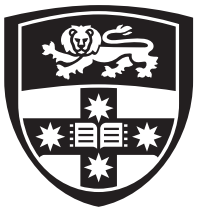


THE UNIVERSITY OF
SYDNEY

PHD THESIS

DISCIPLINE OF BUSINESS ANALYTICS

ENHANCING GRAPH REPRESENTATION LEARNING: DATA-AWARE ADVANCES IN GRAPH NEURAL NETWORKS

Author: **Ye Xiao**

Supervisor: Prof. Junbin Gao

Prof. Andrey Vasnev

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

The University of Sydney

BUSINESS SCHOOL

2026

Contents

List of Figures

List of Tables

CERTIFICATE OF ORIGINALITY

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Ye Xiao

Abstract

Graph-structured data is a fundamental data modality across diverse domains, in which the relations among entities can be as crucial as the entities themselves. However, the irregular nature of it renders traditional Euclidean-based deep learning methods inadequate for modeling its intricate interdependent structure. Graph Neural Networks (GNNs) have emerged as the de facto paradigm for learning effective representations that capture relational information, offering favorable performance through their message-passing mechanism. Despite their development, many recent advances address different tasks primarily in a model-centric view, revolving around more sophisticated designs to enhance representation learning. Considering that the evolving demand for classification further drives models toward more complex architectures, this thesis aims to contribute to the advancement of effective representation learning by placing data at the center of methodological design. To this end, the thesis delves into various data-aware schemes to efficiently and delicately exploit the potential of the graph, thereby enhancing the graph representation learning capabilities of GNNs. To be specific, the thesis follows a unified data-aware principle and investigates it via three practical perspectives of the classification problem of graphs, across three core levels of the representation learning framework, respectively. At the objective level, the thesis investigates multi-label graph classification and introduces a plug-in regularization that augments classification with Laplacian eigenmaps on the constructed metagraph, pulling structurally proximate samples together to preserve generalization on a global scale. At the data level, it studies binary graph classification under se-

vere class imbalance and exploits the potential of available samples by developing a training scheme specialized for supervised graph anomaly detection, with graph augmentation and prototype alignment that can solidify the decision margin in anomaly detection. At the architecture level, it examines classification in continual task streams and proposes a capacity-agnostic GNN framework that perceives external context stimuli and decouples task-specific subnetworks from each other under the parameter-isolation paradigm, substantially boosting the representation learning efficiency under the expanding model capacity scenario in continual learning. To contribute to a more practical basis for representation learning, the thesis advances techniques that yield more robust, effective, and comprehensive representations in real-world graph applications.

Acknowledgements

This thesis marks the end of my PhD journey; however, it carries the traces of people who walked alongside me, throughout the highs and lows along the way. I am sincerely grateful to all who supported me academically and personally during this journey.

First and foremost, I wish to express my deepest gratitude to my supervisor, Professor Junbin Gao, who has been my guide into academia since my Master's studies, awakening my passion for pursuing a higher level of mastery in machine learning, and continuing to illuminate my scholarly path throughout my PhD, even after it. I feel incredibly fortunate to have benefited from his open-door mentorship and profound wisdom, which have instilled in me an unwavering commitment to academic excellence and consistently propelled me forward. It is thanks to his forbearance, encouragement, and tireless guidance, all like the beacon through my darkest stretches, that I have been able to steady my footing and achieve genuine contributions in the academic field of machine learning. Beyond academia, Professor Junbin Gao has been more like a life mentor, whose words, actions, and way of thinking have enlightened me to become a more self-reliant person when facing challenges, which is a treasured mindset I will carry with me throughout my life.

My sincere gratitude also extends to my associate supervisor, Professor Andrey Vasnev, for his generous support, constructive feedback, and thoughtful perspectives, which have been invaluable in shaping my research and advancing my academic development. I am also deeply thank-

ful for all friends, colleagues, and staff at the Business School, especially in the Discipline of Business Analytics, who have been offering generous support and assistance both within and beyond academia, enabling me to move forward smoothly in this journey. Importantly, I gratefully acknowledge the support of the Australian Government through a Research Training Program (RTP) Scholarship, which provided financial support that enabled me to undertake my doctoral studies and focus on the research presented in this thesis.

Finally, none of this would have been possible without the unconditional and unwavering love, faith, and support of my family-my parents, my brother, and my grandparents. They have been a quiet source of strength throughout my PhD journey, endowing me with the courage to confront and overcome formidable challenges head-on. To everyone who has touched this path through momentous milestones and quiet gestures, with generosity, patience, and warmth, please accept my heartfelt gratitude.

Authorship Attribution Statement

This thesis was conducted at the University of Sydney, under the supervision of Prof. Junbin Gao and Prof. Andrey Vasnev, between 2022 and 2025. For works included in the thesis, I made the main contributions by conceiving the idea, designing the research, conducting the experiments, and drafting the paper.

This thesis is based on the following publications:

- **Ye Xiao**[*], Ruikun Li[*], Andrey Vasnev, & Junbin Gao. Enhanced Loss Function based on Laplacian Eigenmaps for Graph Classification. In *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, 2023.
- **Ye Xiao**[*], Ruikun Li[*], Andrey Vasnev, & Junbin Gao. On Leveraging Anomalies with Reference Alignment in Graph-level Anomaly Detection. In *Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2025.
- **Ye Xiao**[*], Ruikun Li[*], Zhenyu Yang, Andrey Vasnev, & Junbin Gao. Capacity-agnostic Parameter Isolation for Continual Graph Learning. In *Proceedings of International Conference on Machine Learning (ICML)*, 2026.

The author has also made substantial contributions to the following research works that are not included in this thesis, for which the author is the co-first author and has an equal contribution with the first author:

- Ruikun Li*, **Ye Xiao***, Xiaoxiao Ma*, Andrey Vasnev, & Junbin Gao. Gdendrite: On Heterophilous Graph Contexts Mining with Versatile Neural Dendrites Framework. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2025.
- Ruikun Li*, Dai Shi*, **Ye Xiao***, & Junbin Gao. Ufgtime: Mining Intertwined Dependencies in Multivariate Time Series via an Efficient Pure Graph Approach. In *Proceedings of International Conference on Very Large Data Bases (VLDB)*, 2025.

In addition to the statements above, in cases where Ye Xiao is not the corresponding author of a published item, permission to include the published material has been granted by the corresponding author.

Ye Xiao

As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Junbin Gao

Use of Generative Artificial Intelligence

The author used generative artificial intelligence to improve the clarity and presentation of the writing of this thesis. Any edits by generative artificial intelligence were carefully reviewed. All ideas, research, analyses, and conclusions are entirely the author's own.

Ye Xiao

Chapter 1

Introduction

The rise of deep learning, propelled by the relentless gains in computational resources and data at an unprecedented scale in recent decades (LeCun et al., 2015; Sarker, 2021), has redefined how modern applications are built, bringing a growing tide of data-driven explorations. These efforts have delved into developing various sophisticated Deep Neural Networks (DNNs) to recognize and harness patterns inherent in data of various types, giving rise to a wide range of intelligent applications that are capable of providing insights in the relevant domains. For instance, recurrent neural networks (RNNs) are widely adopted in speech recognition (Graves et al., 2013), machine translation (Cho et al., 2014), and time-series forecasting (Qin et al., 2017), and have long been popular for sequence and tabular data modeling (Salehinejad et al., 2017). Convolution neural networks (CNNs) are developed for grid-like data and still constitute a core building block in many practical systems for various computer recognition tasks (Z. Li et al., 2021), such as face recognition (Lawrence et al., 1997), medical treatment (Litjens et al., 2017), and autonomous driving (P. Li et al., 2019). In particular, transformers, largely owing to their strong capacity for capturing long-range dependencies of sequential data, have witnessed booming progress and become the mainstay of numerous artificial intelligence fields (Lin et al., 2022). In recent years, a growing body of advanced variants has continued to push the boundaries of state-of-the-art records, reinforcing a data-driven paradigm in which

models learn effective representations from real-world data to support downstream tasks.

1.1 Relational Worlds Beyond Points

With the development of digital infrastructures and services, much real-world data has presented increasingly heterogeneous and intertwined characteristics, reflecting rich interactions among entities rather than isolated ones. Under these circumstances, graphs, throughout which irregular and non-Euclidean topological structures are intrinsic, have emerged as a natural data representation and have attracted growing attention across broader research and industrial communities (J. Zhou et al., 2020). Graphs, composed of nodes representing entities and edges encoding mutual interactions between them, are ubiquitous around us. Many complex real-world systems can be universally depicted by the language of graph structures, within which intricate pairwise relationships are captured and quantified in abstraction. Some typical examples include molecules with atoms linked by chemical bonds, transportation systems where traffic conditions are monitored by sensors at the ends of roads, and social networks involving friendship, collaboration (Veličković, 2023; Z. Wu et al., 2020; Newman, 2012).

Despite its inherently irregular and non-Euclidean nature, graph data remains tightly connected to other structured data modalities. By explicitly modeling relationships among entities, it enriches these modalities with additional relational dependencies and, in turn, enhances their expressive capacity. For example, an image annotated with objects, attributes, and relationships (e.g, girl-feeding-elephant) can be organized into a scene graph, which has shown effectiveness in the vision-language domain (Krishna et al., 2017). In today's interconnected world, many real-world applications are shaped by the surge of graphs, bringing to light advanced graph-based solutions.

1.2 Learning from Relations

Beyond its regular-structured counterparts (e.g., 1D sequences and 2D images on grids), graph data impose further challenges in deep learning due to their structural complexity, with unordered nodes having varying numbers and features, and non-uniform neighborhood relationships across instances being common (J. Zhou et al., 2020). Therefore, those techniques for regular-structured data inherently assume a fixed and ordered local neighborhood structure, and thus induce inductive biases for Euclidean regularities (Bronstein et al., 2021) but are not naturally compatible with irregular graph topologies. As a result, conventional operations (e.g., recurrence in RNNs, convolution in CNNs, and self-attention in transformers) and enhancement techniques (e.g., rotation, cropping, and shifting) usually lead to suboptimal performance.

To handle such complexity, Graph Neural Networks (GNNs) (Gori et al., 2005) arose as a dominant paradigm in response to this need. Their many variants (Defferrard et al., 2016; Kipf & Welling, 2017; Veličković, Cucurull, et al., 2018; W. Hamilton et al., 2017; K. Xu et al., 2018; F. Wu et al., 2019) have since become a preeminent family for representation learning on graph data. By design, GNNs learn representations through the message-passing mechanism (Gilmer et al., 2017), where nodes collectively aggregate and transform information from their neighbors along edges, embedding graphs from non-Euclidean space into vectors of Euclidean space. With graph data emerging at an unprecedented pace, learning effective representations of graph data has become a fundamental building block for developing practical machine learning applications (F. Chen et al., 2020). In this context, GNNs, owing to their compelling capability to integrate the node information as well as the topological structure, have attracted extensive research interest and seen widespread adoption across diverse downstream tasks on graphs.

Classification tasks highlight the practical value of GNNs across multiple levels. For node classification and link prediction, GNNs have been widely adopted to

build recommender systems, where user-item interactions and user-user relations themselves can be modeled as graphs (Fan et al., 2019). GNNs have also become a powerful tool for financial fraud detection (D. Wang et al., 2019; Z. Liu et al., 2018) that can effectively identify suspicious accounts or transactions and uncover potential malicious transactions before they occur, enabling financial institutions to assess risk better and take preventative actions. For graph-level classification in cheminformatics and AI for medicine, GNNs showcase outstanding performance in predicting biochemical properties of molecules and proteins, such as enzymes versus non-enzymes (Xie et al., 2021; J. B. Lee et al., 2018).

Compared with traditional methods that solely consider immediate relations (i.e., one-hop neighbors), GNNs can leverage higher-order neighborhood information across multiple hops, which has proven beneficial for these scenarios (S. Wu et al., 2022; Z. Liu et al., 2018). More broadly, many real-world systems mentioned in Section 1.1 can be formulated as labeled graphs, making GNNs well-suited for learning rich representations for various classification tasks.

1.3 The Price of Relational Richness

Despite the success of GNNs, applications in practice are expanding to more complex settings, calling for further powerful representation learning capabilities (Khoshraftar & An, 2024). In this context, classification, as one of the most common learning paradigms and a long-serving testbed, has witnessed numerous efforts of GNN design aiming at learning discriminative representations of graphs. The landscape of GNNs in various types of classification tasks, however, becomes more nuanced as these pressures often drive GNN designs toward increasing architectural sophistication, prompting many studies to introduce complex 'task-specific' architectures or mechanisms so as to better exploit higher-order dependencies and complex relational patterns suited to different tasks (Z. Wu et al., 2020; T. Chen & Wong, 2020; S. Wu et al., 2022).

Graphs in the real world can be large and noisy (Pan et al., 2014), exhibiting sparse patterns confined only to small neighborhoods (J. B. Lee et al., 2018). In conventional multi-label graph classification, one typically expects GNNs to map graphs belonging to the same class to similar representations that help distinguish them from graphs of other classes. A typical GNN, therefore, combines graph convolutional layers with graph pooling (readout) layers, where the former learn node representations from node features and underlying graph structure, and the latter addresses the inconsistent number of nodes across graphs by aggregating node representations into a holistic but informative graph-level representation. In order to better capture and extract such patterns into representations, some approaches were dedicated to enhancing the message passing of the graph convolutional layers, such as incorporating substructure-aware or much deeper propagation, to better perceive structural information (G. Li et al., 2020; Corso et al., 2020; Bouritsas et al., 2022; D. Zeng et al., 2023). Some researchers have explored the graph pooling layer in various settings, such as hierarchical structures, to preserve comprehensive information in compacted graph representations (Z. Ying et al., 2018; J. Lee et al., 2019; J. Wu et al., 2022; C. Liu et al., 2022).

Graphs in the real world may be scarce and highly heterogeneous, with severely skewed and dispersed distributions that give rise to rare, hard-to-characterize patterns (Akoglu et al., 2015; Pang et al., 2021). This challenge has, in turn, motivated extensive research on graph anomaly detection, which can actually be formulated as binary graph classification with severe class imbalance. In this situation, popular GNN designs, particularly those relying on standard aggregation schemes, are prone to over-smoothing, producing representations where normal and anomalous graphs are nearly indistinguishable and thus restricting their practical utility (Qiao et al., 2025). Nonetheless, few advanced GNN techniques specifically designed for supervised graph anomaly detection have been proposed. G. Zhang, Yang, et al. (2022) designed a GNN that couples a graph convolution with a random-walk kernel to jointly exploit anomalous node attributes and anomalous substructures, detecting

anomalous graphs in a dual-discriminative manner. X. Ma et al. (2023) proposed a deep evolutionary graph mapping framework that maps each graph into a new feature space based on its similarity to a set of learned representative nodes, yielding representations that are both discriminative and interpretable for anomalous graphs. Dong et al. (2024) recognized spectral energy differences between normal and anomalous graphs as the Rayleigh Quotient, and designed a GNN framework that combines a Rayleigh Quotient learning module with a Chebyshev wavelet GNN and RQ-pooling.

Graphs in the real world may arrive in continually expanding task streams, with evolving features, structures, and classes that induce persistent distribution shifts across tasks (Yuan et al., 2026; X. Zhang, Song, & Tao, 2024). Implementing classification under this scenario, termed as continual graph learning, exposes GNNs to a so-called catastrophic forgetting problem (French, 1999), wherein they may overwrite previously useful parameters when continually optimized on new tasks, resulting in severe performance degradation on earlier tasks. Critically, the coupled evolution of node features and graph structures substantially complicates the learning of stable and effective representations (Q. Zhu et al., 2022; S. Hu et al., 2025), further accelerating the obsolescence of previously acquired knowledge and consequently rendering GNNs particularly vulnerable. In general, solutions for tackling this challenge fall into three paradigms, which are regularization-based, replay-based, and parameter isolation-based (X. Zhang, Song, & Tao, 2024; Febrinanto et al., 2023). In continual graph learning, however, the mechanisms of these approaches must contend not only with catastrophic forgetting but also with graph topology (H. Liu et al., 2021; Y. Xu et al., 2020; F. Zhou & Cao, 2021; X. Zhang, Song, Chen, & Tao, 2024; X. Zhang et al., 2022; P. Zhang et al., 2023), inevitably incurring higher computational and memory complexity. This issue is particularly pronounced in parameter isolation–based methods, which typically require enlarged model capacity to encapsulate more tasks (Yuan et al., 2026).

1.4 Beyond Model-Centric Design

Machine learning researchers often pursue higher model accuracy even when the improvement is marginal and comes with substantial complexity overhead (Sculley et al., 2015). Indeed, many works on GNNs gain higher benchmark performance on the aforementioned classification tasks, but they do so at the price of architecturally heavy designs. This deviates from the prevailing trend toward efficient designs in the development of DNNs for Euclidean data (Xiao et al., 2022). Although some efforts on efficiency have been witnessed and demonstrated that such architectural complexity is often misaligned with practical utility (R. Ying et al., 2018; F. Wu et al., 2019; He et al., 2020), the need-driven complexity cannot be entirely eliminated due to the necessity to accommodate diverse tasks. Moreover, many approaches that incorporate efficiency-oriented designs (e.g., pruning, distillation, sparsification, sampling, and compression) entail an inherent trade-off, sacrificing a certain degree of predictive accuracy, even if only marginal, in exchange for higher efficiency (Cheng et al., 2017; Shi et al., 2025; S. Wu et al., 2022; Kaler et al., 2022). In general, as summarized in recent surveys, a large body of work on GNNs remains predominantly model-centric (J. Zhou et al., 2020; Z. Wu et al., 2020; Zha et al., 2025). That is, most current methods treat graph data largely as a fixed input and thus prioritize increasingly complex architectural innovations to enhance graph representation learning (Y. Guo et al., 2024), rather than incorporating graph data into the design of the overall representation learning framework. Ultimately, further advances in graph representation learning require moving beyond purely model-centric designs and exploiting the intrinsic properties of graph data to improve both effectiveness and efficiency.

1.5 Contributions

The demand of going beyond model-centric design leads to the central question of this thesis: **How can graph data be systematically incorporated into the representation learning framework to unlock the full potential of GNNs in various classification tasks?** There are a few pioneering studies investigating this issue, particularly for various complex classification tasks mentioned in Section 1.3. Revolving around the central question, this thesis systematically decomposes it into three concrete research questions and advances GNNs' graph representation learning capability by explicitly incorporating a data-aware principle into methodological development across three key levels of the representation learning framework. These three research questions respectively focus on multi-label graph classification at the objective level, supervised graph-level anomaly detection at the data level, and continual graph learning at the architecture level. In each case, the choice reflects the main leverage point of the problem: the learning objective function for conventional classification, the severely imbalanced and shifting data distribution for anomaly detection, and the architecture's capability to retain and transfer knowledge in continual learning.

Research Question 1 (Objective Level): How can the standard objective function in GNN-based graph classification be enhanced to better exploit inter-graph information among graphs?

We propose LEELoss to enhance the supervised objective by incorporating graph-structure priors that explicitly model inter-graph relationships, leading to more robust and generalizable graph representations without relying on sophisticated GNN architectures. In response to Research Question 1, this work makes the following contributions:

- Integrating graph-structure priors at both the node and graph levels into a supervised learning objective, allowing the objective to exploit original struc-

tural information beyond the aggregated representations produced by message passing and pooling.

- Investigating the role of Laplacian eigenmaps in representation learning for graph classification and extending it from its conventional node-level formulation to a graph-level design that is more aligned with graph classification.

- Demonstrating that the objective block of GNNs remains under-explored, and showing that a carefully tailored loss function can consistently strengthen multiple vanilla GNN backbones for graph classification.

Conventional message passing and graph pooling mainly operate within individual graphs and do not explicitly capture global relations among graphs. To address this limitation, we propose Laplacian Eigenmaps Enhanced Loss (LEELoss), which integrates graph-structure information into Laplacian eigenmaps at both node and graph levels. By constructing suitable similarity matrices, LEELoss steers the learned representations to better preserve intrinsic structures and inter-graph relationships through the learning objective. Experiments show that LEELoss consistently improves the graph classification performance of multiple vanilla GNNs in graph-level classification.

Research Question 2 (Data Level): How to help models generalize better to unseen anomalies by fully exploiting and leveraging scarce anomalous graph samples?

We address this question by developing a specialized training scheme, GARA, for supervised graph-level anomaly detection, which reshapes the data distribution seen by the model via tailored graph augmentation and reinforces a more robust decision margin under severe class imbalance. In response to Research Question 2, this work makes contributions summarized as follows:

- Developing an intuitive and effective training scheme for supervised graph-

level anomaly detection that leverages anomalous graphs to help the model learn more discriminative graph representations and generalize to unseen environments, in contrast to existing methods that largely focus on fitting a fixed set of known anomalous patterns.

- Investigating the feasibility of mitigating class imbalance by explicitly augmenting anomalous graphs. By incorporating graph-level mixup as a principled graph augmentation strategy, this study addresses the limited use of graph-level augmentation in supervised graph-level anomaly detection.

- Achieving outstanding empirical performance without architectural sophistication. Experiments also demonstrate the robustness of this framework under the flipping phenomenon across different anomaly-downsampling directions, which represents a more challenging special-case scenario in anomaly detection.

Supervised graph-level anomaly detection is fundamentally constrained by the scarcity and limited diversity of labeled anomalies. While augmentation techniques that can potentially alleviate this issue are rarely exploited due to the higher graph-level complexity, even with modified losses to mitigate class imbalance, some sophisticated models are still trained on a narrow set of anomalous patterns and thus generalize poorly to unseen cases. To address this limitation, we propose the Graph Augmentation-based Reference Alignment framework (GARA), which consists of two components: the Augmented-Task Generation Module and the Reference Alignment Module. These two modules respectively reshape the data distribution seen by the model via tailored graph augmentation and encourage anomalous and normal graphs to align with normal references, reinforcing a more robust decision margin under a severe class imbalance scenario. Evaluation of these two modules demonstrates that GARA effectively enriches anomalous graphs and achieves outstanding performance even under the more challenging special-case scenario.

Research Question 3 (Architecture Level): How to sustain computational efficiency of the parameter-isolation paradigm under the model's capacity expansion while enabling flexible and extensive cross-task parameter reuse?

We develop the CAGNN framework to enable flexible cross-task parameter reuse while keeping training and inference as efficient as a lightweight vanilla GNN. In response to Research Question 3, this work makes the following contributions:

- Identifying an efficiency bottleneck of existing parameter isolation-based methods in continual learning, which is an efficiency failure induced by continual capacity expansion for accommodating more tasks, and highlighting computational efficiency as a key dimension for evaluating isolation-based methods.
- Presenting a biological neuron-inspired GNN framework that leverages task-specific context to guide efficient subnetwork construction and yields a capacity-agnostic design for efficiency, simultaneously addressing catastrophic forgetting and improving computational efficiency.
- Establishing theoretical guarantees for the effectiveness of the proposed architecture, and introducing a regularization term that facilitates the model's scalability to future tasks.

Parameter isolation is a prevailing paradigm for continual learning, yet it inevitably demands enlarged model capacity as task streams grow and parameter usage becomes saturated. In practice, many isolation-based methods still rely on full-network propagation, where task-specific computation is carried out over the entire shared backbone, and masked parameters continue to participate in propagation. With the model capacity being expanded to accommodate more future tasks, the capacity-driven overhead accumulates and eventually leads to another failure beyond forgetting. We identify it as a critical efficiency bottleneck of most existing parameter isolation-based methods, and propose the Capacity-Agnostic Graph Neural Net-

work (CAGNN) with Context-aware Hierarchical Masking (CHM) as the solution. CAGNN possesses a neuron-inspired architecture that can perceive task contextual information for the construction of subnetworks and is structurally compressible. By leveraging task-specific context as prior knowledge, CHM efficiently identifies task-specific subnetworks, supports parameter sharing across tasks for greater scalability, and stores masks in a memory-friendly manner. Crucially, CHM decouples task-specific subnetworks from the holistic network, setting aside unused parameters from other tasks. Therefore, CAGNN is able to train and infer as efficiently as a vanilla GNN while possessing larger parameter capacity. CAGNN not only preserves knowledge learned from previous tasks, but also saves training time by up to 2 times compared to baseline methods under the same parameter capacity budget, achieving state-of-the-art performance in handling both catastrophic forgetting and computational complexity.

Recognizing that data serves as the fundamental substrate for representation learning, this thesis moves beyond further burdening GNNs with increasing architectural complexity and follows a unified principle across these three perspectives, developing advanced data-aware GNN frameworks that collectively contribute to a more practical basis for the representation learning of graph data in more complex real-world applications.

1.6 Thesis Outline

This thesis is organized into six chapters, with the main content of each chapter briefly described as follows:

Chapter 1: Introduction

This chapter introduces the research background and motivation, outlines a key research gap in graph representation learning, and presents the three research questions that frame the thesis.

Chapter 2: Background

This chapter reviews the background of graph neural networks (GNNs) and graph representation learning, establishing fundamental preliminaries that form the foundation for the technical developments in subsequent chapters.

Chapter 3: Enhancing Objective Function by Laplacian Eigenmaps

This chapter addresses Research Question 1. It investigates objective-level designs for graph classification, aiming to strengthen the learning objective to obtain more discriminative and robust graph representations.

Chapter 4: Leveraging Precious Anomalies for Class Imbalance

This chapter addresses Research Question 2. It focuses on supervised graph-level anomaly detection and proposes a data-level training scheme to improve generalization to unseen cases by better leveraging scarce anomalies.

Chapter 5: Promoting Model Efficiency under Capacity Expansion

This chapter addresses Research Question 3. In this chapter, an under-explored problem regarding model efficiency in continual graph learning is discussed and highlighted, and a novel capacity-agnostic framework is presented that contributes to advancing the parameter-isolation paradigm.

Chapter 6: Conclusions

This chapter concludes the thesis by summarizing the main findings and contributions of the three research works, and discusses potential future directions from data-aware advances toward data-centric advances.

