BEYOND LOCAL NEIGHBORHOODS: LEVERAGING INFORMATIVE NODES FOR IMPROVED GRAPH NEURAL NETWORKS PERFORMANCE

A Dissertation
Submitted to
the Temple University Graduate Board

In Partial Fulfillment
of the Requirements for the Degree
DOCTOR OF PHILOSOPHY

---

by
Peiyu Liang
December 2024

Examining Committee Members:

Xubin He, Advisory Chair, Computer and Information Sciences
Yu Wang, Computer and Information Sciences
Hongchang Gao, Computer and Information Sciences
Zhigen Zhao, External Reader, Statistics, Operations and Data Science

ABSTRACT

Many real-world datasets, such as those from social and scientific domains, can be represented as graphs, where entities are depicted as nodes and their relationships as edges. To analyze the properties of individual entities (node classification) or the community as a whole (graph classification), graph neural networks (GNNs) serve as a powerful tool. Most GNNs utilize a message-passing scheme to aggregate information from neighboring nodes. This localized aggregation allows the network to learn representations that incorporate the context of each node, thereby enhancing its ability to capture complex local structures and relationships.

Despite their success, many GNNs heavily rely on local 1-hop neighborhood information and a stacked architecture of K layers. This dependency can result in poor handling of long-range dependencies and lead to issues like information over-squashing. Consequently, there is a pressing need for advanced methodologies that can systematically aggregate more informative nodes beyond the default graph structure to achieve more accurate classification results.

In this thesis, we highlight the challenges of information over-squashing and the limited capacity of existing GNNs to capture long-range dependencies, focusing on addressing these issues through innovative informative node selection and end-to-end learning strategy using three approaches. Our first approach, *Two-view GNNs with adaptive view-wise structure learning strategy*, posits that more informative nodes should have proximal node representations within a graph structure constructed on such attributes. We reconstruct a new graph structure based on the proximity of node representations and simultaneously learn a graph object from both the newly constructed and default graph structures for relationship reasoning. Additionally, we employ an adaptive strategy that learns inter-structure relationships based on classifier performance. While this approach achieves more accurate classifications, it is still limited by relying on a single or two graph structures. Our second approach, *Cauchy-smoothing GCN (CauchyGCN)*, utilizes the default graph structure but regards more informative nodes as those closely embedded in the embedding space. CauchyGCN

develops a new layer-wise message-passing scheme that follows the properties of the Cauchy distribution, preserving smoothness between closely embedded nodes while penalizing distant 1-hop neighbors less severely. This approach shows competitive results compared to other advancements. From our first two approaches, we observe that (1) Understanding the graph requires learning from multiple perspectives, and informative nodes could reside beyond the default graph structure. (2) Preserving smoothness among informative nodes is beneficial for effective learning. Our third approach, *Topological-induced Graph Transformer (TOPGT)*, defines the additional useful graph structures as topological structures and leverages a self-attention mechanism to assess the importance of closely embedded nodes. This approach achieves state-of-the-art performance compared to existing methods in the domain.

Finally, we summarize our contributions through the three approaches that address the challenges that this thesis highlights. Additionally, I discuss potential future work to explore and utilize informative node information beyond local neighborhoods, aiming to develop large pre-trained GNNs capable of tackling various downstream tasks across different domains.

To my family and husband, for their unconditional love and support.

ACKNOWLEDGMENTS

My PhD would not have been possible without all the support I have received from my advisors, friends, family, and husband.

I would like to thank my advisor, Dr. Xubin He, for his support and guidance throughout my PhD journey. I have learned so much from Dr. He as a researcher, from focusing on the big picture to presenting my work clearly and effectively. His invaluable feedback has helped me grow not only in research but also in many other aspects of my life.

I would like to thank the remaining members of my dissertation committee: Dr. Hongchang Gao, Dr. Yu Wang, and Dr. Zhigen Zhao, for their valuable insights that contributed to the development of this thesis and for supporting my growth and success throughout my PhD journey.

I am grateful to Dr. Hongchang Gao for introducing me to the field of Graph Neural Networks. His mentorship has laid the groundwork for my interest in this area.

I would like to express my sincere gratitude to Dr. Yuzhou Chen. I feel truly lucky to have been mentored by him and to have collaborated with him during the final year of my PhD. Dr. Chen has consistently supported and encouraged my research ideas, reinforcing my commitment to continue my work beyond graduation.

Finally, I want to express my gratitude to my family and my husband for their unwavering support and unconditional love. Thank you for always believing in me and cheering me on, helping to create every opportunity for my success in life.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Introduction

Graphs, which consist of nodes representing entities and edges representing relationships, are fundamental structures in various real-world domains. Researchers have explored a range of machine learning and deep learning techniques to address tasks such as node classification (classifying individual entities within a large network) and graph classification (classifying entire graph communities). Among these techniques, graph neural networks (GNNs) have emerged as a highly effective tool for graph learning, demonstrating exceptional performance across diverse applications. These include recommendation systems [1], social networks [2], chemistry [3], transportation [4], computer vision [5], and natural language processing [6]. Figure 1.1 illustrates examples of these applications [1].

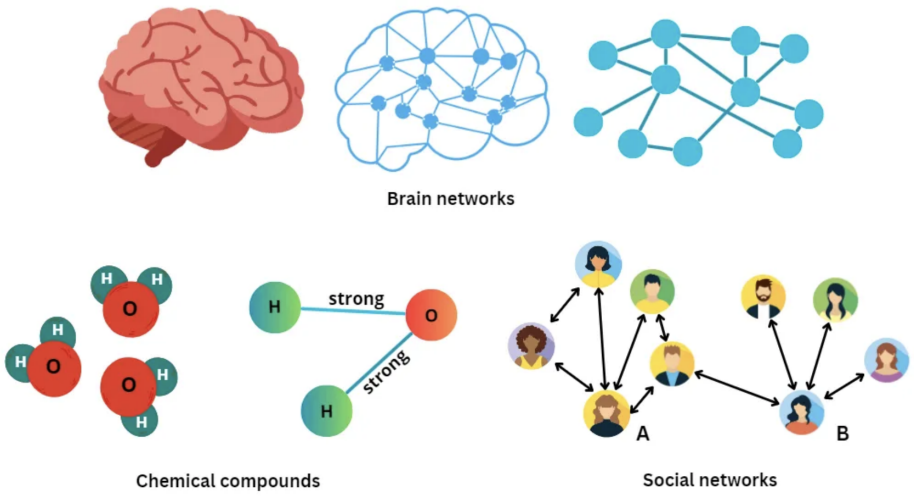


Figure 1.1.: Application examples of graph neural networks [1].

Like other deep learning architectures, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs), GNNs incorporate essential elements like linear layers, residual connections, activation functions, and dropout to enhance their

[1]https://medium.com/@bscarleth.gtz/introduction-to-graph-neural-networks-an-illustrated-guide-c3f19da2ba39

training processes. However, GNNs stand out in the realm of graph learning due to their unique graph convolutional layers, which enable them to effectively leverage the underlying graph structure to model both input and output data. The core functionality of graph convolutional layers in GNNs revolves around two primary components. First, they employ a structural message-passing scheme that computes node representations by aggregating information from neighboring nodes within the local 1-hop neighborhood. This approach allows GNNs to capture the immediate context of each node, providing a rich representation that reflects local interactions and relationships. Second, GNNs typically consist of a stack of K layers, which enables the aggregation of information not just from direct neighbors but also from nodes that are up to K hops away. This multi-hop aggregation significantly enhances the expressivity of the network, allowing it to model more complex dependencies and interactions across the graph. By connecting nodes that are further apart, GNNs facilitate a broader communication mechanism, enabling the capture of long-range dependencies that are crucial for tasks such as graph classification and node classification. In addition to these core components, GNNs also integrate various techniques to improve performance and robustness. For example, normalization layers are often included to stabilize training, while dropout layers help prevent overfitting by randomly deactivating a subset of neurons during training.

One of the significant limitations of GNNs is their strong inductive bias towards the underlying graph structure. This bias necessitates that all nodes learn exclusively from their K-hop neighbors, often neglecting the unique attributes of individual nodes and the consistency of labels. As a result, GNNs can struggle to capture essential relationships, particularly in cases where information from nodes more than one hop away is required. In large-scale graph scenarios, this challenge is exacerbated, making it difficult for GNNs to effectively model long-range dependencies. Moreover, as K increases, GNNs can face issues related to over-smoothing. This phenomenon occurs when the number of layers K exceeds the graph's radius, leading to increasingly homogeneous node representations. In such cases, nodes that may belong to distinct classes can become indistinguishable, as their features converge towards similar val-

ues. This problem is illustrated in Figure 1.2, where nodes are fully interconnected yet belong to different classes—represented by distinct colors such as red, blue, and yellow. With a graph radius of 3 and the application of 4-layer graph convolutions, the nodes in this example exemplify how over-smoothing can render them nearly indistinguishable. As the layers deepen, the nuanced differences in node features diminish, making it challenging for the model to differentiate between classes. This convergence undermines the model's ability to effectively enrich node representations thus limiting its performance in various downstream tasks, such as node classification and graph classification.

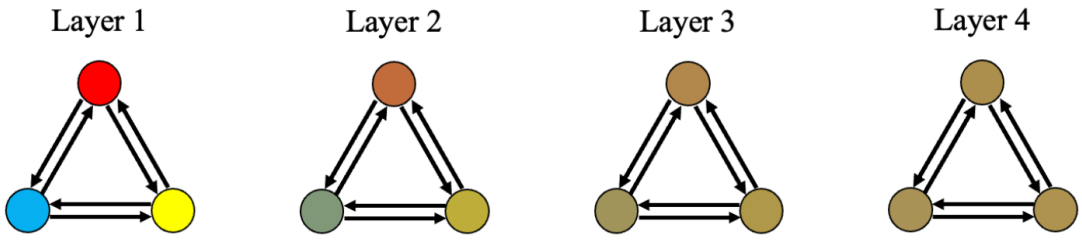


Figure 1.2.: Illustration of over-smoothing problem in graph neural networks.

Addressing these issues is critical for advancing the capabilities of GNNs. Existing work is exploring various strategies, including the integration of attention mechanisms, which allow models to weigh the importance of different neighbors and prioritize relevant information [7]. Additionally, incorporating techniques for enhancing representation diversity and exploring innovative architectures that break away from traditional layer stacking can help mitigate the effects of over-smoothing and improve the capture of long-range dependencies [8, 9]. By refining the design of GNNs to reduce the impact of these limitations, one can enhance their applicability and effectiveness across a broader range of complex graph-based tasks.

1.2 Motivation

While GNNs have demonstrated considerable effectiveness in various tasks, many existing methods still heavily depend on the original properties of the graph, which can constrain their expressiveness. On one hand, the connectivity patterns derived from the original graph reflect factual relationships among entities, but this structure does not guarantee labeling consistency among all connected nodes. As GNNs operate, the graph convolutional layers project nodes with similar proximities into a shared embedding space based on information from their neighbors. In scenarios where nodes are sparsely connected and possess different class labels, this approach can lead to the transfer of noisy or misleading information, ultimately resulting in incorrect classifications. On the other hand, graphs are inherently rich in higher-order structures and diverse connection patterns that may not be fully captured by the original graph topology. Relying exclusively on these properties limits the ability of GNNs to uncover the intrinsic characteristics of the data. For instance, reconsidering the graph structure to connect nodes based on their input similarity rather than merely their factual relationships could enhance the model's performance. This perspective shifts the focus from a purely structural representation to a more nuanced understanding of the relationships between entities. Recent advancements have highlighted the value of identifying recurring and significant neighboring patterns for better graph learning. By recognizing these patterns, GNNs can better adapt their learning processes, allowing for more informed aggregation of features from relevant nodes. Additionally, the exploration of topological features provides insights into higher-order connectivity patterns at both the node and subgraph levels. This exploration can unveil complex interactions that go beyond immediate neighbors, enhancing the model's capacity to capture the multifaceted nature of graph data. Furthermore, incorporating techniques such as attention mechanisms or dynamic graph restructuring can enable GNNs to prioritize and adaptively focus on the most informative connections, thus mitigating the limitations imposed by the original graph structure. By leveraging these advanced methodologies, GNNs can achieve a more effective and robust representation of the

underlying data, ultimately leading to improved performance in a variety of applications, from social network analysis to drug discovery. The ongoing exploration of these dimensions in graph learning is crucial for unlocking the full potential of GNNs and expanding their applicability across diverse domains.

1.3 Contribution

In this thesis, we propose three novel approaches that aim to explore and leverage informative features from nodes beyond the local 1-hop neighbors. These methods are designed to enhance the expressiveness of GNNs for classification tasks, encompassing both node-level and graph-level classifications. Figure 1.3 provides a high-level overview of our three approaches aim at addressing the structural bias of the graph (i.e., message passing beyond neighbors of one hop) in GNNs.

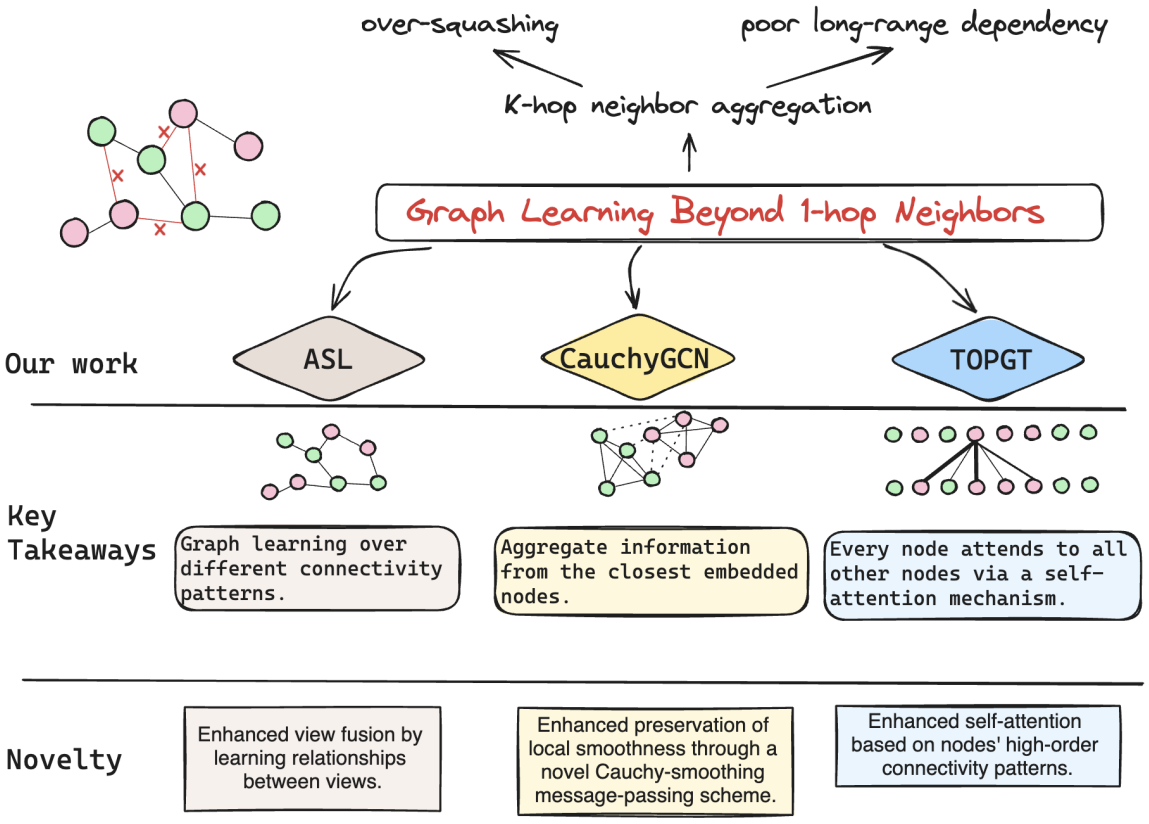


Figure 1.3.: High-level overview of our three approaches.

The contribution of these three approaches can be summarized as follows.

1. Adaptive Structure Learning (ASL) aims at graph learning by leveraging different connectivity patterns. ASL considers both the original graph structure and the node proximity-augmented structures to extract additional informative features from intrinsically proximal nodes that are not limited to 1-hop connections. This approach optimizes the capabilities of two-view GNNs in graph classification tasks while also capturing the inter-view relationships through a novel adaptive view-wise structure learning strategy.

2. Cauchy-smoothing-based graph convolutional network (CauchyGCN) aims at graph learning by focusing on closely embedded and mutually connected nodes in the embedding space, extending beyond just 1-hop neighbors. It also relaxes penalties for distant nodes that lack proximity in the embedding space. This approach balances local and global smoothness preservation through a novel Cauchy-smoothing message-passing scheme, thereby enhancing the performance of message-passing GNNs in node classification tasks.

3. Topology-induced Graph Transformer (TOPGT) focuses on graph learning by enabling every node to attend to all other nodes over the graph using learnable attention weights through a self-attention mechanism, effectively breaking away from the original graph structure. This approach enhances the attention between nodes by taking into account not only their node features and subgraph structures but also their higher-order topological features. As a result, TOPGT achieves state-of-the-art performance in graph classification tasks.

1.4 Synopsis of the Thesis

Throughout this thesis, we focus on discussing the expressive limitations of GNNs related to the inductive bias of the graph structure. We present three approaches to address this issue by exploring informative features extending beyond local 1-hop neighbors. This thesis is organized as follows.

Chapter 2 provides an overview of background and related work. In this section, Section 2.1 presents useful graph notations relevant to graph learning. Section 2.2 introduces two key graph-related tasks that are the primary focus of the methods discussed in this thesis. Section 2.3 describes the evolution of GNNs from convolutional neural networks, highlighting the initial categorization of these approaches. Section 2.4 discusses one of the pivotal paradigms in GNNs: message-passing-based GNNs. Finally, Section 2.5 covers one of the most recent paradigms that effectively addresses the major limitations of traditional GNNs.

In Chapter 3, we present our Adaptive Structure Learning (ASL) approach within the framework of two-view GNNs. Section 3.1 outlines the scope of this approach, discusses the existing challenges in two-view GNNs, and highlights our contributions. Section 3.2 reviews related work in multi-view GNNs and the techniques on which our approach is based. Section 3.3 details our adaptive structure learning strategy that aims to improve graph classification through augmented graph structures. Section 3.4 presents a comprehensive empirical study, comparing our methods with state-of-the-art approaches in graph classification performance, method analysis, and ablation studies. Finally, Section 3.5 summarizes the contributions and achievements of this approach.

In Chapter 4, we introduce our Cauchy-smoothing-based GCN (CauchyGCN) approach within the framework of message-passing GNNs. Section 4.1 outlines the scope of this approach, discusses the major limitations in the field, and highlights our contributions. Section 4.2 reviews Laplacian smoothing-based GNNs, focusing on their layer-wise propagation rules, message-passing schemes, and advancements in the domain. Section 4.3 details our methodology, including the motivation to leverage the Cauchy distribution as a theoretical basis. We propose a novel Cauchy-smoothing-based message-passing scheme that enhances local smoothness while addressing challenges present in Laplacian smoothing. Additionally, we introduce a clustering analysis to improve global smoothness for model optimization. Section 4.4 presents a comprehensive empirical study that compares our methods with state-of-the-art approaches in terms of node classification performance, robustness analysis,

and propagation depth analysis. Finally, Section 4.5 summarizes the contributions and achievements of this approach.

In Chapter 5, we present our Topology-induced Graph Transformer (TOPGT) approach. Section 5.1 addresses the limitations of existing graph transformers, outlines the motivation behind our approach, and highlights our contributions to the field. Section 5.2 reviews topological data analysis, introducing relevant notations used in graph learning and advancements in graph transformers. In Section 5.3, we detail our novel topology-induced graph transformer, discussing the process of extracting topological information, our topology-induced structural encoding strategy, and the topology-aware self-attention mechanism. Section 5.4 presents a comprehensive empirical study comparing our methods with traditional GNNs and the latest graph transformers in terms of graph classification performance. We also conduct ablation studies on all components of our methods and analyze their robustness under edge attacks. Finally, Section 5.5 summarizes the contributions and achievements of this approach.

Lastly, we conclude our research and list future research in Chapter 6.

CHAPTER 2
BACKGROUND AND RELATED WORK

2.1 Graph Notation

An attributed graph can be represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_i : i = 1, \ldots, |\mathcal{V}|\}$ is a node set and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is an edge set. We use $N = |\mathcal{V}|$ to denote the size of the node set. In our study, we consider a node feature matrix $X = [x_1, \ldots, x_N] \in \mathbb{R}^{N \times f_0}$, where $x_i \in \mathbb{R}^{1 \times f_0}$ is the f_0-dimensional feature vector for each node v_i. The edges between nodes are captured by an $N \times N$ non-negative symmetric adjacency matrix A with entries $\{a_{ij}\}_{1 \leq i,j \leq N}$, such that $a_{ij} = a_{ji} = 1$ if $(v_i, v_j) \in \mathcal{E}$, and $a_{ij} = a_{ji} = 0$, otherwise. When considering the adjacency matrix with a self-loop effect, we denote it as $\tilde{A} = A + I$, with entries $\{\tilde{a}_{ij}\}_{1 \leq i,j \leq N}$, where I is an identity matrix of size N. A diagonal degree matrix $\tilde{\mathcal{D}} = diag(\tilde{d}_1, \ldots, \tilde{d}_N)$ is based on $\tilde{A}$, where $\tilde{d}_i = \sum_j \tilde{a}_{ij}$. Laplacian matrix $L = I - A$ is an important component of GNNs from the spectral domain, which we will describe later. A normalized Laplacian matrix that represents the graph $\mathcal{G}$ can be written to $\tilde{L} = I - \hat{\tilde{A}}$, where $\hat{\tilde{A}} = \tilde{\mathcal{D}}^{-\frac{1}{2}} \tilde{A} \tilde{\mathcal{D}}^{-\frac{1}{2}}$ is the normalized adjacency matrix with entries $\{\hat{\tilde{a}}_{ij}\}_{1 \leq i,j \leq N}$. Moreover, we use $\mathcal{N}_i$ to indicate a 1-hop neighboring node set of the center node v_i. It is important to note that this neighborhood definition is permutation invariant, meaning it remains unchanged regardless of the order of nodes in the graph.

2.2 Graph-related Tasks

Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, there are various downstream tasks, including classification and regression at the node, edge, and graph levels. Each of these tasks leverages the attribute information inherent in graphs using different learning strategies. In this thesis, we focus specifically on node and graph classification. Both tasks significantly benefit from the relational information encoded within graph structures and node features. GNNs excel at capturing complex dependencies and interactions, leveraging the interconnectedness of nodes to enhance classification accu-

racy. By aggregating information from local neighborhoods and recognizing patterns across the graph, GNNs can discern subtle relationships that may not be apparent through traditional machine learning approaches. This capability is vital in various applications, including social network analysis, bioinformatics, and fraud detection, where understanding the intricate relationships between entities is key to accurate classification.

