*, 3418:16–61, 2005.

[119] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[120] M Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. *Advances in neural information processing systems*, 23, 2010.

[121] Maciej Kurant and Patrick Thiran. Extraction and analysis of traffic and topologies of transportation networks. *Physical Review E*, 74(3):036114, 2006.

[122] Vito Latora and Massimo Marchiori. Vulnerability and protection of infrastructure networks. *Physical Review E*, 71(1):015103, 2005.

[123] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

[124] Haoyang Li, Xin Wang, and Wenwu Zhu. Curriculum graph machine learning: a survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 667–6682, 2023.

[125] Juanhui Li, Harry Shomer, Haitao Mao, Shenglai Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. Evaluating graph neural networks for link prediction: Current pitfalls and new benchmarking. *Advances in Neural Information Processing Systems*, 36, 2024.

[126] Qiuwei Li, Zhihui Zhu, and Gongguo Tang. Alternating minimizations converge to second-order optimal solutions. In *International Conference on Machine Learning*, pages 3935–3943. PMLR, 2019.

[127] Xianming Li and Jing Li. Angle-optimized text embeddings. *arXiv preprint arXiv:2309.12871*, 2023.

[128] Xiaohe Li, Lijie Wen, Yawen Deng, Fuli Feng, Xuming Hu, Lei Wang, and Zide Fan. Graph neural network with curriculum learning for imbalanced node classification. *arXiv preprint arXiv:2202.02529*, 2022.

[129] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen.

Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31:820–830, 2018.

[130] Yichuan Li, Kaize Ding, and Kyumin Lee. Grenade: Graph-centric language model for self-supervised representation learning on text-attributed graphs. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

[131] Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *Proceedings of ICLR'16*, 2016.

[132] Chen Ling, Junji Jiang, Junxiang Wang, and Zhao Liang. Source localization of graph diffusion via variational autoencoders for graph inverse problems. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 1010–1020, 2022.

[133] Chen Ling, Junji Jiang, Junxiang Wang, My T Thai, Renhao Xue, James Song, Meikang Qiu, and Liang Zhao. Deep graph representation learning and optimization for influence maximization. In *International Conference on Machine Learning*, pages 21350–21361. PMLR, 2023.

[134] Or Litany, Tal Remez, Emanuele Rodola, Alex Bronstein, and Michael Bronstein. Deep functional maps: Structured prediction for dense shape correspondence. In *Proceedings of the IEEE international conference on computer vision*, pages 5659–5667, 2017.

[135] Ao Liu, Qiong Wu, Zhenming Liu, and Lirong Xia. Near-neighbor methods in random preference completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[136] Xiaozhong Liu, Jinsong Zhang, and Chun Guo. Full-text citation analysis: A new method to enhance scholarly networks. *Journal of the American Society for Information Science and Technology*, 64(9):1852–1863, 2013.

[137] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[138] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 779–787, 2021.

[139] Diego Marcheggiani and Ivan Titov. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2017.

[140] Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans. Graph.*, 36(4):71–1, 2017.

[141] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019.

[142] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45, 2015.

[143] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.

[144] Mohamed Mokbel, Mahmoud Sakr, Li Xiong, Andreas Züfle, Jussara Almeida, Taylor Anderson, Walid Aref, Gennady Andrienko, Natalia Andrienko, Yang Cao, et al. Mobility data science: Perspectives and challenges. *ACM Transactions on Spatial Algorithms and Systems*.

[145] Mohamed Mokbel, Mahmoud Sakr, Li Xiong, Andreas Züfle, Jussara Almeida, Taylor Anderson, Walid Aref, Gennady Andrienko, Natalia Andrienko, Yang Cao, et al. Mobility data science (dagstuhl seminar 22021). In *Dagstuhl reports*, volume 12. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[146] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019.

[147] Richard G Morris and Marc Barthelemy. Transport on coupled spatial networks. *Physical review letters*, 109(12):128703, 2012.

[148] Seth A Myers, Aneesh Sharma, Pankaj Gupta, and Jimmy Lin. Information network or social network? the structure of the twitter follow graph. In *Proceedings of the 23rd international conference on world wide web*, pages 493–498, 2014.

[149] Mark EJ Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[150] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023. PMLR, 2016.

[151] OpenAI. Text-embedding-3-small model, 2023. URL `https://openai.com`.

[152] Bo Pan, Zheng Zhang, Yifei Zhang, Yuntong Hu, and Liang Zhao. Distilling large language models for text-attributed graph learning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 1836–1845, 2024.

[153] Dmitry Paranyushkin. Infranodus: Generating insight using text network analysis. In *The world wide web conference*, pages 3584–3589, 2019.

[154] Ruchi Parekh and Giorgio A Ascoli. Neuronal morphology goes digital: a research hub for cellular and system neuroscience. *Neuron*, 77(6):1017–1038, 2013.

[155] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[156] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*.

[157] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[158] Saulo DS Reis, Yanqing Hu, Andrés Babino, José S Andrade Jr, Santiago Canals, Mariano Sigman, and Hernán A Makse. Avoiding catastrophic failure in correlated networks of networks. *Nature Physics*, 10(10):762–767, 2014.