Chapter 2

Background

To provide a foundation for the understanding of the three research works, this chapter introduces the fundamental background knowledge of graph representation learning and graph neural networks (GNNs), providing the key concepts and the problem settings relevant to this thesis. It also formalizes notations and terminologies, and presents the essential preliminaries that will be consistently applied throughout subsequent chapters, serving as the basis for the proposed methods and analyses. Note that in this thesis, we denote scalars by italicized lowercase letters (e.g., x), vectors by italicized bold lowercase letters (e.g., $\boldsymbol{x}$), matrices by italicized bold uppercase letters (e.g., $\boldsymbol{X}$), and sets by calligraphic letters (e.g., $\mathcal{X}$) for clarity and consistency.

2.1 Graphs in Machine Learning

Graphs are an expressive tool for representing entities and their pairwise relationships. Consisting of a collection of nodes (vertices) representing entities and edges (links) encoding mutual interactions between them, graphs provide an abstract but universal description of a wide range of real-world complex systems, such as transaction networks, social networks, citation networks, and molecular structures, and so on. According to the taxonomy of graphs, they can be categorized along several

orthogonal dimensions (A single graph can belong to multiple categories at once). Graphs can be static or dynamic depending on whether their topology evolves over time; Graphs can be directed or undirected depending on whether the direction of interactions matters; Graphs can be weighted or unweighted depending on whether their edges encode interaction strength; According to their node attributes, graphs can be homogeneous or heterogeneous if they have nodes of one types or multiple types; From a structural perspective, graphs can also be bipartite graphs where nodes are partitioned into two disjoint sets with edges only connecting across sets, or hypergraphs which possess hyperedges that can simultaneously connect multiple nodes to model higher-order interations. This thesis centers on elevating graph data to a design principle of different levels of the representation learning framework. Building on this perspective and for the sake of consciousness, the thesis is presented primarily in the context of static graphs, which are the most commonly used setting across many graph learning tasks and benchmarks.

2.1.1 Definition of Graph

Mathmatically, an graph $\mathcal{G}$ is defined as a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_0, v_1, ..., v_{|\mathcal{V}|}\}$ is set of $|\mathcal{V}|$ nodes and $\mathcal{E} = \{e_0, e_1, ..., e_{|\mathcal{E}|}\} \in \mathcal{V} \times \mathcal{V}$ is set of $|\mathcal{E}|$ edges of $\mathcal{G}$, where $|\mathcal{V}|$ and $|\mathcal{E}|$ are the number of nodes and edges. For an edge connecting node $v \in \mathcal{V}$ and node $u \in \mathcal{V}$, e can be denoted as a tuple $(v, u) \in \mathcal{E}$. For an edge connecting node $v \in \mathcal{V}$ and the node itself, the edge $e = (v, v) \in \mathcal{E}$ is referred to as a self-loop. Two nodes that are directly connected by an edge are called one-hop neighbors of each other, and the neighborhood of v usually collects one-hop neighbors and can be defined as $\mathcal{N}(v) = \{u \in \mathcal{V} \mid (u, v) \in \mathcal{E}\}$. Nodes that can be reached via two consecutive edges (i.e., through an intermediate node) are called two-hop neighbors, and so on for higher-order neighborhoods. According to the homophily assumption (McPherson et al., 2001), nodes that are connected usually share similar properties and are more likely to belong to the same class.

The information encoded by a graph can be broadly decomposed into two complementary components: attribute information and structure information. Attribution information is typically associated with $\mathcal{V}$ and represented by a node feature matrix $\boldsymbol{X} \in \mathbb{R}^{N \times F}$, where $N = |\mathcal{V}|$, F is the number of feature, its i-th row corresponds to v_i. Structural information is most commonly expressed by an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$, where each row and column index a node in $\mathcal{V}$. For unweighted graphs, its (i, j)-th entry $\boldsymbol{A}_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$ and $\boldsymbol{A}_{ij} = 0$ otherwise. For weighted graphs, $\boldsymbol{A}_{ij} \in \mathbb{R}$ if there exists an edge, quantifying the strength, frequency, or distance of the relationship. This representation also naturally accommodates undirected graphs where $\boldsymbol{A}$ is symmetric, and directed graphs where $\boldsymbol{A}$ is asymmetric. Equivalently, a graph can be read as $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A})$. Figure 2.1 demonstrates a simple graph example and how its structural information is presented by the adjacency matrix.

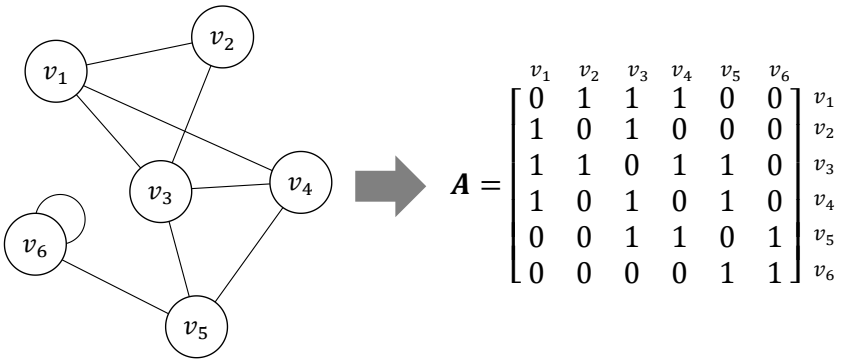


Figure 2.1: An example of an undirected and unweighted graph (Left) with six nodes and one self-loop, and its adjacency matrix $\boldsymbol{A}$ (Right).

2.1.2 Graph Laplacian

Graph structure can also be characterized by the graph Laplacian, which is obtained by degree-based transformation on the adjacency matrix and contributes useful algebraic and spectral properties in machine learning.

Definition 2.1.1 (Graph Laplacian)**.** Given a graph $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A})$, the unormalized graph Laplacian, denoted as $\boldsymbol{L}$, is defined as:

$$\boldsymbol{L} := \boldsymbol{D} - \boldsymbol{A} \tag{2.1}$$

where the degree matrix $\boldsymbol{D}$ is diagonal with entries $\boldsymbol{D}_{ii} = \sum_{j=1}^{N} \boldsymbol{A}_{ij}$, summarizing the number of one-hop neihbors of each node.

Apart from the unnormailzed form of $\boldsymbol{L}$, two widely used normalized variants are the symmetric normalized Laplacian and the random walk normalized Laplacian. The symmetric normalized Laplacian is given by:

$$\boldsymbol{L}_{sym} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2} \tag{2.2}$$

where $\boldsymbol{I}$ is the identity matrix with the same size of $\boldsymbol{A}$. The random walk normalized Laplacian is given by:

$$\boldsymbol{L}_{rw} = \boldsymbol{D}^{-1}\boldsymbol{A} = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{A} \tag{2.3}$$

Both variants retain key properties of $\boldsymbol{L}$. For undirected graphs with nonnegative edge weights, $\boldsymbol{L}$ is symmetric such that $\boldsymbol{L} = \boldsymbol{L}^{\top}$, and the following quadratic form holds for $\forall \boldsymbol{x} \in \mathbb{R}^{N\times 1}$:

$$\begin{aligned} \boldsymbol{x}^{\top}\boldsymbol{L}\boldsymbol{x} &= \frac{1}{2}\sum_{i}\sum_{j}\boldsymbol{A}_{ij}(\boldsymbol{x}_i - \boldsymbol{x}_j)^2 \\ &= \sum_{(i,j)\in\mathcal{E}}(\boldsymbol{x}_i - \boldsymbol{x}_j)^2 \end{aligned} \tag{2.4}$$

Equation (2.4) immediately tells that $\boldsymbol{L}$ is positive semi-definite and has real and nonnegative eigenvalues, ordered as $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Moreover, it highlights that the graph Laplacian $\boldsymbol{L}$ measures the smoothness of a graph signal $\boldsymbol{x}$, quantifying the extent to which a graph signal $\boldsymbol{x}$ varies across connected nodes.

These properties of $\boldsymbol{L}$ offer a concise lens for understanding how graph structure shapes graph representation learning, and form the theoretical basis of graph spectral domains (e.g., Harmonic functions on graphs (X. Zhu et al., 2003) and Laplacian Eigenmaps (Belkin & Niyogi, 2003)). This line of theory laid the foundation of the earliest spectral graph neural network, which applies spectral filters to graph signals via the eigen-decomposition of $\boldsymbol{L}$ (Bruna et al., 2013), and ultimately gave rise to modern message-passing GNNs, where graph representations are learned by iteratively aggregating information from local neighborhoods (Kipf & Welling, 2017; Gilmer et al., 2017). Generally, the graph Laplacian provides a useful theoretical explanation for both the strengths and limitations of message-passing GNNs, within which many widely adopted graph convolutional operations can be implicitly interpreted by Laplacian smoothing. This is equivalent to low-pass filtering that encourages representations across adjacent nodes to be consistent, and explains the denosing functionality of the message-passing mechanism. It also explains the well-known over-smoothing phenomenon that those deep message-passing GNNs usually suffer from, where representations of nodes tend to become increasingly similar and thus indistinguishable.

2.2 Graph Representation Learning

Graph representation learning concerns learning low-dimensional vector representations of graph data that capture both graph features and topological information for downstream tasks. It can be viewed as the graph-specific instantiation of the representation learning of a broader view. Because graphs inherently couple entities through edges, representation learning on graphs typically relies on graph-specific models and has gained increasing attention in machine learning literature (W. L. Hamilton, 2020; F. Chen et al., 2020). Similar to the broad representation learning, the effectiveness of graph representation learning is ultimately governed by the interplay among the quality of the underlying graph data, the way

graphs are constructed and processed, the model that explicitly encodes graph information, and the learning objective that the model is optimized for, which together determine how and what information is retained in learned representations. Research on graph representation learning can be organized by where methodologies are placed in the representation learning framework. Therefore, this section reviews from three perspectives, which are data, model architecture, and the learning objective, to summarize the background knowledge relevant to this thesis. Before introducing these three parts, this section first reviews the major representation learning tasks on graphs (e.g., node-level, edge-level, and graph-level) under different training paradigms.

2.2.1 Representation Learning Tasks on Graphs

2.2.1.1 Training Paradigm

First, graph representation learning tasks can be classified by the type of supervision available, which are supervised learning, unsupervised learning, and semi-supervised learning.

Supervised Learning Supervised learning assumes that labels are provided for the prediction target. Based on supervisory signals, the model is trained to directly optimize the predictive performance by learning representations, which are then fed into a prediction head (usually a fully connected layer or a multi-layer perceptron (MLP)) for generating predictive values. Typical supervised learning tasks for representation learning are classification and regression, while the former is more prevalent in the graph domain.

Unsupervised Learning Unsupervised learning aims to learn representations (commonly called embedding in unsupervised learning) without relying on explicit labels of the prediction target, but instead directly from the data itself, such as graph structure and node features. Among unsupervised learning branches, the

most popular subset is called self-supervised learning. Self-supervised learning constructs pretext supervision signals by generating pseudo labels from the input data, and enables the model to capture more implicit but informative patterns. Typical unsupervised learning tasks for graph representation learning include binary classification, where patterns of negative samples are usually modelled to provide contrastive views (e.g., anomaly detection and link prediction), and graph clusterings (Y. Liu et al., 2022).

Semi-Supervised Learning Representation learning on graphs further accounts for another prevalent one, called semi-supervised learning, which stands in the middle of the above two most conventional training paradigms. Semi-supervised learning is most notable in node and edge prediction on a single observed graph, where the full graph (node features and edge connectivity) is available but with only a small portion of nodes being labeled. The learning objective of semi-supervised learning is to encourage representation to generalize from labeled to unlabeled nodes through the shared graph structure.

2.2.1.2 Learning Subject

Second, graph representation learning tasks can be more specifically categorized by the learning subject, according to the different granularities of the graph.

Node Level Node-level learning treats each node as an instance and aims to learn a representation for each node. At the node level, typical tasks are node classification, where node representations are mapped to label probabilities via a classification head; node regression, where continuous targets are predicted from learned node representations through a regression head; and node clustering, which groups nodes into coherent communities based on similarity in the learned representation space. Nevertheless, among node-level tasks, most scholarly attention usually falls into node classification, often under the semi-supervised learning paradigm intro-

duced above (Y. Liu et al., 2022). Given a graph $\mathcal{G}$ with a feature matrix $\boldsymbol{X}$ and an adjacency matrix $\boldsymbol{A}$, the pipeline for a node-level task is presented as the following process:

$$\underbrace{\mathcal{G}(\boldsymbol{X}, \boldsymbol{A})}_{\text{Input Graph}} \xrightarrow[\text{Encoders}]{\text{Graph}} \underbrace{\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_N\}}_{\text{Node Representations } \boldsymbol{H}} \implies \underbrace{\boldsymbol{h}_i \xrightarrow[\text{Head}]{\text{Prediction}} y_i}_{\text{Prediction}} \tag{2.5}$$

where $\boldsymbol{h}_i$ is a representation vector of v_i and y_i is its corresponding prediction target.

Edge Level Edge-level tasks are similar to node-level classification tasks in most situations, but are slightly different in terms of their prediction targets. They both learn node representations for the downstream tasks. Rather than on an individual node, edge-level learning focuses on predicting properties of node pairs by estimating the likelihood that an edge exists or will form between two nodes. A typical approach to represent edge representation is to combine node representations using a concatenation or scoring function. The most common edge-level tasks are edge classification and link prediction. Link prediction aims to determine whether an edge should exist between a pair of nodes, and is typically treated as a binary classification problem. While edge classification assumes edges are given and focuses on predicting the type or attributes of each edge. It is usually based on the representations of the connected nodes and, when available, explicit edge features. The pipeline for an edge-level task is presented as follows:

$$\underbrace{\mathcal{G}(\boldsymbol{X}, \boldsymbol{A})}_{\text{Input Graph}} \xrightarrow[\text{Encoders}]{\text{Graph}} \underbrace{\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_N\}}_{\text{Node Representations } \boldsymbol{H}} \implies \boldsymbol{h}_i, \boldsymbol{h}_j \rightarrow \underbrace{s(\boldsymbol{h}_i, \boldsymbol{h}_j) \xrightarrow[\text{Head}]{\text{Prediction}} y_{ij}}_{\text{Prediction}} \tag{2.6}$$

where $s(\cdot, \cdot)$ refers to the scoring function or the concatenation operation, and y_{ij} is the predicted edge label between v_i and v_j.

Graph Level Graph-level learning treats each entire graph as an instance and aims to produce a single graph representation vector of the entire graph for prediction. In addition to modeling the structural information within each graph, graph-

level tasks operate over a collection of graphs and are therefore closer to conventional learning settings in Euclidean space, where each graph is treated as an independent instance. However, compared with node-level and Euclidean-based learning, where each instance has a fixed size (e.g., feature vector or image pile), graph-level tasks must handle varying graph sizes and structures. Therefore, graph-level learning usually favors a pipeline that combines a graph encoder with a pooling strategy to obtain a uniform-dimensional graph representation before the prediction head. This setting underlies many graph classification benchmarks, where graph pooling is essential for summarizing local node patterns into a global one (W. L. Hamilton, 2020). Similar to node-level tasks, graph-level tasks include graph classification, graph regression, and graph clustering. The following illustrates the pipeline for a graph-level task:

$$\underbrace{\mathcal{G}(\boldsymbol{X}, \boldsymbol{A})}_{\text{Input Graph}} \xrightarrow[\text{Encoders}]{\text{Graph}} \underbrace{\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_N\}}_{\text{Node Representations } \boldsymbol{H}} \Longrightarrow \text{POOL}(\boldsymbol{H}) \rightarrow \underbrace{\boldsymbol{h}_{\mathcal{G}} \xrightarrow[\text{Head}]{\text{Prediction}} y_{\mathcal{G}}}_{\text{Prediction}} \tag{2.7}$$

here POOL$(\cdot)$ is a graph pooling operation on node representation matrix $\boldsymbol{H}$, $\boldsymbol{h}_{\mathcal{G}}$ and $y_{\mathcal{G}}$ are a graph representation vector and predicted graph label.

Three research questions discussed in this thesis revolve around classification at the node level and the graph level. To be specific, Chapter 3 focuses on conventional supervised graph-level classification. Chapter 4 addresses supervised graph-level binary classification with severe class imbalance, that is, supervised graph-level anomaly detection. Chapter 5 considers both semi-supervised node-level and supervised graph-level classification under the continual learning setting.

2.2.2 Data-Centric Techniques on Graphs

Graph representation learning relies critically on abundant and high-quality data to effectively capture the underlying information of graphs. In practice, however, real-world graphs usually exhibit various problematic attributes, including label scarcity,

heterogeneousness, data incompleteness, data noise, data complexity, and data distribution shift, and thus introduce data-level challenges that may hinder effective representation learning (J. Zhou et al., 2025). To illustrate, label scarcity may cause models to overfit the limited labeled instances and fail to generalize beyond the observed supervision. In the graph representation learning pipeline, the data level mainly concerns how the graph signals, before being fed into the model, can be shaped through various data-centric techniques to mitigate the above challenges, thereby enabling more effective representation learning. As researchers' focus has increasingly shifted toward a data-centric paradigm, graph data, not only the model, has become a primary lever of performance (Zha et al., 2025). In what follows, some fundamental methods on the data level will be introduced.

2.2.2.1 Graph Sampling

A first family of data-centric methods on graphs is graph sampling, whose purpose is to make graph representation learning scalable by controlling the computational and memory overhead induced by complex neighborhood structure. This is particularly important for large-scale or dense graphs, because typical GNNs introduce the complexity of $\mathcal{O}(NF^2 + |\mathcal{E}|F)$, and its receptive field can grow rapidly when recursively aggregating information from neighborhoods across layers, ultimately leading to neighborhood explosion. Technically, sampling methods operate by associating nodes or edges with sampling probabilities and removing those with a higher possibility, and they are heuristically grouped into random sampling and importance sampling.

Random sampling treats sampled nodes as approximately drawn from a uniform distribution. A fundamental strategy is uniform neighbor sampling, which samples a fixed number of neighbors for each node, such that:

$$\mathcal{S}(v) \subseteq \mathcal{N}(v) \tag{2.8}$$

For each node u in the sampled neighbor set $\mathcal{S}(v)$, it follows:

$$u \sim \text{Uniform}(\mathcal{N}(v)), \quad q(u|v) = \frac{1}{d_v} \tag{2.9}$$

where $q(u|v)$ is the probability measure for a particular neighbor u given a vertice v, and $d_v = |\mathcal{N}(v)|$ is the degree of v. This sampling strategy has been incorporated as a core idea for the architecture design in GraphSAGE (W. Hamilton et al., 2017), where each node aggregates information from a uniformly sampled neighborhood. Beyond each seed node, Cluster-GCN (Chiang et al., 2019) further applies random sampling in units of subgraphs through partitioning a graph into several clusters, and randomly selects and combines a fixed number of clusters into a subgraph for training. Despite random sampling's simplicity, it may ignore some informative neighbors.

Importance sampling assigns non-uniform sampling probabilities to preserve informative neighbors and can thus reduce the variance from sampling, making representation learning more stable. The general formulation of it, according to FastCGN (J. Chen et al., 2018) and LADIES (Zou et al., 2019), can be achieved by the following unbiased estimation:

$$\mathbb{E}\left[\frac{1}{|\mathcal{S}(v)|}\sum_{u\in\mathcal{S}(v)}\frac{\phi(v,u)}{q(u|v)}\right] = \sum_{u\in\mathcal{N}(v)}\phi(v,u), \quad u \in \mathcal{S}(v) \ \sim q(u|v) \tag{2.10}$$

where $\phi(v,u)$ is the contribution of neighbor u to vertice v. Although sampling is sometimes described as a data-level technique, importance sampling in scalable GNNs is more appropriately viewed as an algorithmic component of the architecture (e.g., as part of the sampling-based message-passing mechanism).

2.2.2.2 Graph Generation

Data-centric methods also introduce graph generation, which instead aims to learn a distribution over graphs and synthesize new graphs with both high fidelity and

diversity. In literature (Y. Guo et al., 2024), graph generation differs mainly by what object they generate, including node sequence, adjacency matrix, latent node embedding, and spectral components. One core method discussed in Chapter 4 is most closely related to graphon-based techniques, which can be interpreted as a non-parametric adjacency matrix generation.

Definition 2.2.1 (Graphon)**.** A graphon is a symmetric measurable function $W : [0,1]^2 \mapsto [0,1]$, where symmetric means $W(x,y) = W(y,x)$ for any x, y.

Definition 2.2.2 (Associated Graphon of a Graph)**.** For a graph $\mathcal{G}$ with N nodes, its associated graphon $W_{\mathcal{G}} : [0,1]^2 \mapsto [0,1]$ is constructed by partitioning $[0,1]$ into n equal-length intervals $I_1, I_2, ..., I_N$ and defining $W_{\mathcal{G}}$ to be 1 on each block $I_i \times I_j$ whenever the e_{ij} exists, and 0 otherwise.

Generally, the associated graphon of a graph $W_{\mathcal{G}}$ is essentially the adjacency matrix represented as a step function, which specifies edge probabilities. For graphon-based graph generation, the adjacency matrix $\boldsymbol{A}$ of a synthetic graph can be generated by sampling edges independently conditioned on random variables $\{U\}$, which determine how likely a node v connects to others through the graphon (Diaconis & Janson, 2007):

$$\begin{aligned} \boldsymbol{A}_{ij} \mid U_i, U_j &\sim \text{Bernoulli}(W(U_i, U_j)) \\ U_i, U_j &\overset{\text{i.i.d.}}{\sim} \text{Uniform}(0,1) \end{aligned} \tag{2.11}$$

For the node sequence-based method, it simplifies a graph into an ordered sequence and models the sequence using autoregressive methods (J. You et al., 2018):

$$p(\mathcal{G}) = \prod_{i=1}^{N+1} p(S_i^\pi | S_{<i}^\pi) \tag{2.12}$$

where $p(\mathcal{G})$ is the distribution of a graph $\mathcal{G}$ and $S^\pi = (S_1^\pi, ..., S_N^\pi)$ is the node sequence mapped from $\mathcal{G}$. Latent node embedding generation considers generating a graph

in an indirect way, via representing the adjacency matrix $\boldsymbol{A}$ by node embeddings $\boldsymbol{z}$, such as:

$$\boldsymbol{A}_{ij} \mid \boldsymbol{z}_i, \boldsymbol{z}_j \sim \text{Bernoulli}(\sigma(\boldsymbol{z}_i^\top \boldsymbol{z}_j)) \tag{2.13}$$

where $\sigma(\cdot)$ is a squashing function. VGAE is a practical example that utilizes GNNs to learn the node representations into a multivariate Gaussian distribution and reconstructs the graph structure through link prediction tasks (Kipf & Welling, 2016). For spectral component-based graph generation, such as SPECTRE (Martinkus et al., 2022), it primarily involves utilizing an adversarial network and a diffusion model to generate the eigenvalues and eigenvectors of graphs that encode a graph's global structure. Overall, these methods provide different trade-offs between expressivity, scalability, and invariance to node permutations. In this thesis, we emphasize the graphon as a particularly efficient and effective method, which enables other data-centric techniques such as mixing and interpolation in the function space for augmentation.

2.2.2.3 Graph Augmentation

Among data-centric methods, data augmentation plays a prominent role in enhancing representation learning at the data level. By modifying existing data or synthesizing new data, data augmentation serves as an implicit regularizer that can mitigate the model's overfitting, and has been widely employed in computer vision and natural language processing (J. Zhou et al., 2025). However, conventional augmentation techniques designed for Euclidean data are often ill-suited to graphs due to their irregular structure. To address this, graph augmentation has been developed to enrich both node features and topological information, and has now been widely applied in various graph representation learning tasks. Especially in graph contrastive learning, augmentation is crucial for constructing contrasting views that help the model capture positive patterns more accurately (Y. You et al., 2020;

Trivedi et al., 2022). As illustrated in Figure 2.2, in graph contrastive learning, one can create multiple augmented views of each original graph. Two views derived from the same source graph are treated as a positive pair, while views coming from different source graphs are treated as negative pairs. A contrastive objective then encourages the model to bring positive pairs closer and separate negative pairs in the representation space, thereby enhancing the distinguishability between graphs.

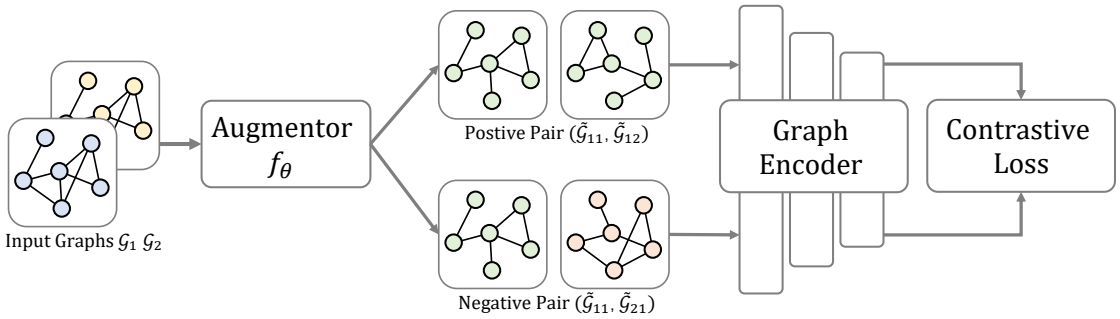


Figure 2.2: An example of how graph augmentation enhances graph contrastive learning.

In abstraction, graph augmentation can be expressed as a transformation function on graphs:

$$f_\theta : \mathcal{G} = (\boldsymbol{X}, \boldsymbol{A}) \mapsto \tilde{\mathcal{G}} = (\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{A}}) \tag{2.14}$$

In terms of whether θ is optimized before or during training, graph data augmentation techniques can be broadly classified as learnable and non-learnable methods. Because graph augmentation spans a wide range of techniques with diverse formulations, in this section, we primarily focus on Graph Mixup methods, on which the training scheme introduced in Chapter 4 is built.