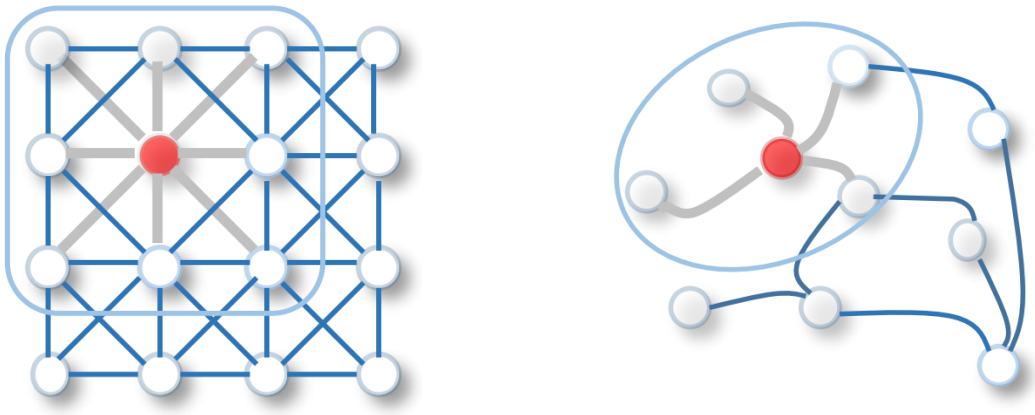
Node classification focuses on predicting the labels of individual nodes based on their features and the relationships defined by the default graph structure. This task often employs semi-supervised learning techniques, which are particularly advantageous when there is a scarcity of labeled data, for where only a small subset of nodes in a graph is labeled with their corresponding classes, while the majority remain unlabeled. By utilizing both labeled and unlabeled nodes, models can enhance their performance through inferring the labels of unlabeled nodes based on their connections. For instance, in a social network context, node classification might involve predicting users' interests or preferences based on their connections with others and the attributes outlined in their profiles. Here, the GNN aggregates features from neighbors to capture not only the local context but also the influence of K-hop community structures, leading to more accurate predictions.

Graph classification, on the other hand, involves assigning labels to entire graphs, necessitating a comprehensive understanding of the graph's topology and attributes. This task typically employs supervised learning strategies, where models are trained on a dataset of labeled graphs. The goal is to learn representations that encapsulate the relevant structural and feature-based information for effective classification. An example of graph classification can be found in cheminformatics, where one might predict the category of chemical compounds based on their molecular structures. GNNs captures not just the individual atoms (nodes) and bonds (edges) but also the subgraph structures, motifs, or higher-order relationships that are crucial for determining the chemical properties and functionalities of the compounds.

As the complexity of the graphs increases, the ability of GNNs to learn from both node-level and graph-level features allows for more nuanced and effective representa-

tions, ultimately improving performance across a wide range of graph-based tasks. As research in this area progresses, ongoing innovations will likely continue to enhance the capabilities of GNNs, making them even more adept at tackling the challenges posed by real-world graph data.

2.3 From Convolutional Neural Networks to Graph Neural Networks



(a) 2D Convolution (b) Graph Convolution

Figure 2.1.: 2D Convolution vs. Graph Convolution [10].

GNNs emerged as a natural extension of convolutional neural networks (CNNs) to handle graph-structured data. As shown in Figure 2.1, while CNNs excel in handling grid-like data (such as images) by leveraging spatial hierarchies and local connectivity, GNNs extend the convolutional concept to allow for the aggregation of information from a node's neighbors to learn the complex relationships and dependencies within graph structures. Since then, GNNs can be broadly categorized into two approaches, spectral-based and spatial-based.

Spectral-based GNNs with a mathematical foundation in graph signal processing. This approach learns a graph filter g applied to a graph signal X, represented as $g \cdot X$. This graph convolution is processed using Fourier transform $\mathcal{F}$ to project X and g in an orthonormal space, resulting in the equation $\mathcal{F}^{-1}(\mathcal{F}(g)\mathcal{F}(X)) = U(U^\top g \cdot U^\top X)$.

The basis U is obtained through the eigendecomposition of the normalized graph Laplacian matrix $\tilde{L}$ (see definition in 2.1), where $U \in \mathbb{R}^{N \times N}$ represents the matrix of eigenvectors arranged by eigenvalues order, and Λ is a diagonal matrix of eigenvalues. The eigenvectors of the normalized Laplacian matrix form an orthonormal space, satisfying $U^\top U = I$. The first work on spectral convolutional networks was introduced by [11]. However, due to the high computing complexity and non-robust structure of eigendecomposition of the normalized graph Laplacian matrix, many variants studies [12–15] after [11] aim to learn faster and optimal matrices for convolution operations.

Spatial-based GNNs focus on learning directly from the graph's structure without relying on spectral properties. This approach utilizes the local neighborhoods of each node to perform message passing, where each node aggregates information from its neighbors to update its representation. The fundamental idea is to define a message function that computes the influence of neighboring nodes, often employing mechanisms such as attention or weighted averaging. For example, a typical update rule might be expressed as $h_i^{(k)} = \sum_{j \in \mathcal{N}_i} \Theta h_j^{(k-1)} + b$, where $h_j^{(k)}$ is the updated feature of node i at iteration k, and for $k = 0$, $h_j^{(0)} = x_j$. Here, $\mathcal{N}_i$ represents the neighbors of i (see definition in Section 2.1), Θ is a learnable weight matrix, and b is a bias term. This paradigm allows for flexibility and scalability, as it can handle graphs of varying sizes and structures. Notable models in this category include GraphSAGE [16] and GAT [7], which have demonstrated effectiveness in a range of applications.

2.4 Message-passing-based GNNs

Graph Convolutional Networks (GCN) [14] is a pivotal advancement in GNNs, offering a computationally efficient approach to graph convolution. This is achieved through a localized first-order approximation (i.e., $P = 1$) of Chebyshev polynomials $T_p(x)$. The graph filter defined can be defined as follows,

$$\begin{aligned} g_{\theta'} \cdot x &\approx \sum_{p=0}^{P=1} \theta'_p T_p(\tilde{L}) x \\ &\approx \theta'_0 x + \theta'_1 (L - I) x, \end{aligned} \tag{2.1}$$

where $\tilde{L} = \frac{2}{\lambda_{max}} L - I$ and λ_{max} is the max egienvalue of Λ. By letting $\theta = \theta'_0 = -\theta'_1$, Eq.(2.1) can be written to,

$$g_\theta \cdot x \approx \theta \left(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x. \tag{2.2}$$

Finally, by normalizing the adjacency matrix A, the graph convolution layer in GCN can be expressed as,

$$H = \hat{\tilde{A}} X \Theta, \tag{2.3}$$

where $\hat{\tilde{A}}$ is defined in Section 2.1. Here, H represents the updated feature, which results from projecting the input features X into a learnable feature space Θ through spectral graph signal processing. Meanwhile, GCN bridge the gap between spectral-based and spatial-based GNNs by allowing their spectral approach, represented in Eq.(2.3), to be expressed in a spatial context,

$$h_v = \Theta^\top \left(\sum_{u \in \{\mathcal{N}_v \cup v\}} \hat{\tilde{A}}_{v,u} x_u \right) \quad \forall v \in \mathcal{V}. \tag{2.4}$$

This spatial approach interprets convolution as a process of aggregating information over localized 1-hop neighborhoods within the graph structure $\hat{\tilde{A}}$. To learn from q-hop neighborhoods for $q > 1$, sequential updates are required, following Eq. 2.4, which involves stacking q graph convolutional layers.

Since then, a series of advancements have led to incremental improvements in GCN, many of which derive from the spatial approach due to its efficiency, scalability, and general applicability. Most of these innovations aim to address the limitations posed by GCN. For instance, GCN is constrained by its shallow architecture; stacking more than three graph convolutional layers can result in over-smoothing [9, 17, 18], causing node representations to become too similar and diminishing their ability to effectively capture long-range dependencies [19]. This issue arises from the message-passing scheme in GCN, which operates under the flawed assumption that all nodes that are 1-hop neighbors of a central node are equally important. To tackle this problem, several solutions have been proposed, such as skip connections [8,20], flexible neighborhood extensions [9,21], and relaxing the penalization on distant neighbors [7,

22]. While these approaches help preserve original node information and mitigate over-smoothing, they can still fall prey to the "curse of structural inductive bias."

2.5 Transformer-based Graph Neural Networks

Recently, transformers [1–3, 5, 6] have revolutionized both natural language processing (NLP) and computer vision (CV) by addressing key limitations of traditional models. In NLP, traditional models like recurrent neural networks [23] struggled with long-range dependencies and sequential processing, leading to inefficiencies and difficulties in capturing context. Transformers, with their self-attention mechanism, enable the model to weigh the importance of different words in a sentence simultaneously, facilitating better understanding of context and relationships. Similarly, in CV, transformers have introduced a new way to process images by treating them as sequences of patches, allowing for enhanced feature extraction and global context awareness. This shift has resulted in significant improvements in performance across various tasks, enabling models to learn more effectively from large datasets and achieve state-of-the-art results.

Building on the success of transformers in NLP and CV, there has been a significant shift towards applying these models to graph-based data, effectively addressing the limitations of message-passing GNNs, which often suffer from structural inductive bias. Specifically, the self-attention mechanism in transformers provides a more flexible approach by enabling the model to consider relationships across the entire graph rather than being confined to local connections. This capability directly alleviates the structural inductive bias that plagues traditional message-passing GNNs, allowing for a richer and more comprehensive understanding of graph structures. For a more detailed discussion of this domain, see Section 5.2.2.

CHAPTER 3
TWO-VIEW GNNS WITH AN ADAPTIVE VIEW-WISE STRUCTURE LEARNING STRATEGY FOR ACCURATE GRAPH CLASSIFICATION

Graphs inherently possess multiple underlying structures. Beyond the default structure, which often represents factual relationships between nodes, augmented structures can provide unique informative local information. To enhance graph learning, it is crucial to effectively utilize this diverse structural information. While several multi-view GNNs exist for early or late view-wise fusion, many overlook the important correspondence between nodes and views. This chapter introduces a novel multi-view GNNs with an *adaptive view-wise structure learning strategy*, focusing specifically on *two-view GNNs*. Our method aims to simultaneously capture valuable local information from both views while also understanding the interrelationships between views through node correspondences and view-wise distances. Empirical studies on graphs from various domains, including molecular, computer vision, and bioinformatics, demonstrate that our method delivers competitive performance, irrespective of the view-wise fusion technique used.

3.1 Introduction

Drawing from the capabilities of GNNs, multi-view frameworks are designed to leverage multiple perspectives of data, thereby enhancing the model's ability to learn from diverse information sources. Within this framework, GNNs are trained in parallel and independently to capture view-specific representations, which are subsequently fused into a unified representation.

While multi-view GNNs theoretically have the potential to utilize more than two views for enhanced learning, several practical challenges often limit the focus to two-view GNNs. First, incorporating multiple views increases the complexity of the model, necessitating sophisticated methods for managing and integrating diverse informa-

tion. This complexity can lead to difficulties in training, tuning hyperparameters, and achieving stable convergence. Second, the computational cost associated with processing and aggregating multiple views can be prohibitive, especially for large-scale graphs. Each additional view demands more resources for storage and computation, which can hinder scalability and efficiency. Third, there is often a trade-off between the richness of information and the risk of overfitting; with too many views, the model may capture noise rather than useful patterns, leading to poorer generalization performance. Thus, two-view GNNs strike a balance between complexity and expressiveness, resulting in a greater number of studies focusing on this specific configuration. Moreover, two-view GNNs represent a particularly intriguing case, as they emphasize two distinct yet complementary views of the same data. By enabling a more nuanced representation of the underlying relationships in the graph, two-view GNNs can capture both local and global patterns that might be overlooked in single-view scenarios. Integrating information from these two perspectives allows them to effectively mitigate challenges such as over-smoothing and information loss. Examples include the identification of molecular networks characterized by two underlying molecular structures [24], the prediction of brain diseases using functional magnetic resonance imaging (fMRI) and diffusion tensor imaging (DTI) data sources [25], and the analysis of user activities on social networks based on both online and offline interactions [26].

Two-view GNNs, while effective in leveraging complementary perspectives of data, encounter several noteworthy challenges that can impact their performance and robustness: 1) *Labeling variability*: One significant challenge arises from labeling variability across views. When view-wise representations are learned independently using unshared GNN parameters, nodes that belong to different classes within a given feature space may generate inconsistent representations. This divergence can lead to difficulties when fusing these representations, as the resulting unified representation may become inundated with noise, reducing the overall accuracy and reliability of the model. Ensuring that both views contribute effectively to a coherent classification outcome requires strategies to align labels and representations. 2) *Node correspon-*

dence ambiguity: Another challenge lies in node correspondence ambiguity. Different views often exhibit distinct graph structures, complicating element-wise fusion due to unclear node correspondences across views. For instance, modality-wise pooling—a common fusion technique—selects specific features from view-specific representations on an element-wise basis. However, without a clear mapping of nodes between the views prior to fusion, there is a risk of underutilizing valuable information and introducing noise into the unified representation. This ambiguity can significantly undermine the potential benefits of integrating two views. 3) *Inter-view correlation neglect and knowledge underutilization*:Although graph representations from different views should inherently exhibit some level of correlation, neglecting to capture these inter-view correlations can hinder the utilization of complementary knowledge. Each view holds unique insights about the underlying data, and failing to recognize their relationships means missing out on opportunities to enrich the learning process. Effective models should strive to exploit these correlations, integrating knowledge from different views to enhance overall understanding and performance. For instance, applying techniques that explicitly model inter-view dependencies could improve representation learning by capturing the shared information that exists between the views.

Addressing these challenges is crucial for the advancement of two-view GNNs, enabling their full potential in the downstream applications. By developing strategies that enhance representation alignment, clarify node correspondences, and leverage inter-view correlations, one can improve the robustness and effectiveness of two-view GNN architectures. In this chapter, we present an optimal transport-based adaptive view-wise structural learning approach for two-view GNNs, focusing to address the aforementioned challenges:

1. To mitigate labeling variability, we propose an unsupervised learning method that focuses on the distances between views.

2. We resolve the the ambiguity in node correspondence by leveraging optimal transport metric for view-wise alignment.

3. To tackle the neglect of inter-view correlations and knowledge underutilization, we adopt a dual approach involving node-level and graph-level considerations. At the node level, we see corresponding nodes as inter-view correlations, intending to minimize the distance between these nodes to preserve shared knowledge. At the graph level, in addition to unsupervised minimization of inter-view distances, we guide the model to maintain balance by adapting to the classifier's performance. This prevents views from diverging too far, neglecting shared knowledge, or converging too closely, underutilizing complementary knowledge.

Our method demonstrates its effectiveness by achieving improved graph classification results on six benchmark datasets. We also conduct experimental evaluations and ablation studies to further validate our approach.

3.2 Related Work

In this section, we provide a brief overview of related work in multi-view graph neural networks and the optimal transport distance metric.

3.2.1 Multi-view Graph Neural Networks

Multi-view GNNs are designed to learn a graph object from multiple perspectives, aiming to enhance prediction accuracy through the combination of diverse data. Typically, each view is trained independently to learn the unique local information under each view of the graph. Existing methods in multi-view GNNs showcase a variety of fusion strategies, each with its strengths and weaknesses. For instance, Song et al. [25] develop a voting strategy that balances outputs from different graph signals, allowing for a more nuanced integration of diverse perspectives. Ktena et al. [27] utilize a Hadamard product layer to capture view-wise similarity, effectively combining features while maintaining the individual characteristics of each view. Zhang et al. [28] take a different approach by concatenating graph-level knowledge to ensure

that all modalities contribute to the final representation, thus preserving valuable information across views.

Despite these advancements, many existing multi-view GNNs often overlook the interrelationships between views, which can significantly limit their effectiveness. The independent training of views may result in missed opportunities for synergy, where the shared information and complementary insights across views could enhance the overall learning process. This neglect of inter-view relationships can lead to suboptimal fusion, where the combined representation fails to capture the richer context that could be derived from understanding how different views interact with one another.

3.2.2 Optimal Transport

Optimal transport (OT) is a powerful distance metric that quantifies the minimum cost required to transform one probability distribution into another. Its application in deep learning has gained considerable traction in recent years, demonstrating significant advantages across various domains. For instance, in content-based image retrieval, OT has shown robustness in partial matching and flexibility in handling variable-length signals, outperforming traditional histogram matching methods that measure similarity in few-shot learning scenarios [29, 30]. Additionally, OT has been effectively employed to measure transfer costs between word embeddings in informative retrieval contexts [31]. Recently, OT has attracted significant interest in the realm of node classification, particularly concerning graph alignment [32–34]. In this setting, OT is used to align corresponding nodes or topological structures between graphs, facilitating unlabeled node predictions. However, integrating OT into deep learning frameworks presents optimization challenges, leading many existing studies to rely on off-the-shelf machine learning models or single-layer neural networks.

In this work, we aim to tackle these optimization hurdles by integrating OT into deep graph learning frameworks. By leveraging the unique properties of OT, we seek to enhance graph-level classification tasks within two-view GNNs. This approach not

only promises to improve the accuracy of node classification but also expands the potential of OT in capturing complex relationships within graph structures.

3.3 Adaptive View-wise Structure Learning Strategy

Our adaptive view-wise structure learning strategy for two-view GNNs aims to obtain more informative insights by leveraging the complementary information provided by two graph structures and adaptively learning their interrelationships. The proposed network construction and objective function are outlined in Section 3.3.1 and Section 3.3.2, respectively. The optimal transport-based approach is detailed in Section 3.3.3

3.3.1 Network Construction

Figure 3.1 depicts the network architecture of our proposed method. Our network consists of two independent GNNs, with one processing the default graph structure $\mathcal{G}_1 = (X, A)$ and the other processing an augmented graph structure based on node similarity $\mathcal{G}_2 = (X, A')$. To learn view-wise node embeddings that follow the input graph structure, each GNN employs three graph convolutional layers, allowing each central node to learn from at most 3-hop neighbors. The model parameters for a GNN are denoted as $\Theta = \{\theta_1, \theta_2, \theta_3\}$, with associated hidden dimension size of $\{d_{\theta_1}, d_{\theta_2}, d_{\theta_3}\}$. The convolution operation in GNNs can be implemented using various techniques, including ChebNet [13] and GCN [14]. The view-specific representations learned by GNNs are then fused using a specified technique, as in most existing multi-view GNNs. However, our method also incorporates a view-wise structure learning layer to capture and learn the interrelationships between the views. This allows for a more comprehensive understanding of the graph's underlying structure and potentially improves classification performance (results can be found in Section 3.4.2).

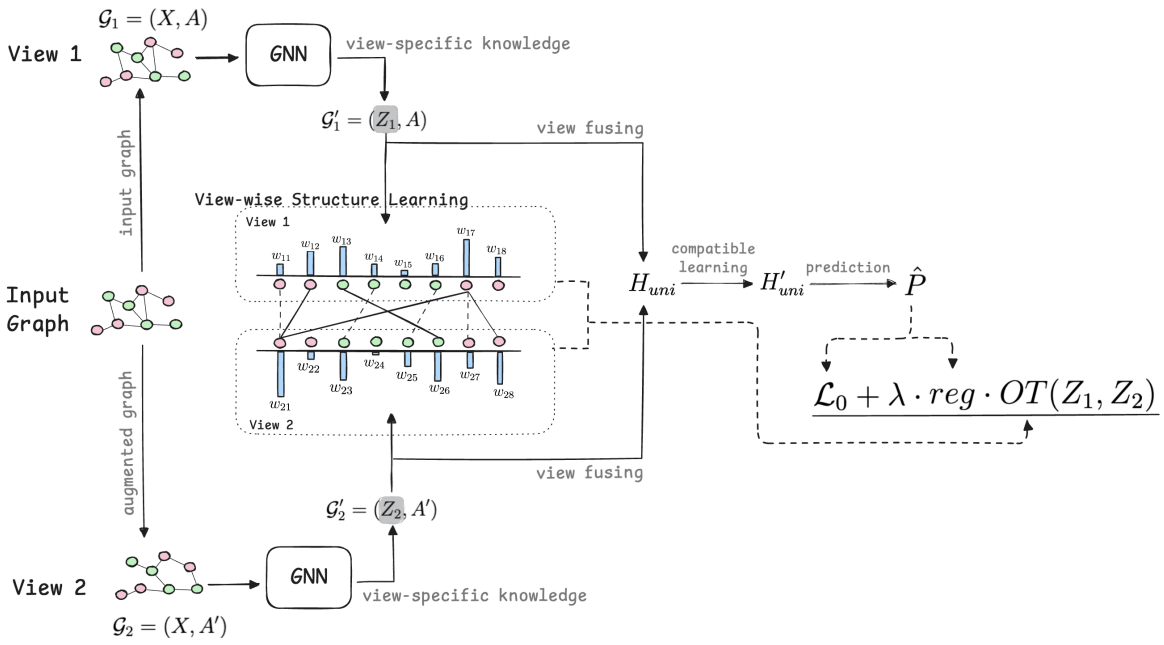


Figure 3.1.: An overview of the proposed adaptive view-wise structure learning strategy in two-view GNNs.

3.3.2 Objective Function

Let the predictive model's objective function be defined as:

$$\mathcal{O}(y) = \underset{y}{\operatorname{argmin}}\{\mathcal{L}_0 + \mathcal{L}_{reg}\} \ . \tag{3.1}$$

The first term, $\mathcal{L}_0$, is the supervised loss, often measured using cross-entropy to penalize the distance between predicted and ground truth class labels. For a sample with true class label Y_T and predicted probability distribution $\hat{P}_T$, the cross-entropy loss is:

$$\mathcal{L}_0 = -\sum_{c=1}^{C} Y_T log(\hat{P}_T) \ . \tag{3.2}$$

In this work, we denote Y_T as the true class label and $\hat{P}_T$ as the predicted probability of the true class label for a sample. The second term, $\mathcal{L}_{reg}$ additionally regularizes model parameters. It is required to be a differentiable function that can be logical inferences for end-to-end optimization. In our work, we design this term for learning the inter-view relationship in an unsupervised learning manner, $\mathcal{L}_{ASL}$. By taking the learned representation Z_1 and Z_2 from two views, $\mathcal{L}_{ASL}$ is defined as:

$$\mathcal{L}_{ASL}(Z_1, Z_2) = reg \cdot OT(Z_1, Z_2) \ . \tag{3.3}$$

Here, the term reg is adaptive and relies on classifier performance through a momentum algorithm. While we minimize the structural distance between views in an unsupervised manner, reg strikes a balance, ensuring it is neither too small to miss out on complementary knowledge nor too large to disregard shared knowledge. To achieve this balance, we link reg to the classifier's performance, quantified by the probability $\hat{P}_T$ of being the true class. If $\hat{P}_T$ suggests that the current unified representation is accurate (a high value of $\hat{P}_T$), we optimize the classifier without imposing strict constraints on the inter-view relationship learning. Conversely, if $\hat{P}_T$ indicates inaccuracies (a high value of $(1 - \hat{P}_T)$), we optimize both the classifier and the inter-view relationship learning for improved performance. Considering an exponential decay rate for the moment estimate, $\gamma \in [0, 1)$, Algorithm 1 presents pseudo-code for computing reg during training epochs $t \in [0, MAX\_EPOCH)$. $OT(Z_1, Z_2)$ represents

our view-wise structure learning strategy based on the optimal transport approach. We provide more details in Section 3.3.3.

Algorithm 1: An algorithm to compute reg in ASL.