[159] Douglas LT Rohde and David C Plaut. Language acquisition in the absence of

explicit negative evidence: How important is starting small? *Cognition*, 72(1): 67–109, 1999.

[160] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in neural information processing systems*, 30, 2017.

[161] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769, 2016.

[162] David I Shuman, Benjamin Ricaud, and Pierre Vandergheynst. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis*, 40(2):260–291, 2016.

[163] Richard Socher, Brody Huval, Bharath Bath, Christopher D Manning, and Andrew Ng. Convolutional-recursive deep learning for 3d object classification. *Advances in neural information processing systems*, 25:656–664, 2012.

[164] Olaf Sporns, Giulio Tononi, and Gerald M Edelman. Theoretical neuroanatomy: relating anatomical and functional connectivity in graphs and cortical connection matrices. *Cerebral cortex*, 10(2):127–141, 2000.

[165] Kaustubh Supekar, Vinod Menon, Daniel Rubin, Mark Musen, and Michael D Greicius. Network analysis of intrinsic functional brain connectivity in alzheimer's disease. *PLoS Comput Biol*, 4(6):e1000100, 2008.

[166] Charles Sutton, Andrew McCallum, et al. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373, 2012.

[167] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, 2015.

[168] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 491–500, 2024.

[169] Jiliang Tang and Huan Liu. Unsupervised feature selection for linked social media data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 904–912, 2012.

[170] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[171] Waldo R Tobler. A computer movie simulating urban growth in the detroit region. *Economic geography*, 46(sup1):234–240, 1970.

[172] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[173] Dimitrios Tsiotas and Serafeim Polyzos. The complexity in the study of spatial

networks: an epistemological approach. *Networks and Spatial Economics*, 18 (1):1–32, 2018.

[174] U.S. Geological Survey. USGS TNM Hydrography (NHD). `https://apps.nationalmap.gov/services/`, 2019. Accessed: June 7, 2019.

[175] Martijn P Van Den Heuvel and Hilleke E Hulshoff Pol. Exploring the brain network: a review on resting-state fmri functional connectivity. *European neuropsychopharmacology*, 20(8):519–534, 2010.

[176] David C Van Essen, Stephen M Smith, Deanna M Barch, Timothy EJ Behrens, Essa Yacoub, Kamil Ugurbil, Wu-Minn HCP Consortium, et al. The wu-minn human connectome project: an overview. *Neuroimage*, 80:62–79, 2013.

[177] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. URL `https://api.semanticscholar.org/CorpusID:13756489`.

[178] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations*.

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

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

[181] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36, 2024.

[182] Jichuan Wang and Xiaoqian Wang. *Structural equation modeling: Applications using Mplus*. John Wiley & Sons, 2019.

[183] Junxiang Wang, Fuxun Yu, Xiang Chen, and Liang Zhao. Admm for efficient deep learning with global convergence. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 111–119, 2019.

[184] Junxiang Wang, Junji Jiang, and Liang Zhao. An invertible graph diffusion neural network for source localization. In *Proceedings of the ACM Web Conference 2022*, pages 1058–1069, 2022.

[185] Junxiang Wang, Hongyi Li, Zheng Chai, Yongchao Wang, Yue Cheng, and Liang Zhao. Toward quantized model parallelism for graph-augmented mlps based on gradient-free admm framework. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[186] Junxiang Wang, Hongyi Li, and Liang Zhao. Accelerated gradient-free neural network training by multi-convex alternating optimization. *Neurocomputing*, 487:130–143, 2022.

[187] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–67, 2018.

[188] Pengyang Wang, Yanjie Fu, Hui Xiong, and Xiaolin Li. Adversarial substructured representation learning for mobile user profiling. In *Proceedings of the*

*25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 130–138, 2019.

[189] Pengyang Wang, Kunpeng Liu, Lu Jiang, Xiaolin Li, and Yanjie Fu. Incremental mobile user profiling: Reinforcement learning with spatial knowledge graph for modeling event streams. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 853–861, 2020.

[190] Shiyu Wang, Xiaojie Guo, and Liang Zhao. Deep generative model for periodic graphs. *Advances in Neural Information Processing Systems*, 35, 2022.

[191] Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):4555–4576, 2021.

[192] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, and Bryan Hooi. Curgraph: Curriculum learning for graph classification. In *Proceedings of the Web Conference 2021*, pages 1238–1248, 2021.

[193] Zheng Wang, Ce Ju, Gao Cong, and Cheng Long. Representation learning for spatial graphs. *arXiv preprint arXiv:1812.06668*, 2018.

[194] Stanley Wasserman and Joseph Galaskiewicz. *Advances in social network analysis: Research in the social and behavioral sciences*. Sage Publications, 1994.

[195] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[196] Xiaowen Wei, Xiuwen Gong, Yibing Zhan, Bo Du, Yong Luo, and Wenbin Hu. Clnode: Curriculum learning for node classification. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 670–678, 2023.