Definition 2.2.3 (Graph Mixup)**.** Given two graphs $\mathcal{G}_i = (\boldsymbol{X}_i, \boldsymbol{A}_i)$ and $\mathcal{G}_j = (\boldsymbol{X}_j, \boldsymbol{A}_j)$ and their corresponding labels y_i and y_j, graph mixup first aligns them into an augmentation space $\mathbb{G}$, and then performs linear interpolation on them to

generate an augmented graph $\tilde{\mathcal{G}}$ with the corresponding mixed label $\tilde{y}$. Specifically:

$$\begin{aligned}(\hat{\mathcal{G}}_i, \hat{\mathcal{G}}_j) &= \text{Align}(\mathcal{G}_i, \mathcal{G}_j) \\ \tilde{\mathcal{G}} &\sim p(\tilde{\mathcal{G}} \mid (1-\lambda)\hat{\mathcal{G}}_i + \lambda\hat{\mathcal{G}}_j) \\ \tilde{y} &= (1-\lambda)y_i + \lambda y_j\end{aligned} \tag{2.15}$$

where $\text{Align}(\cdot,\cdot)$ is the alignment operation, $\lambda \in [0,1]$ controls the weight of two graphs, and $\hat{\mathcal{G}}_i, \hat{\mathcal{G}}_j \in \mathbb{G}$.

Existing graph mixup methods mainly differ from each other in how they align two graphs into the augmentation space $\mathbb{G}$ before performing interpolation. For instance, ifMixup (H. Guo & Mao, 2023a) pads graphs of unequal sizes by adding virtual nodes with zero features, enabling mixing directly in the topological space. IGM (Jia et al., 2024) maps graph pairs into a shared graph representation space using a graph encoder and graph graph pooling. G-Mixup (Han et al., 2022) estimates class-wise graphons and samples synthetic graphs from the mixed graphons. S-Mixup (Ling et al., 2023) computes a soft assignment matrix via a graph matching network to align a graph to the other.

There are various other advanced graph augmentation techniques, which can be more specifically categorized into structure-oriented augmentation, feature-oriented augmentation, and label-oriented augmentation (Y. Guo et al., 2024). For structure-oriented methods, one other example is edge perturbation, such as DGI (Veličković, Fedus, et al., 2018), which modifies the graph structure by randomly adding or dropping edges, following:

$$\tilde{\boldsymbol{A}} = \boldsymbol{A} \oplus \boldsymbol{C}, \quad \boldsymbol{C}_{ij} \sim \text{Bernoulli}(\rho) \tag{2.16}$$

where $\boldsymbol{C}$ is the perturbation matrix determined by the perturbation ratio and $\oplus$ refers to the XOR operation. Graph diffusion, such as personalized PageRank (Page et al., 1999) and heat kernel (Kondor & Lafferty, 2002), is another effective method

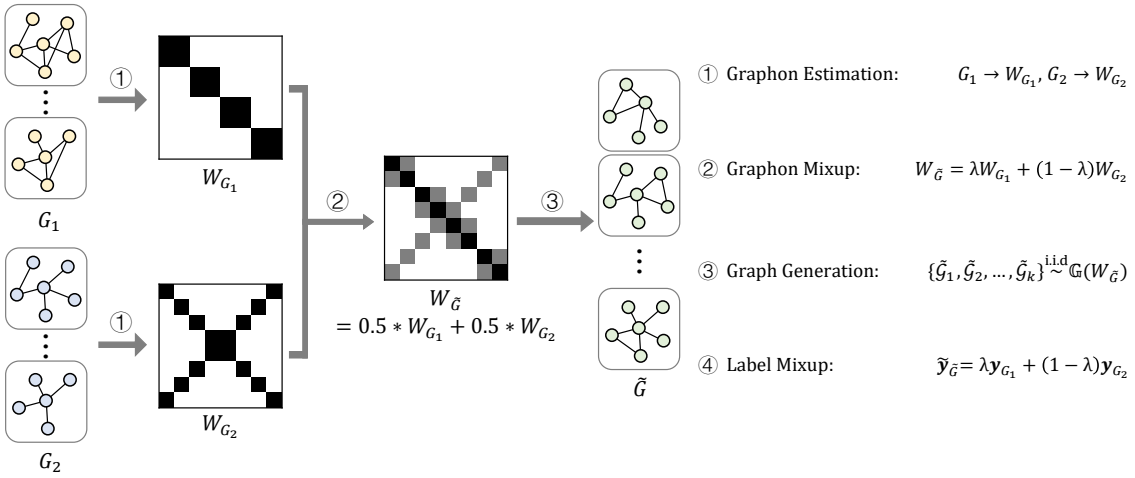


Figure 2.3: An overall workflow of G-Mixup. It contains three key steps: (1) Estimate a graphon for each class. (2) Interpolate graphons of different classes. (3) Generating synthetic graphs from mixed graphons. For a binary classification task with two classes of non-attributed graphs G_1 and G_2 with corresponding labels 0 and 1, G-Mixup generates class-wise graphons W_{G_1} and W_{G_2} for two classes, respectively, mapping graphs of different sizes and different classes into the same augmentation space $\mathbb{G}$. It then interpolates W_{G_1} and W_{G_2} to obtain a mixed graphon $W_{\tilde{G}}$, and performs sampling on $W_{\tilde{G}}$ to generate synthetic graphs that possess structural patterns of two classes. Ultimately, given $\lambda = 0.5$, the corresponding label $\tilde{y}$ of synthetic graphs will be 0.5.

that inserts global topological information into the original graph by multi-step random walks or diffusion, reconnecting nodes that are not directly connected but are considered strongly related within multi-hop neighbors. A general formulation of graph diffusion can be expressed by:

$$\tilde{\boldsymbol{A}} = \sum_{k=0}^{\infty} \gamma_k \boldsymbol{T}^k \tag{2.17}$$

where $\boldsymbol{T} \in \mathbb{R}^{N \times N}$ is the transition matrix generalized from $\boldsymbol{A}$, γ_k controls the global-local information ratio, and a common constraint is $\sum_{k=0}^{\infty} \gamma_k = 1$ to guarantee the convergence of $\tilde{\boldsymbol{A}}$. Intuitively, $[\boldsymbol{T}^k]_{ij}$ represents the probability of reaching node v_j from node v_i in k steps. Note that the graph sampling and graph generation

techniques discussed above are also widely used as components of structure-oriented augmentation. For feature-oriented and label-oriented techniques, they are often closer to generic augmentations. In contrast, modifying the graph structure $\boldsymbol{A}$ directly alters the path the message passes, making structure-oriented augmentation more graph-specific and thus worth a more detailed discussion.

2.2.3 Model Architecture

Model architecture plays a pivotal role in representation learning as it directly governs how representations are computed from input graph signals. Due to the topological diversity of graphs in non-Euclidean spaces, conventional Deep Neural Networks (DNNs), which lack the architecture or mechanisms to model relational dependencies, are often less effective in learning graph representations. Early efforts addressed these challenges primarily through network embedding (Cai et al., 2018) and graph kernel methods (Kriege et al., 2020), within which the former learns a mapping function that projects nodes as low-dimensional vectors while preserving graph topological information, and the latter utilizes a kernel function to represent graph representations by similarity measures. Once such vector representations are crafted, conventional DNNs can process graph data in a standard Euclidean feature space. These traditional methods, however, often run into computational bottlenecks and, more importantly, rely on a two-stage decoupled pipeline rather than end-to-end learning (Z. Wu et al., 2020).

Consequently, there has been a wealth of work devoted to learning graph representations with deep neural networks in an end-to-end fashion, giving rise to the modern family of graph neural networks (GNNs) (Defferrard et al., 2016; Kipf & Welling, 2017; Gilmer et al., 2017; K. Xu et al., 2018; F. Wu et al., 2019). In the graph representation learning pipeline with GNNs, the architecture level concerns the information propagation mechanism that aggregates and transforms node information over the graph, the graph pooling operations that summarize node-level

representations into graph-level ones, and the structure design that adapts to different task requirements and graph settings. In what follows, this thesis reviews these three aspects.

2.2.3.1 Graph Neural Networks

Spatial GNNs Modern GNNs excel at modeling irregular, non-Euclidean graph data largely because of their neural message-passing mechanism (Gilmer et al., 2017), where nodes exchange and aggregate vector-shaped messages along edges and iteratively update their representations through network propagations.

Definition 2.2.4 (Graph Message Passing)**.** Let a graph be $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the node feature matrix $\boldsymbol{X} = \{\boldsymbol{x}_v | v \in \mathcal{V}\} \in \mathbb{R}^{N \times F}$ and the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$, and the output node representation at layer ℓ (initialized by $\boldsymbol{h}_v^{(0)} = \boldsymbol{x}_v$) of a message-passing GNN be $\boldsymbol{h}_v^{(\ell)} \in \mathbb{R}^D$, where D is the hidden dimension of layer ℓ. For each layer $\ell = 0, 1, \ldots, L-1$, message passing is defined by:

$$\begin{aligned} \boldsymbol{m}_{\mathcal{N}(v)}^{(\ell)} &= \mathrm{AGG}^{(\ell)}\left(\{\boldsymbol{h}_u^{(\ell)} : u \in \mathcal{N}(v)\}\right) \\ \boldsymbol{h}_v^{(\ell+1)} &= \mathrm{UP}^{(\ell)}\left(\boldsymbol{h}_v^{(\ell)}, \boldsymbol{m}_{\mathcal{N}(v)}^{(\ell)}\right) \end{aligned} \tag{2.18}$$

where $\mathcal{N}(v)$ denotes the neighborhood of v, the $\mathrm{AGG}^{(\ell)}(\cdot)$ is a permutation-invariant aggregation function (e.g., sum, mean, and max or attention-based), and $\mathrm{UP}(\cdot)$ is the update function.

Practically, instantiating the above message-passing formulation requires specifying the layer-wise aggregation function $\mathrm{AGG}^{(\ell)}(\cdot)$ and update function $\mathrm{UP}^{(\ell)}(\cdot)$. An example following the early GNN formulations (Merkwirth & Lengauer, 2005; Scarselli et al., 2008) considers the summation aggregation, where each node $v \in \mathcal{V}$ updates its representation by combining its own state with the aggregated neigh-

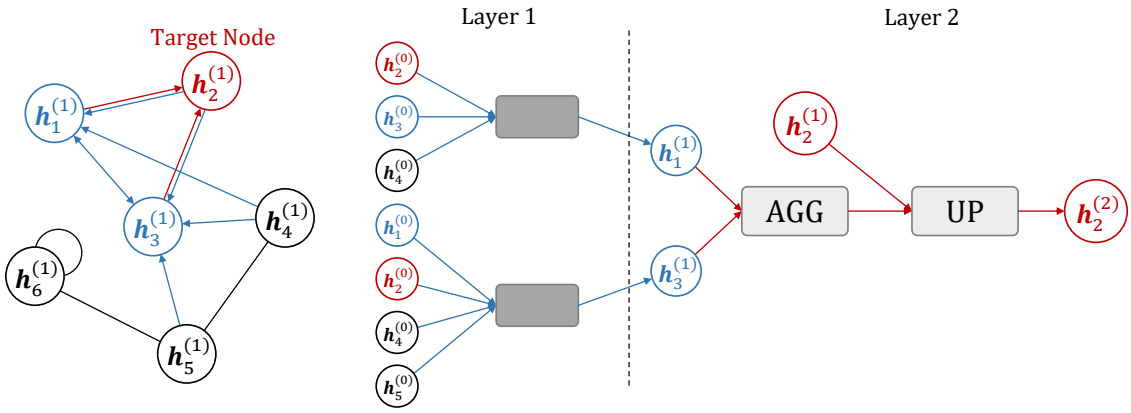


Figure 2.4: An illustration of the two-layer message passing. After two rounds of propagation, the representation of a vertex $\boldsymbol{h}_2^{(2)}$ is formed by aggregating information from its 1-hop neighbors $\{\boldsymbol{h}_1^{(2)}, \boldsymbol{h}_2^{(2)}\}$, which have already incorporated information from their own neighborhoods, bringing 2-hop context to the vertex. The same process extends naturally to deeper layers for a larger receptive field (multi-hop information perception). In this way, successive message passing in multi-layer GNNs first aggregates local information and then progressively refines node representations by integrating information from increasingly distant neighborhoods.

borhood information:

$$\boldsymbol{h}_v^{(\ell+1)} = \sigma\left(\boldsymbol{W}_{\mathcal{V}}^{(\ell)}\boldsymbol{h}_v^{(\ell)} + \boldsymbol{W}_{\mathcal{N}}^{(\ell)}\sum_{u\in\mathcal{N}(v)}\boldsymbol{h}_u^{(\ell)} + \boldsymbol{b}^{(\ell)}\right) \tag{2.19}$$

where $\boldsymbol{W}_{\mathcal{V}}^{(\ell)}, \boldsymbol{W}_{\mathcal{N}}^{(\ell)} \in \mathbb{R}^{D^{(\ell)}\times D^{(\ell+1)}}$ are learnable weight matrices, $\mathbf{b}^{(\ell)}$ is an optional bias term, and $\sigma(\cdot)$ denotes a nonlinear activation function (e.g., tanh, ReLU, or sigmoid). Under this instantiation, the two core components in Definition 2.2.4 become:

$$\begin{aligned}
\mathrm{AGG}^{(\ell)}(\cdot) : \boldsymbol{m}_{\mathcal{N}(v)}^{(\ell)} &= \sum_{u\in\mathcal{N}(v)}\boldsymbol{h}_u^{(\ell)} \\
\mathrm{UP}^{(\ell)}(\cdot) \ : \ \boldsymbol{h}_v^{(\ell+1)} &= \sigma\left(\boldsymbol{W}_{\mathcal{V}}^{(\ell)}\boldsymbol{h}_v^{(\ell)} + \boldsymbol{W}_{\mathcal{N}}^{(\ell)}\boldsymbol{m}_{\mathcal{N}(v)}^{(\ell)} + \boldsymbol{b}^{(\ell)}\right)
\end{aligned} \tag{2.20}$$

From a matrix-view, letting $\boldsymbol{H}^{(\ell)} \in \mathbb{R}^{N\times D^{(\ell)}}$ collect all node representations at

layer ℓ, the same update can be written for a graph as:

$$\boldsymbol{H}^{(\ell+1)} = \sigma\left(\boldsymbol{H}^{(\ell)}\boldsymbol{W}_{\mathcal{V}}^{(\ell)} + \boldsymbol{A}\boldsymbol{H}^{(\ell)}\boldsymbol{W}_{\mathcal{N}}^{(\ell)} + \mathbf{1}(\boldsymbol{b}^{(\ell)})^{\top}\right) \tag{2.21}$$

where the adjacency matrix $\boldsymbol{A}$ is consistent with the definition of $\mathcal{N}(v)$, and $\mathbf{1} \in \mathbb{R}^N$ is an all-ones vector. Ultimately, one can consider adding self-loops as sharing parameters between $\boldsymbol{W}_{\mathcal{V}}$ and $\boldsymbol{W}_{\mathcal{N}}$, thus rewriting Equation (2.21) as:

$$\boldsymbol{H}^{(\ell+1)} = \sigma\left((\boldsymbol{A}+\boldsymbol{I})\boldsymbol{H}^{(\ell)}\boldsymbol{W}^{(\ell)} + \mathbf{1}(\boldsymbol{b}^{(\ell)})^{\top}\right) \tag{2.22}$$

This simplification implicitly incorporates the update into the aggregation operation, and can mitigate the model overfitting, but at the cost of model expressivity (W. L. Hamilton, 2020).

Spectral GNNs The message-passing mechanism and implementation introduced above correspond to spatial GNNs. Historically, however, the early graph neural model was first developed from a spectral perspective, where Bruna et al. (2013) introduced spectral graph convolutions by defining learnable filters in the graph Fourier domain to compute node representations.

Definition 2.2.5 (Spectral Graph Convolution)**.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with with the node feature matrix $\boldsymbol{X} \in \mathbb{R}^{N\times F}$ and symmetric normalized Laplacian $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$. Given the eigen-decomposition $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\top}$, where the graph Fourier basis $\boldsymbol{U} \in \mathbb{R}^{N\times N}$ contains the orthonormal eigenvectors and spectra $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$ contains eigenvalues, the spectral graph convolution $\star$ is defined by applying a learnable spectral filter $g_{\boldsymbol{\theta}}$ in the graph Fourier domain such that:

$$g_{\boldsymbol{\theta}} \star \boldsymbol{X} := \boldsymbol{U}\, g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})\, \boldsymbol{U}^{\top}\boldsymbol{X} \tag{2.23}$$

where $g_{\boldsymbol{\theta}}(\cdot)$ acts diagonally on $\boldsymbol{\Lambda}$.

The operation in Equation (2.23) introduces the computational complexity of

$\mathcal{O}(N^3)$ due to explicit eigen-decomposition. To avoid such a problem, practical spectral GNNs typically parameterize $g_{\boldsymbol{\theta}}$ using a truncated Chebyshev polynomial expansion of order K (Defferrard et al., 2016), which approximates the graph convolution as the following, reducing the computational costs to $\mathcal{O}(K|\mathcal{E}|)$:

$$
\begin{aligned}
g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda}) &\approx \sum_{k=0}^{K-1} \theta_k \, T_k(\tilde{\boldsymbol{\Lambda}}) \\
g_{\boldsymbol{\theta}} \star \boldsymbol{X} &\approx \sum_{k=0}^{K-1} \theta_k \, T_k(\tilde{\boldsymbol{L}}) \boldsymbol{X}
\end{aligned}
\tag{2.24}
$$

where $\tilde{\boldsymbol{\Lambda}} = 2\boldsymbol{\Lambda}/\lambda_{\max} - \boldsymbol{I}$, $T_k(\tilde{\boldsymbol{L}}) \in \mathbb{R}^{N \times N}$ is the Chebyshev polynomial of order k on the scaled Laplacian $\tilde{\boldsymbol{L}} = 2\boldsymbol{L}/\lambda_{\max} - \boldsymbol{I}$, and $\theta \in \mathbb{R}^K$ the Chebyshev coefficient vector.

2.2.3.2 Graph Pooling

Typically, L-layer GNNs produce node representations $\boldsymbol{H}^{(L)} = \{\boldsymbol{h}_v^{(L)} | v \in \mathcal{V}\} \in \mathbb{R}^{N \times D^{(L)}}$, which is readily applicable to node-level tasks. For graph-level tasks, however, a compact and uniform-dimensional graph representation is required to summarize the whole graph $\mathcal{G}$. A graph representation $\boldsymbol{h}_{\mathcal{G}} \in \mathbb{R}^{1 \times D^{(L)}}$ can usually be derived by graph poolings (or readout operation) that condense a set of node representations to a single vector. Depending on whether the pooling explicitly constructs multi-scale coarsened graphs, graph pooling methods are commonly categorized into global graph pooling and hierarchical graph pooling strategies (J. Lee et al., 2019; Z.-P. Li et al., 2021; C. Liu et al., 2022).

Global Graph Pooling Global pooling, also called flat pooling, summarizes node representations into $\boldsymbol{h}_{\mathcal{G}}$ in a single step. Although it does not explicitly consider a hierarchical structure, it can still capture structural patterns implicitly through the node representations produced by message passing. The generalized form of the

global pooling can be written as:

$$\boldsymbol{h}_{\mathcal{G}} = \text{POOL}\left(\{\boldsymbol{h}_v^{(L)} | v \in \mathcal{V}\}\right) \tag{2.25}$$

where POOL$(\cdot)$ is permutation invariant with respect to node ordering. Common instances include:

$$\begin{aligned} \textsc{Sum}: \quad & \boldsymbol{h}_{\mathcal{G}} = \sum_{v \in \mathcal{V}} \boldsymbol{h}_v^{(L)} \\ \textsc{Mean}: \quad & \boldsymbol{h}_{\mathcal{G}} = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \boldsymbol{h}_v^{(L)} \\ \textsc{Max}: \quad & \boldsymbol{h}_{\mathcal{G}} = \max_{v \in \mathcal{V}} \boldsymbol{h}_{vi}^{(L)}, \quad i = 1, .., D^{(L)} \end{aligned} \tag{2.26}$$

Due to their inherent simplicity, the global pooling is computationally efficient and is widely used as the default readout for graph classification, especially when graphs are of moderate size.

Hierarchical Graph Pooling Straightforwardly compressing all node representations into a single vector may discard useful information (e.g., motifs and communities) (J. Lee et al., 2019). Hierarchical pooling addresses this issue by learning a differentiable coarsening procedure that progressively coarsens the graph into a smaller one, yielding hierarchical representations for a graph (Z. Ying et al., 2018). Specifically, at the coarsening stage t, hierarchical pooling computes an assignment matrix $\boldsymbol{S}^{(t)} \in \mathbb{R}^{N_t \times N_{t+1}}$ from the current graph:

$$\boldsymbol{S}^{(t)} = f_{\text{POOL}}^{(t)}\left(\boldsymbol{A}^{(t)}, \boldsymbol{H}^{(t)}\right) \tag{2.27}$$

where N_t is the node (cluster) number of the coarsened graph at stage t. Using $\boldsymbol{S}^{(t)}$, one can obtain a coarsened adjacency matrix and node feature matrix by:

$$\boldsymbol{A}^{(t+1)} = (\boldsymbol{S}^{(t)})^\top \boldsymbol{A}^{(t)} \boldsymbol{S}^{(t)}, \qquad \boldsymbol{H}^{(t+1)} = (\boldsymbol{S}^{(t)})^\top \boldsymbol{H}^{(t)} \tag{2.28}$$

where $\mathbf{A}^{(t)} \in \mathbb{R}^{N_t \times N_t}$ and $\mathbf{H}^{(t)} \in \mathbb{R}^{N_t \times D}$. This coarsening procedure can be interleaved with GNN layers on each coarsened graph, and the final graph representation is usually obtained by applying a global pooling on the coarsened graph of the last layer. In general, the hierarchical pooling can be read as:

$$\boldsymbol{A}^{t+1}, \boldsymbol{H}^{t+1} = \mathrm{POOL}(\boldsymbol{A}^{(t)}, \boldsymbol{H}^{(t)}) \tag{2.29}$$

2.2.3.3 The Dendritic Neuron Model

Structure design at the architecture level further determines how a model is parameterized and organized to suit task requirements and graph settings. For instance, H_2GCN (J. Zhu et al., 2020) employs a skip-connection design that concatenates layer-wise representations while incorporating multi-hop neighborhoods, which helps retain the ego signal in learning representations in heterophilous graphs. The structural advance developed in Chapter 5 draws inspiration from biological neurons, whose distal dendrites enable a routing mechanism for them to perceive contextual stimuli and make context-aware decisions, thereby adapting to diverse environments (Yang et al., 2016). In this regard, we briefly introduce two relevant dendritic neuron models, the Dendritic Neural Network (X. Wu et al., 2018) and the Active Dendrite Neuron (Iyer et al., 2022), even though they were not originally designed for graph representation learning but for simulating these advantageous properties observed in real neurons. Note that some of their detailed properties related to the proposed method will be specifically discussed in Chapter 5. Here, we clarify the terminology for the dendritic neuron model. We refer to the feedforward part (e.g., linear transformation) in conventional deep neural networks as the biological neuron's soma (cell body), and the dendritic structure as the biological neuron's dendrites.

Dendritic Neural Networks Dendritic Neural Networks (DENNs) incorporate the dendritic structure into the feedforward part to enrich the expressivity of the

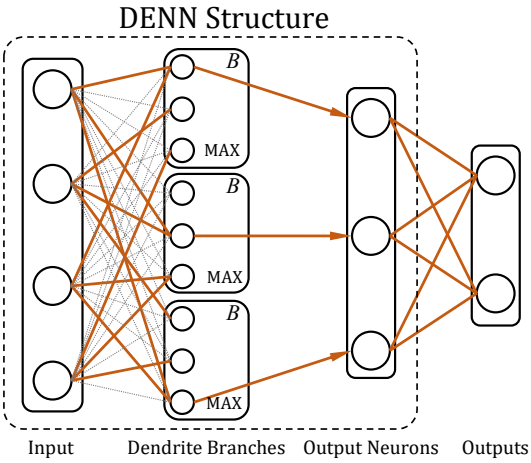


Figure 2.5: An overview of the DENN structure, which is incorporated into the feedforward part and possesses intragroup sparsity. B refers to the branch number corresponding to each output neuron, and MAX selects the strongest branch response for each output neuron.

model. Instead of directly connecting output neurons with input units and performing a dense linear transformation, DENNs assign multiple dendritic branches to each output neuron, while each input unit selectively connects to one branch for each output neuron. Such mutually exclusive synaptic connectivity is commonly implemented by a binary mask and introduces the intragroup sparsity of dendrites. Combined with selecting the strongest branch response at the neuron output, the overall operation simulates the dendritic nonlinearity. In this context, the structure design of DENNs is closer to dendrites themselves, rather than a neuron that possesses dendrites.

Formally, given the input feature vector $\boldsymbol{x} \in \mathbb{R}^D$, a DENN layer constructs B dendritic branches for each output neuron, all parameterized by a weight matrix $\boldsymbol{W} \in \mathbb{R}^{D \times D_{out} \times B}$ and a binary mask $\boldsymbol{M}$ of the same size. Then, the branch responses $\boldsymbol{D} \in \mathbb{R}^{B \times Dout}$ across output neurons can be computed as:

$$\boldsymbol{D} = (\boldsymbol{M} \odot \boldsymbol{W})^\top \boldsymbol{x} \tag{2.30}$$

where $\odot$ denotes the Hadamard product. The final activation of output neurons is obtained by a branch-selection operation (max-pooling) such that:

$$\boldsymbol{y} = \max_{b}(\boldsymbol{D}) + \boldsymbol{b} \tag{2.31}$$

where $\max_b(\cdot)$ is taken element-wisely to select strongest branch singal b over the B branches of each output neuron. Compared with a standard fully connected layer, a DENN layer improves representational capacity by allowing different branches to capture different input patterns, while the masking scheme promotes intra-group sparsity and can reduce redundant connections.