Input: Exponential decay rate for the moment estimate, $\gamma \in [0, 1)$

Output: reg value at training epoch t

for $t \in [0, MAX\_EPOCH)$ **do**
  Get the predicted probability of being the true class $\hat{P}_T$ from the $\hat{P}$;
  if *time step* $t = 0$ **then**
    Compute: $reg \leftarrow (1 - \hat{P}_T)$;
  else
    Compute: $reg \leftarrow \gamma \cdot reg + (1 - \gamma) \cdot (1 - \hat{P}_T)$;
  end
  $t \leftarrow t + 1$;
end

As a result, the objective function of the proposed method is:

$$\min \mathcal{L} \triangleq \mathcal{L}_0 + \lambda \cdot \mathcal{L}_{ASL}(Z_1, Z_2) \ , \tag{3.4}$$

with $\lambda \in (0, 1)$ being a hyper-parameter.

3.3.3 Optimal Transport-based Approach

Inter-view relationships are often overlooked in existing two-view GNNs. Due to unshared weights and late fusion, labeling variability and ambiguous node correspondences can challenge the unified representation. As such, the view-wise structure learning strategy in two-view GNNs is desired and must satisfy the following conditions: 1) graph-level representations from different views should not be assigned to different classes if they represent the same object, 2) corresponding nodes between views must have stronger connections than non-corresponding nodes, and 3) the strategy should be efficiently solvable using stochastic gradient descent.

To this end, we leverage the idea of optimal transport as the view-wise structure learning strategy to learn the inter-view relationships. The optimal transport metric based on Z_1 and Z_2 is calculated as,

$$OT(Z_1, Z_2) = \sum_{i \in Z_1} \sum_{j \in Z_2} c_{ij} \tilde{f}_{ij} \ , \tag{3.5}$$

where c_{ij} is the pairwise distance and $\tilde{f}_{ij}$ is the optimal transport (structural learning) flow between corresponding node u and v. We now introduce the computational specifics of $OT(Z_1, Z_2)$ and provide a detailed discussion of its role as a view-wise structure learning strategy in two-view GNNs. This strategy addresses the three key needs outlined in the previous section.

The first step involves establishing node correspondences between views. This process hinges on a distance matrix that represents the dissimilarities between nodes. The dimensions of this matrix depend on the number of nodes present in both views. We employ the cosine distance metric to compute c_{ij} in Eq. (3.5),

$$\begin{aligned} c_{ij} &= 1 - cos(\boldsymbol{z}_{1i}, \boldsymbol{z}_{2j}) \ , \\ &= 1 - \frac{\boldsymbol{z}_{1i}^T \boldsymbol{z}_{2j}}{\|\boldsymbol{z}_{1i}\| \|\boldsymbol{z}_{2j}\|}, \quad \forall i, j \in \mathbb{R}^N \ , \end{aligned} \tag{3.6}$$

where $\boldsymbol{z}_{1i} \in \mathbb{R}^{d_{\theta_3}}$ is the feature vector for node u in Z_1 and $\boldsymbol{z}_{2j} \in \mathbb{R}^{\theta_3}$ is the feature vector for node v in Z_2. Smaller values of c_{ij} signify the correspondence between nodes (i, j) and the intention of a structural learning flow between them for shared knowledge preservation.

After successfully identifying corresponding nodes using the distance matrix, the next factor to consider is the node weights, which play a crucial role in determining the structural learning flow between these corresponding nodes. To illustrate this, let's consider a practical example: imagine we have movies A, B, and C, with pairs (A, B) and (A, C) representing corresponding movies. Now, when recommending a movie to a user based on their preference for movie A, it makes more sense to prioritize the more popular (or possibly other specific merits) movie if both candidate movies have the same similarity to movie A. Therefore, in cases where distances are equal within a set of corresponding nodes, nodes with higher weights should receive a greater

allocation of structural learning flow to their corresponding nodes, particularly if those corresponding nodes also possess higher weights.

Unlike the approach used in [34, 35], which assigns a uniform weight to all nodes in node classification tasks, we incorporate two essential factors when determining node weights in graph classification tasks. First, we take into account each node's feature vector. Each node is assigned a feature score (FS) that summarizes its feature vector, defined as $FS = \sum_{d_{\theta_3}}(Z.)$. Second, to facilitate fusion for a more grounded unified representation, we assess each node's contribution score (CS) to the unified representation based on the chosen fusion technique. In the case of,

- View-wise maximum pooling operation [36], which selects the maximum values element-wise from Z_1 and Z_2, i.e., $(H_{uni})_{i,o} = \max((Z_1)_{i,o}, (Z_2)_{i,o})$. In this operation, we assign a contribution score of one to the selected element in the unified representation. Consequently, the total contribution score of node i in Z_1 and Z_2 is computed as follows:

$$\begin{aligned}(CS_{Z_1})_i &= \sum_{o=1}^{d_{\theta_3}} 1_{[(Z_1)_{i,o} \geq (Z_2)_{j,o}]}, \quad \forall i \in Z_1 \ , \\ (CS_{Z_2})_i &= \sum_{o=1}^{d_{\theta_3}} 1_{[(Z_1)_{i,o} < (Z_2)_{j,o}]}, \quad \forall i \in Z_2 \ ,\end{aligned} \tag{3.7}$$

  where on every feature dimension, $(CS_{Z_1})_{io} + (CS_{Z_2})_{io} = 1$.

- Concatenation ($H_{uni} = Z_1||Z_2$) or Hadamard product ($H_{uni} = Z_1 \odot Z_2$) operation, where all features are fused into the unified representation. Hence, we assign a contribution score of 0.5 to every element. Consequently, the total contribution score of node i in Z_1 and Z_2 is equal to $(CS)_i = \sum_{o=1}^{d_{\theta_3}} \frac{1}{2}$. Still, on every feature dimension, $(CS_{Z_1})_{io} + (CS_{Z_2})_{io} = 1$.

Next, we proceed to normalize the score $(CS)_i$ of each node to the value of the feature dimension d_{θ_3} in the final layer of the GNN:

$$(\widehat{CS})_i = \frac{(CS)_i}{d_{\theta_3}} \ . \tag{3.8}$$

As a result, a node weight is calculated as follows:

$$w = \max\{0, FS \times \widehat{CS}\} \ . \tag{3.9}$$

Here, we employ the $\max(\cdot)$ function to ensure non-negative weights.

Given the two crucial components, distances c and weights w, the optimal transport can determine which nodes should engage in structural learning with one another and establish the magnitude of the structural learning flow between them. This can be accomplished by solving a linear programming problem to obtain the optimal transport (structural learning) flow $\tilde{f}$ that is disclosed in Eq. (3.5). Consider nodes i and j in Z_1 and Z_2 with weights w_i and w_j respectively, $\tilde{f}_{ij}$ can be solved as follows:

$$\underset{f_{ij}}{\text{minimize}} \sum_{i \in Z_1} \sum_{j \in Z_2} c_{ij} f_{ij} \ , \tag{3.10}$$

$$\begin{aligned} \text{s.t} \quad & f_{ij} \geqslant 0, \ \sum_{i \in Z_1} f_{ij} \leqslant w_j, \ \sum_{j \in Z_2} f_{ij} \leqslant w_i, \\ & \sum_{i \in Z_1} \sum_{j \in Z_2} f_{ij} = min\{\sum_{i \in Z_1} w_i, \sum_{j \in Z_2} w_j\}. \end{aligned}$$

This gives us, $\tilde{f}_{ij} > 0$ if nodes should engage in structural learning, otherwise, $\tilde{f}_{ij} = 0$. After obtaining the optimal structural learning flow $\tilde{f}_{ij}$, the optimal transport between Z_1 and Z_2 is calculated as $OT(Z_1, Z_2) = \sum_{i \in Z_1} \sum_{j \in Z_2} c_{ij} \tilde{f}_{ij}$.

The objective function in Eq. (3.4) is a bilevel optimization problem, where the outer-level optimization problem is to optimize model parameters of the graph neural network, and the inner-level one is to optimize the structural learning flow between corresponding nodes for computing the optimal transport. To optimize this challenging problem, we design an end-to-end training strategy that efficiently solves it using the stochastic gradient descent algorithm. Since it is a convex optimization problem, we can solve it with an existing convex solver efficiently in the forward pass. However, it is challenging to compute the gradient in the backward pass. In particular, when computing the gradient of the loss function $\mathcal{L}$ with respect to the model parameter θ, we need to compute $\frac{\partial \tilde{f}(\theta)}{\partial \theta}$. Therefore, to enable the end-to-end training, we should compute $\frac{\partial \tilde{f}}{\partial \theta}$ efficiently.

To address this challenge, we construct the Lagrangian function for Eq. (3.10) as follows:

$$\mathcal{J}(f(\theta), \eta, \nu) = c^T f(\theta) + \eta^T P(f(\theta)) + \nu^T Q(f(\theta)) \ , \tag{3.11}$$

where $\eta > 0$ and ν are dual variables, $P(f(\theta))$ denotes the inequality constraints, and $Q(f(\theta))$ represents the equality constraints. Then, according to the KKT condition, i.e., $G \triangleq \nabla \mathcal{J}(\tilde{f}(\theta), \tilde{\eta}, \tilde{\nu}) = 0$ where $\tilde{f}(\theta)$, $\tilde{\eta}$, and $\tilde{\nu}$ denotes the optimal solution of Eq. (3.11), it is easy to obtain $\frac{\partial \tilde{f}}{\partial \theta}$ by taking the gradient of G with respect to θ, which is shown as follows:

$$\frac{\partial \tilde{f}(\theta)}{\partial \theta} = -(\frac{\partial G}{\partial \tilde{f}(\theta)})^{-1} \frac{\partial G}{\partial \theta} \ , \tag{3.12}$$

where the right-hand side is easy to compute based on Eq. (3.11) (See Eqs.(9-10) in [37]). As a result, by plugging this step into the backpropagation procedure, we train our graph neural network in an end-to-end manner. In essence, we enable inter-view relationship awareness before the fusion layer, tackle the fusing challenges, and implicitly enhance the unified representation for better performance in the down-stream task.

3.3.4 Interpretation of the Optimal Transport-based Approach

Let $\Pi = \{(i, j) | \tilde{f}_{ij} > 0\}$ be the set of corresponding nodes that should engage in structural learning through the solution of $OT(Z_1, Z_2)$. The unsupervised learning objective of $OT(Z_1, Z_2)$ in Eq. (3.4) is aiming at minimizing the distances among the corresponding nodes in Π. This is because only $(i, j) \in \Pi$ exhibit meaningful structural learning flows, as $\tilde{f}_{ij} > 0$.

Moreover, minimizing $OT(Z_1, Z_2)$ can resolve the issue of two concerns in existing two-view GNNs. First, as Z_1 and Z_2 are the inputs of OT, minimizing $OT(Z_1, Z_2)$ is equivalent to the minimization of distance between Z_1 and Z_2. This offers a resolution to the issue of the two modalities falling apart within a given feature space. Second, minimizing the distances among the nodes in Π implicitly mitigates the challenge of potential loss of shared knowledge between modalities.

3.4 Empirical study

To assess the effectiveness of our proposed optimal transport-based adaptive view-wise structure learning approach for two-view GNNs, we conducted extensive experiments using six benchmark graph classification datasets. We first provide details about datasets and experimental setup. Then, we present graph classification results against state-of-the-art approaches across the six benchmark datasets. Lastly, we offers a comparative analysis of our proposed method.

3.4.1 Datasets and Experimental Setup

Datasets. We use six benchmark datasets for graph classification from TU-Datasets [38]. In our dataset selection, we span diverse domains, including the recognition of small molecular networks such as MUTAG, BZR_MD, PTC_MR, and ER_MD. We also delve into the realm of computer vision by considering the recognition of Cuneiform signs, and we explore brain disease prediction, as exemplified by KKI. Detailed statistics for these datasets are provided in Table 3.1.

In dataset preparation, we begin with single-modality datasets $\mathcal{G} = (X, A)$. We use them as the first modality $\mathcal{G}_1 = (X, \mathcal{L}_1(A))$. To create the second modality, we take a different approach compared to techniques like node permutation [34] or the

Table 3.1.: Statistics of the graph classification datasets used in ASL for two-view GNNs.

Datasets	**Graphs**	**Classes**	**Avg. Vertices**	**Features**
MUTAG	188	2	17.93	7
BZR_MD	306	2	21.30	8
PTC_MR	344	2	14.29	18
ER_MD	446	2	21.33	10
Cuneiform	267	30	21.27	3
KKI	83	2	26.96	190

addition of noise [39], which are commonly used in MVGNNs for node classification tasks. Instead, we generate a diverse graph topology $\mathcal{S}$ based on the Mahalanobis distances matrix ($\mathcal{D} \in \mathbb{R}^{N \times N}$) between node features with a randomly generated transformation matrix ($M \in \mathbb{R}^{f_0 \times f_0}$), and normalize it by a standard Gaussian distribution [24]. Such that, $\mathcal{S} = exp(-\mathcal{D}/2)$, with $d(x_i, x_j) = \sqrt{(x_i - x_j)^T M (x_i - x_j)}$, for all $d(x_i, x_j) \in \mathcal{D}$. As a result, the second input graph $\mathcal{G}_2 = (X, \mathcal{L}_2(A, S))$ is constructed, featuring a semi-synthetic graph topology.

Implementation Details and Configurations. We adopt a data partitioning approach following prior studies [24, 40, 41]. Our methodology incorporates a 10-fold cross-validation technique. During training and evaluation of our proposed method, we utilize a mini-batch size of 32. The data preprocessing steps are consistent with those outlined in part '*Datasets*'.

In line with the network architecture discussed in Section 3.3.1, each Graph Neural Network (GNN) in our model consists of three graph convolutional layers. These layers have embedding dimensions set as $d_{\theta 1} = 16, d_{\theta 2} = 64, d_{\theta 3} = 128$. ReLU activation functions and dropout layers with a rate of 0.1 follow each convolutional layer. We employ a convolution operation as described in ChebNet [13], with a Chebyshev degree (K) of 6, or in GCN [14]. The fusion layer utilizes modality-wise max pooling to generate a unified representation. In the compatibility learning layer, we apply a linear layer with an embedding dimension of 128, followed by a ReLU activation layer and a dropout layer with a rate of 0.1. The read-out layer employs graph-level max pooling among the nodes to create the graph-level representation for class prediction.

Parameter tuning of the moment estimate parameter γ (in Algorithm 1) with a range of $\{0.01, 0.02, \ldots, 0.99, 1.\}$, and the hyperparameter λ in Eq.(3.4) over the values $\{5e^{-2}, 1e^{-3}, 5e^{-3} \ldots, 1e^{-5}, 5e^{-5}\}$.

We evaluate the method's performance using classification accuracy as the primary metric. We employ the Adam optimizer with a learning rate of $5e-3$ and train for a total of 200 epochs (T = 200). To solve the Linear Programming (LP) problem in the optimal transport metric, we leverage the GPU-accelerated convex optimization solver QPTH [42] and compute gradients during the backward pass.

3.4.2 Comparison with State-of-the-art Methods

The experimental results for graph classification are presented in Table 3.2. We observe that our method consistently outperforms all baseline methods across all datasets. In the following, we analyze the results in two scenarios, i.e., single-view and two-view.

Table 3.2.: Comparison of graph classification results for ASL in two-view GNNs against baseline methods.

	Datasets					
Method	MUTAG	BZR_MD	PTC_MR	ER_MD	Cuneiform	KKI
GIN (1st view) [17]	90.5 ± 7.4	66.3 ± 11.3	71.5 ± 8.2	68.1 ± 6.1	-	80.6 ± 17.6
InfoGraph (1st view) [43]	91.4 ± 0.1	79.4 ± 0.1	72.1 ± 0.1	78.2 ± 0.1	88.4 ± 0.1	65.6 ± 0.2
ChebNet (1st view) [13]	91.5 ± 5.8	78.7 ± 6.5	70.4 ± 5.0	78.0 ± 4.4	87.9 ± 5.1	78.9 ± 12.6
ChebNet (2nd view) [13]	92.6 ± 4.8	79.0 ± 8.7	69.0 ± 4.9	76.9 ± 4.6	87.9 ± 5.1	81.1 ± 13.2
ChebNet-Multigraph [40]	93.1 ± 4.7	81.6 ± 7.2	73.6 ± 2.7	79.6 ± 4.2	88.7 ± 6.0	83.3 ± 11.4
ChebNet-MVAGC [24]	93.6 ± 4.6	81.4 ± 7.3	72.2 ± 4.7	80.6 ± 4.0	88.3 ± 6.2	81.1 ± 12.2
ChebNet+ASL (ours)	**94.7 ± 4.7**	**82.6 ± 7.5**	**74.8 ± 4.5**	**81.2 ± 3.4**	**89.5 ± 6.0**	**86.7 ± 12.9**
GCN (1st view) [14]	89.9 ± 4.9	79.3 ± 6.5	73.4 ± 6.8	79.8 ± 4.2	88.7 ± 6.5	82.2 ± 11.3
GCN (2nd view) [14]	89.9 ± 4.9	78.7 ± 6.8	72.6 ± 7.8	79.8 ± 3.1	88.7 ± 5.6	78.9 ± 16.1
GCN-Multigraph [40]	91.5 ± 3.4	79.4 ± 7.3	74.3 ± 4.7	79.0 ± 4.0	88.4 ± 5.4	81.1 ± 12.2
GCN-MVAGC [24]	91.0 ± 4.1	79.7 ± 5.4	72.2 ± 5.6	81.0 ± 4.1	89.1 ± 5.4	84.4 ± 10.2
GCN+ASL (ours)	**91.5 ± 4.2**	**82.7 ± 6.5**	**74.6 ± 7.3**	**82.5 ± 3.3**	**89.5 ± 4.9**	**85.6 ± 10.0**

In single-view scenario, baselines include GIN (1st view), InfoGraph (1st view), ChebNet (1st view), GCN (1st view), ChebNet (2nd view),and GCN (2nd view).

First, we evaluate the performance of ChebNet and GCN, both renowned as foundational methods in the GNN domain. Their popularity arises from the simplicity of construction and efficient learning, especially in the case of GCN. Our findings demonstrate that the classification accuracy achieved by ChebNet and GCN in learning either the 1st or 2nd view comparable with that of other methods, namely GIN and InfoGraph when handling the 1st modality. This highlights our rationale for

selecting ChebNet and GCN as the base methods in the GNN component of our proposed method.

Second, we compare our method with single-view methods. Our observations indicate that the classifiers trained by our methods, ChebNet+ASL and GCN+ASL, consistently outperform those of the single-view methods. This evidence strongly suggests that a unified representation, as trained by our approach, is better for meeting downstream tasks compared to view-specific single-view representations.

In two-view scenario, baselines include ChebNet-Multigraph, ChebNet-MVAGC, GCN-Multigraph, and GCN-MVAGC.

To demonstrate the effectiveness of the proposed adaptive view-wise structural learning, we conduct a comparative analysis against other two-view GNNs, namely Multigraph and MVAGC. We categorize the compared methods into two groups: ChebNet-like and GCN-like, aligning with the respective graph convolution operation employed. The results of our method consistently outperforms both Multigraph and MVAGC. This performance improvement ranges from 0.4% in the Cuneiform dataset using GCN-like methods to a substantial 5% increase in classification accuracy for the KKI dataset using ChebNet-like methods. Our findings strongly support the effectiveness of adaptive view-wise structural learning in two-view GNNs, leading to enhanced unified representation and improved classification results.

3.4.3 Method Analysis

In this section, we conduct various experiments to evaluate the effectiveness of our method by exploring multiple design variants, including the selection of inter-view distance metrics and the adaptive effects.

We conducted a performance comparison of inter-view distance computation when employing alternative distance metrics, including Manhattan distance, Euclidean distance, and cosine distance, as replacements for the optimal transport metric in Eq. (3.5). It is important to emphasize that these alternative distance metrics assume a constant graph structure, which leads them to solely measure the distance between

two views without accounting for the identification of corresponding nodes. Furthermore, these alternatives operate under the assumption of a uniform structural learning flow between nodes, without considering feature importance or the significance of contribution to the unified representation. As depicted in Table 3.3, none of the alternative distance methods demonstrated superior classification performance compared to optimal transport. This comparison demonstrates the effectiveness of optimal transport in both computing view-wise distances and serving as the foundation of our view-wise structure learning strategy. Optimal transport's ability to identify corresponding nodes and allocate the optimal structural learning flow between them is a key factor in the success of our approach.

Table 3.3.: Ablation study of different distance metrics for ASL in two-view GNNs.

	Distance Metrics			
Datasets	Manhattan	Euclidean	cosine	Optimal transport
MUTAG	88.9 ± 5.4	88.9 ± 4.9	88.9 ± 4.9	$\mathbf{91.0 \pm 4.1}$
BZR_MD	80.0 ± 5.9	81.1 ± 6.8	81.1 ± 6.4	$\mathbf{81.7 \pm 5.8}$
PTC_MR	67.8 ± 5.0	71.7 ± 6.2	72.0 ± 5.1	$\mathbf{75.3 \pm 7.9}$
ER_MD	78.1 ± 3.1	79.2 ± 5.3	79.4 ± 3.9	$\mathbf{80.1 \pm 2.9}$
Cuneiform	83.5 ± 11.8	83.9 ± 9.1	85.1 ± 10.3	$\mathbf{86.5 \pm 8.3}$
KKI	$\mathbf{83.3 \pm 14.3}$	82.2 ± 10.2	80.0 ± 10.9	$\mathbf{83.3 \pm 10.2}$

We also investigate the influence of the adaptive effect designed in the objective function, i.e., the *reg* term in Eq.(3.3). As discussed in Section 3.3.2, the proposed optimal transport-based AMoSL is adapted to the classifier's performance. However, we also explore a radical scenario where the adaptive effect is eliminated by setting the regularization parameter reg=1, allowing unconditional unsupervised learning on structure distance. Table 3.4 presents the results of structure distance and model performance under this radical scenario. We use ✗to represent reg = 1, for methods without the adaptive effect, and ✓to represent methods with the adaptive effect. In this experiment, the network is set up to use the ChebNet graph convolution opera-

Table 3.4.: Ablation study of the adaptive effectiveness for ASL in two-view GNNs .

Datasets	**Fusion techniques**	**Distance**		**Accuracy**	
		✗	✓	✗	✓
MUTAG	max	3.75	14.73	89.4 ± 6.2	**91.0 ± 4.1**
	concat	4.87	11.35	91.0 ± 5.8	**91.5 ± 4.8**
BZR_MD	max	4.00	14.14	81.1 ± 6.4	**81.7 ± 5.8**
	concat	4.44	12.55	81.1 ± 6.0	**82.4 ± 7.0**
PTC_MR	max	24.25	29.96	72.0 ± 5.5	**75.3 ± 7.9**
	concat	4.76	37.84	71.1 ± 4.7	**71.6 ± 6.0**
ER_MD	max	3.19	13.39	78.3 ± 4.4	**80.1 ± 2.9**
	concat	3.75	11.32	78.7 ± 4.2	**79.6 ± 5.0**
Cuneiform	max	24.92	30.83	84.6 ± 11.3	**86.5 ± 8.3**
	concat	20.11	14.70	86.5 ± 6.5	**88.0 ± 8.3**
KKI	max	13.96	35.86	82.2 ± 12.4	**83.3 ± 10.2**
	concat	11.26	37.35	84.4 ± 11.3	**85.6 ± 13.2**

tion with $K = 1$, while other configurations remain consistent with those described in Section **??**. When structure distance learning is always allowed, we observe significantly small structure distances, which excessively emphasize the importance of minimizing distances between view representations. This converges the views too close in the feature space and hinders the unified representation from utilizing complementary knowledge from different views. As a result, none of the radical scenarios can train classifiers that outperform the proposed method. This underscores the significance of maintaining a balance between views, thus emphasizing the importance of the adaptive design in the proposed method.