[197] Daphna Weinshall, Gad Cohen, and Dan Amir. Curriculum learning by transfer learning: Theory and experiments with deep networks. In *International Conference on Machine Learning*, pages 5238–5246. PMLR, 2018.

[198] Chao Wen, Yinda Zhang, Zhuwen Li, and Yanwei Fu. Pixel2mesh++: Multi-view 3d mesh generation via deformation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1042–1051, 2019.

[199] Zhihao Wen and Yuan Fang. Augmenting low-resource text classification with graph-grounded pre-training and prompting. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 506–516, 2023.

[200] Douglas B West. *Introduction to Graph Theory*. Prentice Hall, 2001.

[201] Richard Lee Wheeden and Antoni Zygmund. *Measure and integral*, volume 26. Dekker New York, 1977.

[202] Kelin X Whipple. Bedrock rivers and the geomorphology of active orogens. *Annu. Rev. Earth Planet. Sci.*, 32:151–185, 2004.

[203] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples for graph data: deep insights into attack and defense. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 4816–4823, 2019.

[204] Q Wu, C Brinton, Z Zhang, M Cucuringu, A Pizzoferrato, and Z Liu. Equity2vec: End-to-end deep learning framework for cross-sectional asset pricing. In *2nd ACM International Conference on AI in Finance*, 2021.

[205] Qiong Wu, Wen-Ling Hsu, Tan Xu, Zhenming Liu, George Ma, Guy Jacobson, and Shuai Zhao. Speaking with actions - learning customer journey behavior.

In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 279–286, 2019. doi: 10.1109/ICOSC.2019.8665577.

[206] Qiong Wu, Adam Hare, Sirui Wang, Yuwei Tu, Zhenming Liu, Christopher G Brinton, and Yanhua Li. Bats: A spectral biclustering approach to single document topic modeling and segmentation. *ACM Transactions on Intelligent Systems and Technology*, 2021.

[207] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5): 1–37, 2022.

[208] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[209] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[210] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

[211] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[212] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

[213] Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, et al. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 36:17238–17264, 2023.

[214] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. Graphformers: Gnn-nested transformers for representation learning on textual graph. *Advances in Neural Information Processing Systems*, 34:28798–28810, 2021.

[215] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[216] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134*, 2023.

[217] Li Yi, Hao Su, Xingwen Guo, and Leonidas J Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2282–2290, 2017.

[218] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.

[219] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 4800–4810, 2018.

[220] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5708–5717, 2018.

[221] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823, 2020.

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

[223] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 793–803, 2019.

[224] Delvin Ce Zhang, Menglin Yang, Rex Ying, and Hady W Lauw. Text-attributed graph representation learning: Methods, applications, and challenges. In *Companion Proceedings of the ACM on Web Conference 2024*, pages 1298–1301, 2024.

[225] Zheng Zhang and Liang Zhao. Representation learning on spatial networks. *Advances in Neural Information Processing Systems*, 34:2303–2318, 2021.

[226] Zheng Zhang and Liang Zhao. Unsupervised deep subgraph anomaly detection. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 753–762. IEEE, 2022.

[227] Zheng Zhang and Liang Zhao. Self-similar graph neural network for hierarchical

graph learning. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*, pages 28–36. SIAM, 2024.

[228] Zheng Zhang, Hossein Amiri, Zhenke Liu, Andreas Züfle, and Liang Zhao. Large language models for spatial trajectory patterns mining. *arXiv preprint arXiv:2310.04942*, 2023.

[229] Zheng Zhang, Hossein Amiri, Dazhou Yu, Yuntong Hu, Liang Zhao, and Andreas Zufle. Transferable unsupervised outlier detection framework for human semantic trajectories. *32nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2024.

[230] Zheng Zhang, Yuntong Hu, Bo Pan, Chen Ling, and Liang Zhao. Taga: Text-attributed graph self-supervised learning by synergizing graph and text mutual transformations. *arXiv preprint arXiv:2405.16800*, 2024.

[231] Zheng Zhang, Sirui Li, Jingcheng Zhou, Junxiang Wang, Abhinav Angirekula, Allen Zhang, and Liang Zhao. Non-euclidean spatial graph neural network. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*, pages 154–162. SIAM, 2024.

[232] Zheng Zhang, Junxiang Wang, and Liang Zhao. Curriculum learning for graph neural networks: Which edges should we learn first. *Advances in Neural Information Processing Systems*, 36, 2024.

[233] Zheng Zhang, Allen Zhang, Ruth Nelson, Giorgio Ascoli, and Liang Zhao. Representation learning of geometric trees. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4374–4385, 2024.

[234] Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. Learning on large-scale text-attributed graphs via variational infer-

ence. In *The Eleventh International Conference on Learning Representations*, 2022.

[235] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*, pages 11458–11468. PMLR, 2020.

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

[237] Tianyi Zhou, Shengjie Wang, and Jeff Bilmes. Robust curriculum learning: from clean label detection to noisy label self-correction. In *International Conference on Learning Representations*, 2020.

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

[239] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2847–2856, 2018.