Active Dendrite Neuron Active Dendrite Neuron (ADN) models dendrites as a context-aware activation that modulates the neuron's somatic signals, whose structure design is closer to the biological view that distal dendrites can selectively amplify or suppress somatic signals, thereby enabling context-specific adaptation and reducing interference across somatic signals. By design, ADN takes a primary input $\boldsymbol{x} \in \mathbb{R}^D$ by the feedforward part $f(\cdot)$ and an additional contextual information $\boldsymbol{c} \in \mathbb{R}^C$ by dendrite part $d(\cdot)$, where $f(\cdot)$ simulates the neuron soma and performs linear transformation on $\boldsymbol{x}$ and $d(\dot{c})$ simulates dendritic activation and generate branch responses from $\boldsymbol{c}$. Specifically, these two parts are calculated as:

$$\begin{aligned} f(\boldsymbol{x}) &: \boldsymbol{y} = \boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{b} \\ d(\boldsymbol{c}) &: \boldsymbol{d} = \max_{b \in B} \boldsymbol{U}^\top \boldsymbol{c} \end{aligned} \tag{2.32}$$

where $\boldsymbol{W} \in \mathbb{R}^{D \times D_{out}}$ is the weight matrix of the feedforward part, and $\boldsymbol{U} \in \mathbb{R}^{C \times C_{out} \times B}$ is the weight matrix of the dendrite part.

Similarly, ADN typically activates the strongest branch by $\max_b(\cdot)$ and converts its response into a signal modulating factor through a sigmoid function $\sigma(\cdot)$. Finally, the neuron outputs of one ADN layer are defined by modulating the feedforward

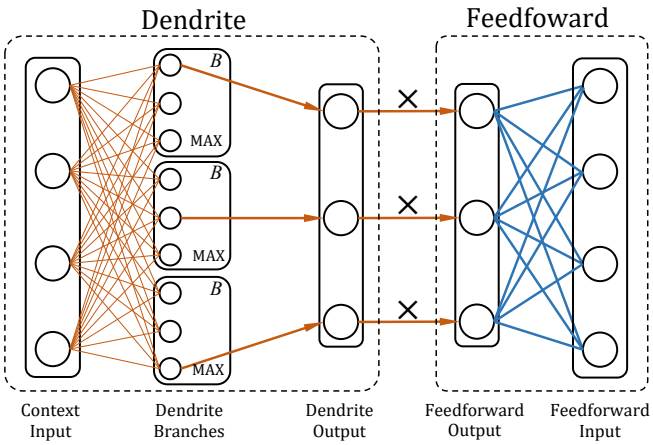


Figure 2.6: An overview of the structure of ADN, which comprises the feedforward part and the dendrite part. The dendrite part takes contextual information as input and produces context-aware activation on feedforward outputs. B refers to the branch number corresponding to each output neuron, and MAX selects the strongest branch response for each output neuron.

outputs with corresponding modulating factors using the Hadamard product:

$$\hat{\boldsymbol{y}} = \left(\boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{b}\right) \odot \sigma\left(\max_b \boldsymbol{U}^\top \boldsymbol{c}\right) \tag{2.33}$$

This formulation highlights the key role of active dendrites, that is, activating context-aware patterns within deep neural network training. However, it does not model the sparse synaptic connectivity of dendrites.

2.2.4 Learning Objective

A well-designed model, even with high-quality graph data, does not automatically acquire meaningful graph representations. It also demands a learning objective that provides an explicit training signal and guides it to learn effective representations readily for downstream tasks. The objective level concerns how supervision signals and inductive biases are encoded in optimization targets (e.g., loss function and regularization term) to drive the predicted outputs toward desired targets as optimization proceeds. Taking supervised learning as an example, the loss quantifies

the discrepancy between model predictions and supervision signals during the forward pass, and its gradients are then computed through backpropagation to update model parameters via an optimizer. While it may appear routine, a carefully designed learning objective is essential for further improving the model's capability and downstream performance. In the graph representation learning, the choice of the objective depends strongly on the training paradigm. The rest of this section first reviews common objective functions of different training paradigms and then introduces two objective functions that are also relevant to the thesis.

2.2.4.1 Supervised and Semi-Supervised Objectives

For node-level and graph-level classification, as well as link prediction under supervised or semi-supervised settings, cross-entropy is the most widely used objective function. It can be viewed as the negative log-likelihood of a categorical distribution parameterized by the classifier's predicted probabilities:

$$\mathcal{L}_{\text{CE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_{i=1}^{C} y_i \log \hat{y}_i \tag{2.34}$$

where $\hat{\boldsymbol{y}}$ is the predicted class probabilitiy vector of training samples, $\boldsymbol{y}$ is the one-hot encoded gournd truth labels, and C is the number of classes. In practice, it is often implemented with task-specific adaptations (e.g., class reweighting or additional constraints) to improve generalization.

2.2.4.2 Unsupervised and Self-Supervised Objectives

When labels are unavailable or scarce, graph representations are often learned from the graph itself using unsupervised or self-supervised objective functions. Two representatives are reconstruction loss and contrastive loss. Meanwhile, Laplacian Eigenmaps and one-class classification objectives are also widely used.

Reconstruction Loss Reconstruction loss learns a graph encoder that produces graph representations which are predictive of observed graph structure or features, and also a decoder that maps graph representations back to the original data space. A generic formulation of the reconstruction loss is:

$$\boldsymbol{H} = f_\theta(\mathcal{G}), \qquad \hat{\boldsymbol{A}} = g_\phi(\boldsymbol{H}), \qquad \mathcal{L}_{\text{Rec}} = \ell(\hat{\boldsymbol{A}}, \boldsymbol{A}) \tag{2.35}$$

where f_θ is an encoder, g_ϕ is a decoder, and $\ell(\cdot, \cdot)$ can be a binary cross-entropy loss for graph structure reconstruction or a mean squared-error loss for feature reconstruction.

Contrastive Loss In contrastive learning, one usually constructs positive and negative pairs, and learn representations by pulling positives together while pushing negatives apart via the contrastive loss. Given two stochastic views $\boldsymbol{x}_i$ and $\boldsymbol{x}'_i$ of the same instance i, and a graph encoder yielding representations $\mathbf{z}_i$ and $\mathbf{z}'_i$, the widely used contrastive loss is the InfoNCE loss, which is defined as the following:

$$\mathcal{L}_{\text{NCE}} = -\sum_{i=1}^{B} \log \frac{\exp\left(\text{sim}(\mathbf{z}_i, \mathbf{z}'_i)/\tau\right)}{\sum_{j=1}^{B} \exp\left(\text{sim}(\mathbf{z}_i, \mathbf{z}'_j)/\tau\right)} \tag{2.36}$$

where $\text{sim}(\cdot, \cdot)$ is a similarity measure (e.g., cosine similarity), τ is a temperature hyperparameter, and B is the batch size.

Laplacian Eigenmaps The core idea of the Laplacian Eigenmaps is intuitive: if two points are adjacent and similar in the original space, then they should also be close to each other in the low-dimensional representation. Laplacian Eigenmaps (Belkin & Niyogi, 2001, 2003) achieve this by constructing a proximity graph with a weighted adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ for N data points, using $\epsilon-$ neighborhoods or $k-$nearest neighbors. Given the degree matrix $\boldsymbol{D}$ and the graph symmetric normalized Laplacian matrix $\boldsymbol{L}$, Laplacian Eigenmaps solves the following equations

to find representations:

$$\frac{1}{2}\sum_{i,j}(y_i - y_j)^2 \boldsymbol{A}_{ij} = \boldsymbol{y}^\top \boldsymbol{L}\boldsymbol{y}$$

$$\boldsymbol{y}^* = \underset{\text{s.t. } \boldsymbol{y}^\top \boldsymbol{D}\boldsymbol{y}=1}{\arg\min} \boldsymbol{y}^\top \boldsymbol{L}\boldsymbol{y} \tag{2.37}$$

Here, $\boldsymbol{y} = [y_1, ..., y_N]^\top$ collects one dimension of representations of all N data points (one-dimensional representations), and y_i is the scalar representation value of the i-th data point. The constraint aims to prevent trivial solutions and leads to the generalized eigenvalue problem $\boldsymbol{L}\boldsymbol{y} = \lambda \boldsymbol{D}\boldsymbol{y}$, where $\boldsymbol{y}^*$ is given by the eigenvector of $\boldsymbol{D}^{-1}\boldsymbol{L}$, corresponding to the smallest non-zero eigenvalue. This solution can also be interpreted by the Laplacian smoothing, where eigenvectors associated with smaller non-zero eigenvalues are smoother and therefore exhibit less variation across adjacent nodes in the representation space. Generally, this objective function penalizes mapping nearby data points far apart, thereby encouraging representations to be closer in terms of the neighborhood structure.

One-Class Classification Objective For binary classification with severe class imbalance (i.e., anomaly detection), one-class classification (OCC) provides a simple yet effective objective that learns a compact description of normality in the representation space. Let $f_\theta(\mathcal{G})$ denote the learned representation of a graph and $\boldsymbol{c}$ be a fixed center, the OCC objective can be defined as:

$$\mathcal{L}_{\text{OCC}} = \|f_\theta(\mathcal{G}) - \boldsymbol{c}\|_2^2 \tag{2.38}$$

which maps normal samples into a hypersphere with the center $\boldsymbol{c}$. At inference time, an anomaly score can be defined as $s(\mathcal{G}) = \|f_\theta(\mathcal{G}) - \boldsymbol{c}\|_2^2$, where larger values indicate a higher degree of abnormality.

Chapter 3

Enhancing Objective Function by Laplacian Eigenmaps

Graph-level classification typically follows a pipeline, where GNNs first compute node representations through message passing among nodes, and the readout operation then summarizes them into a graph-level representation. Apparently, the inductive bias of standard message-passing GNNs is largely local, as it only explicitly models intra-graph interactions. In this context, the most direct signals that encourage discriminative graph representations across different graphs come from the objective function, which shapes graph representations according to supervision of the label. Although recent progress in GNNs has steadily improved graph classification performance, most of these advances, however, are driven by architectural innovations. Meanwhile, the learning objective, typically cross-entropy, is often left largely unchanged. Such a generic objective may not provide sufficiently explicit guidance for inter-graph interactions that help construct the relative geometry among graph representations.

This chapter responds to the **Research Question 1: How can the standard objective function in GNN-based graph classification be enhanced by better exploiting inter-graph information among graphs?** and presents a simple yet effective regularizer that augments the commonly used cross-entropy.

Leveraging the smoothness property of Laplacian Eigenmaps, the proposed regularization term encourages more inter-graph-aware representations, while remaining plug-and-play for existing GNN architectures.

3.1 Introduction

Graph representation learning has attracted great attention in the recent decade as ballooning data is being generated in the form of graphs. Compared with regular lattice information such as images, audio, or text, graphs simultaneously carry intricate object interactions over high-dimensional features, challenging researchers to conduct feature engineering to generate adequate graph representations. Since the rise and wide application of Graph Neural Networks (GNNs), the situation has changed, and research effectiveness on downstream prediction tasks has been significantly improved. The essence of the machine-learning applications of various GNNs (e.g., node classification (Kipf & Welling, 2017), link prediction (M. Zhang & Chen, 2018), and graph classification (Z. Wang & Ji, 2020)) depends highly on the output of different but sequential blocks of GNNs. A basic GNN learns node representations and is normally ready for node classification and link prediction tasks, as these tasks involve only node-level information. Graph-level classification problems, where more compact representations on the graph level are necessary, further entail a readout function or graph pooling layers that enable GNNs to integrate the node representation vectors and graph topological information into a graph representation vector. At last, with a classification head and a carefully crafted objective function, GNNs perform these tasks in an end-to-end manner (Z. Wu et al., 2020).

Although having yielded outstanding experimental results, many recent distinguished works for graph classification concentrate on model architecture, where different building blocks are explicitly formulated. For example, some research on different message passing schemes dig into the graph convolutional operation (node information transmission and neighborhood aggregation algorithms) from the spec-

trum or space domain (Kipf & Welling, 2017; W. Hamilton et al., 2017; Levie et al., 2018; K. Xu et al., 2018), and some are devoted to the readout layer and propose various graph pooling strategies that can filter less informative features or capture different-order statistics of graphs (Z. Ying et al., 2018; J. Lee et al., 2019; Z. Wang & Ji, 2020). However, this limits their view to relationships within graphs, as primary signals that encourage their learned graph representations to be distinguished from each other come only from the supervision of labels in the objective function. Considering that the objective function is usually set as routine, those works seem to be indifferent to its room for improvement in the graph representation learning and show less effort in explicitly modeling the intrinsic structure and relationships among graphs. Given this situation, we concentrate on the objective level of those works to investigate whether enhancing the objective function can facilitate the graph classification performance of an arbitrary GNN framework without altering its internal building blocks.

Considering a wide range of applications demonstrating the Laplacian Eigenmaps' efficiency in mining the local geometry of the high-dimensional data (Brun et al., 2003; Z.-L. Sun et al., 2005; Mi et al., 2013; Z. Chen et al., 2019), this work proposes approaches based on Laplacian Eigenmaps (Belkin & Niyogi, 2003), namely **L**aplacian **E**igenmaps **E**nhanced **Loss** (**LEELoss**), to enhance the representation learning capability of models through regularizing the objective function. To be specific, we implement our approaches by applying the Laplacian Eigenmaps as a regularization term on the cross-entropy loss to jointly optimize graph representations. In this work, we specifically discuss the property of Laplacian Eigenmaps for supervised representation learning, and implement the LEELoss from the node-level perspective, and then extend it to the graph-level perspective. For the node-level implementation, we borrow the idea from the work by H. Zhu et al. (2021) to set Laplacian Eigenmaps into a contrastive setting. For the graph-level implementation, we construct graph similarity matrices from graph priors and adapt LEELoss to the graph level by treating each graph as a node in a meta graph, so that the regulariza-

tion aligns with the inductive bias of message passing, offering a higher-dimensional view of the relationships among graphs. Attributed to the extra discriminative capability powered by Laplacian Eigenmaps, GNNs are expected to achieve higher graph classification accuracy and faster convergence without introducing substantial architectural complexity.

The contributions of this chapter are summarized as follows:

- We propose the LEELoss by implementing Laplacian Eigenmaps for supervised graph classification tasks, which is simple and intuitive to enhance graph representation learning.
- We construct the graph similarity matrix to model inter-graph interactions and extend Laplacian Eigenmaps from their original node level to the graph level, offering further signals beyond graph labels.
- Extensive experiments on seven benchmark datasets have shown that the LEELoss facilitates the representation learning capability of popular vanilla GNN models, revealing further potentials of the objective function when it is carefully regularized or redesigned, other than previous blocks of a GNN framework.

3.2 Related Works

In this section, we review recent advances in GNNs and Laplacian Eigenmaps that provide useful insights for designing the proposed LEELoss. The mechanisms and mathematical formulations of GNNs and Laplacian Eigenmaps are presented in Section 2.2.3.1 and Section 2.2.4.2, respectively.

3.2.1 Graph Neural Network Frameworks

Feature engineering for graphs is a complicated and costly process. However, the rapid expansion of end-to-end deep learning paradigms enables the model itself to

learn representations without many handcrafted features, which can significantly boost the efficiency of downstream tasks. Different from some classical graph embedding methods, including dimension reduction, random walk, and matrix factorization, GNNs naturally enable end-to-end representation learning on graphs with arbitrary structures, preserve higher-order neighborhood information, and support a broad range of learning objectives (Goyal & Ferrara, 2018; F. Chen et al., 2020).

In recent years, many popular graph modeling techniques based on GNNs have come to light and achieved outstanding performance. For instance, the Graph Convolution Network (GCN) (Kipf & Welling, 2017), which is a spectral-based approach, capitalizes on the first-order Chebyshev polynomial and symmetric normalized Laplacian to extract feature information of proximity and has been proven to be a simple yet efficient framework. Since GCN, many advanced GNNs have been proposed from the aspect of the spatial domain; GraphSAGE (W. Hamilton et al., 2017) generates variations of nodes by learning functions of sampling and aggregating node features to simulate data for the unseen nodes; Graph Attention Network (GAT) (Veličković et al., 2017) utilizes the attention mechanism, which measures the weight of edges according to features of the corresponding node pair, to gather more information from nodes of high significance; K. Xu et al. (2018) proposed the Graph Isomorphism Network (GIN) which successfully recognizes some isomorphic graph structures that previous GNNs fail to distinguish and share the similar capability with Weisfeiler-Lehman graph isomorphism test.

3.2.2 Laplacian Eigenmaps

Laplacian Eigenmaps (Belkin & Niyogi, 2003) is a geometrically motivated nonlinear manifold learning method that can construct a low-dimensional representation for high-dimensional data and retain the inferred graph structure from data. Based on the assumption that the data resides on a low-dimensional manifold embedded in a high-dimensional space, Laplacian Eigenmaps preserve the feature and struc-

ture information of data by keeping the correspondence of local Euclidean distance between data representations if those data points are tightly related in the global space. Compared with many other manifold learning techniques, such as isometric mapping (Tenenbaum et al., 2000), locally linear embedding (Roweis & Saul, 2000), and diffusion maps (Coifman et al., 2005), Laplacian Eigenmaps are more celebrated among researchers for their computational simplicity and availability on arbitrary manifolds and have been broadly applied for dimension reduction and feature extraction (B. Li et al., 2019).

In the computer vision domain, Laplacian Eigenmaps have been used for human brain image processing and showed their property of clustering via visualizing fiber trace of the human brain with a color close to its neighbors' colors (Brun et al., 2003). W. Luo (2011) implemented Laplacian Eigenmaps into face recognition tasks as a feature extraction technique to quantify and preserve the locality of information, proving their advantages over some other methods, such as principal component analysis and locality-preserving projections. In the graph domain, Z. Chen et al. (2019) proposed a Teacher-Student mechanism to regularize the representation learning process, utilizing the main idea of Laplacian Eigenmaps to force the node to be distributionally similar to its neighboring nodes. Laplacian Eigenmaps have also been reformulated as an objective function for unsupervised contrastive learning for node classification tasks and have demonstrated their superior representation learning capability when concatenated with multiple GNN frameworks (H. Zhu et al., 2021).

3.3 Methodology

Considering the architecture of GNN models, the primary objective of this work is to enhance the graph-level classification capability of any existing vanilla GNN model without modifying its structure or intrinsic properties across different blocks. In this section, we first discuss how Laplacian Eigenmaps can be leveraged as an

objective function to enhance the graph representation learning capability of GNNs, and provide an overview of the implementation of our proposed LEELoss.

3.3.1 Laplacian Eigenmaps for Representation Learning on Graphs

Apparently, when considering one-dimensional representations from Laplacian Eigenmaps to D-dimensional representation, Equation (2.2.4.2) can be rewrote as:

$$\sum_{i,j} \| \boldsymbol{y}_i - \boldsymbol{y}_j \|^2 \boldsymbol{A}_{ij} = \mathrm{Tr}(\boldsymbol{Y}^\top \boldsymbol{L}\boldsymbol{Y})$$

$$\boldsymbol{Y}^* = \underset{\text{s.t. } \boldsymbol{Y}^\top \boldsymbol{D}\boldsymbol{Y}=\boldsymbol{I}}{\arg\min} \mathrm{Tr}(\boldsymbol{Y}^\top \boldsymbol{L}\boldsymbol{Y}) \tag{3.1}$$

where $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_N]^\top \in \mathbb{R}^{N \times D}$ and $\boldsymbol{y}_i$ represents the D-dimensional representation of node i. Similarly, the optimal node representation $\boldsymbol{y}_i^*$ of the matrix $\boldsymbol{Y}$ can be obtained by solving the generalized eigenvalue problem with D smallest non-zero eigenvalues.

When utilizing Laplacian Eigenmaps as an objective function for representation learning tasks instead of their original dimensionality reduction tasks, we note that the clustering property of Laplacian Eigenmaps can support GNN encoders to learn representations based on the similarity in terms of pair-wise nodes' Euclidean distance, other than their common application of constructing low-dimensional representations. That is to say, given a graph $\mathcal{G}$ with a node feature matrix $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ and an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ as input, a multi-layer GNN encoder can map it into a node representation matrix $\boldsymbol{H} \in \mathbb{R}^{N \times D}$, which collect learned representations of all nodes, and D is equal to the neuron number of the output layer. By simply considering the representation matrix $\boldsymbol{Y}$ from Equation (3.1) as $\boldsymbol{H}$ from the GNN encoder, we have an objective function on the basis of Laplacian Eigenmaps

for unsupervised classification tasks:

$$\sum_{i,j} \| \boldsymbol{h}_i - \boldsymbol{h}_j \|^2 \boldsymbol{A}_{ij} = \mathrm{Tr}(\boldsymbol{H}^\top \boldsymbol{L}\boldsymbol{H}) \tag{3.2}$$

With the adjacency matrix $\boldsymbol{A}$ as an indicator of the node's similarity, this objective function favors similar representations for adjacent nodes and will force them to be closer through the network propagation. In supervised learning, however, considering the conventionally used cross-entropy loss function guides the representation learning primarily via supervision signals from the label, absorbing Equation (3.2) into the supervised setting would be a feasible option to provide further supervision signals based on the graph Laplacian matrix $\boldsymbol{L}$. Nevertheless, integrating graph priors (e.g., $\boldsymbol{L}$) into Equation (3.2) to enhance interactions between learned representations is not easy at the graph level. In this context, we focus on the graph classification problem and thus conduct supervised graph representation learning while leveraging the advantages of Laplacian Eigenmaps to better exploit graph data.

Following the work by J. Zeng & Xie (2021), who used contrastive learning to regularize models in classification, we can consider the following form of the optimization formulation to solve this problem:

$$\mathcal{L} = \mathcal{L}_{(c)}(G, \boldsymbol{y} \mid \text{Graph Encoder, Classification Head}) + \lambda \mathcal{L}_{(l)}(G \mid \text{Graph Encoder}) \tag{3.3}$$

where $\mathcal{L}_{(c)}$ denotes the conventional classification loss function and $\mathcal{L}_{(l)}$ denotes the Laplacian Eigenmaps loss function, λ is a hyperparameter controlling the effect of $\mathcal{L}_{(l)}$, and G and $\boldsymbol{y}$ refer to a set of input graphs and their corresponding labels, respectively. The Graph Encoder can be any GNN framework, and the Classification Head can be a fully connected layer. Note that both loss functions share the same Graph Encoder.

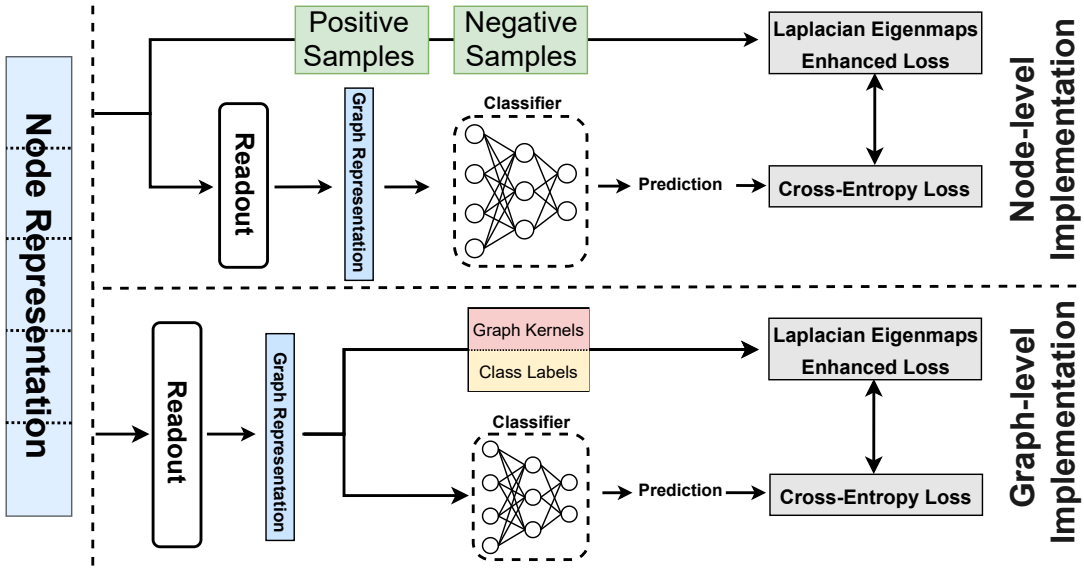


Figure 3.1: Illustration of Laplacian Eigenmaps Enhanced Loss. The top right is the process of node-level implementation, and the bottom is the process of graph-level implementation.

By design, the first part takes training graphs as input and performs prediction on graph representations. The second part serves as a data-aware auxiliary task to refine the representation learning process. As a consequence, the GNN model will produce representations with stronger expressivity, presenting more satisfactory graph classification performance. Figure 3.1 visualizes the specific process of LEELoss in node-level implementation and graph-level implementation, respectively.

3.3.2 Laplacian Eigenmaps Enhanced Loss

In this section, we introduce specific implementation details of LEELoss on the node level and graph level, respectively.