3.5 Chapter Summary

In this chapter, we propose a two-view GNN framework to mitigate the graph structure bias inherent in single-view GNNs. Two-view GNNs, posits that more informative nodes should have proximal node representations within a graph structure constructed on such attributes. We reconstruct a new graph structure based on the proximity of node representations and simultaneously learn a graph object from both the new constructed and default graph structures for relationship reasoning. Additionally, by introducing an adaptive view-wise structure learning strategy, we aim to align view-wise representations and effectively learn inter-view relationships.

Through extensive evaluation, we demonstrate the effectiveness of our approach. The improved graph classification results highlight the benefits of learning from augmented structures, which provide more informative local information. Additionally, our adaptive view-wise structure learning strategy outperforms existing two-view GNNs.

CHAPTER 4
CAUCHYGCN: PRESERVING LOCAL SMOOTHNESS IN GRAPH CONVOLUTIONAL NETWORKS

In Graph Convolutional Networks (GCNs), a message-passing scheme explicitly learns and reasons node representations via aggregation and propagation of neighboring information over the default graph structure. Most existing message-passing schemes are grounded in Laplacian smoothing, which seeks to maintain the similarity of node representations in the hidden feature space (local smoothness) among neighboring nodes, ensuring their labeling consistency (global smoothness). This often leads to Laplacian smoothing imposing strict penalization on distant neighbors. Because some distant neighbors are inter-class or represent some necessary intra-class patterns, strict penalization of distant neighbors can fail to preserve local smoothness effectively as expected thus introducing noise, mixing representations, and failing to capture valuable hidden patterns. Although recent research has introduced various strategies, including graph filters, k-hop jumps, and bounded penalties to tackle this issue, these methods often fall short of explicitly capturing and preserving the local smoothness over the default graph structure. This chapter presents CauchyGCN, which enhances the preservation of local smoothness. CauchyGCN comprises two key components: 1) a Cauchy smoothing message-passing scheme that explains and preserves local smoothness in each hidden layer, and 2) an unsupervised clustering analysis that simultaneously improves the classifier's capacity to learn both local and global smoothness. We conduct comprehensive experiments using five benchmark datasets to assess the performance of CauchyGCN in semi-supervised node classification tasks compared to state-of-the-art GCNs.

4.1 Introduction

While significant advancements have been made in message-passing schemes, many still rely heavily on Laplacian smoothing [44,45]. Laplacian smoothing operates under the assumption that neighboring nodes belong to the same class, and it enforces strict

penalization on distant neighbors intending to maintain the proximity of representations among neighboring nodes (local smoothness) to ensure labeling consistency (global smoothness). However, the strict penalization of distant neighbors often results in Laplacian smoothing being less effective than expected at preserving local smoothness. Several factors contribute to this ineffectiveness. First, distant neighbors could belong to different classes [46]. For instance, in widely used graph datasets like Cora [47] and CiteSeer [48], the inter-class neighbors account for approximately 19% and 26%, respectively. Message passing between inter-class neighbors introduces noise and leads to the mixing of representations [49, 50]. Second, some distant neighbors may represent critical intra-class patterns in sparsely connected regions of the graph [22]. Excessive penalization of these patterns fails to capture valuable hidden proximal patterns in local smoothness. Designing a new message-passing scheme beyond Laplacian smoothing is thus essential to solving these issues.

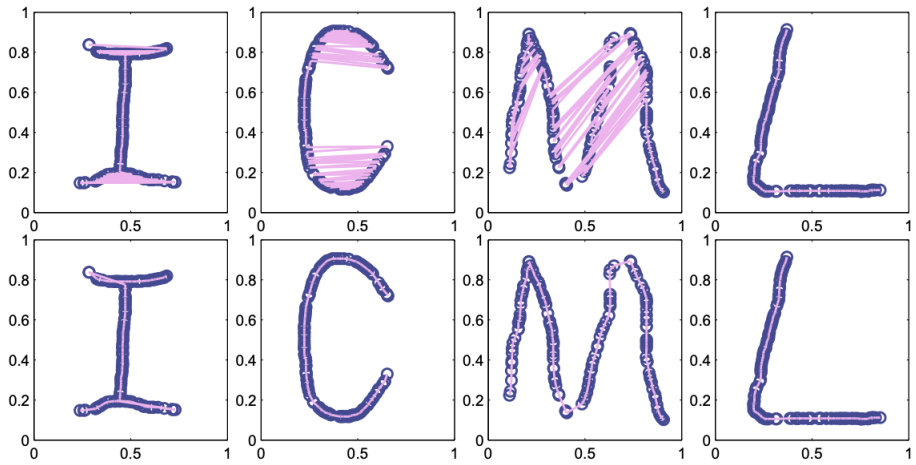


Figure 4.1.: Comparison of the smoothness preservation between Laplacian smoothing (top)and Cauchy smoothing (bottom) [51].

To tackle the abovementioned challenges in Laplacian smoothing-based GCNs, we propose CauchyGCN which focuses on capturing and preserving local smoothness in an interpretable approach. CauchyGCN achieves local smoothness preservation from two key components in GCNs: the layer-wise message-passing scheme and unsuper-

vised clustering analysis. Specifically, we design Cauchy smoothing, which leverages the desired properties of Cauchy distribution, to capture and preserve the proximal patterns and similarity relationships over the underlying graph structure. In contrast to the emphasis on the variant of the distant neighbors in Laplacian smoothing, Cauchy smoothing emphasizes the variant of the close neighbors. Figure 4.1 illustrates the comparison of smoothness preservation between Laplacian smoothing and Cauchy smoothing. We further combine Cauchy smoothing and Laplacian smoothing and strike a balance between them as a new message-passing scheme in CauchyGCN. Our message-passing scheme reduces noise that might lead to the mixing of inter-class node representations and mitigates the penalization of non-smooth intra-class variations, while also enhancing the capture and preservation of valuable local smoothness, encompassing both proximal patterns and similarity relationships. Moreover, we introduce an end-to-end unsupervised clustering analysis, which simultaneously enhances global smoothness and improves both inter-cluster distinction and intra-cluster cohesion.

Our main contributions are as follows: 1) This is the first approach of leveraging Cauchy distribution to preserve local smoothness in GNNs through a new design message-passing scheme. 2) We jointly learn node representations and clustering analysis to improve both local and global smoothness. 3) Extensive experiments demonstrate that CauchyGCN achieves competitive performance in semi-supervised node classification compared to existing methods.

4.2 Related Work

In this section, we present the work most closely related to ours, including an overview of graph convolutional layers, message-passing schemes, and advancements in message-passing GNNs.

4.2.1 Graph Convolutional Layer

Considering a node representation x_i, a general graph convolutional layer in GCNs involves projection and propagation layers [52],

$$\begin{aligned} x_i^{(k)} &= x_i^{(k-1)}\Theta^{(k)} \qquad \text{(Projection)} \ , \\ h_i^{(k)} &= f^{(k)}\left(x_i^{(k)}, x_{\mathcal{N}_i}^{(k)}\right) \quad \text{(Propagation)} \ , \end{aligned} \tag{4.1}$$

where $k \in [1, K]$ stands for the layer index with K being the last layer. Here, the projection layer updates the feature vector of node i, denoted as $x_i^{(k-1)} \in \mathbb{R}^{d_{\theta_{k-1}}}$, from the previous layer $k-1$ using a layer-wise trainable projection matrix $\Theta^{(k)} \in \mathbb{R}^{d_{\theta_{k-1}} \times d_{\theta_k}}$ to produce a new feature vector, $x_i^{(k)} \in \mathbb{R}^{d_{\theta_k}}$. In this context, $d_{\theta_{k-1}}$ and d_{θ_k} represent the hidden feature spaces in layers $k-1$ and k, respectively. The propagation layer employs a specified message-passing scheme $f(\cdot)$ to further update $x_i^{(k)}$ by incorporating neighboring information $x_{\mathcal{N}_i}^{(k)}$ from the current feature space, while $h_i^{(k)}$ represents the output feature vector of node i. The corresponding layer-wise propagation rule [14],

$$\mathcal{O} = \arg\min_{h_i^{(k)}} \underbrace{\|h_i^{(k)} - x_i^{(k-1)}\|^2}_{\mathcal{L}_0} + \lambda \underbrace{F^{(k)}\left(x_i^{(k)}, x_{\mathcal{N}_i}^{(k)}\right)}_{\mathcal{L}_{reg}}, \tag{4.2}$$

optimizes the network to capture significant graph relations within the current hidden feature space. In Eq. (4.2), $\mathcal{L}_0$ enforces the network to maintain label consistency by controlling the distance between the node representation in the two consecutive layers (layer $k-1$ and layer k), which can be beneficial for tasks like node classification. $\mathcal{L}_{reg}$ with the hyperparameter $\lambda \in [0, 1]$ ensures that the message-passing process adheres to certain smoothness patterns and denoising constraints. In the forward pass, obtaining an optimal or sub-optimal solution for $h_i^{(k)}$ via $\mathcal{L}_{reg}$ requires an appropriate optimization technique for node feature vector $x_i^{(k)}$. For instance, when $F^{(k)}$ is convex and differentiable, the message-passing scheme can be represented as $f(x_i^{(k)}) = \frac{\partial F^{(k)}}{\partial x_i^{(k)}}$. In the following sections, we use X (equivalent to $x^{(k-1)}$) to denote the input node feature matrix and H (equivalent to $h^{(k)}$) to denote the output node feature matrix in layer k.

4.2.2 Laplacian smoothing-based GCNs

Laplacian smoothing [44, 45] is designed to enhance the smoothness of node representation over the graph structure, where the magnitude of $tr(H^T\tilde{L}H)$ is largely influenced by the differences between distant neighbors. Laplacian smoothing is defined as,

$$tr(H^T\tilde{L}H) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\hat{\tilde{a}}_{ij}(h_i - h_j)^2 \ , \tag{4.3}$$

where h_i is the feature vector of node i, corresponding to the i-th row of the ndoe feature matrix H. $\{\hat{\tilde{a}}_{ij}\}_{1\leq i,j\leq N}$ are entries of the normalized adjacency matrix with self-loop effect (see definition in Section 2.1). When applied to GCNs, let $\mathcal{L}_{reg}$ in Eq.(4.2) adheres to Laplacian smoothing [14], such that,

$$\mathcal{O}_{\text{GCN}} = \arg\min_{H} \|H - X\|^2 + \lambda \cdot tr(H^T\tilde{L}H) \ . \tag{4.4}$$

This encourages the local smoothness of node representations during the message-passing process at each layer, with particular emphasis on the smoothness between distant neighbors (see Proposition Laplacian smoothing). Eq. (4.4) corresponds to the propagation rule in GCN [14], denoted as $\mathcal{O}_{\text{GCN}}$, and stands as the fast and vanilla benchmark in realm of GCNs.

The proposition of Laplacian-smoothing [45] shows as the following: Laplacian-smoothing focuses on preserving local smoothness between the node i and its distant neighbor j during gradient descent. The update rule is given by: $(h_i - h_j)^* = (h_i - h_j) - 2\eta\tilde{L}(h_i - h_j)$, where η is the learning rate. The correction term $2\eta\tilde{L}(h_i - h_j)$ penalizes the difference between the node representations of node i and j. Consequently, the strict penalization is imposed on the substantial difference between h_i and h_j.

The message-passing scheme corresponding to the propagation rule in Eq. (4.4) can be obtained by performing a single gradient descent step at X [14],

$$\begin{aligned} H^*_{\text{GCN}} &= X - \eta\frac{\partial\mathcal{O}_{\text{GCN}}}{\partial H}|_{H=X} \\ &= x - \eta\left(2\left(H - X\right) + 2LH\right) \\ &= (D^{-1/2}\tilde{A}D^{-1/2} + L - 2\eta L)X = \hat{\tilde{A}}X \ , \end{aligned} \tag{4.5}$$

with a step size of $\eta = \frac{1}{2}$. As Laplacian smoothing represents a smooth and convex l_2-based regularization, the updated node representation H^* can be obtained as a closed-form solution.

4.2.3 Other Advancements

Because neighboring nodes can belong to different classes or be distantly embedded, the fast and vanilla framework offered by GCN [14] introduces noise and mixes representations during the message-passing process as it assumes the constant importance of neighboring information. To tackle this issue, a line of studies relax this extreme assumption and strive to preserve local smoothness based on the original data's locality from either spectral or spatial approaches.

Spectral approaches are theoretically based on graph signal processing, where advancements delve into refining graph filter definitions to enhance the processing of the graph's frequency domain through graph Fourier transforms. PPNP/ APPNP [9] involves adjusting a node's neighborhood through teleport probability using personalized PageRank [53] on the graph filter. GNN-LF/HF [21] further refine the graph filter in PPNP/APPNP into low-/high-pass filtering kernels for k-hop neighborhoods, thereby enhancing its adaptability to accommodate arbitrary coefficients of polynomial filters by introducing more adjustable factors.

Spatial approaches directly exploit the graph structure, where advancements aim at improving the capture of connectivity among neighboring nodes. GAT [7] leverages attention mechanisms to emphasize the importance between neighbors. GraphSAGE [16] sampling on the neighbors and learns different local patterns through different aggregation methods. JKNet [8] extends its reach beyond 1-hop neighborhoods using hierarchical neighborhood information. UGNN [54] employs the node's intra-class rate as a neighbor-information scalar. ElasticGNN [22] introduces a smoothing strategy based on the l_1-norm with a soft-thresholding operator to restrict distant neighboring information.

Different from these advancements, CauchyGCN distinguishes itself by capturing and preserving local smoothness of the nonlinearity but proximity pattern among closely embedded and mutually connected neighbors, adhering to the Cauchy distribution. We also introduce a weight factor following a Gaussian distribution to explain which neighbors should preserve their local smoothness.

4.3 Preserving Local Smoothness via Cauchy Smoothing

In this section, we begin by introducing Cauchy smoothing for capturing and preserving local smoothness in an interpretable approach through a new message-passing scheme. We then detail the new message-passing scheme, which involves both Cauchy smoothing and Laplacian smoothing and strikes a balance between these two smoothing strategies. After that, we present an end-to-end unsupervised clustering analysis aimed at further improving both global and local smoothness. Lastly, we elucidate the optimization process of CauchyGCN.

4.3.1 Cauchy Smoothing

While Laplacian smoothing prioritizes smoothness between distant neighbors, it overlooks smoothness between closely embedded and more mutually connected neighbors. We propose a complementary strategy to address this limitation and enhance local smoothness.

Cauchy distribution has found applications in various domains [51, 55] to align objects that carry proximal representations,

$$\phi(x, x_0, \sigma) = \frac{1}{\pi}\left[\frac{\sigma}{(x - x_0)^2 + \sigma^2}\right] \ . \tag{4.6}$$

Here, x_0 signifies the location factor, $\sigma > 0$ is a scale parameter, and $\frac{1}{\pi}$ is a constant that is omitted in our proposed method. The key properties of the Cauchy distribution are detailed in the proposition of Cauchy distribution. Considering the distance d_{ij}

between two data points i and j, Figure 4.2 illustrates decay functions, comparing the Cauchy distribution with other distributions. The decay functions include:

$$\begin{aligned} \textit{Cauchy}: &\quad \frac{1}{d_{ij}^2 + \sigma^2} \\ \textit{Laplacian}: &\quad -d_{ij}^2 \\ \textit{Gaussian}: &\quad e^{-d_{ij}^2/\sigma^2} \\ \textit{Exponential}: &\quad e^{-d_{ij}^2/\sigma} \\ \textit{Linear}: &\quad -d_{ij} \end{aligned}$$

As d_{ij} increases, the absolute value of the decay function for the Laplacian and Lin-

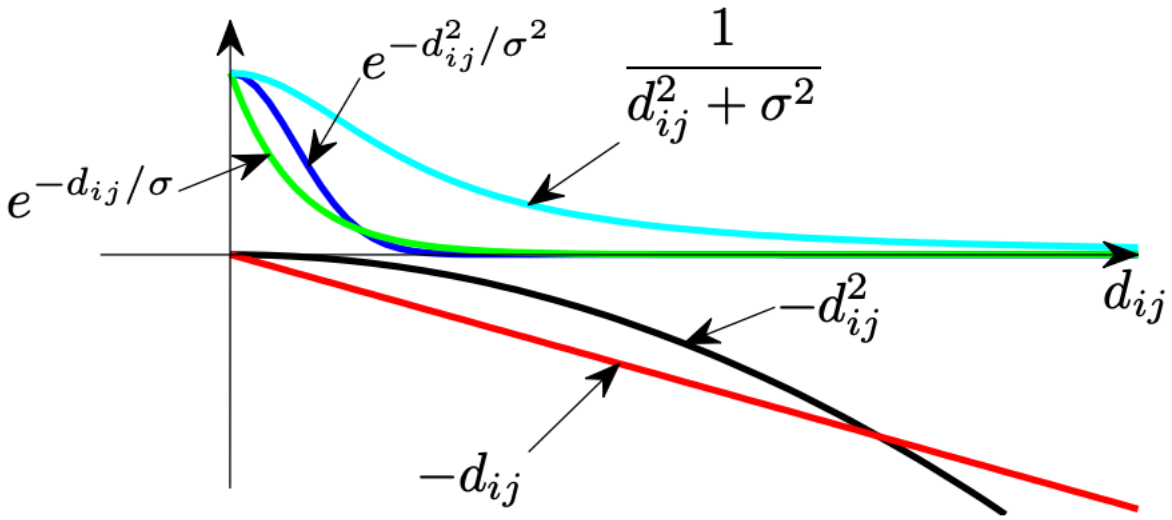


Figure 4.2.: Comparison of decay function between Cauchy distribution and other distributions [51].

ear distribution increases. In contrast, for the Cauchy, Gaussian, and Exponential distributions, smaller values of d_{ij} correspond to larger absolution values of the decay function. This suggests that Cauchy, Gaussian, and Exponential distributions are strong candidates for preserving smoothness between closely embedded data points. Notably, the Cauchy distribution exhibits a smoother transition as the distance be-

tween two data points increases. Moreover, it stands in stark opposition to the Gaussian distribution.

Therefore, we introduce Cauchy-based smoothing strategy to preserve the local smoothness of the nonlinearity but proximity pattern among closely embedded and mutually connected neighbors. Consider a central node i and j as one of its neighboring nodes. Our objective is to maximize the magnitude of the Cauchy distribution, aiming to minimize the distance between their representations (h_i and h_j),

$$\max_{h_i} \frac{\gamma}{(h_i - h_j)^2 + \gamma^2} \ . \tag{4.7}$$

The proposition of Cauchy distribution shows as the following: Let γ represent a positive constant. In the Cauchy distribution ϕ, the maximum occurs at $\frac{1}{\pi\gamma}$ when $x = x_0$. The magnitude of ϕ decreases as the difference between x and x_0 increases.

Additionally, we introduce a weight factor w_{ij} to reason and determine which neighbors should have their local smoothness preserved. This assessment takes into account both the graph structure and the hidden node proximity pattern,

$$w_{ij} = a_{ij} \cdot e^{-d_{ij}/\sigma^2} \ . \tag{4.8}$$

The term a_{ij} reflects the connectivity of neighbors in the graph structure. $d_{ij} = \|h_i - h_j\|^2$ measures the distance between neighboring nodes under the hidden topological pattern, adhering to a Gaussian distribution. Because the Gaussian distribution decays more rapidly than the Cauchy distribution when their parameters γ in Eq. (4.7) and σ in Eq. (4.8) are identical, the weight factor compels Cauchy smoothing to preserve local smoothness among closely embedded neighbors following the Gaussian distribution but ensures proper strict penalizations following the Cauchy distribution.

As a result, the Cauchy smoothing is defined as,

$$\max_{h_i} \frac{w_{ij} \cdot \gamma}{(h_i - h_j)^2 + \gamma^2} \equiv \min_{h_i} \frac{-w_{ij} \cdot \gamma}{(h_i - h_j)^2 + \gamma^2} \ . \tag{4.9}$$

The proposition of Cauchy-smoothing illustrates the insights into preserving local smoothness among closely embedded and mutually connected neighbors via Cauchy smoothing.

The proposition of Cauchy-smoothing shows as the following: Cauchy-smoothing focuses on preserving local smoothness between node i and its close neighbor j during gradient descent. The update rule is given by: $(h_i - h_j)^* = (h_i - h_j) + 2\eta \frac{w_{ij} \cdot \gamma (h_i - h_j)}{((h_i - h_j)^2 + \gamma^2)^2}$, with η representing the learning rate. The correction term $2\eta \frac{w_{ij} \cdot \gamma (h_i - h_j)}{((h_i - h_j)^2 + \gamma^2)^2}$ penalizes the disparity between the node representations of node i and j only when there is a sufficiently small difference between h_i and h_j, with the condition that the weight factor $1 \geq w_{ij} > 0$ follows a Gaussian distribution.

4.3.2 Message-Passing Scheme in CauchyGCN

In terms of preserving local smoothness, Laplacian smoothing emphasizes the variance among distant neighbors, while Cauchy smoothing emphasizes the variance among close neighbors. It is reasonable to incorporate both Laplacian smoothing (Eq. (4.3)) and Cauchy smoothing (Eq. (4.7)) to capture local smoothness from distinct perspectives while necessitating a balancing estimator to help mitigate the strict penalization from either aspect. Therefore, the layer-wise propagation rule in CauchyGCN is defined as:

$$\mathcal{O}_c = \arg\min_H \|H - X\|^2 + \lambda_1 \cdot tr(H^T \tilde{L} H) - \lambda_2 \sum_i \sum_j \frac{w_{ij} \cdot \gamma}{\|h_i - h_j\|^2 + \gamma^2} \ , \qquad (4.10)$$

where $\lambda_1 \in [0, 1]$ and $\lambda_2 \in [0, 1]$ are parameters that scale the penalties from Laplacian smoothing and Cauchy smoothing, respectively.