3.3.2.1 Node-level Implementation

We define the fundamental formulation of LEELoss, which is constructed on the node level, by straightforwardly appending Laplacian Eigenmaps behind the cross-

entropy loss:

$$\text{LEELoss} = \mathcal{L}_{CE}(G, \boldsymbol{y}) + \lambda \cdot \text{Tr}(\boldsymbol{H}^\top \boldsymbol{L}\boldsymbol{H}) \tag{3.4}$$

It is obvious that LEELoss directly shapes node representations $\boldsymbol{H}$ to support more effective node representations. Sequentially, effective node representations always contribute to more robust graph representations, which then lead to satisfactory classification outcomes. In addition, we note that Contrastive Laplacian Eigenmaps (COLES) (H. Zhu et al., 2021) further integrate a negative sampling mechanism into Laplacian Eigenmaps and have shown outstanding performance in node-level unsupervised learning tasks. The objective function of COLES is defined as the following equation:

$$\begin{aligned}
\boldsymbol{H}^* &= \underset{\text{s.t. } \boldsymbol{H}^\top\boldsymbol{H}=\boldsymbol{I}}{\arg\min}\ \text{Tr}(\boldsymbol{H}^\top \boldsymbol{L}^{(+)}\boldsymbol{H}) - \frac{\eta}{m}\sum_{m=1}^{m}\text{Tr}(\boldsymbol{H}^\top \boldsymbol{L}_m^{(-)}\boldsymbol{H}) \\
&= \underset{\text{s.t. } \boldsymbol{H}^\top\boldsymbol{H}=\boldsymbol{I}}{\arg\max}\ \text{Tr}(\boldsymbol{H}^\top \Delta\boldsymbol{A}\boldsymbol{H}) \qquad (3.5) \\
&\text{where}\ \ \Delta\boldsymbol{A} = \boldsymbol{A}^{(+)} - \frac{\eta}{m}\sum_{m=1}^{m}\boldsymbol{A}_m^{(-)} \qquad (3.6)
\end{aligned}$$

Here m refers to the number of negative samples. $\boldsymbol{L}^{(+)}$ is a symmetric graph Laplacian matrix generated based on the adjacency matrix $\boldsymbol{A}$ of the input graph. $\boldsymbol{L}_m^{(-)}$ is also a symmetric graph Laplacian matrix but is purposely constructed via a randomized adjacency matrix $\boldsymbol{A}_m$, which is generated based on the negative sampling from the random graph sampling theory (Erdős & Rényi, 1960). Note that $\boldsymbol{L}^{(+)} = \boldsymbol{I} - \mathrm{A}^{(+)}$, $\boldsymbol{L}_m^{(-)} = \boldsymbol{I} - \boldsymbol{A}_m^{(-)}$, and both $\boldsymbol{A}^{(+)} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ and $\boldsymbol{A}_m^{(-)} = \boldsymbol{D}^{-1/2}\boldsymbol{A}_m\boldsymbol{D}^{-1/2}$ are symmetric normalized adjacency matrices. $0 \leq \eta \leq 1$ is a hyperparameter controlling the weight of the negative part.

Introducing a contrastive mechanism, COLES becomes a block contrastive loss and has access to blocks of similar data that give rise to better generalization capability of the model. COLES is also a Wasserstein distance-based method that is resistant to situations of poor data distribution overlap. Given these advantages,

we have a direct adoption of COLES in LEELoss:

$$\begin{aligned} \text{LEELoss}_{(node)} &= \mathcal{L}_{CE}(G, \boldsymbol{y}) - \lambda \cdot \text{Tr}(\boldsymbol{H}^\top \Delta \boldsymbol{A} \boldsymbol{H}) \\ &= \mathcal{L}_{CE}(G, \boldsymbol{y}) + \lambda \cdot \Big(\eta \cdot \text{Tr}(\boldsymbol{H}^\top \frac{1}{m} \sum_{m=1}^{m} \boldsymbol{A}_m^{(-)} \boldsymbol{H}) - \text{Tr}(\boldsymbol{H}^\top \boldsymbol{A}^{(+)} \boldsymbol{H}) \Big) \end{aligned} \tag{3.7}$$

Consequently, $\text{LEELoss}_{(node)}$ favors similar representation pairs in normal Laplacian Eigenmaps but penalizes dissimilar pairs in randomized Laplacian Eigenmaps.

3.3.2.2 Graph-level Implementation

Appropriate node representations, however, are not sufficient for a classifier to perform prediction on graph labels. Within a GNN framework for graph classification, there exists a readout function layer, typically graph pooling, that helps the model learn a representation from the graph perspective in the manner of summarizing a stack of node representations into a vectorized graph representation. Most importantly, graph representations generated by different graph pooling strategies further encapsulate topological information, which is crucial for the classifier to precisely recognize and distinguish among graphs. Heuristically, directly refining graph representations would be more reasonable for graph-level tasks. Therefore, we can have an extension for the fundamental LEELoss from its original node level to the graph level.

To implement such an extension, one can view Laplacian eigenmaps from a higher-dimensional perspective. We suppose that, for a given set of N graphs $G = \{\mathcal{G}_1, ..., \mathcal{G}_N\}$, we can gather a set of node representation matrices $\mathcal{H} = \{\boldsymbol{H}_1, ..., \boldsymbol{H}_N\}$ readily to be processed by the readout layer. With a graph pooling strategy (e.g., mean pooling, sum pooling, SortPool (M. Zhang et al., 2018), DiffPool (Z. Ying et al., 2018)), we can further flatten $\mathcal{H}$ into a graph representation matrix $\boldsymbol{H}_g = [\boldsymbol{h}_{g1}, ...\boldsymbol{h}_{gN}]$, where $\boldsymbol{h}_{gi}$ is the representation of $\mathcal{G}_i$. If we consider a meta graph, which is essentially the high-dimensional manifestation of a batch of individual graphs that each node in this meta graph corresponds to a graph in G, it is natural for us to

construct a Laplacian matrix $\boldsymbol{L}_g$ for the meta graph. Under these circumstances, we consider the natural substitution of the node representation matrix $\boldsymbol{H}$ in Equation (3.4) by the graph representation matrix $\boldsymbol{H}_g$:

$$\text{LEELoss}_{(graph)} = \mathcal{L}_{CE}(G, \boldsymbol{y}) + \lambda \cdot \text{Tr}(\boldsymbol{H}_g{}^\top \boldsymbol{L}_g \boldsymbol{H}_g) \tag{3.8}$$

Remarkably, the major hurdle of this implementation is the acquisition of the graph similarity matrix $\boldsymbol{A}_g$ (adjacency matrix of the meta graph), which is used for calculating $\boldsymbol{L}_g$. Unlike the adjacency matrix, which naturally represents node-level interactions of a graph, $\boldsymbol{A}_g$ should be artificially created with graph priors to measure the relationship among individual graphs.

Label-Based Graph Similarity Matrix For supervised graph classification tasks where label information is involved in the learning process, we intend to view label information as evidence to establish edges among graphs, such that graphs with the same label are deemed to connect:

$$[\boldsymbol{A}_g]_{ij} = \begin{cases} 1, \text{ if } y_{\mathcal{G}_i} = y_{\mathcal{G}_j} \ (i \neq j) \\ 0, \text{ if } y_{\mathcal{G}_i} \neq y_{\mathcal{G}_j} \text{ or } i = j \end{cases} \tag{3.9}$$

where $y_{\mathcal{G}_i}$ is the label of graph $\mathcal{G}_i$. In the way that using labels to construct $\boldsymbol{A}_g$, we have both classification loss and Laplacian eigenmaps loss involving both explicit and implicit label information, respectively.

Kernel-Based Graph Similarity Matrix Graph kernels, which are a family that directly defines a similarity metric on graphs in the manner of calculating an inner product in a Hilbert feature space, have long been a popular tool to accurately measure the similarity between graphs (Siglidis et al., 2020). Formally, a graph kernel is defined as a symmetric, positive semidefinite function k on a set of graphs

G such that $k : G \times G \mapsto \mathbb{R}^{|G|\times|G|}$:

$$k(\mathcal{G}_i, \mathcal{G}_j) = \langle \phi(\mathcal{G}_i), \phi(\mathcal{G}_j) \rangle_{\mathcal{H}} \tag{3.10}$$

where $\mathcal{G}_i$, $\mathcal{G}_j \in G$, ϕ is a feature map, and $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is an inner product operation in a Hilbert feature space. Having the effectiveness and access to multiple powerful kernel methods, we can use graph kernels to construct our graph similarity matrix $\boldsymbol{A}_g$, within which $[\boldsymbol{A}_g]_{ij} = k(\mathcal{G}_i, \mathcal{G}_j)$. Note that $[\boldsymbol{A}_g]_{ij} = 0$ for $i = j$. Here, we consider two different graph kernels.

Weisfeiler-Lehman Optimal Assignment Kernel Given two graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, the Weisfeiler-Lehman optimal assignment kernel (Kriege et al., 2016), which is extended from the Weisfeiler-Lehman subtree kernel based on the theory of valid assignment kernels, is defined as:

$$k(\mathcal{G}, \mathcal{G}') = \max_{B \in \mathfrak{B}(\mathcal{V}, \mathcal{V}')} \sum_{(v, v' \in B)} \sum_{i=0}^{h} \delta(\ell_i(v), \ell_i(v')) \tag{3.11}$$

where $\mathfrak{B}(\mathcal{V}, \mathcal{V}')$ represents the collection of bijections between $\mathcal{V}$ and $\mathcal{V}'$, v is a node in the node set V, h represents the number of iterative relabeling procedure of the Weisfeiler-Lehman kernel, $\ell_i(v)$ is the label of node v after i^{th} label refinement step. Weisfeiler-Lehman optimal assignment kernel measures similarity between graphs relying on the base kernel $k(v, v') = \sum_{i=0}^{h} \delta(\ell_i(v), \ell_i(v'))$, which is induced by a hierarchical relabeling procedure, to calculate the degree on similar neighboring node of node v and v'. Generally, the involvement of the optimal assignment kernels offers the Weisfeiler-Lehman optimal assignment kernel a more accurate description of graphs' similarity than its original one.

Shortest-Path Kernel Given two graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, and a procedure that correspondingly decomposes them into shortest-path graphs $\mathcal{S} = (\mathcal{V}_s, \mathcal{E}_s)$ and $\mathcal{S}' = (\mathcal{V}'_s, \mathcal{E}'_s)$, where each edge is assigned a label equivalent

to the shortest distance between its endpoint nodes in their original graphs, the shortest path kernel (Borgwardt & Kriegel, 2005) measures graph similarity through an exclusive comparison operation, following the formula below:

$$k(\mathcal{S}, \mathcal{S}') = \sum_{e \in \mathcal{E}_s} \sum_{e' \in \mathcal{E}'_s} k^{(1)}_{walk}(e, e') \tag{3.12}$$

where $k^{(1)}_{walk}$ is a 1-length walk synthetic kernel designed with a comparison mechanism that is based on labels of edges e and e', and also the node label of the endpoints of those edges. Generally, the shortest-path kernel first transforms graphs into their shortest-path versions and then compares shortest paths in pairs in shortest-path graphs.

To conclude, although the graph similarity matrix calculated by graph kernel methods does not indicate the pair-wise graph relationship of the same accuracy level as that using label information, it further considers the unseen graphs' relation. Because entries of the kernel-based graph similarity matrix are valued from 0 to 1, they can reflect the degree of similarity between graphs. Most importantly, unlike the traditional adjacency matrix, which is sparse, or the label-based graph similarity matrix whose entry values are set to either 0 or 1, the kernel-based graph similarity matrix is a dense one, and each entry of it can, to some extent, imply the potential relationship among graphs instead of naively treating their relationship in a binary manner. In this way, using graph kernels to construct a graph similarity matrix $\boldsymbol{A}_g$ considers unseen cases, and this method is believed to enable GNNs a better generalization capability and overfitting resistance. Noticeably, graph kernel methods render all graphs connected with different degrees, which is not an ideal case. Hence, noises may arise within $\boldsymbol{A}_g$ and disturb the learning efficiency of our proposed method. An approach to this problem is sampling according to the probability distribution (e.g., Gumbel-softmax sampling (Jang et al., 2016) and argtopk sampling). Since every entry value of $\boldsymbol{A}_g$ calculated by graph kernels can also be viewed as the probability of how likely a graph connects to the other, performing

probability sampling on elements of $\boldsymbol{A}_g$ will help remove those entries with low probability and thus alleviate the negative impact brought by noise.

Given the availability of the graph similarity matrix $\boldsymbol{A}_g$, we propose another similar method, which borrows ideas from both the information aggregation in message passing and Laplacian Eigenmaps. Recalling the fundamental principle that GNNs aggregate and transfer node information, different types of GNNs learn node representations in a graph by aggregating neighboring nodes' information and then updating their corresponding representations. By setting l aggregation layers, GNNs are capable of capturing graph structural information within the l-hop neighborhood. Considering the random walk normalized Laplacian $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{A}$ into Equation (3.2), we derive:

$$\mathrm{Tr}(\boldsymbol{H}_g{}^{\top}\boldsymbol{L}_g\boldsymbol{H}_g) \tag{3.13}$$

$$= \mathrm{Tr}(\boldsymbol{H}_g{}^{\top}(\boldsymbol{I} - \boldsymbol{D}_g^{-1}\boldsymbol{A}_g)\boldsymbol{H}_g) \tag{3.14}$$

$$= \mathrm{Tr}(\boldsymbol{H}_g^{\top}\boldsymbol{H}_g) - \mathrm{Tr}(\boldsymbol{H}_g^{\top}\hat{\boldsymbol{A}}_g\boldsymbol{H}_g) \tag{3.15}$$

where $\hat{\boldsymbol{A}}_g$ is degree-normalized $\boldsymbol{A}_g$. Focusing on the latter part, one can easily observe that the term $\hat{\boldsymbol{A}}_g\boldsymbol{H}_g$ reflects the way that the node-level information is aggregated in the message passing of GCN. Intuitively, it is easy for us to interpret that the term $\boldsymbol{H}_g{}^{\top}\hat{\boldsymbol{A}}_g\boldsymbol{H}_g$ intrinsically aggregates 1-hop neighboring information for a graph and compares representations between that graph and the average of its neighbors. However, the value of the term $\mathrm{Tr}(\boldsymbol{H}_g{}^{\top}\hat{\boldsymbol{A}}_g\boldsymbol{H}_g)$ will be out of scale, as the more similar two graph representations are, the larger their inner product is. Considering the scale and direction of the cross-entropy loss, we invert this term, ensuring that its value is of the same scale as the cross-entropy loss. Ultimately, we define this method as the following equation:

$$\mathrm{LEELoss}_{(agg)} = \mathcal{L}_{CE}(G, \boldsymbol{y}) + \lambda \cdot (\mathrm{Tr}(\boldsymbol{H}_g{}^{\top}\hat{\boldsymbol{A}}_g\boldsymbol{H}_g))^{-1} \tag{3.16}$$

Similar to the underlying motivation of Laplacian Eigenmaps, the latter term will also force representations of graphs to be similar according to the relationship information encapsulated in $\boldsymbol{A}_g$.

3.4 Experimental Evaluation

To validate the effectiveness of our proposed method, LEELoss, we evaluate its performance with three classic vanilla GNN models, which are GCN (Kipf & Welling, 2017), GraphSAGE (W. Hamilton et al., 2017), and GIN (K. Xu et al., 2018), and make a comparison with their original setting, which is with the cross-entropy loss. The objective of this experimental evaluation is to solve the following two questions:

- **Question 1**: Can the proposed LEELoss enhance the representation learning capability of vanilla GNNs on graph-level classification tasks, by simply appending after the conventional cross-entropy loss?
- **Question 2**: Which variants of LEELoss is the final choice? Can explicitly introducing inter-graph information for graph-level classification bring better performance for these GNN models?

3.4.1 Experimental Setup

3.4.1.1 Datasets

We use 7 datasets, which originate from the graph classification benchmarks (Yanardag & Vishwanathan, 2015), to evaluate the effectiveness of LEELoss. These datasets are categorized into two major classes: Molecules (MUTAG, PTC, PROTEINS, NCI1) and Social Networks (IMDB-BINARY, COLLAB, REEDIT-BINARY). To be consistent with the original settings for the molecule datasets and social networks datasets (K. Xu et al., 2018), whose experimental goal is to let the model learn a representation depending on the network structure rather than input node features, the input node features of these datasets are set as categorical labels and

are artificially created, respectively. For social network datasets, REEDIT-BINARY has the same feature for all nodes, and IMDB-BINARY and COLLAB use the one-hot encoding of node degree as their node features.

Table 3.1: Hyperparameter Search Space

Hyperparameter	**Search space**
Batch Size	{32, 64, 128}
Hidden Dimension	{16, 32, 64}
λ	{0.0001, 0.001, 0.01, 0.1, 1}
Graph Similarity Matrix	{Label, WL, SP}[1]
Sampling Type	{None, Gumbel, Argtopk}[2]
η	{0.1, 1}

[1] Label, WL, and SP are the corresponding abbreviations of the aforementioned graph similarity construction methods.
[2] Sampling type for the kernel-based graph similarity matrix.

3.4.1.2 Configuration

To clearly identify the effectiveness of LEELoss, we keep the basic configuration of the compared pair the same. That is to say, for different datasets, we fix the fine-tuned settings of original GNN frameworks for models with LEELoss to assess the performance of our proposed method under the same and relatively fair environment as its counterpart. For models involved, we utilize a 5-layer setup (including the input layer) and append batch normalization (Ioffe & Szegedy, 2015) on each hidden layer. Adam optimizer (Kingma & Ba, 2014) is also applied, where the learning rate is initially set to 0.01 and decayed by 0.5 per 50 epochs. To avoid overfitting, the number of epochs for different datasets ranges from 100 to 350 according to their learning curves, as well as their dataset sizes, displayed by K. Xu et al. (2018). Regarding other hyperparameters, the number of negative samples m is fixed to 5, and Table 3.1 shows the search space for the rest. In our experiment, we perform 10-fold cross-validation following Yanardag & Vishwanathan (2015) and Niepert et al. (2016). Due to the small dataset size, the cross-validation only uses training sets

and validation sets. Our cross-validation strategy records the best performance of each fold and reports the average and standard deviation of the maximum validation accuracy across 10 folds.

3.4.2 Performance Analysis

Table 3.2: Graph classification accuracies (%) of LEELoss$_{(\text{node})}$, LEELoss$_{(\text{graph})}$, and LEELoss$_{(\text{agg})}$ versus the original cross-entropy loss with GCN, GraphSAGE, and GIN on 7 datasets.

GCN				
Dataset	Original	LEELoss$_{(node)}$	LEELoss$_{(graph)}$	LEELoss$_{(agg)}$
MUTAG	91.46±6.8	**91.46±5.9**	**91.99±4.9**	**92.57±5.4***
PTC	70.96±5.3	**71.55±5.6**	**72.71±5.2***	**71.27±6.5**
PROTEINS	78.34±3.3	**78.97±2.8**	**78.62±3.0**	**79.87±3.0***
NCI1	76.47±1.5	76.30±1.7	**76.59±1.6***	76.11±2.1
COLLAB	80.38±1.6*	**80.38±2.0***	79.68±1.4	80.06±1.1
IMDB-B	77.90±1.8	**77.90±1.8**	**78.50±2.5***	**78.30±2.0**
REDDIT-B	50.00	50.00	50.00	50.00
GraphSAGE				
MUTAG	90.44±6.1	**90.96±4.7**	**92.02±5.9**	**92.51±4.9***
PTC	73.85±3.8	**74.13±5.5**	**75.91±5.2***	**74.42±2.5**
PROTEINS	78.97±3.3	**79.15±3.3**	**78.97±3.6**	**79.33±3.5***
NCI1	75.11±1.4	74.79±1.4	**75.18±1.5***	74.11±1.4
COLLAB	OOM	OOM	OOM	OOM
IMDB-B	77.50±2.7	76.40±2.4	**78.00±2.9***	**78.00±2.4***
REDDIT-B	OOM	OOM	OOM	OOM
GIN				
MUTAG	94.68±4.7	94.12±5.7	**95.15±6.2**	**95.76±4.6***
PTC	70.06±4.5	**72.10±4.8**	**72.38±4.6**	**73.86±6.1***
PROTEINS	78.70±3.6	**79.42±2.7**	**79.61±3.4**	**79.69±3.6***
NCI1	83.80±1.6*	82.77±1.7	83.36±2.1	82.63±1.3
COLLAB	80.24±1.2	**80.38±1.4**	78.78±1.5	**80.46±1.6***
IMDB-B	77.80±1.9	**78.60±1.9**	**78.50±2.4**	**78.80±2.1***
REDDIT-B	80.64±2.8*	OOM	79.95±2.3	78.90±2.9

[1] LEELoss results that outperform the original counterpart are highlighted in bold.
[2] For each dataset, the best performance within each backbone across losses is marked with *.
[3] OOM refers to out of memory.

Although the proposed method is not a specific model, test performance still qualifies the generalization capability that directly determines future downstream

applications of a model and thus indicates to what extent a model can be improved when applied with LEELoss. In response to **Question 1**, Table 3.2 compares prediction accuracies between the original cross-entropy loss and LEELoss, which includes $\text{LEELoss}_{(node)}$, $\text{LEELoss}_{(graph)}$, and $\text{LEELoss}_{(agg)}$. Note that in ($\text{LEELoss}_{(graph)}$ and $\text{LEELoss}_{(agg)}$), we consider that label-based and different kernel-based graph similarity matrices are of types for the graph-level implementation, and only report the best result between them.

As can be seen, it is quite obvious that the proposed method has better prediction accuracies in most cases, which demonstrates that LEELoss is indeed rich in capacity to facilitate the representation learning process of chosen models. Especially in MUTAG, PTC, PROTEINS, and IMDBBINARY datasets whose sizes are relatively small, both node-level and graph-level implementations of LEELoss consistently show an overwhelming performance over the original one on three GNN frameworks. The consistency of LEELoss on these datasets implies that it is probably more suitable for datasets with a smaller scale. However, although certain specific methods have achieved higher scores, LEELoss yields weaker average performance for large-scale datasets, including NCI1, COLLAB, and REDDITBINARY. The occurrence of such an opposite situation is expected, as more data introduces too many uncertainties for the representation learning based on Laplacian Eigenmaps to handle. Overall, six out of six datasets with GCN, five out of five datasets with GraphSAGE, and five out of seven datasets with GIN have recorded their highest prediction accuracies by LEELoss, demonstrating the robust expressive power of LEELoss. Remarkably, among different implementations of LEELoss, the graph-level implementation seems to be superior to its node-level counterpart, as graph classification problems favor more graph-level information. Within graph-level implementation, $\text{LEELoss}_{(agg)}$ is more compatible with GIN and performs better than the other two. Also, $\text{LEELoss}_{(node)}$ demands a higher computational cost, leading to the out-of-memory problem in REDDITBINARY. Noticeably, GCN fails to converge on REDDITBINARY due to its weaker ability to distinguish isomorphic

graphs (K. Xu et al., 2018).

3.4.3 Ablation Analysis

Table 3.3: Comparison results of separate implementations to the combination implementation, and ultimately to none of them.

Loss	MUTAG	PTC	PROTEINS	IMDB-B
LEELoss$_{(graph)}$	95.76±4.6	73.86±6.1	79.69±3.6	78.80±2.1
LEELoss$_{(node)}$	94.12±5.7	72.10±4.8	79.42±2.7	78.60±1.9
LEELoss$_{(node+graph)}$	94.12±4.4	71.80±4.9	79.24±2.8	78.40±2.2
w/o LEELoss	94.68±4.7	70.06±4.5	78.70±3.6	77.80±1.9

In response to **Question 2**, Table 3.3 demonstrates the experimental results of the ablation study on four major datasets, where our proposed methods are significantly beneficial, with the implementation of the combination gradually reduced to the original cross-entropy loss. As expected, the simultaneous application of node-level and graph-level implementations results in a longer computation time, but it can still achieve a better or comparable classification capability compared to its original counterpart. However, it is clear from the results that the effectiveness of the combination is less than that of cases where the node-level or graph-level implementation is exclusively considered. Considering its corresponding high training performance on these datasets, this is likely due to overfitting resulting from the overlapped function of the node-level and graph-level modules. Therefore, we conclude that the individual module of the node level or graph level will suffice for our loss function enhancement.

3.4.4 Hyperparameter Sensitivity Analysis

In addition, we observe how the performance of LEELoss varies with the weight λ and the choice of graph kernel. Note that we fix other variables of LEELoss$_{(agg)}$ to fairly justify the influence of these two parameters, and the results in this experiment

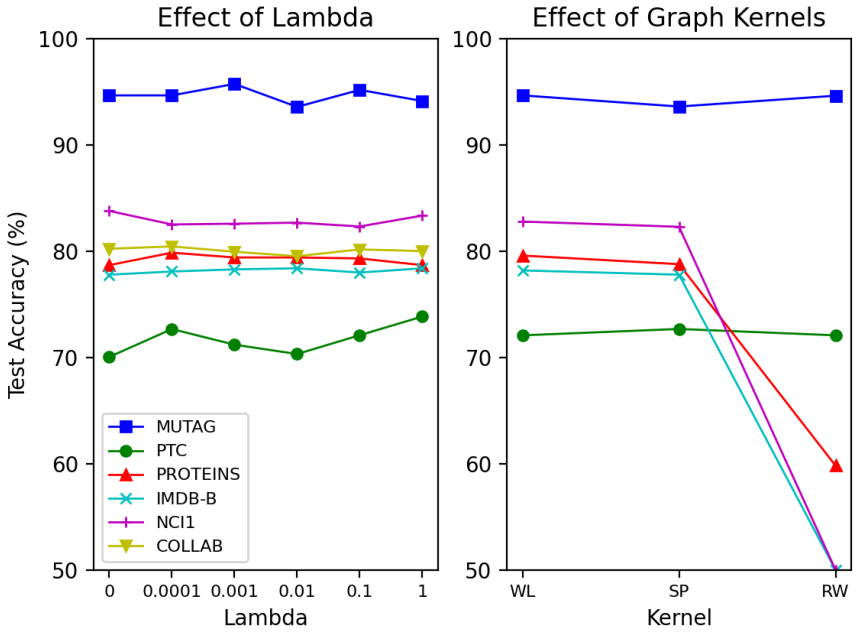


Figure 3.2: Effect of the weight λ of the regularization term and different graph kernels on test accuracy.

do not represent its best performance. Figure 3.2 respectively shows test accuracies by different λ and different graph kernels. As we can see, the left diagram shows a pattern that our regularization term has essential facilitation on model performances around λ of 0.0001 and 0.1. Starting from $\lambda = 0$, which is equivalent to the original objective function case, λ tends to exert a positive influence on the model performance as it increases. Normally, as the λ continuously increases to 1, model performance should drop since the regularization term itself is originally designed for unsupervised learning and dominates the classification loss of a supervised model. According to the right diagram, we find that different graph similarity matrices based on different graph kernels do not show obvious tendencies of influence on the classification accuracy. We also test an extra Geometric Random Walk Kernel, but it fails to converge when applied to large-scale datasets due to the out-of-memory problem caused by its high computational cost.

3.5 Conclusion

This work investigates enhancing models' original objective function using graph priors without altering models' structure or intrinsic properties. Highlighting that Laplacian Eigenmaps naturally shorten the distance between similar objects in the representation space, we present LEELoss by applying Laplacian Eigenmaps as a regularization term of the cross-entropy loss. Since the graph classification learning process involves representations of different states, we implement LEELoss from the node level and then extend it to the graph level to investigate the representation learning capability promoted by Laplacian Eigenmaps of different perspectives. To take advantage of Laplacian Eigenmaps in graph-level implementation, various measures are used to construct a graph similarity matrix that captures different inter-graph relationships. We conduct comprehensive experiments, demonstrating that the proposed method improves graph classification performances of three classic GNN frameworks on most datasets. Most importantly, we realize that exploiting appropriate data priors in the objective function other than the previous blocks can also facilitate representation learning, enabling a stronger generalization capability of models. Particularly for the graph-level classification problem, where GNNs struggle to explicitly model interactions between graphs, introducing additional graph priors at the objective level would be a promising and lightweight way to enhance graph representation learning.

Chapter 4

Leveraging Precious Anomalies for Class Imbalance

Graph-level anomaly detection (GLAD) remains relatively underexplored compared with node-level anomaly detection, and the supervised setting is even less studied than the dominant unsupervised paradigm. This gap largely stems from a long-standing assumption that anomalies are rare, weakly labeled, or even unobservable in practice. Nevertheless, recent studies have begun to revisit supervised GLAD by exploiting a limited number of labeled anomalous graphs and have reported encouraging experimental results. Most existing approaches, however, alleviate the severe class imbalance mainly through reweighting the conventional objective function to balance the model's focus on anomalies. More critically, many of them still follow the standard graph classification recipe that treats anomalies as well-defined targets, but further places emphasis on modeling the detailed information about them. These approaches are to some extent vulnerable because anomalous graphs often contain incomplete, biased, highly diverse information. Therefore, overfitting to their patterns can hurt the robustness and generalization of the model. This led us to shift our focus to the data itself and investigate its potential to enhance graph representation learning in graph-level anomaly detection tasks.

This chapter responds to the **Research Question 2 (Data Level): How**

to help models generalize better to unseen anomalies by fully exploiting and leveraging scarce anomalous graph samples? and develops a training framework that consists of two modules that leverage anomalies to address the class imbalance and improve generalizability from a data-level perspective, without requiring sophisticated layers tailored to anomaly patterns.

4.1 Introduction

Graph-level anomaly detection (GLAD) aims to recognize individual graphs that exhibit patterns significantly deviating from those of the majority. GLAD has applications in various fields, such as detecting rare chemical compounds in biochemistry, identifying fraudulent activities in transaction networks, and recognizing abnormal behavior, including bot activity, in social networks (Akoglu et al., 2015; X. Ma et al., 2021). Many recent studies on GLAD, leveraging the ability of Graph Neural Networks (GNNs) to capture topological information and graph features effectively, have made significant progress and established GNN-based methods as the mainstream paradigm in the field of graph anomaly detection (Zhao & Akoglu, 2023; Zhao et al., 2022; Qiu et al., 2022; X. Luo et al., 2022; R. Ma et al., 2022; Y. Liu et al., 2024). Furthermore, these mainstream GLAD studies often highlight the high cost of labeling anomalies, which exacerbates the class imbalance problem, and assume that anomalies are generally unknown in advance (X. Ma et al., 2021; Pang et al., 2021; Kim et al., 2022). As a result, these studies discard anomalous samples and adhere to the unsupervised learning paradigm. Since anomalies are only marginally observed in such tasks, unsupervised GLAD methods are limited in utilizing labeled anomaly data (Pang et al., 2021). As illustrated in Figure 4.1(Right), this lack of diversity in training data renders the model suboptimal in constructing the boundary for normal and anomalous graphs.

However, although it is widely acknowledged that incorporating any available labeled data, even a small amount, can significantly improve anomaly detection

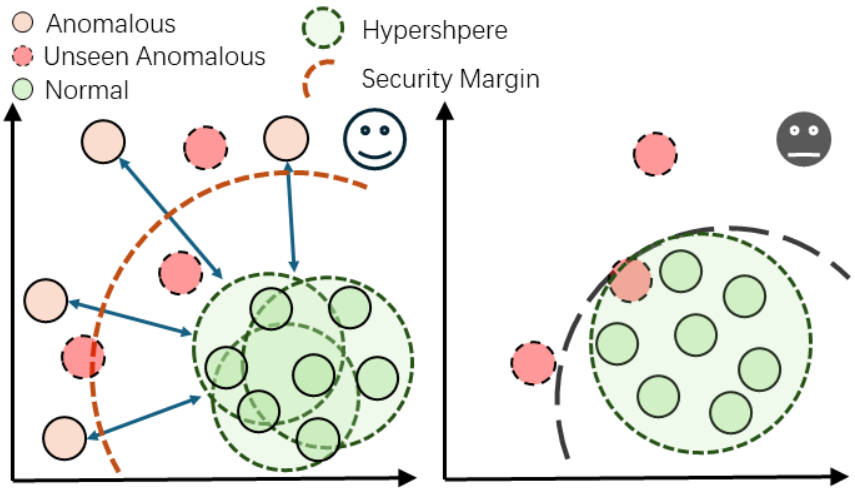


Figure 4.1: A brief example of reference alignment that provides a contrastive view.

performance (Aggarwal & Aggarwal, 2017; X. Ma et al., 2021, 2023), and that the inherent antagonism between normality and anomaly allows anomaly detection to be framed as a binary classification problem where one class is rare (F. Liu et al., 2022; H. Wang et al., 2022; Lim et al., 2018; Z. Xu et al., 2022), only a few state-of-the-art works have attempted to leverage labeled anomaly data and introduce GLAD into the supervised learning paradigm (G. Zhang, Yang, et al., 2022; X. Ma et al., 2023; Dong et al., 2024). These methods primarily employ revised loss functions to mitigate a so-called class imbalance problem brought by the scarcity of labeled anomalies and have achieved notable performance. Considering the higher-dimensional complexity of the graph-level tasks, they bypass augmentation techniques as commonly adopted in other domains (Pang et al., 2023; F. Liu et al., 2022; D. Wang et al., 2019; Lim et al., 2018) for addressing the class imbalance issue. Some cutting-edge studies at the node level have utilized augmentation techniques of different modalities to increase the sample size and improve model robustness (F. Liu et al., 2022; Ding et al., 2021). However, the non-Euclidean nature and the higher-dimensional complexity of graphs render these typical augmentation techniques, which investigate the point information within an individual graph, less applicable to graph-level tasks. Additionally, anomalous graphs are heterogeneous (Pang et al., 2021) in that they arise

from many different types and causes, challenging the models' generalizability on unseen cases. Therefore, two key challenges arise for current label-leveraged supervised GLAD methods. ***Challenge 1***: The structural irregularity across graphs has made graph-level augmentation techniques underexplored in GLAD, despite their potential to address the fundamental scarcity of labeled anomalies. ***Challenge 2***: The extreme scarcity of anomalous samples available hinders these GLAD methods from modeling detailed graph patterns to generalize to unseen cases (Pang et al., 2021; Y. Gao et al., 2023).

Recognizing these challenges, in this work, we propose a GLAD framework that enhances the use of precious anomalies, which we term **G**raph **A**ugmentation-based **R**eference **A**lignment framework (**GARA**). GARA is trained in a supervised manner and comprises two distinct components: (1) an augmented-task generation module based on the Mixup technique (H. Zhang et al., 2018) to create a variety of balanced learning batches. Considering that graph generation (Ding et al., 2022), which generates new graphs by capturing key motifs from observed graphs, has proven to be effective for enlarging the scale of training samples in graph-level classification, and that GLAD tasks can be regarded as a special case of binary classification, we investigate this method for augmenting anomalous graphs and then constructing a balanced training environment. (2) A reference alignment module that aligns normal and anomalous graphs with jointly learned normal references. To tackle ***Challenge 2*** that current GLAD methodologies overemphasize the training on existing uncomprehensive anomalies and compromise the models' generalizability to unseen ones, the reference alignment module contracts the normal boundary while expanding the security margin with multiple views of the normal distribution, enhancing resilience to unseen anomalies, all without undermining the contributions of predominant normalities when leveraging anomalous samples in training (Figure 4.1(Left)). To summarize, this work contributes to GLAD from the following points:

- We develop GARA, an intuitive and effective GLAD framework that considers a contrastive perspective using normal references to exploit the potentialities of scarce anomalous graphs, distinguishing it from existing supervised GLAD models, which mainly revolve around modeling existing anomalous patterns.
- We investigate the feasibility of enriching the pool of the anomalous class via anomaly augmentation and employ an anomaly-specialized graph generation strategy, specifically incorporating graph mixup techniques, to mitigate the class imbalance in the GLAD problem.
- Comprehensive experiments on real-world datasets demonstrate the superiority of GARA, which achieves outstanding performance with an intuitive design.

4.2 Related Works

In this section, we briefly introduce the background of graph-level anomaly detection and recent advanced GNN-based approaches for this problem. We also review the Graph Mixup methods and discuss some of their characteristics. Definition and an example of the Graph Mixup are presented in Section 2.2.2.3.

4.2.1 Graph-Level Anomaly Detection

Real-world anomalies inherently exhibit characteristics of sample scarcity and heterogeneity (Pang et al., 2021; Akoglu et al., 2015). Although anomalies can be grouped into a single category in practical implementations, they are much rarer and can exhibit significant diversity in their classes or attributes. This results in a severely skewed and dispersed distribution when compared to that of normal instances. In this context, anomaly detection has been extensively studied for downstream applications, aiming to identify abnormal patterns or objects that deviate from the mass distribution. In the realm of various anomaly detection tasks,

graph-level anomaly detection (GLAD) significantly challenges researchers. These challenges arise not only from the non-Euclidean nature of graphs but also from the higher dimensional complexity involved, rendering the detection of anomalous graphs more intractable than that of anomalies with conventionally tabular or grid-like structures (e.g. textual data represented as bag-of-words (X. Sun et al., 2018) and hyperspectral image represented as fined-grained spectral vectors (X. Ma et al., 2020)), and that of node-level and edge-level counterparts that investigate individual graphs (X. Wang et al., 2021; Ding et al., 2021; Song et al., 2021; G. Zhang, Li, et al., 2022).

Despite the relatively fewer research efforts devoted to it in the past than its node-level counterparts (X. Ma et al., 2021; Kim et al., 2022), GLAD still holds significant and impactful applications in real-world scenarios, ranging from recognizing rare chemical compounds in biochemistry to identifying fraudulent transaction networks in e-commerce (Jiang & Ma, 2018; Akoglu et al., 2015), and is drawing gradually increasing attention. Recent advancements in GLAD works, such as OGCIN (Zhao & Akoglu, 2023), GLAM (Zhao et al., 2022), GLocalKD (R. Ma et al., 2022), and OCGTL (Qiu et al., 2022), primarily revolve around the unsupervised paradigm, concentrating on learning normal patterns. The principle of unsupervised approaches, which solely train on normal samples and identify unusual instances by matching them with the modeled normal distribution, offers an alignment view in GLAD and demonstrates a certain extent of robustness in such scenarios, even without the supervision from labels. Few works, including iGAD (G. Zhang, Yang, et al., 2022), GmapAD (X. Ma et al., 2023), and RQGNN (Dong et al., 2024), consider anomalies in training. Nevertheless, these approaches often focus on exploiting the differences between normal and existing anomalous graphs, easily failing to generalize to unseen cases. Specifically, the first utilizes a random walk kernel to investigate anomalous substructures, the second considers informative nodes as a class indicator via the evolutionary mapping algorithm, and the third involves the Rayleigh quotient as class-wise information. Even capturing sophisticated anomalous pat-

terns well, they tackle the class imbalance problem by adjusting the loss function's attention on two classes without considering augmentation techniques, which are prevalent in node-level tasks. Therefore, the above oversights leave a significant gap in the supervised GLAD framework.

4.2.2 Graph Mixup

Mixup is a data augmentation technique that interpolates the features and labels of two random samples to generate synthetic training examples and has shown significant improvement in the model's generalizability (Ding et al., 2022). Traditional Mixup (H. Zhang et al., 2018), however, is designed for regular grid-like data and is vulnerable to graph data that is located in non-Euclidean space. For instance, a simple adjustment to an edge in a graph may significantly destroy a graph property and change its semantics. In the graph domain, several existing mixup approaches primarily focus on both the dimensions of the graph adjacency matrix and the node feature matrix and utilize various tricks to circumvent this challenge. Manifold Mixup (Verma et al., 2019) mixes graphs in the embedding space, where the uniform representations of graphs can be interpolated with each other. ifMixup (H. Guo & Mao, 2023b) aligns graphs with arbitrary node order to perform interpolation. However, both of them are unable to preserve graph motifs, which are vital for labeling generated graphs. G-Mixup (Han et al., 2022) and S-Mixup (Ling et al., 2023) are the two most advanced candidates for graph mixup.

To be specific, G-Mixup is a non-training approach that utilizes graphon to perform graph generation. It first estimates the graphon of each class and then interpolates between graphons to obtain a mixed one, which serves as a probability matrix for sampling synthetic adjacency matrices. The major issue for G-Mixup is the mixup of node feature matrices since it was originally designed for non-attributed graphs. One proposed solution is to assign the mean feature of all graphs in a class as the feature of the corresponding graphon. In contrast, S-Mixup is a training-based

approach that utilizes a graph-matching network to train assignment matrices of pair graphs. It uses the assignment matrices to align the shape of pair graphs so that two graphs can be mixed up together. Compared to the first two methods, G-Mixup and S-Mixup are capable of keeping the critical information of graphs and are implemented at the input level.

4.3 Methodology

In GLAD tasks, graphs are traditionally categorized into two classes, i.e., normal (class 0) and anomalous (class 1). In the following context, we use the symbols '−' and '+' to represent the normal and anomalous graphs, respectively. The overall framework of GARA consists of two core components: the Augmented-Tasks Generation Module and the Reference Alignment Module. Overviews of the structure of these two modules are depicted by Figure 4.2 and Figure 4.3, respectively. Incorporating the Mixup technique, the initial augmented-task generation module generates additional anomalous training samples to construct random balanced batches $\mathcal{T}_i$. With this balanced setting, the reference alignment module applies the reference contrastive loss $\mathcal{L}_{RC}$, which evaluates the fair distances between graph representations of normal graphs $\boldsymbol{H}_G^-$ and anomalous graphs $\boldsymbol{H}_G^+$ to normal references $\boldsymbol{r}_{G^-}$. In conjunction with the cross-entropy loss function $\mathcal{L}_{CE}$, this module outputs predicted labels to identify anomalies.

4.3.1 Augmented-Tasks Generation Module

4.3.1.1 Anomalous Graphs Augmentation

Because supervised GLAD aligns with a special case of graph-level classification, the utilization of graph generation can offer valuable insights into potential solutions for the class imbalance problem in GLAD. Particularly based on the Mixup (H. Zhang et al., 2018), our augmented-task module conducts anomaly-specialized graph aug-

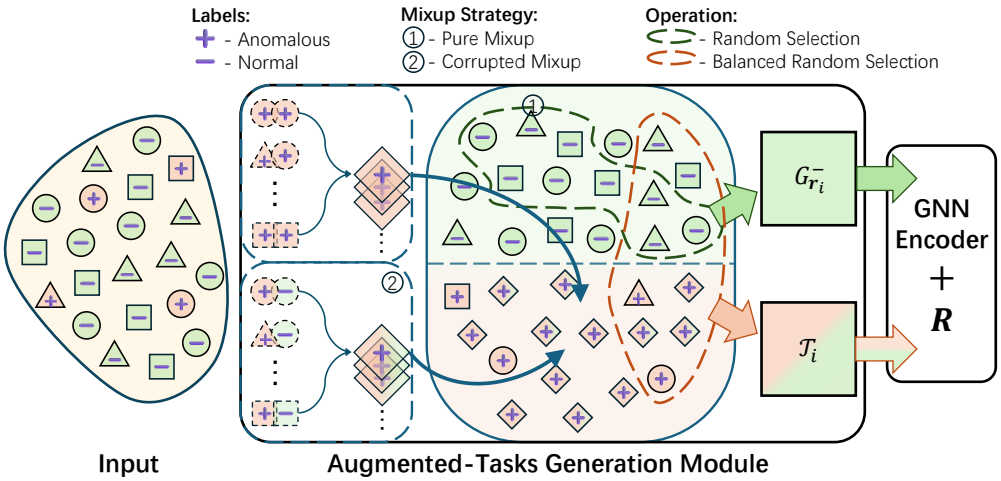


Figure 4.2: GARA: Augmented-tasks Generation Module.

mentation to generate synthetic anomalies within the input feature space, enriching anomalous training samples and thereby mitigating the sample scarcity associated with the anomalous class. Given a random pair of samples $(\boldsymbol{x}_i, y_i)$ and $(\boldsymbol{x}_j, y_j)$, the formal mathematical expression of Mixup is:

$$\widetilde{\boldsymbol{x}} = \lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j, \quad \widetilde{y} = \lambda y_i + (1-\lambda) y_j \tag{4.1}$$

where the random variable $\lambda \sim \text{Beta}(\alpha, \alpha)$ controls the weight of ingredients. Considering the irregularity of graphs, where their adjacency matrices $\boldsymbol{A}$ and features $\boldsymbol{X}$ are normally in different shapes, we follow the Equation (2.15) and use $\mathcal{M}$ to represent the whole mixup operation on graphs. Specifically, for an attributed graph set $G = \{\mathcal{G}_i\}_{i=1}^{N}$, we define an augmented graph by graph mixup as the following equation:

$$\widetilde{\mathcal{G}} = \mathcal{M}\left(\{\mathcal{G}_i, \mathcal{G}_j\} \,|\, G, \lambda\right) \tag{4.2}$$

where the corresponding augmented adjacency matrix and features, respectively, are:

$$\widetilde{\boldsymbol{A}} = \mathcal{M}(\boldsymbol{A}_i, \boldsymbol{A}_j, \lambda), \quad \widetilde{\boldsymbol{X}} = \mathcal{M}(\boldsymbol{X}_i, \boldsymbol{X}_j, \lambda) \tag{4.3}$$

Different from the conventional graph classification problem, the GLAD problem suffers from severe class imbalance. This issue poses a major question regarding the mix-up strategy in the GLAD task. Typically, graph mixup employs a strategy of mixing graphs from different classes (Han et al., 2022; Ling et al., 2023) using the random variable λ. To preserve the representative motifs of anomalous graphs, we consider two strategies from inter-class and intra-class perspectives and investigate their performance in detection.

Corrupted Mixup Strategy. Given a random anomalous graph $\mathcal{G}_i$ as the base from the anomalous graph set G^+, the corrupted mixup strategy expects more diverse anomalous graphs and slightly corrupts $\mathcal{G}_i$ with a random normal graph $\mathcal{G}_j$ from the normal graph set G^- to generate augmented anomalies:

$$\widetilde{\mathcal{G}}^+ = \mathcal{M}\left(\mathcal{G}_i | \, G^+, \mathcal{G}_j | \, G^-, \lambda\right) \quad \text{with} \quad 0 \leq \lambda < 0.5 \tag{4.4}$$

The random λ drawn from the beta distribution may diminish the significance of augmented graphs as anomalies, as a large weight assigned to the normal ingredient introduces excessive noise or incorrect information. Instead of a random λ, we fixed λ as a hyperparameter with a small value on normal graphs, and (1-λ) on anomalous graphs so that anomalous graphs have a more focused contribution relative to normal graphs.

Pure Mixup Strategy. The pure mixup strategy expects pure anomalous graphs and mixes graphs within the anomalous class:

$$\widetilde{\mathcal{G}}^+ = \mathcal{M}\left(\{\mathcal{G}_i, \mathcal{G}_j\} | \, G^+, \lambda\right) \tag{4.5}$$

Remaining the random λ, this strategy generates different combinations of observed anomalies with the guarantee of the number as well as the purity of augmented anomalies.

With the aim of providing a sufficient amount of anomalies for the subsequent construction of learning tasks, both strategies assign labels of augmented graphs to be that of the anomalous class, i.e., $\widetilde{y} = y^+$. Notably, although augmented graphs smooth the dispersed distribution of the anomalous class and prevent the model from overfitting to observed anomalies, we still highlight the importance of observed anomalies since they provide essential contrasts to normal graphs. Given the substantial disparity in quantity between the two classes in the original sample, both mixup strategies are designed to work collaboratively with the oversampling, which selects observed anomalies with replacement, for generating the augmented anomalous graph set.

4.3.1.2 Balanced Learning Task

We construct a variety of balanced learning tasks to simulate diverse data backgrounds. To be specific, given an initial training set $G = \{G^-, G^+\}$ and an augmented graph set $\widetilde{G}^+$ after anomaly augmentation, we first merge them such that $G' = \{G^-, G^+_{aug}\}$, where $G^+_{aug} = G^+ \cup \widetilde{G}^+$. Considering the dispersed distribution of the anomalous class against the normal one, we iteratively perform random masking on graphs within G^- to create numerous subsets G^-_b, which serve as backgrounds with different normal distributions. Each subset $G^-_{b_i}$ is then combined with anomalies to create a distinct learning scene $S_i = \{G^-_{b_i}, G^+_{aug}\}$ and each learning scene S_i constitutes the data pool $\mathcal{T}_i$ for a balanced learning task. Through randomly sampling graph subsets $G^-_{\mathcal{T}_i}$ and $G^+_{\mathcal{T}_i}$ where $G^-_{\mathcal{T}_i} \subset G^-_{b_i}$, $G^+_{\mathcal{T}_i} \subset G^+_{aug}$ and $|G^-_{\mathcal{T}_i}| = |G^+_{\mathcal{T}_i}| = \frac{Batch\ size}{2}$, we have a balanced batch $\mathcal{T}_i = \{G^-_{\mathcal{T}_i}, G^+_{\mathcal{T}_i}\}$.

Ultimately, we will use these generated learning tasks as training batches for the subsequent GNN encoder. It is noticeable that due to the overlapping across learning tasks, the augmented-tasks generation module usually anticipates the number of learning tasks to be larger than that of batches from the conventional mini-batch method, i.e., $|\mathcal{T}| > \frac{|G|}{Batch\ size}$. As highlighted in (Pham et al., 2020; Shimizu et al., 2018), utilizing balanced batches rather than the conventional mini-batch method

proves effective for imbalanced classification. Most importantly, we introduce shifting normal distributions with further randomness in the sample selection of learning tasks to create customized backgrounds. Carefully customized training backgrounds, consequently, encourage the model to generalize well to unseen environments.

4.3.2 Reference Alignment Module

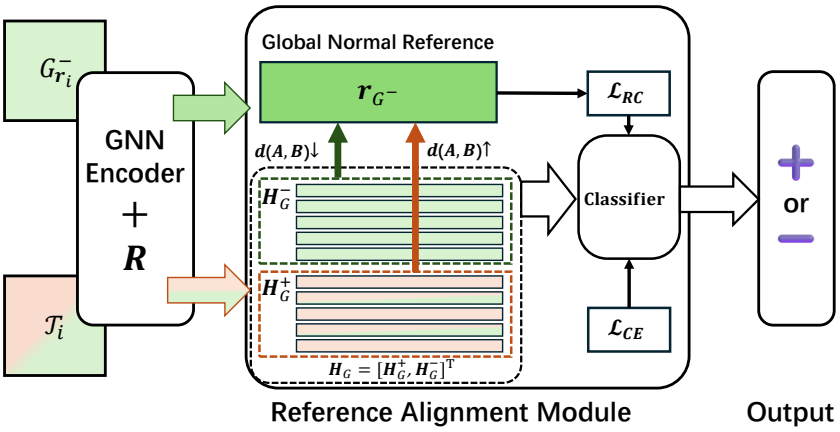


Figure 4.3: GARA: Reference Alignment Module.

One point of view that supports unsupervised anomaly detection is that considering anomalies as one category for the binary classification problem suffers from the heterogeneity of anomalies (H. Wang et al., 2022), which results in poor generalizability of the model. Specifically, the inherent heterogeneity of anomalies may make the model harder to train, and the relatively low heterogeneity of accessible anomalous samples does not provide the model with a comprehensive view of real-world anomalies. Driven by the one-class classification (OCC)(Noumir et al., 2012), we highlight the role that normal instances play in anomaly detection and propose the reference alignment module to bridge this gap. Akin to the unsupervised anomaly detection paradigm, the reference alignment module aligns normal but further anomalous graph representations with shifting normal references within a shared parameter space, thus allowing dynamic centers for normal instances and reinforcing their boundary in a contrastive manner.