Now, we need to solve the layer-wise propagation rule in Eq. (4.10) to illustrate the message-passing scheme of CauchyGCN. However, the difficulty is introduced since the two smoothing strategies are coupled by the components of H. To address this issue, we find inspiration in splitting methods [56], particularly those designed for solving optimization problems of the form $f(x) = l(x) + u(x) + \varphi(Ax)$. This problem is equivalent to the one presented in Eq. (4.10). By reformulating the problem, we convert it from a three-function problem to a more manageable two-function form, expressed as $f(x) = \mathcal{L}(x) + \epsilon\varphi(x)$, where $\mathcal{L}(x) = l(x) + u(x)$ and $\epsilon \in [0, 1]$ is a scalar. Following the splitting method, Eq. (4.10) is reformulated to: $\mathcal{L}(H) = \|H - X\|^2 + \lambda_1 \text{tr}(H^T \tilde{L} H)$ and $\varphi(H) = \lambda_2 \sum_i \sum_j \frac{w_{ij} \cdot \gamma}{\|h_i - h_j\|^2 + \gamma^2}$, aiming to find

an optimal solution of H. This allows us to approach the optimization step-by-step. First, we solve the optimization problem in $\mathcal{L}(H)$, which intriguingly aligns with solving the Laplacian smoothing problem outlined in Eq. (4.5). The one gradient descent step at X is:

$$\begin{aligned} F^* &= X - \eta \frac{\partial \mathcal{L}(H)}{\partial H} | H = X \\ &= X - \eta \lambda_1 L X \\ &= \frac{\lambda_1}{(\lambda_1 + \lambda_2)} \hat{\tilde{A}} X + (\frac{\lambda_2}{(\lambda_1 + \lambda_2)} X \ . \end{aligned} \tag{4.11}$$

where the step size $\eta = \frac{1}{2(\lambda_1+\lambda_2)}$. The optimal solution F^* obtained through λ_1-scaled Laplacian smoothing is subsequently utilized to solve the second function, $\varphi(F^*)$. Before delving into the search for the optimal solution of Cauchy smoothing, it is prudent to calculate the derivative of $\varphi(F^*)$ with respect to F^*.

$$Z^* = \frac{\partial \varphi(F^*)}{\partial F^*} = -2\lambda_2 \sum_i \sum_j \frac{w_{ij} \cdot \gamma(f_i^* - f_j^*)}{\left(\|f_i^* - f_j^*\|^2 + \gamma^2\right)^2} \ , \tag{4.12}$$

where w_{ij} has $d_{ij} = \|f_i^* - f_j^*\|^2$. Finally, let us take another gradient descent on F^* for Cauchy smoothing,

$$\begin{aligned} H^* &= F^* - \eta\epsilon \frac{\partial \varphi(F^*)}{\partial F^*} = F^* - \eta\epsilon Z^* \\ &= \lambda \hat{\tilde{A}} X + (1-\lambda) \left(X - \epsilon \sum_i \sum_j \frac{w_{ij} \cdot \gamma(f_i^* - f_j^*)}{\left(\|f_i^* - f_j^*\|^2 + \gamma^2\right)^2} \right) \ , \end{aligned} \tag{4.13}$$

where the step size $\eta = \frac{1}{2(\lambda_1+\lambda_2)}$, $\epsilon \in [0, 1]$ is a scalar from the splitting method, and H^* denotes the output node representation at the current layer.

Let $\lambda = \frac{\lambda_1}{\lambda_1+\lambda_2}$, we introduce a more scalable parameter λ to enhance the equilibrium between the two smoothing strategies. For special cases, when $\lambda_2 = 0$ leads a fully Laplacian smoothing as $\lambda = 1$. Conversely, when $\lambda_1 = 0$ leads a complete Cauchy smoothing as $\lambda = 0$. Furthermore, it is important to note that all node representations in this context are normalized by the square root of the degree matrix ($\tilde{D}$). We denote the distance matrix of all neighboring nodes as ΔH, as such $\Delta H = \sum_{a_{ij} \neq 0} \frac{h_i}{\sqrt{\tilde{d}_i}} - \frac{h_j}{\sqrt{\tilde{d}_j}}$. The graph convolution process in CauchyGCN follows the

general framework of graph convolution outlined in Eq. (4.1), and a concise process is provided in Algorithm 2.

Algorithm 2: Graph convolution in CauchyGCN

Input: Node Representation X; Adjacency Matrix $\hat{\tilde{A}}$; Layer-specific trainable weight matrix Θ; Activation Function $\sigma(.)$; Hyper-parameters $\lambda \in [0, 1]$, $\gamma \geq 0$, and $\epsilon \in [0, 1]$

Output: Output Node Representation H

Initialize $k \leftarrow 1$, $X^{(0)} \leftarrow X$, $\Theta^{(1)} \leftarrow \Theta$;

for $k \in [1, K]$ **do**

  Projection layer: $X^{(k)} \leftarrow X^{(k-1)}\Theta^{(k)}$

  if $\lambda \neq 0$ **then**

    Laplacian smoothing: $F^{(k)} \leftarrow \lambda\hat{\tilde{A}}X^{(k)} + (1-\lambda)X^{(k)}$

  else

    $F^{(k)} \leftarrow X^{(k)}$

  end

  if $\lambda \neq 1$ **then**

    $\Delta F^{(k)} \leftarrow \sum_{a_{ij}\neq 0} \frac{f_i^{(k)}}{\sqrt{\tilde{d}_i}} - \frac{f_j^{(k)}}{\sqrt{\tilde{d}_j}}$

    $w^{(k)} \leftarrow \hat{\tilde{A}} \cdot e^{-\Delta F^{(k)}/\sigma^2}$

    Cauchy smoothing: $Z^{(k)} \leftarrow \frac{\gamma \cdot w^{(k)} \Delta F^{(k)}}{\left((\Delta F^{(k)})^2 + \gamma^2\right)^2}$

  else

    $Z^{(k)} \leftarrow 0$

  end

  Message-passing: $H^{(k)} \leftarrow \lambda\hat{\tilde{A}}X^{(k)} + (1-\lambda)(X^{(k)} - \epsilon Z^{(k)})$

  Activation + Dropout layer: $H^{(k)} \leftarrow dropout(\sigma(H^{(k)}))$

  Update $k \leftarrow k + 1$

end

4.3.3 Clustering Analysis

We introduce a clustering analysis at the output layer to smooth both inter-class clustering and intra-class clustering, simultaneously enhancing the learning of local and global smoothness. Specifically, we employ the unsupervised deep clustering techniques in [57], which utilize the KL divergence to compare the clustering distribution (Q) with an auxiliary self-training target distribution (P). The KL divergence is calculated as follows:

$$KL(P\|Q) = \sum_{i=1}^{N}\sum_{c=1}^{C} p_{ic} \cdot log\frac{p_{ic}}{q_{ic}}, \tag{4.14}$$

where $c \in [1, C]$ denotes a class, with C being the number of classes in the dataset, and $i \in [1, N]$ denotes a node.

The clustering distribution (Q) captures the similarity between the node representation $X_i^{(k)}$ and cluster centroid $\{\mu_c\}_{c=1}^{C}$. Since the node representations are partially learned from the Cauchy distribution, we utilize the Cauchy distribution to compute the clustering distribution, enabling it to also capture heavy tails. The probability q_{ic} of assigning node i to cluster c is computed as follows:

$$q_{ic} = \frac{\left(1 + \|h_i^{(k)} - \mu_c\|^2\right)^{-1}}{\sum c' \left(1 + \|h_i^{(k)} - \mu_{c'}\|^2\right)^{-1}}, \tag{4.15}$$

where we assume the scale hyper-parameter $\gamma = 1$ for the Cauchy distribution. Recall, $H^{(k)}$ is the node representation from the last layer.

The target distribution P is designed to enhance the performance of the classifier and correct centroids. The distribution P is defined as follows:

$$p_{ic} = \frac{q_{ic}^2/s_c}{\sum_{c'} q_{ic'}^2/s_{c'}} \tag{4.16}$$

where $s_c = \sum_i q_{ic}$ represents the soft cluster frequencies.

The challenge lies in determining the initial values for centroids μ and updating them throughout each training epoch in CauchyGCN. The initialization strategy of centroids, as outlined in [57], assumes a high accuracy of the classifier during the initial training epoch. However, this assumption presents a challenge for CauchyGCN, given

that the classifier's accuracy at the first epoch is typically quite low. Moreover, while the K-means clustering method seems intuitive, its lack of differentiability complicates the optimization problem during backpropagation. To circumvent the challenges, we compute the centroids μ_c by averaging the node representations associated with high confidence in each class label c. At the first training epoch, we initialize centroids μ_c as follows:

$$\mu_c = \frac{1}{N_c} \sum_{i=1}^{N_c} H_{i \in c}^{(k)} \ . \tag{4.17}$$

In this approach, N_c denotes the number of nodes in cluster c. The condition $i \in c^K$ for node representation H^K indicates that node i is assigned to class c due to having the highest confidence in this class among all potential classes. The optimization of μ_c will be discussed in Section 4.3.4.

4.3.4 Optimization of CauchyGCN

The objective function of CauchyGCN in the output layer is jointly optimizing the performance of semi-supervised node classification and unsupervised clustering analysis,

$$\mathcal{O}_{CGCN} = \underbrace{- \sum_{l \in \mathcal{Y}_{label}} \sum_{f=1}^{F} Y_{lf} ln H_{lf}^{L}}_{\text{semi-supervised node classification}} + \underbrace{\kappa KL(P \| Q)}_{\text{clustering analysis}} \tag{4.18}$$

where $\mathcal{Y}_{label}$ is the set of ground truth labels of the labeled nodes and $\kappa \in [0, 1]$ is a hyper-parameter that control the effect of $KL(P\|Q)$. The gradient of $\mathcal{O}_{CGCN}$ with respect to $h_i^{(k)}$ is computed as:

$$\frac{\partial \mathcal{O}_{CGCN}}{\partial h_i^{(k)}} = 2 \sum_c (p_{ic} - q_{ic}) \frac{h_i^{(k)} - \mu_c}{1 + \|h_i^{(k)} - \mu_c\|^2} \ . \tag{4.19}$$

This is then passed down to the backpropagation to optimize the projection matrix Θ, i.e., $\partial \mathcal{O}_{CGCN} / \partial \Theta$. The gradient of $\mathcal{O}_{CGCN}$ with respect to cluster centroid μ_c is computed as:

$$\frac{\partial \mathcal{O}_{CGCN}}{\partial \mu_c} = -2 \sum_i (p_{ic} - q_{ic}) \frac{h_i^{(k)} - \mu_c}{1 + \|h_i^{(k)} - \mu_c\|^2} \ . \tag{4.20}$$

We manually update μ_c to the next training epoch as:

$$\mu_c' \leftarrow \mu_c - \beta \frac{\partial \mathcal{O}_{CGCN}}{\partial \mu_c} \ , \tag{4.21}$$

$\beta \in [0, 1]$ denotes the gradient step. This process is not involved in the backpropagation process that optimizes the projection matrix Θ.

4.4 Empirical study

We evaluate the performance of CauchyGCN on several benchmark datasets for semi-supervised node classification, comparing its performance to state-of-the-art methods.

4.4.1 Datasets

We validate our CauchyGCN on semi-supervised node classification tasks using the following real-world citation networks, Wikipedia-based article networks, and co-authorship networks: 1) Cora, CiteSeer, and PubMed [58] are citation networks widely used in node classification literature, where nodes are bag-of-words representations of documents and edges are citation links. 2) Wiki-CS [59] is a Wikipedia-based article network in the field of Computer Science, where nodes are word embeddings of the articles, edges are hyperlinks, and classes are assigned to 10 relative fields. 3) Coauthor-CS [60] is a co-authorship network based on the Microsoft Academic Graph, where nodes are paper keywords for each author's papers, edges depict co-authorship, and classes are assigned to the 15 most active fields. We conduct 10 runs on all datasets with the splitting method in [22] for training.

4.4.2 Settings and Baselines

To ensure fair comparisons, all methods are fixed under the following settings: the same data splitting method; 16 hidden feature size; 0.1 learning rate; 5e-4 weight decay rate; 0.5 dropout rate; 300 training epochs; and all performance is reported as

the mean and standard variance of 10 runs in terms of semi-supervised node classification accuracy (%). The propagation depth, which refers to the number of graph convolutional layers, varies depending on the methods used. PPNP [55] and GNN-HF-closed [21] have a propagation depth of one. ChebNet [13], GCN [14], GAT [7], APPNP [9], GNN-HF-iter [21], ElasticGNN [22], and CauchyGCN have a propagation depth of two. All compared methods were implemented using the hyperparameters specified in the respective literature. In the case of CauchyGCN, the layer-wise message-passing scheme has the balancing parameter λ in the range of $[0.2, 0.7]$, the scale γ in Cauchy distribution is set to 1, the scalar of KL divergence in the optimization problem κ is chosen to be less than 0.1, and gradient step of centroids μ is selected from the range $0 < \beta \leq 0.5$.

4.4.3 Analysis of Node Classification Performance

The results of semi-supervised node classification are in Table 4.1, with **Bold** is used to show the best results. Here are some notable observations: *Importance of recognizing the closely embedded neighbors*: GAT demonstrates superior performance over GCN and ChebNet in Cora, CiteSeer, and Wiki-CS, suggesting the crucial importance of paying more attention to information aggregated and propagated from closely embedded neighbors. PPNP, APPNP, GNN-HF-closed, and GNN-HF-iter outperform GCN across most datasets, indicating that filtering out less frequently related neighboring information can effectively denoise the aggregated information at the center node. These findings emphasize the significance of utilizing the underlying graph structure to assess the importance of neighbors, a concept incorporated into CauchyGCN through the modeling of the weight factor w_{ij}.

Importance of letting the data represent itself: In contrast, ElasticGNN, the closest contender, also introduces an additional smoothing strategy to preserve local smoothness. However, CauchyGCN consistently outperforms ElasticGNN, particularly on Wiki-CS, a dataset characterized by high-frequency edges between nodes. While ElasticGNN relies on the l_1 norm and a soft-thresholding operator to preserve

closely embedded neighbors, CauchyGCN distinguishes itself by allowing the data itself to determine the preservation of local smoothness through a Cauchy distribution.

It is worth noting that GNN-HF-closed outperforms CauchyGCN on Cora and CiteSeer by 0.3% and 0.1%, respectively. As discussed earlier, GNN-HF improves upon GCN from the spectral aspect by filtering information via a refined graph filter. On the other hand, CauchyGCN improves GCN by introducing a novel smoothing strategy from the spatial perspective. These two perspectives are independent and can be examined for potential integration to further improve the performance of GCNs.

Table 4.1.: Comparison of semi-supervised node classification results for CauchyGCN against baselines.

	Datasets				
Method	**Cora**	**Citeseer**	**Pubmed**	**Wiki-CS**	**Coauthor-CS**
ChebNet [13]	79.6 ± 1.7	69.1 ± 2.3	76.9 ± 1.9	68.2 ± 4.6	91.3 ± 0.4
GCN [14]	80.3 ± 1.6	69.7 ± 1.5	77.3 ± 1.8	74.6 ± 2.6	90.9 ± 0.6
PPNP [9]	81.5 ± 1.1	70.6 ± 1.5	78.6 ± 1.8	75.2 ± 2.2	88.9 ± 1.4
APPNP [9]	80.4 ± 1.8	70.0 ± 1.5	77.7 ± 2.1	75.2 ± 2.7	91.6 ± 0.5
GNN-HF-closed [21]	82.0 ± 1.2	**71.6 ± 1.2**	**79.3 ± 2.2**	70.7 ± 3.2	91.0 ± 0.5
GNN-HF-iter [21]	80.4 ± 1.5	70.7 ± 1.8	77.7 ± 2.3	72.4 ± 3.1	91.9 ± 0.5
GAT [7]	80.7 ± 1.7	70.6 ± 1.5	-	74.7 ± 2.6	-
ElasticGNN [22]	81.3 ± 1.5	70.8 ± 1.5	78.4 ±2.1	43.9 ± 13.1	91.5 ± 0.4
CauchyGCN	**82.2 ± 0.8**	71.3 ± 1.5	79.2 ± 1.7	**75.7 ± 2.7**	**92.0 ± 0.5**

4.4.4 Ablation Study

This section analyzes the performance of CauchyGCN with different configurations, this includes parameters: λ (the balancing factor in Eq. (4.9)) and ϵ (in Eq. (4.13)) in the message-passing scheme, as well as β (in Eq. (4.21)) and κ (in Eq. (4.18)) in clustering analysis. We use a heat map in Fig. 4.3 to illustrate the average classification accuracy of 10 runs across all possible configurations of λ_1 and λ_2. Specifically, λ_1 and λ_2 are the balancing factors in the first and second graph convolutional layers, respectively. All colored cubes denote performance equal to or higher

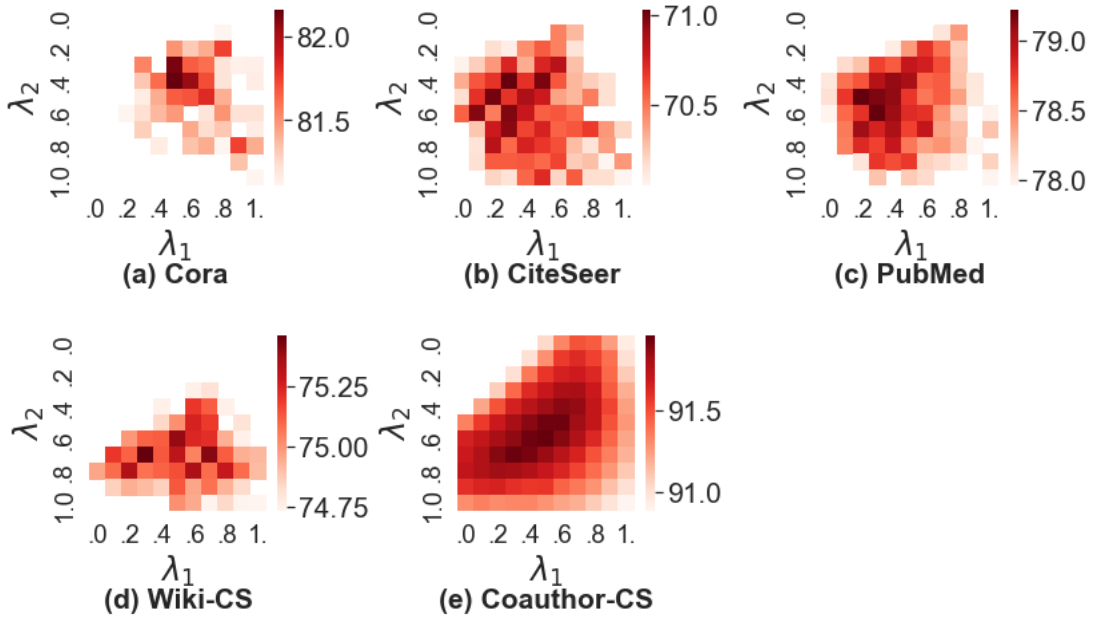


Figure 4.3.: CauchyGCN ablation study on balancing parameters λ in Eq. (4.9).

than that of the fully Laplacian smoothing (GCN) method, where $[\lambda_1, \lambda_2] = [1, 1]$, with darker colors indicating superior performance. These results are conducted under a fixed value of ϵ, β, and κ, which are tuned for the best performance of CauchyGCN shown in Table 4.1. Observing Fig. 4.3: 1) The value of $(1 - \lambda)$ represents the proportion of Cauchy smoothing employed in CauchyGCN, whereas λ denotes the percentage of Laplacian smoothing. Generally, the presence of more colored cubes compared to uncolored ones suggests that CauchyGCN is readily adjustable for superior performance over GCN. Notably, the best performance of CauchyGCN emerges when $[\lambda_1, \lambda_2]$ being $[0.5, 0.4]$ on Cora, $[0.3, 0.4]$ on CiteSeer, $[0.3, 0.6]$ on PubMed, $[0.3, 0.7]$ on Wiki-CS, and $[0.3, 0.7]$ on Coauthor-CS, illustrating a significant reliance and need for Cauchy smoothing in preserving local smoothness. 2) All experiments have a Cauchy-based clustering analysis since the tuned value of β and κ are greater than 0. GCN with clustering analysis, the special case wherein $[\lambda_1, \lambda_2] = [1, 1]$, underperformed CauchyGCN but outperformed or equal to GCN without clustering analysis. This observation underscores the effectiveness of both the Cauchy-based message-passing scheme and the clustering analysis in CauchyGCN.

4.4.5 Robustness Analysis with Graph Structure Attacks

We assess the robustness of CauchyGCN under adversarial attacks [61] on graph structure with varying perturbation ratios. The experimental results, as illustrated in Figure 4.4, encompass different methods evaluated under perturbation rates of 0%/5%/10%/15%/20%. Note that the outcomes for the 0% perturbation rate are not directly comparable to those in Table 4.1 due to differing data splitting techniques. For this study, node is randomly split as 10% for training, 10% for validation, and 80% for testing. The findings from Figure 4.4 indicate the superior performance of CauchyGCN over GCN and ElasticGNN, attributed to the Cauchy smoothing. It efficiently identifies proximal neighbors and preserves local smoothness, even when confronted with noisy and irregular neighboring information.

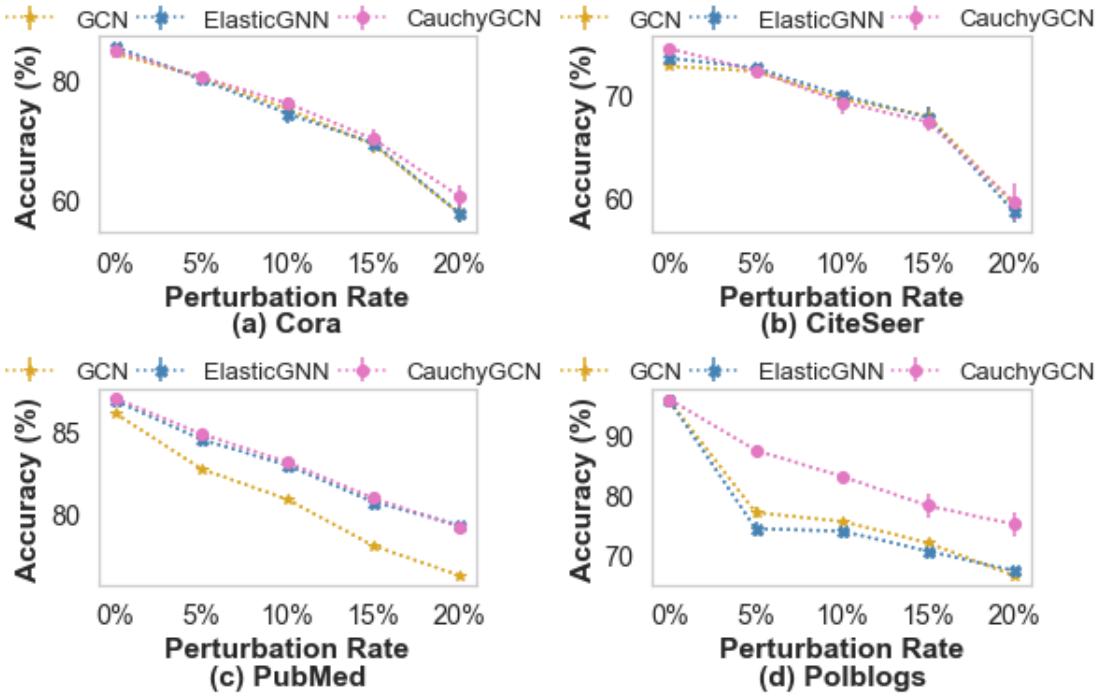


Figure 4.4.: Classification accuracy (%) under different perturbation rates of adversarial graph attack.