4.3.2.1 Shifting Normal References

Given the normal set G^- from the initial training set G and the normal set $G^-_{\mathcal{T}_i}$ from the task $\mathcal{T}_i$, the reference alignment module first constructs a normal reference set $G^-_{r_i}$, where graphs are randomly sampled from the set $\{G^- \backslash G^-_{\mathcal{T}_i}\}$ with the restriction of $|G^-_r| \gg |G^-_{\mathcal{T}_i}|$, for each $\mathcal{T}_i$. Other than the normal patterns encapsulated within each learning task $\mathcal{T}_i$, this module produces a representation $\boldsymbol{r}_{G^-} \in \mathbb{R}^{1\times D}$, or reference, from different normal reference sets $G^-_{r_i}$ to investigate normal graphs from a broader perspective. Note that D is the number of hidden dimensions. Now let $f_\theta(\cdot) : \mathcal{G} \mapsto \mathbb{R}^D$ with learnable parameters θ be a GNN encoder appended by a readout function $\boldsymbol{R}$ that maps a graph into a graph representation, each normal reference $\boldsymbol{r}_{G^-_i}$ is computed as the mean of representations of graphs belonging to the associated normal reference set $G^-_{r_i}$:

$$\boldsymbol{r}_{G^-_i} = \frac{1}{|G^-_{r_i}|} \sum_{\mathcal{G}_j \in G^-_{r_i}} f_\theta(\mathcal{G}_j) \tag{4.6}$$

4.3.2.2 Reference Contrastive Loss

We aim to shorten $d(\boldsymbol{H}^-_G, \boldsymbol{r}_{G^-})$ and increasing $d(\boldsymbol{H}^+_G, \boldsymbol{r}_{G^-})$, where $d(\cdot,\cdot) : \mathbb{R}^D \times \mathbb{R}^D \mapsto [0, +\infty)$ is a distance function, and $\boldsymbol{H}^-_G$ and $\boldsymbol{H}^+_G$ are normal and anomalous graph representations respectively. As a popular paradigm for unsupervised anomaly detection, OCC sheds light on the choice of the distance function and offers us an insight into the issue of graph heterogeneity, that is, learning from the view of normal instead of anomalous against normal. Specifically, one should avoid overly focusing on the local details of anomalies but instead should consider anomalies as instances deviating from the normal distribution and analyze them from the perspective of non-normal, contrasting to normal. Recall from Section 2.2.4.2, the objective of OCC is defined as:

$$\mathcal{L}_{\text{OCC}} = \|f_\theta(\mathcal{G}) - \boldsymbol{c}\|_2^2 \tag{4.7}$$

This loss function utilizes Euclidean distance to map data points into a hypersphere with a fixed center $\boldsymbol{c}$ so that anomalies can be identified if they lie outside the boundary. Based on the OCC objective, we propose the reference contrastive loss, which considers anomalies in training and is defined as the following equation:

$$\mathcal{L}_{RC} = \begin{cases} \dfrac{\|f_\theta(\mathcal{G}) - \boldsymbol{r}_{G^-}\|_2^2}{\varphi} & \text{if } \mathcal{G} \text{ is normal} \\ 1 - \dfrac{\|f_\theta(\mathcal{G}) - \boldsymbol{r}_{G^-}\|_2^2}{\varphi} & \text{if } \mathcal{G} \text{ is anomalous} \end{cases} \tag{4.8}$$

where φ is a normalization factor computed by:

$$\varphi = \sqrt{\sum_{i=1}^{N} \left(\|f_\theta(\mathcal{G}_i) - \boldsymbol{r}_{G^-}\|_2^2\right)^2} \tag{4.9}$$

and N is the number of graphs.

Instead of the fixed center $\boldsymbol{c}$ in Equation (4.7), the reference contrastive loss employs the normal reference $\boldsymbol{r}_{G^-}$ as the center that provides a relative view between normal and anomalous graphs. As previously mentioned, $\boldsymbol{r}_{G^-}$ is jointly learned by the GNN encoder and varies with the learning task $\mathcal{T}_i$ so that graph representations can be continuously revised through dynamic centers, which in turn can be reciprocally updated. Similar to the OCC objective, the first term penalizes the distance between normal graphs and the normal reference and encompasses them within the corresponding hypersphere. The second term incentivizes the distance between anomalous graphs and the normal reference to reinforce the normal hypersphere and simultaneously push it away from anomalous graphs. Additionally, a normalization factor φ is applied to ensure that both distances are on a comparable scale, thereby enabling fair measurement regarding minimizing this loss function. In general, our proposed reference contrastive loss concentrates on leveraging observed anomalies to improve the robustness of the normal graph representations and the security margin between normal and anomalous graphs, such that the model's resilience to the heterogeneity of anomalies can be enhanced.

It is noticeable that the distance term $d(\cdot, \cdot)$ in Equation (4.8) and (4.9), specifically $\|f_\theta(\mathcal{G}) - \boldsymbol{r}_{G^-}\|_2^2$, may produce a nonconvergent case by dying ReLU, which we call a trivial downgrade. In detail, when especially used for a basic GNN encoder (e.g., GCN (Kipf & Welling, 2017)) with ReLU as the activation function (ReLU(x)= $\max(0, x)$), this distance term $d(\cdot, \cdot)$ may learn trivial representations, including the normal reference $\boldsymbol{r}_{G^-}$, and in turn renders the reference contrastive loss $\mathcal{L}_{RC}$ yield an infinite value. We discuss this problem below:

Consider a Euclidean-based distance where the components are within the same space. Then, for input $\boldsymbol{x} = \{x_1, \ldots, x_n\}$ and a simple network having monotonic activation function $\sigma(\cdot)$ that has a lower bound/upper bound with $\inf_x \sigma(\boldsymbol{x}) = 0/\sup_x \sigma(\boldsymbol{x}) = 0$, one can always learn the weight of k-th feature w_k to satisfy $w_k x_i^k < 0$ such that $z_i = \sigma(w_k x_i^k) = 0$. This indicates that representations after a zero-bounded activation function can be downgraded to all 0's vectors, resulting in the Euclidean distance between representations being consistently zero. By mapping all inputs below/above the zero infimum/supremum, the Euclidean-based distance can be consistently approximated to a trial solution as a consequence. To address this trivial downgrade, one should consider activation functions with non-zero bounds, such as Leaky ReLU or GeLU. Alternatively, one might also utilize more sophisticated networks, such as GIN (K. Xu et al., 2018), which are less susceptible to the dying ReLU. Last but not least, we assign a lower bound to the normalization factor φ to prevent division by zero:

$$\varphi' = \max(1, \varphi) \tag{4.10}$$

Ultimately, given the balanced design of our learning tasks, we assume their structure in which the first half is normal and the second half is anomalous. In this

context, our $\mathcal{L}_{RC}$ can be rewritten from the batch perspective as:

$$\mathcal{L}_{RC} = \frac{2}{N}\sum_{j=1}^{N/2}\frac{\|f_\theta(\mathcal{G}_j) - \nabla_{G_i^-}\|_2^2}{\varphi} + \left(1 - \frac{2}{N}\sum_{j=N/2}^{N}\frac{\|f_\theta(\mathcal{G}_j) - \boldsymbol{r}_{G_i^-}\|_2^2}{\varphi}\right) \tag{4.11}$$

where $\boldsymbol{r}_{G_i^-}$ corresponds to each $\mathcal{T}_i$. Ultimately, the reference contrastive loss collaboratively works with a weighted cross-entropy loss to train the model. On top of this, the overall loss function of GARA is formulated as:

$$\mathcal{L} = \mathcal{L}_{CE} + \alpha \cdot \mathcal{L}_{RC} \tag{4.12}$$

where $\mathcal{L}_{CE}$ refers to the weighted cross-entropy loss and α is a hyperparameter controling the weight of $\mathcal{L}_{RC}$. Beyond the binary classification, the reference alignment module extends further and enables the model to separate normal and anomalous graphs from a contrastive perspective that favors anomaly detection. The detailed process of GARA is presented in Algorithm 1.

4.3.3 Time Complexity Analysis

The Augmented-Tasks Generation module adopts two approaches to generate augmented graphs, which are G-Mixup (Han et al., 2022) and S-Mixup (Ling et al., 2023), respectively. Given K training graphs and K synthetic graphs, the computation complexity of the graph generation using G-Mixup involves graphon estimation at least $\mathcal{O}(KN^2)$ and graph generation $\mathcal{O}(KN^2)$. Within the graph generation using S-Mixup, each cross-graph matching operation by the graph matching network (Y. Li et al., 2019) also requires $\mathcal{O}(KN^2)$, and the calculation of the soft alignment matrix for graph generations requires $\mathcal{O}(KN^2)$. Note that even though S-Mixup possesses less theoretical computational cost for a single graph, it requires an additional training stage and usually suffers from $\mathcal{O}(KN^2)$ complexity on both time and space. Particularly on large graphs, S-mixup is less efficient than the G-mixup due to its iterative cross-graph matching operations.

Algorithm 1: GARA

Input : Attributed graph set $G = \{G^-, G^+\}$; Training epochs E; Task number T; Reference sample number R; Batch size B.

Output: Predicted labels $\boldsymbol{Y}$

Initialization
Using G^+ in G as the base, perform $\mathcal{M}$ based on (4.4) or (4.5) and oversampling to generate augmented set $\widetilde{G}^+$ such that $|\widetilde{G}^+| + |G^+| \approx |G^-|$ and merge $\widetilde{G}^+$ with G^+ as G^+_{aug}.
while epoch$< E$ **do**
    Randomly mask data in G^- to generate T subsets G^-_b; Merge them with G^+_{aug} to obtain T learning scenes $S_i = \{G^-_{b_i}, G^+_{aug}\}$
    for $i = 1 \dots T$ **do**
        Randomly sample $\frac{B}{2}$ graphs from $G^-_{b_i}$ and G^+_{aug} to construct a balanced learning task $\mathcal{T}_i = [G^-_{\mathcal{T}_i}, G^+_{\mathcal{T}_i}]$
        Randomly sample R graphs from set $\{G^- \backslash G^-_{\mathcal{T}_i}\}$ to construct a reference set $G^-_{r_i}$
        Learn the normal reference $r_{G^-_i}$ based on (4.6)
        Learn graph representations $\boldsymbol{H} = [\boldsymbol{H}^-_G, \boldsymbol{H}^+_G]$ by f_θ
        Calculate (4.12) using $\boldsymbol{H}$ and $G^-_{r_i}$ and obtain predicted labels $\boldsymbol{y}_i$ by Classifier
        Update f_θ and Classifier by minimizing (4.10)
    end
end
return $\boldsymbol{Y} = \{\boldsymbol{y}_i\}_{i=1}^T$

Given the weight matrix $\boldsymbol{W} \in \mathbb{R}^{F \times F}$, the computational complexity of GIN can be approximated as $\mathcal{O}(NF^2 + N^2F)$. Since the neighborhood aggregation operations are calculated using a sparse operator, it can be rewritten as $\mathcal{O}(NF^2 + |\mathcal{E}|F)$, where $|\mathcal{E}|$ is the edge number. The reference alignment module calculates the normal reference through the message-passing of GIN and therefore yields the same complexity of $\mathcal{O}(NF^2 + |\mathcal{E}|F)$. Despite the additional complexity induced by the Augmented-Tasks Generation Module, its augmented graphs generation process only operates at the initial stage of training and thus can be considered as data preprocessing. Thereby, the computational complexity of GARA with G-Mixup and S-Mixup can be separated into two parts, which are approximately equal to $\mathcal{O}(KN^2)$ before training and $\mathcal{O}(NF^2 + |\mathcal{E}|F)$ for each GIN layer during training, respectively.

4.4 Experimental Evaluation

In this section, we conduct extensive experiments and an ablation study to evaluate the performance of the proposed GARA framework by answering the following questions:

- **Question 1**: Can GARA be comparable or superior to up-to-date GLAD models with a specific training strategy rather than sophisticated model layers?
- **Question 2**: How is the effectiveness of the two key modules of GARA, and how do they contribute to the overall detection performance?
- **Question 3**: When using classification datasets for GLAD, how is GARA's resistance to the performance flip issue?
- **Question 4**: How is GARA's sensitivity to different hyperparameters?

4.4.1 Experimental Setup

4.4.1.1 Datasets

We evaluate the performance of our method on nine commonly used graph-level binary classification datasets, which are obtained from TUDataset (Morris et al., 2020): AIDS, MUTAG, Mutagenicity, NCI1, PROTEINS, IMDB-BINARY, REDDIT-BINARY, KKI, and OHSU. These datasets range from biochemistry to online social networks. Detailed descriptions of the nine datasets used in experiments are provided below, and their corresponding statistics are provided in Table 4.1:

- MUTAG is a small molecule dataset containing nitroaromatic compounds. Class 0 represents their mutagenicity on Salmonella typhimurium, and class 1 represents their non-mutagenicity on Salmonella Typhimurium.
- Mutagenicity is a small molecule dataset of chemical compounds where Class 0 represents mutagen drugs and Class 1 represents non-mutagen ones.

- AIDS is a small molecule dataset where Class 0 are chemical compounds inactive against HIV and Class 1 are active against HIV.
- PROTEINS is a bioinformatics dataset containing graph structures of proteins. Class 0 represents the structures of the enzyme, and Class 1 represents those of non-enzymes.
- NCI1 is a bioinformatics dataset containing chemical compound data published by the National Cancer Institute (NCI). Class 0 represents compounds that can inhibit the growth of non-small cell lung cancer. Class 1 represents those inactive for non-small cell lung cancer.
- IMDB-BINARY is a social network dataset of movie collaboration. Class 0 represents ego-networks extracted from romance movies, and Class 1 represents those from action movies.
- REDDIT-BINARY is a social network dataset where Class 0 contains online discussion threads extracted from Reddit, and Class 1 contains online question/answer threads extracted from Reddit.
- KKI is a bioinformatics dataset where Class 0 represents brain networks with attention deficit hyperactivity disorder and Class 1 represents healthy ones.
- OHSU is a bioinformatics dataset where Class 0 represents Brain networks with hyperactive-impulsive disorder and Class 1 represents healthy ones.

By default, we consider class 0 (C_0) as the anomaly. For the first seven datasets, we follow the method in works (Zhao & Akoglu, 2023; Qiu et al., 2022) to downsample anomalous graphs in the train set to 10% of their original size to simulate real-world anomaly scenarios. It is noticeable that KKI and OHSU are two brain network datasets for an abnormal brain disorder diagnosis called ADHD (Pan et al., 2016) in that the meaning of their labels is consistent with the context of anomaly. Therefore, we retain all their data for training as the work (X. Ma et al., 2023) does.

Table 4.1: Statistics of Datasets

Dataset	#Graphs (Raw)		#Nodes (Avg.)		#Edges (Avg.)		#Features
	C_0	C_1	C_0	C_1	C_0	C_1	
AIDS	400	1600	37.61	10.2	89.9	20.4	38
MUTAG	63	125	13.9	14.9	29.2	44.8	7
Mutagenicity	2401	1936	29.4	31.5	60.6	62.7	14
PROTEINS	663	450	49.9	22.9	188.1	83.1	3
NCI1	2053	2057	25.7	34.1	55.3	73.9	37
IMDB	500	500	20.1	14.9	193.6	192.6	–
REDDIT	1000	1000	641.3	218.0	1471.9	519.1	–
KKI	37	46	190.0	190.0	237.4	239.3	190
OHSU	35	44	190.0	190.0	400.5	381.1	190

The attributes of these datasets are generated using one-hot encoding from node labels, if available; otherwise, they are generated based on node degrees.

4.4.1.2 Baselines

We compare GARA against two groups of competitors, 7 state-of-the-art GLAD baseline methods: (1) ***unsupervised method***, including OCGIN (Zhao & Akoglu, 2023), OCGTL (Qiu et al., 2022), GLocalKD (R. Ma et al., 2022),
SIGNET (Y. Liu et al., 2024), (2) ***supervised method***, including iGAD (G. Zhang, Yang, et al., 2022), GmapAD (X. Ma et al., 2023), and RQGNN (Dong et al., 2024). For those who have not been introduced, OCGIN and OCGTL apply a one-class SVDD classifier (i.e., OCC) to find a hypersphere for normal graphs and identify those lying outside the hypersphere as anomalies. GLocalKD learns normal graphs from local and global perspectives by comparing both graph and node representation pairs. SIGNET considers mutual information between two views of graphs and further achieves the interpretation of anomalies.

4.4.1.3 Configuration

The training frameworks and objectives between unsupervised and supervised GLAD methods entail separate experiment implementations. For unsupervised baselines, we base our results on the unified unsupervised framework from Y. Wang et al. (2024) to report the average performance of 5 iterations. Since these methods identify anomalies with respect to anomaly scores and labels, we report only their ROCAUC. For supervised baselines, we perform 5-fold cross-validation with the splitting setting of 80%/10%/10%/. We select the average performance (ROCAUC and Macro-F1 score) where the model achieves the best average ROCAUC on the validation set. In GARA, we choose GIN (K. Xu et al., 2018) as the base of graph convolution and investigate the performance of the model with the two representative graph mixup methods (i.e., G-Mixup and S-Mixup). We optimize hyperparameters using the automated search toolbox: Optuna (Akiba et al., 2019) for GARA and baselines.

4.4.2 Performance Analysis

In response to **Question 1**, we evaluate the anomaly detection performance of GARA using two commonly used evaluation metrics, which are ROCAUC and F1 score. Table 4.2 provides the performance comparison against seven SOTA baselines on nine benchmark datasets. The experiment results reveal that our proposed GARA achieves exceptional performance, specifically for six datasets in terms of both average ROCAUC and Macro-F1 score. Resting on the assumption that anomalies are unobservable, four unsupervised GLAD baselines tend to be less powerful in detection capability than supervised methods due to the exclusion of anomaly-related information. In particular, by leveraging treasured anomalous labels and extending OCC into a contrastive fashion, GARA takes the lead by approximately 11.67% across nine datasets in terms of average ROCAUC compared with OCGIN, OCGTL, and GlocalKD, which adopt OCC to capture the normal distribution. Notably, attributed to its strong ability to model normal patterns, SIGNET

Table 4.2: Detection ROCAUC and F1 performance on nine datasets. For unsupervised baselines, results are reported as the average performance of 5 iterations. For supervised baselines, experiments are conducted using 5-fold cross-validation with the splitting setting of 80%/10%/10%/, and results are reported according to the average performance where the model achieves the best average ROCAUC on the validation set. The best and second-best results are highlighted by gray and underlining, respectively.

Datasets			**MUTAG**	**Mutagenicity**	**AIDS**	**PROTEINS**	**NCI1**	**IMDB-B**	**REDDIT-B**	**KKI**	**OHSU**
Unsupervised	**OCGIN**	**ROCAUC**	69.31±3.2	59.87±1.7	96.68±0.8	74.52±4.8	65.46±1.3	64.22±3.6	79.10±2.2	52.11±6.5	59.73±6.3
	OCGTL	**ROCAUC**	72.57±3.3	72.17±2.4	99.03±1.2	72.19±3.1	73.61±0.9	65.80±3.3	82.57±1.9	47.57±5.9	60.29±5.1
	GLocalKD	**ROCAUC**	76.27±5.5	68.32±1.9	98.74±1.2	76.03±0.9	70.75±2.1	59.07±2.1	81.88±2.2	53.59±5.9	62.71±5.1
	SIGNET	**ROCAUC**	85.49±4.4	74.81±2.1	97.62±1.1	75.56±2.7	74.32±1.3	73.48±3.5	85.42±2.6	55.72±4.7	62.66±3.8
Supervised	**iGAD**	**ROCAUC**	86.94±6.4	76.05±1.5	99.34±0.5	79.22±2.1	68.76±1.7	72.96±3.6	OOM	70.71±8.7	66.85±1.2
		F1	69.58±3.0	67.23±1.4	77.45±6.9	51.40±5.7	53.27±2.7	64.33±3.1	OOM	63.61±6.9	35.15±0.7
	GmapAD	**ROCAUC**	82.23±3.8	75.71±1.6	99.07±0.7	83.35±2.8	73.97±1.9	81.21±2.7	84.19±3.1	63.31±5.3	72.46±6.1
		F1	74.30±5.8	69.92±2.8	82.77±1.1	64.84±1.9	59.43±2.4	74.27±3.1	77.19±3.7	56.16±8.8	68.62±8.0
	RQGNN	**ROCAUC**	89.89±5.6	84.41±1.5	99.56±0.5	78.50±1.8	81.25±2.4	79.84±4.3	79.20±3.0	66.91±7.4	72.78±9.8
		F1	74.35±7.5	72.99±4.0	85.79±3.0	69.19±4.8	69.38±6.4	57.01±4.9	67.05±2.9	55.90±8.8	53.18±7.2
	GARA-G	**ROCAUC**	88.81±7.0	85.73±1.3	99.50±0.3	79.59±0.9	81.08±1.6	80.10±3.4	88.73±4.4	68.09±8.7	74.72±9.1
		F1	72.66±7.3	77.73±2.3	84.72±1.7	66.98±3.7	71.90±2.0	72.14±5.6	78.92±5.2	54.37±9.2	68.94±9.5
	GARA-S	**ROCAUC**	90.04±5.8	85.71±1.1	99.60±0.4	79.55±2.7	81.51±1.7	80.72±2.8	OOM	73.80±9.5	69.44±6.3
		F1	79.35±5.9	79.24±1.0	84.98±5.4	64.51±6.1	69.78±4.8	73.00±3.1	OOM	56.55±10.3	53.44±8.6

achieves detection performance that is most comparable to that of supervised methods. Nevertheless, it is quite obvious that the supervised GLAD methods generally perform better than unsupervised GLAD methods, highlighting the effectiveness of incorporating anomalous information during GLAD training.

Within the supervised category, GARA has significant advantages compared with iGAD and GmapAD on average, except for PROTEINS and IMDB-BINARY. The RQGNN model demonstrates competitive performance across seven datasets with node labels, but its effectiveness, especially the F1 score, is diminished on IMDB and REDDIT datasets, which lack node labels. GARA of two G-mixup and S-mixup share similar performance on both metrics. It is expected since these two methods exhibit competitive performance in graph classification tasks. However, the latter demands a higher computational cost due to the training of a graph-matching network (Y. Li et al., 2019). Notably, on the KKI and OHSU datasets, where we did not perform the downsampling, GARA detects anomalies better, underscoring the model architectures that are exclusively designed to capture anomalous patterns in the balanced GLAD context.

By reviewing these two evaluation metrics across 9 datasets, GARA exhibits strong consistency and is at least comparable to leaders in those datasets where it does not outperform. GARA introduces a specifically tailored training strategy with the minimal involvement of anomalous samples to handle GLAD tasks, showcasing salient generalizability and flexibility when compared with those with GLAD-specialized model layers.

4.4.3 Albation Analysis

In response to **Question 2**, we conduct an ablation study on two modules of our framework. We begin the experiment from the original GIN and sequentially stack the augmented-tasks generation module and the reference alignment module because the latter is built upon the balanced design of the former. Experiments are

Table 4.3: Ablation study on three datasets. **GA** indicates the Augmented-tasks Generation Module and **RA** indicates the Reference Alignment Module.

Metric	Variants	KKI	PROTEINS	Mutagenicity
ROCAUC	**w/o GA & RA**	50.43±6.4	70.25±1.1	77.83±0.8
	w/ GA	67.02±6.7 (32.8%↑)	74.73±1.1 (6.4%↑)	83.16±0.9 (6.8%↑)
	w/ GA & RA	68.09±8.7 (35%↑)	79.59±0.9 (13.3%↑)	85.73±1.3 (10.2%↑)
F1	**w/o GA & RA**	46.75±5.7	62.58±2.4	70.52±1.2
	w/ GA	51.70±6.9 (10.6%↑)	63.25±3.2 (1.1%↑)	76.61±1.9 (8.6%↑)
	w/ GA & RA	54.37±9.2 (16.3%↑)	66.98±3.7 (7.0%↑)	77.71±2.3 (10.2%↑)

implemented on KKI, PROTEINS, and Mutagenicity datasets over five-fold cross-validation, and we use the same weighted cross-entropy loss as well as the same hyperparameter setting for each stage of the experiment to guarantee a fair playing field.

Table 4.3 showcases the improvement of ROCAUC and F1 performance after plugging two modules. We observed that adding the augmented-tasks generation module can boost the detection performance significantly on KKI. Despite the KKI dataset remaining at its original size under our setting, the diverse anomalous samples and balanced learning tasks introduced still enable the model to capture patterns of both classes fairly, indicating this module's capability of enhancing generalizability. However, there is no substantial gain from the reference alignment module. An ablation study on three datasets shows that the model, combined with these two modules, achieves gradual improvement, evidencing the effectiveness of the two modules under conventional anomaly detection scenarios. According to the results, it is evident that the improvement mainly comes from the augmented-tasks generation module, which aims to alleviate the class imbalance problem. The reference alignment module, however, still offers a comparative perspective that further supports the model in anomaly detection. Overall, this ablation study suggests that appropriately designing a training framework for GLAD can also bring about significant improvement in detection performance.

4.4.4 Performance Flip Persistence Analysis

Table 4.4: Detection performance by considering different classes as the anomalous class.

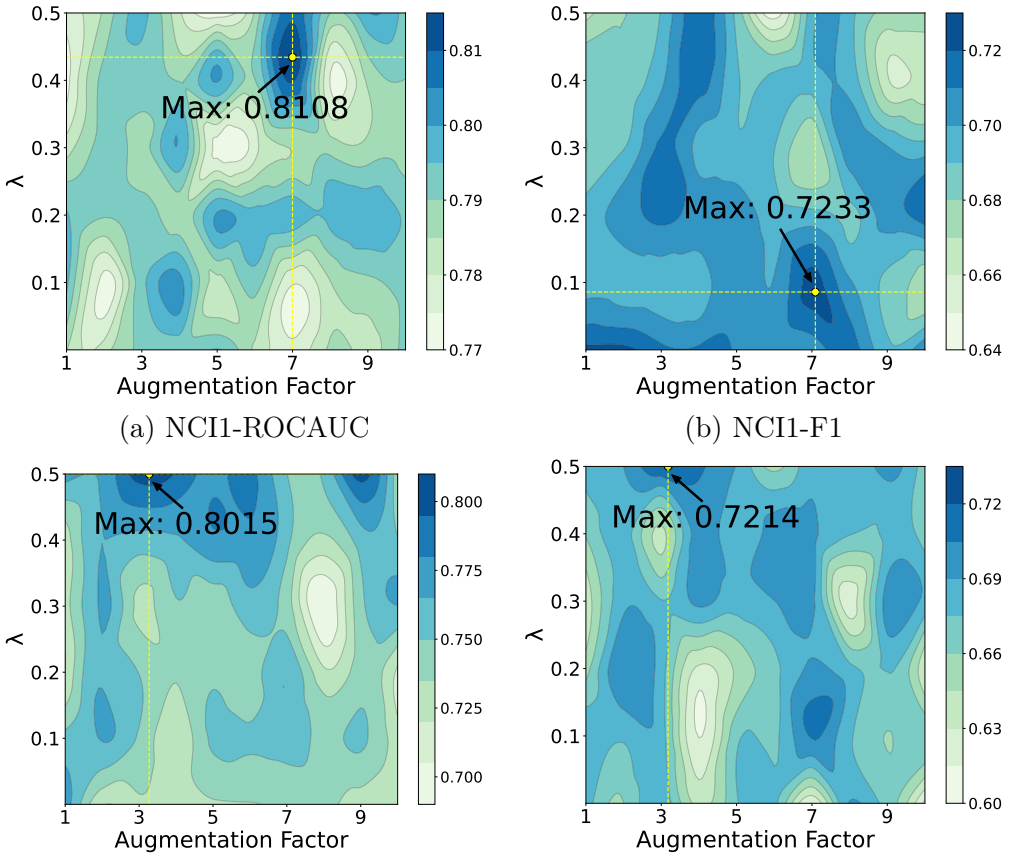
Down Class	MUTAG		Mutagenicity		AIDS	
	X/C_0	**Non-X**/C_1	**X**/C_0	**Non-X**/C_1	**X**/C_0	**Non-X**/C_1
ROCAUC	88.81±7.0	88.53±2.1	85.73±1.3	83.56±0.5	99.60±0.3	99.25±0.6
F1	72.66±7.3	75.53±1.9	77.71±2.3	77.08±0.8	84.72±1.7	84.80±2.7
PROTEINS		**NCI1**		**IMDB-B**		**REDDIT-B**
X/C_0	**Non-X**/C_1	**Non-X**/C_0	**X**/C_1	**X**/C_0	**Y**/C_1	**X**/C_0	**Y**/C_1
79.59±0.9	76.53±3.4	81.08±1.6	80.62±2.2	80.10±3.4	74.15±6.0	88.73±4.4	85.93±1.1
66.98±3.7	71.04±6.4	71.90±2.0	72.24±1.8	72.14±5.6	66.17±6.8	78.92±5.2	77.91±3.6

Performance flip on anomaly detection was first recognized in the work (Zhao & Akoglu, 2023) when using graph classification datasets for evaluation, and is defined as a severe performance gap between two different downsampling directions. This issue is said to be severe within the datasets of **X**&**Non-X** semantics as the **Non-X** class usually exhibits the dispersed distribution in contrast to the dense distribution by the **X** class. Normally, treating the **Non-X** class as the normal class leads to a harder task because its pattern is not centralized and would be difficult to learn. In response to **Question 3**, Table 4.4 presents the ROCAUC and F1 performance of downsampling class 0 (C_0) and class 1 (C_1) as anomalies, respectively.