4.4.6 *Analysis of the Propagation Depth*

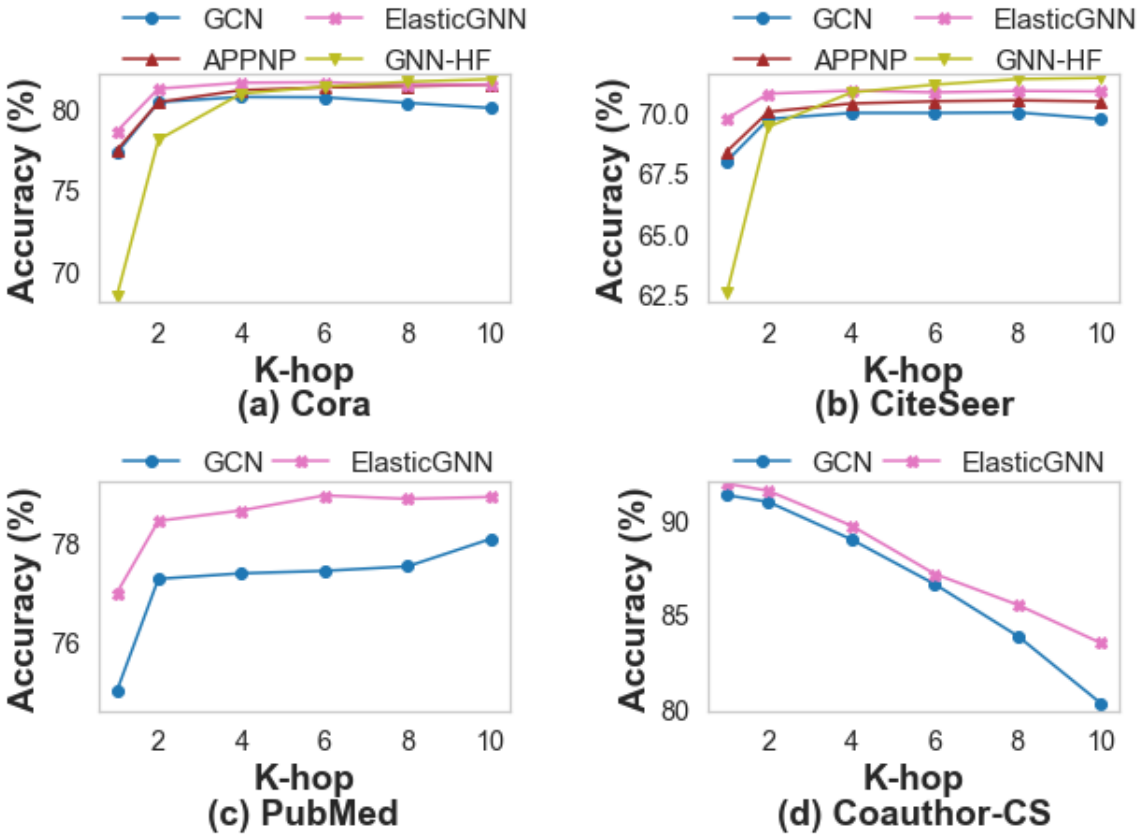


Figure 4.5.: Analysis of the propagation depth of baselines vs CauchyGCN.

GCN is considered a shallow method, utilizing two graph convolutional layers. This means that GCN updates the central node by aggregating information from its neighbors and neighbors' neighbors, resulting in a two-hop propagation step. When extending to consider more than two-hop neighbors, GCN encounters the challenge of over-smoothing the features [8, 49]. To address this challenge, PPNP introduced to predict before propagating, using weight sharing to tackle the issue. Different from GCN-like method in Eq.(4.1), a PPNP-like method is expressed as $H = Propagation\left\langle \left(x_i^{(0)}\Theta^{(0)}\right)\Theta^{(1)}, K, \Lambda \right\rangle$, where Λ is the set of hyperparameters for the message-passing scheme, k denotes the propagation depth, and $\left(x_i^{(0)}\Theta^{(0)}\right)\Theta^{(1)}$ signifies the predict representation, which is then passing to propagation step. However,

PPNP-like methods with propagation step $K > 2$, such as APPNP and GNN-LF/HF-iter, did not demonstrate monotonic increasing performance with deeper propagation in their paper. This suggests that considering more than two-hop neighbors does not promise an improved performance.

We conducted experiments with the PPNP-like methods, including APPNP, GNN-HF, and ElasticGNN, as well as a reconstructed PPNP-like GCN, varying the value of $K \in [1, 2, 4, 6, 8, 10]$. Figure 4.5) illustrates the average testing classification accuracy over 10 runs with respect to the K-hop propagation. The upper limit of the y-axis in panels (a), (c), and (d) represents the accuracy of the shallow CauchyGCN, while in panel (b), it corresponds to the highest accuracy of GNN-HF at 10 hops. APPNP and GNN-HF are out-of-memory after 2-hop propagation when testing with PubMed and Cauother-CS, preventing us from reporting their corresponding results for comparison. Now, we analyze their performances alongside the shallow ($K = 2$) CauchyGCN's performance that is shown in Table 4.1. Among the methods tested on the Cora and CiteSeer datasets, all except for GNN-HF showed convergence in performance after the two-hop propagation, while GNN-HF displayed continuously improving performance as the K-hop propagation increased. Notably, GNN-HF also outperformed CauchyGCN on the CiteSeer dataset at the two-hop propagation step, as illustrated in Table 4.1. Moreover, methods tested on PubMed underperform the CauchyGCN. Methods tested on Coauthor-CS also underperform the CauchyGCN, with a special phenomenon of monotonically decreasing performance as K-hop increases. Based on the above analysis, we design CauchyGCN as a GCN-like shallow method.

4.5 Chapter Summary

Unlike the augmented graph structure approach presented in Chapter 3, this chapter introduces CauchyGCN, which utilizes the default graph structure but prioritizes nodes with similar embeddings as more informative. CauchyGCN develops a new layer-wise message passing scheme that follows the properties of the Cauchy distribu-

tion, preserving smoothness between closely embedded nodes while penalizing distant 1-hop neighbors less severely. Meanwhile, the Cauchy-based unsupervised clustering analysis enhances intra-class smoothness in the output layer, thereby simultaneously improving the classifier's ability to learn both local and global smoothness. Extensive experiments on node classification demonstrate the effectiveness of CauchyGCN, emphasizing the importance of preserving local smoothness and capturing informative information that extends beyond immediate neighbors.

CHAPTER 5
TRANSFORMER WITH TOPOLOGICAL FEATURES: A ROBUST APPROACH FOR LONG-RANGE DEPENDENCY REASONING IN GNNS

Due to neighboring nodes not always sharing the same class, multi-layer GNNs for learning multi-hop neighbors struggle to capture long-range dependencies and express complex relationships within graphs. Recently, incorporating Transformers into GNNs, i.e., Graph Transformers (GT), has significantly addressed these challenges. Specifically, the combination of a Transformer encoder layer with a graph convolution layer allows nodes to attend to long-range dependencies without structural inductive bias. However, the self-attention mechanism in GT primarily focuses on node features and local substructures, often neglecting the crucial high-order connectivity patterns of the graph, i.e., topological features. In this chapter, we introduce Topology-Induced Graph Transformer (TOPGT) that addresses this gap. TOPGT leverages both graph convolution and Transformer layers to learn the local topological features of the graph, enhancing the expressiveness of GNNs concerning these features. Experiments on graph classification tasks on various benchmark datasets show that TOPGT achieves highly competitive results on all datasets and demonstrates the significant advantages of leveraging the topological information of the graph data in feature space and the powerful learning ability based on the transformer architecture.

5.1 Introduction

Recently, Transformer-based models [62–68] have shown promising results in NLP [69] and CV [70] tasks, allowing long-range dependencies to be learned and achieving permutation-invariant attention, motivate the exploration of their potential to address analogous limitations in GNNs [71–76]. In particular, the self-attention mechanism in Transformers offers two key benefits. On the one hand, it overcomes the structural inductive bias inherent to most GNNs, enabling capturing long-range dependencies

and overcoming the over-smoothing [77]. This is due to the self-attention mechanism allowing nodes to attend to other nodes over the graph during message-passing based on specific attention scores, beyond the restriction to their immediate neighbors. On the other hand, the self-attention mechanism offers better expressiveness on the 1-WL test in graph learning. It requires input features to be better identifiable at different aspects of interest, including node features, edge features, and substructures [72]. The attention scores are then calculated based on these features for better relation reasoning among nodes [78]. However, Transformers still struggle in graph learning, where the limitation lies in their inability to extract and reason on the inherent high-order topological features within the graph. As recently shown by [79–81], such topological features, e.g., connected components and holes might be an important step in graph knowledge discovery. For instance, persistent homology [82,83] has been used to study the topological information encoded in the graph [84–86]. Yet, these ideas have never been applied in graph representation learning via Transformer-based architecture.

5.2 Background

5.2.1 Topological Data Aanlysis

Persistent homology (PH) [87,88] is a suite of tools within Topological Data Analysis (TDA) that has shown great promise in a broad range of domains including bioinformatics, material sciences, and social networks [89]. One of the key benefits of PH is that it can capture subtle patterns in the data shape dynamics at multiple resolution scales. PH has been successfully integrated as a fully trainable topological layer into various machine learning and deep learning models [90], addressing such tasks as node classification [91], link prediction [92], molecules and biomolecular complexes representation learning [93], graph classification [94], and spatiotemporal prediction [95]. For instance, [96] builds a neural network based on the DeepSet architecture [97] which can achieve end-to-end learning for topological features. [93] introduces multi-component persistent homology, multi-level persistent homology, and electrostatic persistence for chemical and biological characterization, analysis, and

modeling by using convolutional neural networks. [98] proposes a trainable topological layer that incorporates global topological information of a graph using persistent homology. However, to the best of our knowledge, PH has not yet been employed for Transformer-based models.

Inherently, PH is a subfield in computational topology that allows us to retrieve the evolution of the inherent shape patterns in the data along various user-selected geometric dimensions [99, 100]. Broadly speaking, by "shape" here we mean the properties of the observed object that are preserved under continuous transformations, e.g., stretching, bending, and twisting. (The data can be a graph, a point cloud in Euclidean space, or a sample of points from any metric space). Since one of the most popular PH techniques is to convert the point cloud to a distance graph, for generality we proceed with the further description of PH on graph-structured data. By using a multi-scale approach to shape description, PH enables to address the intrinsic limitations of classical homology and to extract the shape characteristics that play an essential role in a given learning task. In brief, the key idea is to choose some suitable scale parameters α and then to study changes in homology that occur to $\mathcal{G}$ which evolves with respect to α. That is, we no longer study $\mathcal{G}$ as a single object but as a *filtration* $\mathcal{G}_{\alpha_1} \subseteq \ldots \subseteq \mathcal{G}_{\alpha_n} = \mathcal{G}$, induced by monotonic changes of α. To make the process of pattern counting more systematic and efficient, we build an abstract simplicial complex $\mathscr{K}(\mathcal{G}_{\alpha_j})$ on each $\mathcal{G}_{\alpha_j}$, resulting in filtration of complexes $\mathscr{K}(\mathcal{G}_{\alpha_1}) \subseteq \ldots \subseteq \mathscr{K}(\mathcal{G}_{\alpha_n})$. For instance, we can select a scale parameter as a shortest (weighted) path between any two nodes; then abstract simplicial complex $\mathscr{K}(\mathcal{G}_{\alpha_*})$ is generated by subgraphs $\mathcal{G}^{'}$ of bounded diameter α_* (that is, $(k-1)$-simplex in $\mathscr{K}(\mathcal{G}_{\alpha_*})$ is made up by subgraphs $\mathcal{G}^{'}$ of k-nodes with $diam(\mathcal{G}^{'}) \leq \alpha_*$). If $\mathcal{G}$ is an edge-weighted graph $(\mathcal{V}, \mathcal{E}, w)$, with the edge-weight function $w : \mathcal{E} \mapsto \mathbb{R}$, then for each α_j we can consider only induced subgraphs of $\mathcal{G}$ with maximal degree of α_j, resulting in a degree sublevel set filtration.

Equipped with this construction, we trace data shape patterns such as independent components, holes, and cavities which appear and merge as scale α changes (i.e., for each topological feature ρ we record the indices b_ρ and d_ρ of $\mathscr{K}(\mathcal{G}_{b_\rho})$ and $\mathscr{K}(\mathcal{G}_{d_\rho})$,

where ρ is first and last observed, respectively). We say that a pair (b_ρ, d_ρ) represents the birth and death times of ρ, and $(d_\rho - b_\rho)$ is its corresponding lifespan (or persistence). In general, topological features with longer lifespans are considered valuable, while features with shorter lifespans are often associated with topological noise. The extracted topological information over the filtration $\{\mathscr{K}_{\alpha_j}\}$ can be then summarized as a multi-set in $\mathbb{R}^2$ called *persistence diagram (PD)* $\mathcal{D} = \{(b_\rho, d_\rho) \in \mathbb{R}^2 : d_\rho > b_\rho\} \cup \Delta$ (here $\Delta = \{(t,t)|t \in \mathbb{R}\}$ is the diagonal set containing points counted with infinite multiplicity; including Δ allows us to compare different PDs based on the cost of the optimal matching between their points).

Finally, there are multiple options to select an abstract simplicial complex $\mathscr{K}$ [101]. Due to its computational benefits, one of the most widely adopted choices is a Vietoris-Rips (VR) complex. However, the VR-complex uses the entire observed data to describe the underlying topological space and, hence, does not efficiently scale to large datasets and noisy datasets. In contrast, a witness complex captures the shape structure of the data based only on a significantly smaller subset $\mathfrak{L}_v \subseteq \mathcal{V}$, called a set of *landmark* points. In turn, all other points in $\mathcal{V}$ are used as "witnesses" that govern which simplices occur in the witness complex.

5.2.2 Graph Transformers

Studies on GT can be broadly categorized into two main approaches. *Positional/Structural Encoding (PE/SE)*: This approach involves designing manual or learnable PE/SE schemes to inject graph structure information into the Transformers [71, 74, 102–108]. For instance, Graphormer [74] proposes three manual SEs: centrality encoding to capture a node's importance within a graph, spatial encoding to represent a node's relative position to others, and edge encoding to capture information about the graph connectivity. Similarly, SAN [104] utilizes a manual Laplacian eigenvectors PE to provide information about a node's global position. Extending to manual PE/SE, RWPE [102] leverages a trainable parameter to learn a random-walk PE. *GNN and Transformer Integration*: This approach focuses

on combining GNNs and Transformers to leverage the strengths of both architectures [72, 73, 77, 78, 109, 110]. Examples include GraphTrans [73] stacks Transformer layers on the top of GNN layers, GPS [72] utilizes a parallel architecture where GNN and Transformer layers are trained independently before fusing their representations, GraphiT [109] encodes graph kernel into the self-attention mechanism, SAT [78] enhances the graph kernel for better subgraph structure attention, and LGI-GT [77] interleaves GNN and Transformer layers. Our proposed TOPGT falls into both categories and bridges them. TOPGT introduces a topology-induced SE module to encode and capture the node connectivity on topological features. Additionally, it leverages the graph kernel approach proposed by GraphiT to refine the self-attention mechanism in Transformers. As a result, TOPGT not only to attend to the node features and subgraph structures but also to higher-order topological features inherent in the graphs.

5.3 Topology-Induced Graph Transformer

TOPGT's core concept is to leverage global attention for topological information. Section 5.3.1 outlines the process of extracting topological features for each node and constructing a topological connectivity matrix for each graph. In Section 5.3.2, we explain how to integrate this topological information into a structural encoding scheme. Section 5.3.3 details how the extracted topological information is applied to the global self-attention mechanism. Figure 5.1 provides a visual overview of the TOPGT architecture.

5.3.1 Topological Information Extraction

Topological information extraction (TIE) involves a three-step process to identify intrinsic high-order connectivity patterns (i.e., topological features) within the graph: (1) define a q-hop subgraph for each node (where $q \geq 1$), (2) compute the persistence diagrams of each subgraph via the PH, and (3) determine the topological connection pattern for any pair of nodes based on the similarity between their corresponding

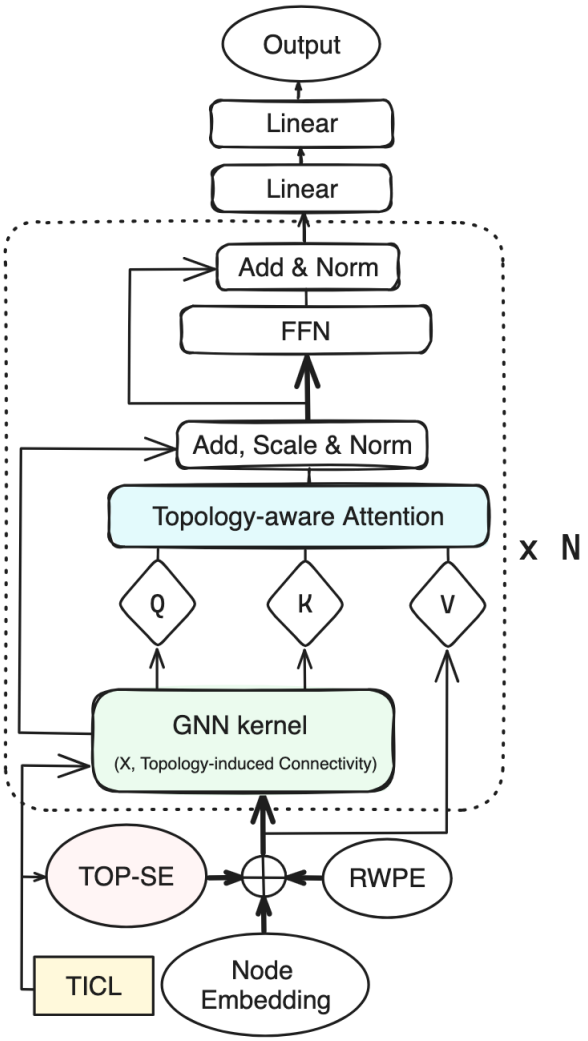


Figure 5.1.: Overview of the architecture of Topology-Induced Graph Transformer.

persistence diagrams. Figure 5.2 shows the top-view of the topological information extraction.

To explore the topological features of each node, we focus on its local connections, represented by subgraphs. This approach offers two key advantages: First, we capture the most relevant context for each node by concentrating on local topological information. Second, we identify recurring patterns within similar substructures across the entire graph, which is beneficial for self-attention mechanisms. Specifically, we define a q-hop subgraph centered at each node v, denoted as $\mathcal{G}_v^q = (\mathcal{V}_v^q, \mathcal{E}_v^q) \subseteq \mathcal{G}$. These subgraphs include the neighbors of the node v within a maximum distance of q-

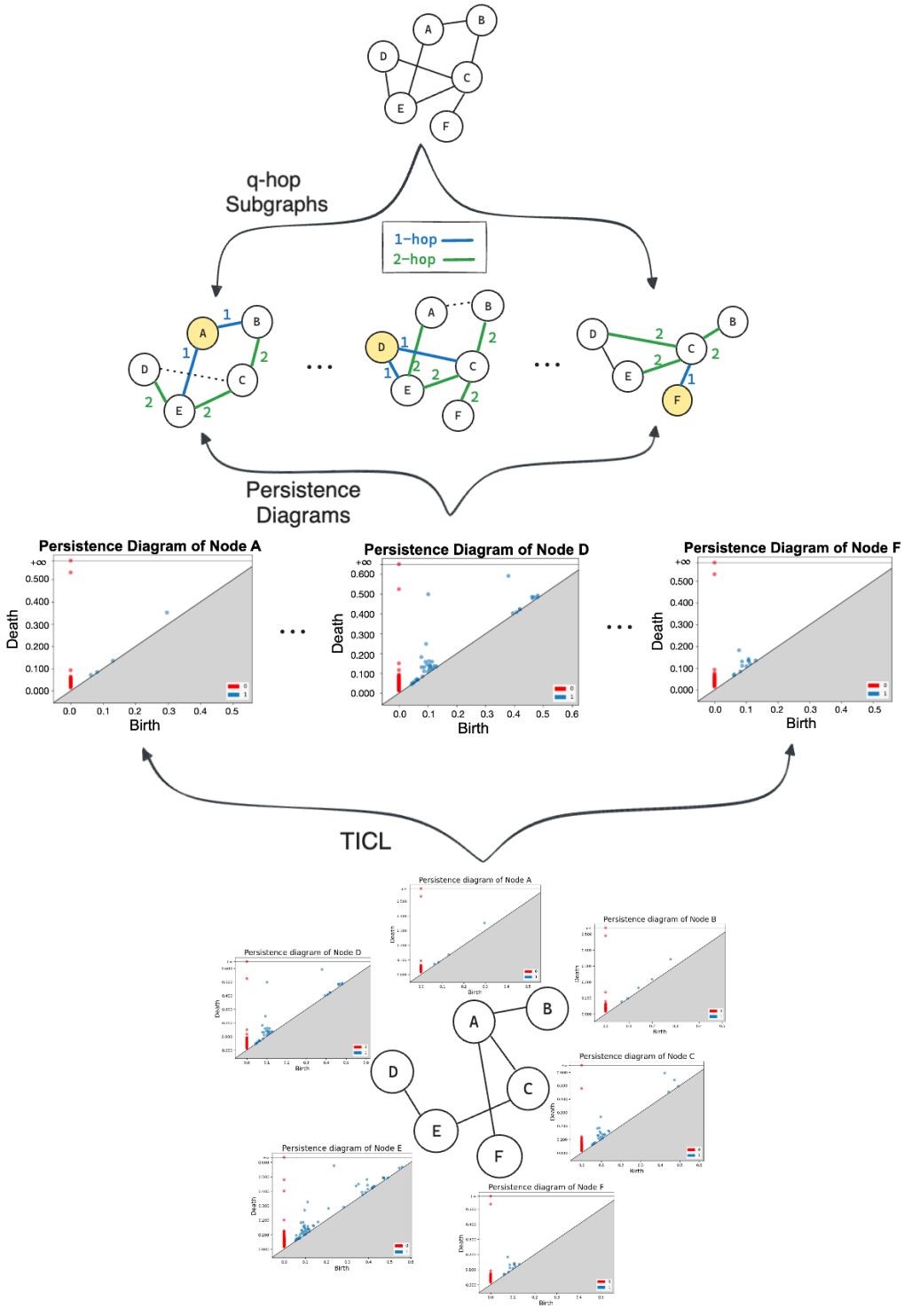


Figure 5.2.: Topology-induced connectivity learning (TICL) module.

hops along the shortest paths. Once we identify the neighboring nodes $\mathcal{V}_v^q$ connected to the node v, we assign weights to the edges $\mathcal{E}_v^q$ based on the features exhibited by the neighboring nodes. The weighted edge matrix $\tilde{\mathcal{E}}_v^q$, is computed as follows: $\|X_{u_i} - X_{u_j}\|/f_0, \forall u_i, u_j \in \mathcal{V}_v^q$. Finally, we denote the weighted q-hop subgraph of the node v as $\tilde{\mathcal{G}}_v^q = (\mathcal{V}_v^q, \tilde{\mathcal{E}}_v^q)$.

The topology-induced connectivity learning (TICL) module is designed to learn connectivity information from the topology-based perspective. For each node v, we compute a persistence diagram, $\mathrm{Dg}_{\tilde{\mathcal{G}}_v^q} = \mathrm{PH}(\tilde{\mathcal{G}}_v^q)$, from its weighted q-hop subgraph $\tilde{\mathcal{G}}_v^q$. We consider various topological estimation techniques, including VR-complexes and weak and strong witness complexes. For the witness complex, to choose landmarks, we establish their selection criteria from the top-τ (in %) of the node degree ranking. By analyzing $\mathrm{Dg}_{\tilde{\mathcal{G}}_v^q}$, we can understand the underlying topological features within the subgraph $\tilde{\mathcal{G}}_v^q$ centered around node v. To construct the topology-induced connectivity of the entire graph $\mathcal{G}$, we utilize the persistence diagrams $\mathrm{Dg}_{\tilde{\mathcal{G}}_v^q}$ extracted from each node's weighted q-hop subgraph. These persistence diagrams capture the topological features of $\tilde{\mathcal{G}}_v^q$, and nodes with similar persistence diagrams are likely to share similar underlying topological properties.

Let $\mathrm{Dg}_{\tilde{\mathcal{G}}_v^q}$ and $\mathrm{Dg}_{\tilde{\mathcal{G}}_u^q}$ be the persistence diagrams for subgraphs $\tilde{\mathcal{G}}_v^q$ of the node v and $\tilde{\mathcal{G}}_u^q$ of the node u, respectively. The distance between these persistence diagrams, denoted by $\mathcal{W}_p(\mathrm{Dg}_{\tilde{\mathcal{G}}_v^q}, \mathrm{Dg}_{\tilde{\mathcal{G}}_u^q})$, is calculated as follow:

$$\mathcal{W}_p(\mathrm{Dg}_{\tilde{\mathcal{G}}_v^q}, \mathrm{Dg}_{\tilde{\mathcal{G}}_u^q}) = \inf_{\gamma \in \Gamma} \left(\sum_{(x,y)\sim\gamma} \|x - y\|_\infty^p \right)^{1/p}, \tag{5.1}$$

where $1 \leq p \leq \infty$, and Γ refers to the set of all couplings in the two input persistence diagrams. In addition, we also show that the vectorization function φ of the persistence diagram is stable as follows.

Theorem of stability of topological summaries: We let X^+ and X^- be two shape objects and let φ be a stable parameter vectorization with the stability equation.

$$d(\varphi(X^+), \varphi(X^-))) \leq \mathcal{C}_\varphi \cdot \mathcal{W}_{p_\varphi}(\mathrm{Dg}_{X^+}, \mathrm{Dg}_{X^-}), \tag{5.2}$$

where $\varphi(X^{\pm})$ denote the corresponding vectorizations for $\mathrm{Dg}_{X^{\pm}}$, $\mathcal{C}_{\varphi}$ denotes a constant.

The proof of the theorem of stability of topological summaries: Let $\mathrm{Dg}_{X^{+}} = \{q_j^{+}\} \cup \Delta^{+}$ (where $q_j^{+} = (b_j^{+}, d_j^{+}) \in \mathrm{Dg}_{X^{+}}$) and $\mathrm{Dg}_{X_{t^{-}}} = \{q_j^{t^{-}}\} \cup \Delta^{-}$ where Δ^{+} and Δ^{-} represent the diagonal with infinite multiplicity of $\mathrm{Dg}_{X^{+}}$ and $\mathrm{Dg}_{X^{-}}$ respectively, and q_j^{+} and q_j^{-} represent the birth and death times a hole σ_j in X^{+} and X^{-} respectively. Let $\Upsilon : \mathrm{Dg}_{X^{+}} \mapsto \mathrm{Dg}_{X^{-}}$ represent a bijective matching. Then the p-th Wasserstein distance can be defined as

$$\mathcal{W}_{p\varphi}(\mathrm{Dg}_{X^{+}}, \mathrm{Dg}_{X^{-}}) = \min_{\Upsilon}(\sum_{j} ||q_j^{+} - \Upsilon(q_j^{+})||_{\infty}^{p})^{\frac{1}{p}}, \quad p \in \mathbb{Z}^{+}.$$

Then, a persistence vectorization $\varphi(\mathrm{Dg}_{X^{+}})$ is stable if $d(\varphi(\mathrm{Dg}_{X^{+}}), \varphi(\mathrm{Dg}_{X^{-}})) \leq \mathcal{C}_{\varphi} \cdot \mathcal{W}_{p_{\varphi}}(\mathrm{Dg}_{X^{+}}, \mathrm{Dg}_{X^{-}})$, and the constant $\mathcal{C}_{\phi} > 0$ is independent of X^{+} and X^{-}. Note that $\mathcal{C}_{\varphi} > 0$ is independent of $X^{\pm}$. The above stability inequality interprets that as the changes in the vectorizations are bounded by the changes in PDs.

Inherently, we consider nodes to be similar based on their topological features if the Wasserstein distance $\mathcal{W}_p(\cdot, \cdot)$ between their persistence diagrams falls below a threshold value r (where $r \in [0, 1]$). Based on these similarities, we then can construct the topology-induced connectivity matrix $\mathcal{E}^{topo} \in \mathbb{R}^{N \times N}$, which is formulated as follows:

$$\mathcal{E}_{vu}^{topo} = \begin{cases} 1, & \text{if } 0 \leq \mathcal{W}_p(\mathrm{Dg}_{\tilde{\mathcal{G}}_v^q}, \mathrm{Dg}_{\tilde{\mathcal{G}}_u^q}) < r \ , \\ 0, & \text{otherwise,} \end{cases} \quad \forall \ v, u \in \mathcal{V}. \tag{5.3}$$

In summary, the set of topology-induced edges $\mathcal{E}^{topo}$ reveals the underlying connectivity among nodes, reflecting their intrinsic high-order connectivity patterns within a localized subgraph through the TICL.

5.3.2 Topology-Induced Structural Encoding

While existing graph transformers aim to overcome the 1-WL test from the structural inductive bias when learning long-range dependencies, PE/SE remains a cru-

cial module to provide access to the canonical relative positions of nodes within the graph [72,102]. Analogous to how word order and surrounding words offer context for a word's meaning in NLP, PE/SE helps graph transformers understand the distance and structural similarity between nodes based on their relative positions/structure within the graph. Note that existing PE/SE schemes mainly rely on the pre-defined graph attributes, e.g., [102] accesses relative positions via random walks on adjacency matrices, and [74] accesses the relative structure through a learnable degree centrality. However, such approaches mostly overlook the richer set of high-order relationships and topological connection patterns, i.e., failing to capture node positions and structures.

Here, topology-induced structural encoding (TOP-SE) is introduced to investigate nodes' substructures deeply and comprehensively. It enables unique encoding of each node's positions/structures that access the topological connectivity within s-hop distance,

$$SE_v^{topo} = [(W_R^{topo})_{vv}^1, (W_R^{topo})_{vv}^2, \ldots, (W_R^{topo})_{vv}^s] \in \mathbb{R}^{1\times s}. \tag{5.4}$$

Instead of using one-hot encoding which can become computationally expensive at $\mathcal{O}(N^2)$, our TOP-SE draws inspiration from RWPE [102]. It defines a random walk diffusion process on the graph's topological connectivity, i.e., $((W_R^{topo})_{vv})$. This offers lower complexity at $\mathcal{O}(sN)$, where s represents the walk length and is much smaller than the number of nodes, $s \ll N$ and $(W_R^{topo})_{vv}$ represents a random walk on $\mathcal{E}^{topo}$ (here we find the optimal hyperparameter s via cross-validation). The random walk strategy explores the node's topological neighborhood, reaching nodes that are close together in the topological space, before eventually returning to the starting node itself. Formally, the random walk transition matrix is defined as $W_R^{topo} = \mathcal{E}^{topo}(\mathcal{D}^{topo})^{-1}$, where $\mathcal{D}^{topo} \in \mathbb{R}^{N\times N}$ is a diagonal matrix represents node degree of the topological information in $\mathcal{E}^{topo}$. Moreover, the walk length s can be set to the average number of nodes per graph in the dataset for simplicity. Alternatively, for optimal performance on specific datasets, s can be tuned within the furthest topological reach of a node. This involves a trade-off between latency and context captured. A smaller s offers lower computational cost but may miss long-range dependencies; conversely, a larger

s allows richer context but comes at the expense of increased parameters, processing time, and potentially high sparsity in SE^{topo}.

To this end, the generalization of SE^{topo} is crucial because the underlying topological features captured in $\mathcal{E}^{topo}$ can be sparse and vary even among graphs from the same distribution. Thus, we enable a learnable TOP-SE to learn a more generalized representation of SE^{topo} and test on the later unseen graphs, which improves the ability to handle the inherent variability of topological information,

$$SE^{topo} = SE^{topo}\Theta_{pe}^{\top} + b_{pe}, \tag{5.5}$$

where $\Theta_{pe} \in \mathbb{R}^{f_m \times s}$ is a learnable embedding matrix, and $b_{pe} \in \mathbb{R}^{f_m}$ is an optional bias term with the output dimension f_m. To fuse the SE^{topo} with node features, we consider additive fusion and concatenation as the primary methods in our TOPGT model.

5.3.3 Topology-Aware Self-Attention Mechanism

A typical transformer layer consists of a self-attention layer followed by a skip-connection and a feed-forward network. GT leverage the power of Transformer layers, particularly the self-attention layer, to overcome the limitation of GNNs in processing information only among immediate neighbors. Because the self-attention layer where allows nodes to attend to all other nodes in the graph, its capability has fueled research into using self-attention for various features within the graph, including node features [72, 73, 77, 109], edge features [71, 77], and substructures [78].

The self-attention mechanism in TOPGT is designed to capture both the above-mentioned features and the high-order topological features of the graph. To achieve this, we propose the topology-aware self-attention mechanism, which is a kernel smoother approach inspired by [78, 109]. Specifically, the key features we want the

self-attention mechanism to attend to are encoded in $\tilde{X}'_v$. These features are typically obtained through a GNN layer,

$$\begin{aligned} \tilde{X}'_v &= \mathrm{GNN}(X', \tilde{\mathcal{E}'}^q_v), \\ \tilde{\mathcal{E}'}^q_v &= \{1_{(v,u)\in\mathcal{E}^q_v}, 1_{(v,u)\in\mathcal{E}^{topo}}\}, \\ X' &= X\Theta^\top_{embed} + PE + SE^{topo}, \end{aligned} \tag{5.6}$$

where $\tilde{\mathcal{E}'}^q_v$ represents both the local context-induced and topology-induced connectivity, X' denotes a composed embedding, which incorporates the node feature (embedded with matrix $\Theta_{embed} \in \mathbb{R}^{f_m \times N}$), the positional encoding PE (we allow any positional encoding to apply here), and the topological-induced structural encoding SE^{topo} (defined in Eq. (5.5)). Then, the similarity between nodes based on these key features is calculated as:

$$e_{vu} = \frac{\langle \Theta_Q \tilde{X'}_v, \Theta_K \tilde{X}'_u \rangle}{\sqrt{f_{QK}}} \ . \tag{5.7}$$

Here, we define Θ_Q as the trainable projection matrix for the central node and Θ_K as the trainable projection matrix of the other nodes. Both matrices have a dimension of $N \times f_{QK}$ to facilitate the dot production (denoted by $\langle \cdot\ ,\ \cdot \rangle$). The resulting similarity score is then scaled by the square root of f_{QK}. Finally, the self-attention mechanism is defined as:

$$Att(v) = \sum_{u \in N^q_v} \frac{\exp(e_{vu})}{\sum_{j \in N^q_v} \exp(e_{vj})} f(X'_u) \ . \tag{5.8}$$

The final attention reflects on the X' with a trainable projection $f(X') = X'\Theta_V$, where $\Theta_V \in \mathbb{R}^{N \times f_{QK}}$ has the identical parameter size as Θ_Q and Θ_K.

5.4 Empirical study

We evaluate the performance of our proposed TOPGT for graph classification tasks. We first compare its performance to the state-of-the-art baselines. Next, we delve deeper into TOPGT by conducting ablation studies on the three design components discussed in Section 5.3. Finally, we study the robustness of our TOPGT model on graphs with structure noise (i.e., edge-noisy graphs).

We evaluate the performance of TOPGT on a comprehensive set of 13 cross-domain graph benchmarks, including (i) 9 small-scale benchmarks, each containing hundreds of graphs, facilitating rapid testing and ensuring TOPGT does not overfit to limited data. This includes chemical compounds BZR, BZR_MD, COX2, COX2_MD, and DHFR from [111], where nodes are atoms and edges represent different chemical bond types, and molecular compound datasets PTC [112] study carcinogenicity on rodents with four experimental groups, female rats (FR), male rats (MR), female mice (FM), male mice (MM); (ii) 3 medium-scale benchmarks, each containing thousands of graphs, including PROTEINS [113], which focuses on protein structure, with nodes representing secondary structure elements and edges indicating connections between neighboring amino acids or three-nearest neighbors that are spatially closed. NCI1 and NCI109 [114] study the activity of anti-cancer drugs on non-small cell lung cancer (NCI1) and ovarian cancer (NCI109); (iii) one large-scale benchmarks from the Open Graph Benchmark (OGB) to assess TOPGT's scalability. OGBG-CODE2 [115] focuses on code summarization, involving 452,741 graphs representing Abstract Syntax Trees of Python code. To ensure fair evaluation, we employ different data split strategies. For small-scale benchmarks and PROTEINS, we use 10-fold cross-validation on a 90%/10% random training/testing split [17]. For NCI1 and NCI109, we conduct 10 independent runs using an 80%/10%/10% random training/validation/testing split [77]. For large-scale benchmarks, we conduct 10 runs with different random seeds [77] and use the default training/testing split.

Our proposed method compares with 10 state-of-the-art GNNs and graph transformers: (i) 3 pure GNNs: These methods rely on messaging-passing for information propagation within the graph. We compare against the Graph Convolutional Network (GCN) [14], which focuses on layer-wise distant neighbor penalization, Graph Isomorphism Network (GIN) [17] that allows learnable weights for the central nodes during message passing, and Long-Range Graph Neural Networks (LRGNN) [20], a deep stacked GNNs with adaptive skip connection schemes for capturing long-range dependencies; (ii) 3 GNNs with topological learning: These methods explicitly incorporate topological information. We evaluate against Topological Graph Neural

Networks (TOGL) [98] that enhances learning with a topological layer for capturing global topological information of graphs, Tensor-view Topological Graph Neural Network (TTG-NN) [116] that captures Tensor-view Topological information and Tensor-view Graph structure information, and Wit-TopoPool [117] that leverages topological pooling and witness complex-based topological embeddings; (iii) 4 GNNs with Transformers: We compare against GraphTrans [73] that stacks multi-layer GNNs with multi-layer Transformers, structure-aware graph transformer (SAT) [78]that use graph kernel transformer approach for structure attention, Local and global operators interleave GT (LGI-GT) [77] that interleaves GNN layers with Transformers, and Graphormer-*SPIS* [118] that explores the structural power of graph Transformer though global Weisfeiler-Lehman test.

All experiments are conducted on an NVIDIA Quadro RTX 8000 GPU with 48GB of memory. We perform a limited hyperparameter search to ensure training efficiency and keep the model simple. TOPGT is trained end-to-end using the AdamW optimizer with a set-tuned learning rate $\{0.0001, 0.0005, 0.001, 0.005\}$. The training process minimizes the cross-entropy loss function to find the optimal model parameters. We set the search depth of the q hop subgraph to be $q = 3$. We provide more experimental results of parameter sensitive analysis in Appendix B3. For positional encoding based on the adjacency matrix, we utilize the Random Walk PE (RWPE) [102]. The initial dimension of both RWPE and our TOP-SE is set equal to the input feature dimension of the nodes.

5.4.1 Graph Classification Results

Table 5.1 and Table 5.2 show the performance of TOPGT against other SOTA methods on small-scale and medium-scale benchmarks, respectively. All results were generated using the same data configuration for consistency. For TOGL, TTG-NN, Wit-TopoPool, GraphTrans (benchmarks NCI1 and NCI109), LGI-GT (benchmarks NCI1 and NCI109), and Graphormer-*SPIS* the reported statistics are from their original papers. The statistics for GCN and GIN are obtained from [117]. We implement

and evaluate the remaining methods using the best hyperparameters in their respective GitHub repositories. Our results show that no single baseline consistently outperforms all other baselines. The dashed results in Table 5.1 and Table 5.2 highlight some baselines that outperform others on a specific dataset. For example, LRGNN, a pure GNN with adaptive skip connections to preserve local structure, achieves the best performance on COX2. Wit-TopoPool, a GNN-based model with a topological pooling layer to capture local topology, achieved the best performance on PTC_MM and PROTEINS. Finally, SAT, a Transformer-based model that employs transformers with global attention to local subgraphs, achieved the best results on eight benchmarks. In particular, TOPGT surpasses all baselines in all 12 datasets. In particular, TOPGT achieves significant SOTA performances of 86.60% on NCI1 and 84.84% on NCI109, demonstrating its generalization efficiency and scalability. Table 5.3 presents our results on the large-scale benchmark OGBG-CODE2, in terms of the mean F1 score ± standard deviation of 10 runs. Given the computational complexity of topology computation algorithms, we restrict TICL to the testing set. We observe that TOPGT outperforms all baselines except K-Subtree SAT, and achieves competitive performance compared to K-Subtree SAT.

Table 5.1.: Comparison of TOPGT with baselines on 9 small-scale benchmarks.

	Datasets								
Method	**COX2**	**COX2_MD**	**PTC_FR**	**PTC_MR**	**PTC_FM**	**PTC_MM**	**BZR**	**BZR_MD**	**DHFR**
GCN [14]	76.53 ± 1.82	-	69.80 ± 4.40	62.26 ± 4.80	62.39 ± 0.85	67.80 ± 4.00	79.34 ± 2.43	-	74.56 ± 1.44
GIN [17]	80.30 ± 5.17	-	66.97 ± 6.17	64.60 ± 7.00	64.19 ± 2.43	67.18 ± 7.35	85.60 ± 2.00	-	82.20 ± 4.00
LRGNN [20]	88.54 ± 3.84	76.95 ± 4.87	74.21 ± 4.87	70.35 ± 3.55	-	76.76 ± 4.98	-	-	87.24 ± 5.08
TTG-NN [116]	86.73 ± 3.41	-	73.23 ± 3.91	68.91 ± 4.02	69.33 ± 2.09	74.11 ± 4.57	87.40 ± 2.62	-	78.72 ± 5.33
Wit-TopoPool [117]	87.23 ± 3.15	-	75.00 ± 3.51	70.57 ± 4.43	71.71 ± 4.86	79.12 ± 4.45	87.80 ± 2.44	-	-
GraphTrans [73]	84.80 ± 2.76	77.59 ± 4.65	73.80 ± 3.45	72.66 ± 4.68	71.62 ± 3.44	75.62 ± 3.02	88.86 ± 3.92	77.59 ± 4.65	77.12 ± 3.39
SAT [78]	88.23 ± 3.83	77.45 ± 4.98	76.37 ± 3.22	73.63 ± 3.95	72.36 ± 2.07	77.09 ± 4.87	90.23 ± 3.35	78.42 ± 7.39	87.31 ± 3.81
LGI-GT [77]	87.58 ± 2.96	73.95 ± 3.87	-	-	-	-	89.39 ± 3.64	78.08 ± 5.51	-
Graphormer-*SPIS* [118]	83.22 ± 2.25	-	-	69.28 ± 5.34	-	-	-	-	-
TOPGT (ours)	**89.74 ± 3.25**	**78.61 ± 6.41**	**78.89 ± 4.16**	**74.13 ± 4.21**	**72.49 ± 2.16**	**79.23 ± 4.62**	**92.34 ± 2.50**	**79.88 ± 6.97**	**88.84 ± 3.54**

Table 5.2.: Comparison of TOPGT with baselines on 3 medium-scale benchamrks.

	Datasets		
Method	**PROTEINS**	**NCI1**	**NCI109**
GCN [14]	70.31 ± 1.93	80.72 ± 2.03	81.70 ± 1.85
GIN [17]	76.16 ± 2.76	-	-
LRGNN [20]	78.93 ± 4.11	-	-
TOGL [98]	76.00 ± 3.90	75.80 ± 1.80	-
TTG-NN [116]	77.62 ± 3.92	-	-
Wit-TopoPool [117]	80.00 ± 3.22	-	-
GraphTrans [73]	78.17 ± 4.33	82.60 ± 1.20	82.30 ± 2.60
SAT [78]	78.48 ± 3.30	83.66 ± 0.89	83.69 ± 0.72
LGI-GT [77]	-	82.18 ± 1.90	83.36 ± 1.89
Graphormer-*SPIS* [118]	79.41 ± 1.46	-	-
TOPGT (ours)	**80.01 ± 3.30**	**86.60 ± 0.72**	**84.84 ± 0.92**

Table 5.3.: Comparison of TOPGT with baselines on a OGBG benchmark.

	Datasets
Methods	**OGBG-CODE2**
GCN [14]	0.1507 ± 0.0018
GCN+virtual node [14]	0.1595 ± 0.0018
GIN [17]	0.1495 ± 0.0023
GIN+virtual node [17]	0.1581 ± 0.0026
Transformer [102]	0.1670 ± 0.0015
GraphTrans [73]	0.1830 ± 0.0024
K-Subtree SAT [78]	**0.1937 ± 0.0028**
GPS [72]	0.1894 ± 0.0024
TOPGT	0.1932 ± 0.0013

5.4.2 Ablation Studies

To gain deeper insights into the contributions of key components of our TOPGT for graph classification tasks, we conduct ablation studies on different simplicial complexes and PE/SE strategies.

Table 5.4.: Ablation study of different simplicial complexes.

	Datasets					
Simplicial Complexes	**COX2**	**PTC_MR**	**BZR**	**DHFR**	**PROTEINS**	**NCI1**
vietoris–rips	**89.74 ± 3.25**	**74.13 ± 4.21**	91.72 ± 3.34	88.10 ± 3.81	**80.01 ± 3.30**	86.27 ± 0.84
weak witness	88.44 ± 3.66	73.85 ± 2.83	**92.34 ± 2.50**	87.84 ± 3.45	79.60 ± 3.13	**86.60 ± 0.72**
strong witness	88.56 ± 4.08	73.44 ± 3.45	90.36 ± 2.03	**88.84 ± 3.54**	79.74 ± 3.16	86.54 ± 0.91

We conduct an ablation study on three different simplicial complexes, i.e., VR-complex, strong witness complex, and weak witness complex. Experiments are conducted across 6 datasets. While pinpointing the single best type of simplicial complex is beyond the scope of this work, the trade-offs are worth noting. Specifically, the VR-complex can capture the overall shape of the noise-free graph, whereas the witness complex is less sensitive to noise. However, performing Exploratory Data Analysis (EDA) to distinguish noise from graph features becomes challenging due to the complex structure of a graph. Note that, our primary objective here is to demonstrate that incorporating topological information can enhance graph representation learning compared to baselines. As shown in Table 5.4, all PH-based models consistently outperform all baselines presented in Table 5.1 and Table 5.2.

To evaluate the effectiveness of TOP-SE in TOPGT, we conduct an ablation study with four variants: (i) without PE/SE, (ii) RWPE, (iii) TOP-SE, and (iv) RWPE + TOP-SE. Table 5.5 shows the best performance achieved after tuning various types of simplicial complex and learning rates in the range $\{0.0001, 0.0005, 0.001\}$. We can observe that, as ablating the PE/SE module (variant (i)), it leads to the lowest performance which aligns with the findings of [72, 102]. While RWPE captures relative node positions and TOP-SE captures the topological connection patterns of nodes, they are not inherently subsets of each other and can capture complementary infor-

mation. The combination of RWPE and TOP-SE (variant (iv)) leads to performance improvements of up to 2% (on PTC_MM).

Table 5.5.: Ablation studies of different learnable positional encoding methods.

	Datasets			
PE/SE	**COX2**	**PTC_MM**	**BZR**	**DHFR**
NONE	86.10 ± 2.24	76.06 ± 4.45	89.86 ± 2.86	87.51 ± 4.54
RWPE	88.35 ± 3.08	77.23 ± 3.53	90.36 ± 2.89	87.64 ± 3.94
TOP-SE	88.45 ± 3.73	77.14 ± 4.69	91.71 ± 3.62	87.58 ± 4.03
RWPE + TOP-SE	**89.74 ± 3.25**	**79.23 ± 4.62**	**92.34 ± 2.50**	**88.84 ± 3.54**

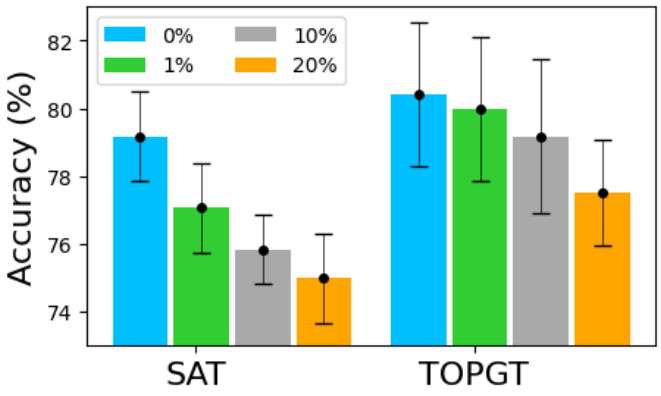
In addition, we ablate specific features from the self-attention mechanism to evaluate the effect of topology-aware self-attention in TOPGT. Table 5.6 shows that incorporating comprehensive information, i.e., node features, structural features, and topological features, can consistently lead to the best performance. Additionally, attending to both node and topology features consistently outperforms attending only to node features over 4 datasets. Furthermore, we notice that structural features are also necessary, i.e., including them alongside node features improves performance on DHFR compared to using node and topology features.

Table 5.6.: Ablation studies of self-attention mechanism on different features.

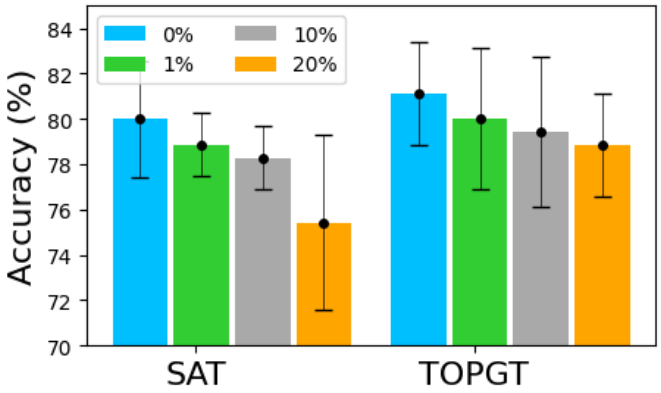
	Datasets			
Feature	**COX2**	**BZR**	**DHFR**	**PROTEINS**
Node	87.80 ± 2.98	90.09 ± 2.45	86.19 ± 2.67	79.65 ± 2.98
Node + Structure	87.91 ± 2.70	92.21 ± 2.42	88.04 ± 2.98	79.33 ± 2.64
Node + Topology	88.25 ± 2.76	92.22 ± 3.11	87.25 ± 2.81	79.87 ± 2.89
Node + Structure + Topology	**89.74 ± 3.25**	**92.34 ± 2.50**	**88.84 ± 3.54**	**80.01 ± 3.30**

5.4.3 Robustness Analysis under Edge Attack

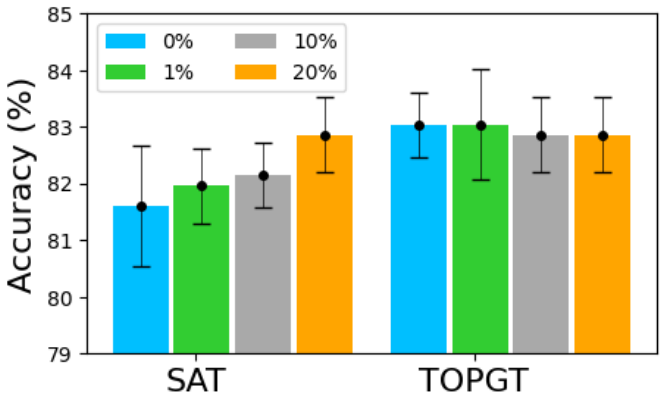
We evaluate the robustness of our TOPGT against random edge attacks that alter the graph structure. This attack randomly removes existing edges and adds fake edges with varying perturbation ratios (0%, 1%, 10%, 20%). It is important to note that due to different data splitting techniques used in this study, the results for the 0% perturbation rate cannot be directly compared to those shown in other tables and figures. Here we split graphs randomly with 80% for training, 10% for validation, and 10% for testing. Figure 5.3 shows the average classification accuracy of graphs (bars) with standard deviation (error bars) for 5 runs for varying perturbation rates under random edge attacks on COX2, PTC_MR, and PROTEINS. From the table, we see that TOPGT consistently exhibits lower variance in prediction accuracy compared to SAT across different perturbation ratios. In summary, incorporating topological information appears to be a key factor in achieving robustness. That is, TOPGT leverages (local) higher-order connection patterns within the graph, leading to less sensitivity to graph perturbations.



(a) COX2.



(b) PTC_MR.



(c) PROTEINS.

Figure 5.3.: Graph classification accuracy for robustness analysis under edge attack.

5.5 Chapter Summary

Motivated by the findings in Chapter 3 and Chapter 4, this chapter defines the additional useful graph structures as topological structures and leverages a self-attention mechanism to assess the importance of closely embedded nodes. Specifically, we introduce a novel framework for graph representation learning that takes advantage of both the topological information among the graph substructures characterized via persistent homology and self-attention mechanisms of Transformer-based architecture. By leveraging the power of the topology-aware self-attention mechanism, our Topology-Induced Graph Transformer (TOPGT) can better capture and model local topological and graph structural information. It has demonstrated capabilities to yield the most competitive graph classification performance. Additionally, TOPGT improves over the state-of-the-art graph Transformer model by significant margins under graph structural attacks, providing more insight into the reliability and robustness of our TOPGT for graph classification.

CHAPTER 6
CONCLUSIONS

In this thesis, I present three approaches to address the graph structure bias in Graph Neural Networks (GNNs) by leveraging more informative node and graph structures.

Our first approach, two-view GNNs with adaptive view-wise structure learning, conducts graph learning on both the default graph structure and an augmented graph structure based on node representation similarity. This dual approach enables the model to learn more informative, view-specific node knowledge while also capturing inter-view relationships, ultimately resulting in more accurate graph classification.

The second approach, CauchyGCN, introduces a Cauchy-based smoothing strategy combined with clustering analysis that targets the preservation of local smoothness. This method relaxes the strict penalization of distant embedded 1-hop neighbors while imposing a stronger penalization on closely and mutually embedded nodes beyond the 1-hop neighbor. As a result, CauchyGCN achieves competitive node classification performance compared to state-of-the-art methods in the field.

Lastly, the third approach integrates transformers into GNNs through graph topological data analysis. In this work, we connect emerging research directions in graph representation learning for transformer-based models with topological information. We introduce a novel framework that enables the learning of topological information and graph structures at both local and global levels. Ultimately, this approach demonstrates its efficacy and robustness through comprehensive comparisons with state-of-the-art baselines in graph classification tasks.

For future research, we aim to explore and leverage informative node knowledge and graph structures beyond local neighborhoods and default configurations to develop large pre-trained GNNs capable of addressing a variety of downstream tasks across different domains.

REFERENCES CITED

[1] F. Monti, M. M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 3700–3710.

[2] M. Sun, X. Zhang, J. Zheng, and G. Ma, "Ddgcn: Dual dynamic graph convolutional networks for rumor detection on social media," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 4, pp. 4611–4619, Jun. 2022. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/20385

[3] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 1263–1272.

[4] A. Derrow-Pinion, J. She, D. Wong, O. Lange, T. Hester, L. Perez, M. Nunkesser, S. Lee, X. Guo, B. Wiltshire, P. W. Battaglia, V. Gupta, A. Li, Z. Xu, A. Sanchez-Gonzalez, Y. Li, and P. Velivckovi'c, "Eta prediction with graph neural networks in google maps," *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:237303762

[5] K. Han, Y. Wang, J. Guo, Y. Tang, and E. Wu, "Vision GNN: An image is worth graph of nodes," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=htM1WJZVB2I

[6] L. Wu, Y. Chen, K. Shen, X. Guo, H. Gao, S. Li, J. Pei, and B. Long, "Graph neural networks for natural language processing: A survey," *CoRR*, vol. abs/2106.06090, 2021. [Online]. Available: https://arxiv.org/abs/2106.06090

[7] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ

[8] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80. PMLR, 10–15 Jul 2018, pp. 5453–5462.

[9] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *International Conference on Learning Representations*, 2019.

[10] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 4–24, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:57375753

[11] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.

[12] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *ArXiv*, vol. abs/1506.05163, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:10443309

[13] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, p. 3844–3852.

[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=SJU4ayYgl

[15] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2019.

[16] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[17] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=ryGs6iA5Km

[18] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: higher-order graph neural networks," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.33014602

[19] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: current limitations and effective designs," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[20] L. Wei, Z. He, H. Zhao, and Q. Yao, "Search to capture long-range dependency with stacking gnns for graph classification," in *Proceedings of the ACM Web Conference 2023*, ser. WWW '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 588–598. [Online]. Available: https://doi.org/10.1145/3543507.3583486

[21] M. Zhu, X. Wang, C. Shi, H. Ji, and P. Cui, "Interpreting and unifying graph neural networks with an optimization framework," in *Proceedings of the Web Conference 2021*, ser. WWW '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1215–1226.

[22] X. Liu, W. Jin, Y. Ma, Y. Li, L. Hua, Y. Wang, M. Yan, and J. Tang, "Elastic graph neural networks," in *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 2021.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986. [Online]. Available: https://api.semanticscholar.org/CorpusID:205001834

[24] N. Adaloglou, N. Vretos, and P. Daras, "Multi-view adaptive graph convolutions for graph classification," in *Computer Vision – ECCV 2020*. Cham: Springer International Publishing, 2020, pp. 398–414.

[25] X. Song, A. Frangi, X. Xiao, J. Cao, T. Wang, and B. Lei, "Integrating similarity awareness and adaptive calibration in graph convolution network to predict disease," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part VII*. Berlin, Heidelberg: Springer-Verlag, 2020, p. 124–133. [Online]. Available: https://doi.org/10.1007/978-3-030-59728-3_13

[26] S. Zhang and H. Tong, "Final: Fast attributed network alignment," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1345–1354.

[27] S. I. Ktena, S. Parisot, E. Ferrante, M. J. L. Martin Rajchl, B. Glocker, and D. Rueckert, "Distance metric learning using graph convolutional networks: Application to functional brain networks," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Cham: Springer International Publishing, 2017, pp. 469–477.

[28] Y. Zhang, L. Zhan, W. Cai, P. Thompson, and H. Huang, "Integrating heterogeneous brain networks for predicting brain disease conditions," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part IV*. Berlin, Heidelberg: Springer-Verlag, 2019, p. 214–222. [Online]. Available: https://doi.org/10.1007/978-3-030-32251-9_24

[29] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, pp. 99–121, 2000.

[30] C. Zhang, Y. Cai, G. Lin, and C. Shen, "Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 12 200–12 210.

[31] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, "From word embeddings to document distances," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning -Volume 37*, ser. ICML'15. JMLR.org, 2015, p. 957–966.

[32] H. P. Maretic, M. El Gheche, G. Chierchia, and P. Frossard, "Fgot: Graph distances based on filters and optimal transport," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 7, 2022, pp. 7710–7718.

[33] M. Heimann, H. Shen, T. Safavi, and D. Koutra, "Regal: Representation learning-based graph alignment," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ser. CIKM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 117–126.

[34] J. Gao, X. Huang, and J. Li, "Unsupervised graph alignment with wasserstein distance discriminator," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ser. KDD '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 426–435.

[35] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2018, p. 499–508. [Online]. Available: https://doi.org/10.1145/3178876.3186116

[36] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *2015 IEEE International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, dec 2015, pp. 945–953. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICCV.2015.114

[37] S. Barratt, "On the differentiability of the solution to convex optimization problems," 2019.

[38] K. Kersting, N. M. Kriege, C. Morris, P. Mutzel, and M. Neumann, "Benchmark data sets for graph kernels," 2016. [Online]. Available: http://graphkernels.cs.tu-dortmund.de

[39] H. Xu, D. Luo, and L. Carin, "Scalable gromov-wasserstein learning for graph partitioning and matching," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

[40] B. Knyazev, X. Lin, M. R. Amer, and G. W. Taylor, "Spectral multigraph networks for discovering and fusing relationships in molecules," *ArXiv*, vol. abs/1811.09595, 2018.

[41] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 4805–4815.

[42] B. Amos and J. Z. Kolter, "OptNet: Differentiable optimization as a layer in neural networks," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, 2017, pp. 136–145.

[43] F.-Y. Sun, J. Hoffman, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," in *International Conference on Learning Representations*, 2019.

[44] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[45] F. R. K. Chung, "Spectral graph theory." Providence, RI: American Mathematical Society, 1997.

[46] Y. Ma, X. Liu, N. Shah, and J. Tang, "Is homophily a necessity for graph neural networks?" in *International Conference on Learning Representations*, 2022.

[47] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, pp. 127–163, 2000.

[48] C. L. Giles, K. D. Bollacker, and S. Lawrence, "Citeseer: An automatic citation indexing system," in *Proceedings of the Third ACM Conference on Digital Libraries*, ser. DL '98, 1998, p. 89–98.

[49] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.

[50] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," in *AAAI Conference on Artificial Intelligence*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:202539008

[51] D. Luo, C. Ding, F. Nie, and H. Huang, "Cauchy graph embedding," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11. Madison, WI, USA: Omnipress, 2011, p. 553–560.

[52] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[53] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking : Bringing order to the web," in *The Web Conference*, 1999.

[54] Y. Ma, X. Liu, T. Zhao, Y. Liu, J. Tang, and N. Shah, "A unified view on graph neural networks as graph signal denoising," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021.

[55] Y. Cao, M. Long, B. Liu, and J. Wang, "Deep cauchy hashing for hamming space retrieval," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1229–1237.

[56] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[57] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48, 2016, pp. 478–487.

[58] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, 2016, p. 40–48.

[59] P. Mernyei and C. Cangea, "Wiki-cs: A wikipedia-based benchmark for graph neural networks," *CoRR*, 2020. [Online]. Available: https://arxiv.org/abs/2007.02901

[60] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," *ArXiv*, 2019. [Online]. Available: https://arxiv.org/abs/1811.05868

[61] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *International Conference on Learning Representations*, 2019.

[62] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[63] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 4195–4205.

[64] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 9, pp. 11 121–11 128, Jun. 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/26317

[65] Y. Li, M. E. Ildiz, D. Papailiopoulos, and S. Oymak, "Transformers as algorithms: Generalization and stability in in-context learning," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 23–29 Jul 2023, pp. 19 565–19 594. [Online]. Available: https://proceedings.mlr.press/v202/li23l.html

[66] J. Von Oswald, E. Niklasson, E. Randazzo, J. Sacramento, A. Mordvintsev, A. Zhmoginov, and M. Vladymyrov, "Transformers learn in-context by gradient descent," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 23–29 Jul 2023, pp. 35 151–35 174. [Online]. Available: https://proceedings.mlr.press/v202/von-oswald23a.html

[67] Y. Bai, F. Chen, H. Wang, C. Xiong, and S. Mei, "Transformers as statisticians: Provable in-context learning with in-context algorithm selection," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 57 125–57 211. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/b2e63e36c57e153b9015fece2352a9f9-Paper-Conference.pdf

[68] K. He, C. Gan, Z. Li, I. Rekik, Z. Yin, W. Ji, Y. Gao, Q. Wang, J. Zhang, and D. Shen, "Transformers in medical image analysis," *Intelligent Medicine*, vol. 3, no. 1, pp. 59–78, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667102622000717

[69] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*

(Long and Short Papers), J. Burstein, C. Doran, and T. Solorio, Eds. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[70] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[71] V. P. Dwivedi and X. Bresson, "A generalization of transformer networks to graphs," *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.

[72] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a General, Powerful, Scalable Graph Transformer," *Advances in Neural Information Processing Systems*, vol. 35, 2022.

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

[74] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu, "Do transformers really perform badly for graph representation?" in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. [Online]. Available: https://openreview.net/forum?id=OeWooOxFwDa

[75] Q. Wu, W. Zhao, Z. Li, D. Wipf, and J. Yan, "Nodeformer: A scalable graph structure learning transformer for node classification," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=sMezXGG5So

[76] C. Liu, Y. Zhan, X. Ma, L. Ding, D. Tao, J. Wu, and W. Hu, "Gapformer: Graph transformer with graph pooling for node classification," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, E. Elkind, Ed. International Joint Conferences on Artificial Intelligence Organization, 8 2023, pp. 2196–2205, main Track. [Online]. Available: https://doi.org/10.24963/ijcai.2023/244

[77] S. Yin and G. Zhong, "Lgi-gt: Graph transformers with local and global operators interleaving," 2023.

[78] D. Chen, L. O'Bray, and K. Borgwardt, "Structure-aware transformer for graph representation learning," in *Proceedings of the 39th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, 2022.

[79] L. Wasserman, "Topological data analysis," *Annual Review of Statistics and Its Application*, vol. 5, pp. 501–532, 2018.

[80] F. Hensel, M. Moor, and B. Rieck, "A survey of topological machine learning methods," *Frontiers in Artificial Intelligence*, vol. 4, p. 681108, 2021.

[81] C. S. Pun, S. X. Lee, and K. Xia, "Persistent-homology-based machine learning: a survey and a comparative study," *Artificial Intelligence Review*, vol. 55, no. 7, pp. 5169–5213, 2022.

[82] H. Edelsbrunner, "Persistent homology: theory and practice," 2013.

[83] A. Zomorodian and G. Carlsson, "Computing persistent homology," in *Proceedings of the twentieth annual symposium on Computational geometry*, 2004, pp. 347–356.

[84] Q. Zhao and Y. Wang, "Learning metrics for persistence-based summaries and applications for graph classification," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[85] Y. Chen, B. Coskunuzer, and Y. Gel, "Topological relational learning on graphs," *Advances in neural information processing systems*, vol. 34, pp. 27 029–27 042, 2021.

[86] M. Horn, E. De Brouwer, M. Moor, Y. Moreau, B. Rieck, and K. Borgwardt, "Topological graph neural networks," in *International Conference on Learning Representations*, 2021.

[87] H. Edelsbrunner, D. Letscher, and A. Zomorodian, "Topological persistence and simplification," in *Proceedings 41st Annual Symposium on Foundations of Computer Science*, 2000, pp. 454–463.

[88] A. Zomorodian and G. Carlsson, "Computing persistent homology," *Discrete & Computational Geometry*, vol. 33, no. 2, pp. 249–274, 2005.

[89] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington, "A roadmap for the computation of persistent homology," *EPJ Data Science*, vol. 6, pp. 1–38, 2017.

[90] C. S. Pun, K. Xia, and S. X. Lee, "Persistent-homology-based machine learning and its applications–a survey," *arXiv preprint arXiv:1811.00252*, 2018.

[91] Q. Zhao, Z. Ye, C. Chen, and Y. Wang, "Persistence enhanced graph neural network," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2896–2906.

[92] Z. Yan, T. Ma, L. Gao, Z. Tang, and C. Chen, "Link prediction with persistent homology: An interactive view," in *International conference on machine learning*. PMLR, 2021, pp. 11 659–11 669.

[93] Z. Cang, L. Mu, and G.-W. Wei, "Representability of algebraic topology for biomolecules in machine learning based scoring and virtual screening," *PLoS computational biology*, vol. 14, no. 1, p. e1005929, 2018.

[94] Y. Chen and Y. R. Gel, "Topological pooling on graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 6, 2023, pp. 7096–7103.

[95] Y. Chen, Y. Gel, and H. V. Poor, "Time-conditioned dances with simplicial complexes: Zigzag filtration curve based supra-hodge convolution networks for time-series forecasting," *Advances in Neural Information Processing Systems*, vol. 35, pp. 8940–8953, 2022.

[96] M. Carrière, F. Chazal, Y. Ike, T. Lacombe, M. Royer, and Y. Umeda, "Perslay: A neural network layer for persistence diagrams and new graph topological signatures," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2786–2796.

[97] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," *Advances in neural information processing systems*, vol. 30, 2017.

[98] M. Horn, E. D. Brouwer, M. Moor, Y. Moreau, B. Rieck, and K. Borgwardt, "Topological graph neural networks," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=oxxUMeFwEHd

[99] H. Edelsbrunner, *A short course in computational geometry and topology*. Springer, 2014, no. Mathematical methods.

[100] A. J. Zomorodian, *Topology for computing*. Cambridge university press, 2005, vol. 16.

[101] G. Carlsson and M. Vejdemo-Johansson, *Topological data analysis with applications*. Cambridge University Press, 2021.

[102] V. P. Dwivedi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, "Graph neural networks with learnable structural and positional representations," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=wTTjnvGphYj

[103] M. S. Hussain, M. J. Zaki, and D. Subramanian, "Global self-attention as a replacement for graph convolution," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 655–665. [Online]. Available: https://doi.org/10.1145/3534678.3539296

[104] D. Kreuzer, D. Beaini, W. L. Hamilton, V. Létourneau, and P. Tossou, "Rethinking graph transformers with spectral attention," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[105] P. Li, Y. Wang, H. Wang, and J. Leskovec, "Distance encoding: design provably more powerful neural networks for graph representation learning," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[106] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, Jun. 2018, pp. 464–468. [Online]. Available: https://aclanthology.org/N18-2074

[107] H. Wang, H. Yin, M. Zhang, and P. Li, "Equivariant and stable positional encoding for more powerful graph neural networks," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=e95i1IHcWj

[108] W. Park, W.-G. Chang, D. Lee, J. Kim, and S. Hwang, "Grpe: Relative positional encoding for graph transformer," in *ICLR2022 Machine Learning for Drug Discovery*, 2022.

[109] G. Mialon, D. Chen, M. Selosse, and J. Mairal, "Graphit: Encoding graph structure in transformers," 2021.

[110] C. Cai, T. S. Hy, R. Yu, and Y. Wang, "On the connection between mpnn and graph transformer," in *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML'23. JMLR.org, 2023.

[111] J. J. Sutherland, L. A. O'Brien, and D. F. Weaver, "Spline-fitting with a genetic algorithm: a method for developing classification structureactivity relationships," *Journal of Chemical Information and Computer Sciences*, vol. 43, no. 6, pp. 1906–1915, 2003, pMID: 14632439. [Online]. Available: https://doi.org/10.1021/ci034143r

[112] N. Kriege and P. Mutzel, "Subgraph matching kernels for attributed graphs," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ser. ICML'12. Madison, WI, USA: Omnipress, 2012, p. 291–298.

[113] K. Borgwardt, underlineCS, S. Schönauer, S. Vishwanathan, A. Smola, and P. Kröger, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21 Suppl 1, pp. i47–56, 01 2005.

[114] N. Wale and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," in *Sixth International Conference on Data Mining (ICDM'06)*, 2006, pp. 678–689.

[115] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: datasets for machine learning on graphs," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[116] T. Wen, E. Chen, and Y. Chen, "Tensor-view topological graph neural network," in *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2024.

[117] Y. Chen and Y. R. Gel, "Topological pooling on graphs," in *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. [Online]. Available: https://doi.org/10.1609/aaai.v37i6.25866

[118] W. Zhu, T. Wen, G. Song, L. Wang, and B. Zheng, "On structural expressive power of graph transformers," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 3628–3637. [Online]. Available: https://doi.org/10.1145/3580305.3599451