It can be observed that there is no evident performance flip across five **X**&**Non-X**-type datasets regarding ROCAUC. Although most of their F1 score remain similar between different downsampling directions, that of MUTAG and PROTEINS gains an increase when treating the **Non-X** class as the anomaly and follows the typical easy task in anomaly detection. For IMDB-BINARY and REDDIT-BINARY, which are **X**&**Y**-type, both of their ROCAUC and F1 performance witness a slight drop when downsampling class 1. Performance flip is not usual in such datasets, and this slight phenomenon may be raised due to their lack of node features. Overall, this improvement can be attributed to conducting GLAD under a supervised setting where the mapping between inputs and outputs is guided by label information. Furthermore, two modules by GARA provide a more general and contrastive

view between normal and anomalous graphs that allows the model to distinguish effectively between their patterns.

4.4.5 Sensitivity Analysis



(a) NCI1-ROCAUC (b) NCI1-F1

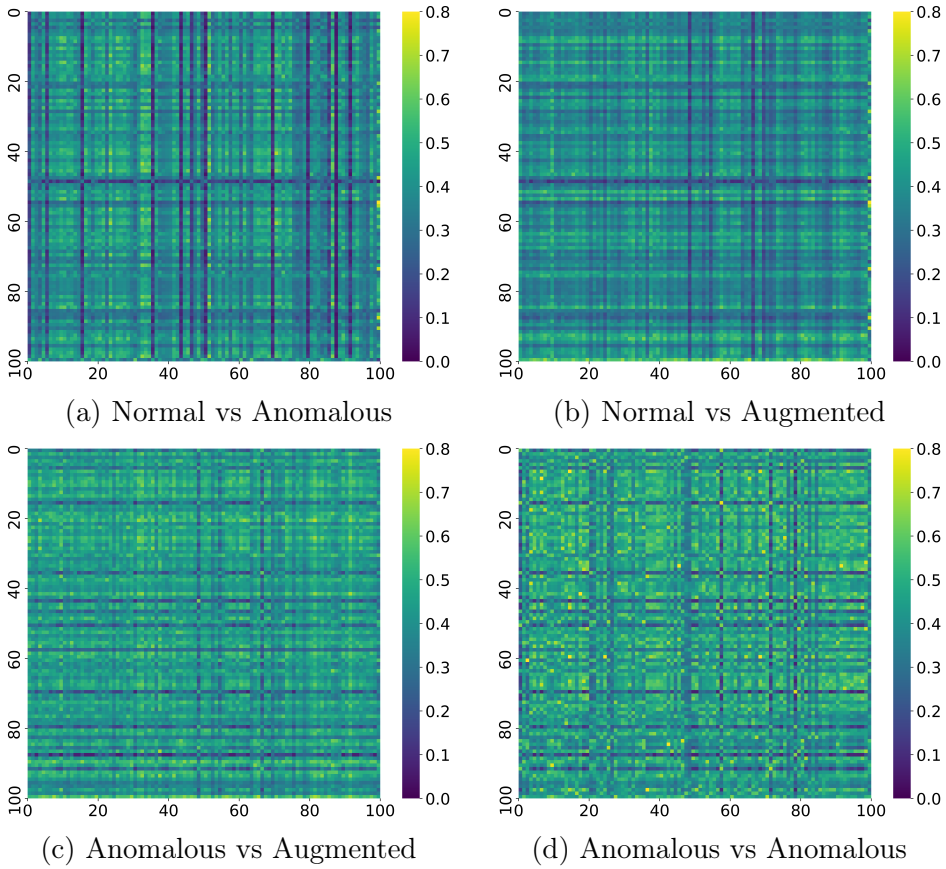
(c) IMDB-ROCAUC (d) IMDB-F1

Figure 4.4: Performance Surface by Two Core Hyperparameters in the Augmented-tasks Generation module.

To assess the specific influence of graph augmentation on the detection performance of our model and answer **Question 4**, we conduct a sensitivity study on NCI1 and IMDB-BINARY datasets over two critical hyperparameters in the augmented-tasks generation module: λ which controls the portion of normal instance in mixed graphs, and Augmentation Factor that designates how many augmented graphs to generate in terms of the multiple of anomalous sample size. Note that $\lambda = 0$ refers to the pure mixup strategy case where λ is assigned a random value between 0 and

1. By observing the Figure 4.4, we notice that, although the ROCAUC change is not obvious on the NCI1 dataset, which is limited in a range of 4.5%, the choice of mixup strategy and number of augmented graphs indeed exerts a certain influence. According to the distributions of the F1 score presented, the F1 score seems to be vulnerable to the λ while not being so for the augmentation multiplier. Overall, the distribution from Figure 4.4 demonstrates that the augmented-tasks generation module facilitates the detection performance. Moreover, the contour surface stretches along both the x-axis and y-axis, indicating that introducing more corrupted augmented anomalous samples into training would be more beneficial.

4.4.6 Augmentation Effectiveness Analysis



(a) Normal vs Anomalous

(b) Normal vs Augmented

(c) Anomalous vs Augmented

(d) Anomalous vs Anomalous

Figure 4.5: Visualization of Graph Similarity.

Graph anomalies are often characterized by rare and subtle structural motifs, and overly aggressive augmentation may introduce noise or dilute anomalous seman-

tics. Therefore, GARA does not aim to replace the scarce observed anomalies with unrestricted synthetic samples, nor does it assume that augmentation alone can fully capture the heterogeneous space of real-world anomalies. Instead, the augmented-task generation module is designed to make more reliable use of limited anomalies by generating smooth local variations around them, without severely diluting genuine anomalous signals. In the corrupted mixup strategy, each augmented anomaly is anchored by an observed anomalous graph, while the contribution of the normal graph is controlled by a small λ to avoid overwhelming the anomalous semantics. In the pure mixup strategy, graphs are mixed within the anomalous class to preserve anomaly purity. Moreover, the original anomalous graphs are always retained in the training pool and continue to provide essential contrasts against normal graphs. The sensitivity study in Figure 4.4 shows that the augmentation strength affects the detection performance, suggesting that semantic preservation depends on a proper degree of corruption.

In addition, we conduct graph similarity analysis to provide structural evidence that the generated anomalies remain less similar to normal graphs and preserve useful anomalous characteristics. To clearly identify the efficacy of augmented graphs, we visually observe the similarity among 100 normal, anomalous, and augmented graphs using the corrupted mixup strategy. Figure 4.5 provides a visualization of their corresponding similarity matrix of the Mutagenicity dataset calculated by the Weisfeiler-Lehman Optimal Assignment Kernel (Kriege et al., 2016). As can be seen, both similarity matrices of **Normal vs Anomalous** and **Normal vs Augmented** possess a larger deeper portion than that of **Anomalous vs Augmented** on average, indicating that the augmented anomalous samples generated by our proposed mixup strategy feature anomalous patterns and are less similar to normal graphs as those original anomalous graphs do (See Figure 4.5(a) (b) and (c)). Figure 4.5(d) provides the similarity matrix between 100 and another 100 anomalous graphs. Notably, native anomaly samples often exhibit a higher degree of similarity among themselves, substantiating the expressivity of augmented graphs as

anomalous graphs. Hence, the role of augmentation in GARA is not to explicitly enumerate all possible anomalous patterns, but to reduce overfitting to a few biased observations while maintaining their contrastive value through reference alignment.

4.5 Conclusion

In this chapter, we realize how the distribution of data in the real world not only challenges people in precisely capturing useful information in data, but also models in generalizing to future unseen cases. Thereby, we highlight the importance of even a small portion of anomalous graphs without fixating on modeling specific patterns of them, but instead enriching their expressivity through data-level techniques to provide stronger and more diverse supervision signals. To better leverage limited anomalies, we design an anomaly-specific training scheme that explores the graph mixup methods in augmenting anomalous samples to alleviate the class imbalance problem and constructs a contrastive view in the context of anomaly detection. This training scheme, instantiated by two specific modules, inherits the graph classification recipe but simultaneously extends the spirit of one-class classification to the supervised paradigm such that abundant normal instances can provide references that enhance detection boundaries. Comprehensive experiments on downsampled real-world datasets elucidate that approaching the graph-level anomaly detection task as a regular classification task, through data-level efforts, can also achieve superior detection performance.

Chapter 5

Promoting Model Efficiency under Capacity Expansion

Continual graph learning studies how to learn a sequence of graph tasks over time while preserving previously acquired knowledge and adapting to new tasks, typically under limited memory and strict efficiency constraints. A prevailing solution is parameter isolation, which allocates task-specific subnetworks to mitigate a so-called catastrophic forgetting problem. Although parameter isolation-based approaches inevitably necessitate architectural design or related mechanisms to achieve this, they do not consider how data can benefit this process in generating subnetworks with efficiency and interpretability, and force the model to infer subnetworks fully implicitly during training. Particularly for graph data, which exhibits irregular and non-Euclidean characteristics, poses further challenges in leveraging it in generating effective task-specific subnetworks. Most importantly, existing parameter isolation-based methods suffer from a huge efficiency bottleneck, as their architecture designs will incur substantial computational overhead if they demand larger model capacity to accommodate more tasks as the task stream grows. This observation motivates us to design a model architecture that can simultaneously address these two challenges when learning representations over continual graph task streams.

In this chapter, we propose a novel GNN framework with a biological neuron-

inspired architecture to answer **Research Question 3: How to sustain computational efficiency of the parameter-isolation paradigm under continual capacity expansion while enabling flexible and extensive cross-task parameter reuse?** To this end, the proposed model leverages graph contextual information as a data prior for the construction of task-specific subnetworks, and further decouples these subnetworks during both training and inference, achieving state-of-the-art effectiveness as well as computational efficiency.

5.1 Introduction

"*Continual learning is the constant development of complex behaviors with no final end in mind* (Ring, 1994)". As a popular continual learning paradigm, parameter isolation-based methods inevitably enlarge model capacity when parameter usage becomes saturated, incurring substantial space and computational overhead, especially on large datasets (Yuan et al., 2026; X. Zhang et al., 2022). Existing methods have largely overlooked the computational complexity issue induced by capacity expansion, but instead prioritize improving the utilization efficiency of the available parameter budget (L. Wang et al., 2024), suffering from a huge efficiency bottleneck. To be specific, they involve the *full-network propagation*, meaning that task-specific computations are performed over the entire shared network, and volumes of unused (i.e., masked) parameters still participate in the calculation during propagation (Figure 5.1 Left). In other words, with disjoint subnetworks and D_{out} output units for each task, the per-task complexity is $\mathcal{O}(D_{in} \cdot (TD_{out}))$, increasing linearly with the expected maximum task number T. Under the assumption of no end in Continual Learning, these methods will eventually suffer from a severe efficiency bottleneck. Under these circumstances, "***improving the computational efficiency for the parameter isolation-based method under model capacity expansion***" becomes a critical demand for this paradigm.

A straightforward solution is to allocate a neatly partitioned parameter block

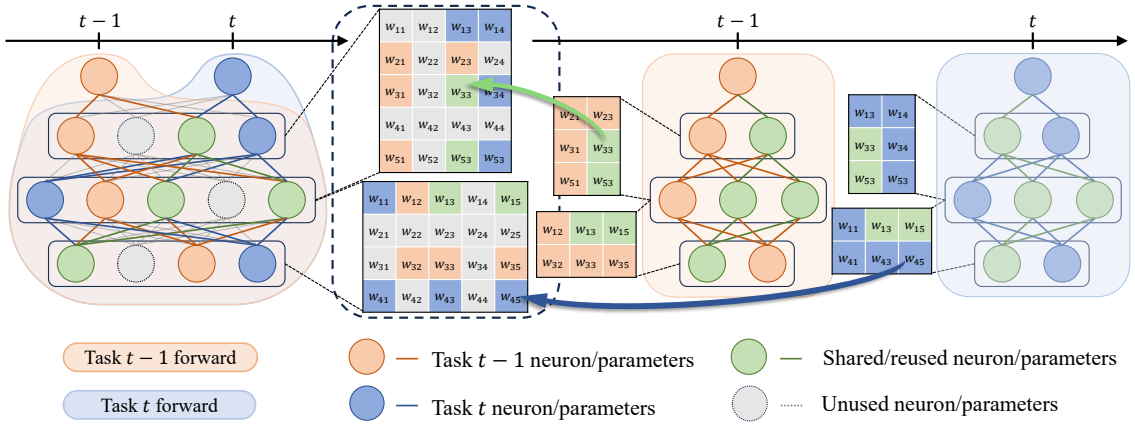


Figure 5.1: Intuitive comparison of full-network and decoupled propagation in parameter-isolation continual learning. Conventional methods, such as PackNet (Mallya & Lazebnik, 2018), Piggyback (Mallya et al., 2018), HAT (Serra et al., 2018), and WSN (Kang et al., 2024), propagate through the entire network despite using only task-specific subnetworks (Left). Our method instead decouples selected subnetworks from unselected parameters while preserving a shared parameter space for knowledge transfer (Right). TAAM (J. Liu & Zhang, 2026) provides a related alternative by assigning task-specific modulators to a frozen backbone, but it isolates new knowledge into separate modulators and only promotes coarse-grained parameter sharing on the entire frozen backbone.

to each task (e.g., neuron-wise selection). However, this method poses challenges to parameter reusability as it reuses parameters on a coarse granularity level, easily causing bias in the transfer of knowledge (Mallya & Lazebnik, 2018; Golkar et al., 2019; P. Zhang et al., 2023; Ko et al., 2025). Parameter reusability is a critical metric of interest; although reducing reuse may avoid bias, it rapidly exhausts free parameters. Fine-grained methods such as Piggyback (Mallya et al., 2018), SupSup (Golkar et al., 2019), and WSN (Kang et al., 2024), learn self-adaptive masks and perform element-wise reuse, allowing for the transfer of knowledge between tasks with fine granularity. Yet, these masks entail continual optimization and offer no guarantee of reuse level, nor a transparent interpretation of why those parameters should be reused. Most importantly, they still incur full General Matrix-Matrix Multiplication operations on the full network due to their irregularity. Ultimately,

it turns back to the original challenge of pursuing effective parameter reuse. Under these circumstances, we reformulate our question as: "***How can we design a parameter isolation-based model that maintains computational efficiency under continual capacity expansion while enabling flexible and extensive cross-task parameter reuse?***".

Inspired by biological neurons that possess distal dendrites to perceive external stimuli from context and adapt to diverse environments (Schachter et al., 2010; Harnett et al., 2012; R. Li et al., 2025) and the underexplored role of parameter isolation in the graph domain (X. Zhang, Song, & Tao, 2024; Ko et al., 2025), we propose the Capacity-Agnostic Graph Neural Network (CAGNN) for continual graph learning, featuring computational efficiency and effective parameter reusability. As a parameter isolation-based method, CAGNN enjoys twofold efficiency through a Context-aware Hierarchical Masking (CHM): **(1)** CHM leverages task-specific context as a prior to construct subnetworks during a brief warm-up, where parameters are shared based on cross-task common patterns, thereby avoiding persistent mask optimization and improving the interpretability of parameter selection; **(2)** By design, CHM supports memory-friendly mask storage and enables training of each task in a decoupled fashion, where every task-specific subnetwork is structurally compressible and can be decoupled from the holistic network (Figure 5.1 Right). After warm-up, CAGNN trains and infers as a compact vanilla backbone, excluding unused parts from other tasks.

We theoretically justify parameter reuse under CHM and introduce a Fisher information-based regularization (Kirkpatrick et al., 2017; H. Liu et al., 2021) to refine subnetwork structures during warm-up. These designs collectively reduce the sensitivity to the model's capacity, alleviating computational complexity brought by capacity expansion. To the best of our knowledge, this is the first work to explicitly study capacity-driven efficiency degradation in parameter isolation under continual model capacity expansion. Empirical analysis verifies that graph contexts can provide sufficient initial bias for subnetwork generation, and comprehensive experiments

show that CAGNN well-handles catastrophic forgetting and improves computational efficiency on node- and graph-level task-incremental learning benchmarks when compared with state-of-the-art parameter isolation-based methods. The contributions are summarized as follows:

- We identify the efficiency bottleneck in parameter isolation-based continual learning induced by capacity expansion, and highlight computational efficiency as a crucial dimension of performance for the parameter isolation-based paradigm.
- We propose a novel biological neurons-inspired GNN framework, which possesses architectural superiority that leverages the task-specific context for efficient subnetwork construction and decouples task-specific subnetworks for capacity-agnostic efficient training and inference.
- We provide a theoretical justification for the proposed architecture and introduce a Fisher information-based regularization to promote effective parameter reuse for greater model scalability for the ongoing task stream.

5.2 Related Works

In this section, we briefly introduce the background of continual graph learning and recent parameter-isolation approaches on graphs. We then review the dendritic neuron model and redefine its structure into a generalized formulation, which underpins the proposed CAGNN framework. Finally, we formalize the task-incremental continual learning problem for the parameter-isolation paradigm. Details of the dendritic neuron model are provided in Section 2.2.3.3.

5.2.1 Continual Graph Learning

Beyond sequential data streams themselves, learning tasks in many real-world applications often unfold in a sequential manner. Instead of being fully accessible and

jointly trainable at the beginning, tasks are usually revealed to the model one by one as new data, new categories, or new scenarios emerge over time. This learning process requires a model to continuously acquire new knowledge from incoming tasks while preserving the knowledge learned from previous ones. Continual Learning (Ring, 1994; L. Wang et al., 2024), also known as Incremental Learning, has emerged as a crucial research field to tackle this scenario. A central challenge in Continual Learning is catastrophic forgetting, where a model continually updates its parameters to adapt to new tasks but unintentionally overwrites previously acquired knowledge. As a consequence, the model may achieve good performance on the current task while suffering from severe performance degradation on earlier tasks (French, 1999; Kirkpatrick et al., 2017). This issue becomes particularly important when old data are no longer accessible due to storage, privacy, or practical constraints, making it difficult to simply retrain the model on all historical tasks.

While most existing studies in Continual Learning focus on independent data such as images and text, the same learning paradigm naturally extends to relational data, represented by graphs. Graph-structured data are inherently dynamic in many real-world scenarios, where new types of nodes, associated edges, or even entire graphs may emerge over time. Therefore, graph learning models are also expected to adapt to newly arriving graph tasks without forgetting previously learned structural and semantic knowledge, giving rise to the paradigm of Continual Graph Learning. For instance, in social or citation networks, entities may establish or cease relationships with existing or new entities as the network evolves (J. Liu et al., 2019; K. Wang et al., 2020). Similarly, in molecular databases, new types of chemical compounds are gradually collected as experimental techniques advance, and novel structures are identified (Huang et al., 2014; Z. Wu et al., 2018). These scenarios highlight the necessity of developing continual learning methods that can effectively handle graph-structured data.

5.2.2 Parameter-Isolation Paradigm on Graphs

Analogous to the principal research streams of Continual Learning, Continual Graph Learning approaches also fall into three broad categories, which are Regularization-based methods, Replay-based methods, and Parameter isolation-based methods, respectively (De Lange et al., 2021; L. Wang et al., 2024; X. Zhang, Song, & Tao, 2024; Febrinanto et al., 2023). Although substantial advances have emerged in recent years (H. Liu et al., 2021; F. Zhou & Cao, 2021; Cui et al., 2023; X. Zhang, Song, Chen, & Tao, 2024; P. Zhang et al., 2023), parameter isolation-based methods on graphs remain significantly underexplored, with far fewer studies in the field compared to the other two (X. Zhang, Song, & Tao, 2024; Ko et al., 2025). As the name implies, parameter isolation-based methods tackle catastrophic forgetting by featuring task-specific subnetworks within an over-parameterized global space, where model parameters are isolated for different tasks to different extents. Rather than mitigating catastrophic forgetting, this paradigm guarantees a minimal level of this problem (can be zero percent of forgetting if parameters of previous tasks are completely separated or frozen), showcasing absolute dominance in model performance (Ko et al., 2025).

Nevertheless, there is a trade-off between the effectiveness of models in solving catastrophic forgetting and their space complexity (X. Zhang, Song, & Tao, 2024), and many existing works adopt various strategies on parameter sparsity, capacity, and reusability to pursue models' scalability for future tasks. Specifically, PackNet (Mallya & Lazebnik, 2018) frees capacity by pruning low-magnitude parameters; Winning Subnetworks (WSN) (Kang et al., 2022, 2024) are grounded in the Lottery Ticket Hypothesis (LTH) (Frankle & Carbin, 2019) to identify sparse, potentially overlapping subnetworks through task-specific masks; Parameter Isolation GNN (PI-GNN) (P. Zhang et al., 2023) progressively rectifies stable information and expands networks for subsequent tasks; Hierarchical Prototype Networks (HPNs) (X. Zhang et al., 2022) employ Atomic Feature Extractors (AFEs) to construct, select and

combine multi-level prototypes, enabling implicit parameter reuse from a prototype viewpoint. Hard Attention to the Task (HAT) (Serra et al., 2018) learns task-specific attention masks over output neurons and protects parameters important for previous tasks. More recently, Task-Aware Adaptive Modulation (TAAM) (J. Liu & Zhang, 2026) freezes a shared GNN backbone and assigns each task a lightweight Neural Synapse Modulator (NSM), which performs node-attentive adaptive modulation. However, it relies on adding task-specific modulators on top of a frozen backbone, which isolates new knowledge into separate expert modules but does not explicitly promote fine-grained parameter sharing across tasks, leading to linearly increasing task-specific parameters as the task stream grows. Notably, PI-GNN (P. Zhang et al., 2023) employs a resampling and compression mechanism to mitigate this problem, but this mechanism demands additional training and is not universally applicable to other methods since it was originally designed for time-incremental contexts (dynamic graphs). In task-incremental contexts, which are the main arena for parameter isolation-based methods, PI-GNN is more accurately classified as a hybrid approach.

5.2.3 Dendritic Neuron Model

A neuron's soma itself provides only a limited membrane area for synaptic input, constraining its responsiveness to external stimuli. However, a neuron's dendrites substantially enlarge this receptive surface and actively transform synaptic signals, enabling integration of inputs from diverse sources and modulation of somatic output (Fiala & Harris, 1999; Harnett et al., 2012; Schachter et al., 2010). Many dendritic neuron models have been developed to mimic this information-processing mechanism in real neurons (S. Gao et al., 2018, 2021; Rall, 1962; Todo et al., 2019). According to earlier advances (Grewal et al., 2021; X. Wu et al., 2018; Iyer et al., 2022; R. Li et al., 2025), the dendritic neuron model can be formulated as the fol-

lowing representation:

$$\hat{\boldsymbol{y}} = \left(\boldsymbol{W}_f^\top \boldsymbol{x} + \boldsymbol{b}\right) \odot \sigma\left(\max_{b \in B} \left(\boldsymbol{W}_d \odot \boldsymbol{M}_d\right)^\top \boldsymbol{c}\right) \tag{5.1}$$

The model is a joint of the feedforward and dendritic unit. The former functions as neuronal soma and computes the base response from input $\boldsymbol{x}$. The latter functions as dendrites and perceives the context $\boldsymbol{c}$ to generate branch-wise modulation signals. The random binary mask $\boldsymbol{M}_d$ determines branch b's connection with each dendrite input unit, across B branches in each output unit without replacement, to enforce synaptic connectivity in dendrite weight $\boldsymbol{W}_d$. The strongest branch response of each dendrite output is selected and passed through a sigmoid gate $\sigma(\cdot)$ to modulate the feedforward output via Hadamard product $\odot$.

5.2.4 Problem Definition

We study task-incremental continual graph learning under the parameter-isolation paradigm. Denoting a graph that has a node set $\mathcal{V}$, edge set $\mathcal{E}$, and node features $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \boldsymbol{X}\}$, where N is the number of nodes and F is the number of node features, we consider node-level and graph-level classification in the task-incremental setting with a sequence of class-disjoint tasks $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_T\}$ arriving to the model sequentially, with task identity available during both training and inference. For each task $\mathcal{T}_t$, parameter isolation learns a binary mask $\mathbf{m}_{\mathcal{T}_t}$ to select a task-specific subnetwork from the shared parameter space $\boldsymbol{\theta}$, i.e., $\boldsymbol{\theta}_{\mathcal{T}_t} = \mathbf{m}_{\mathcal{T}_t}(\boldsymbol{\theta})$. After task $\mathcal{T}_t$ is learned, its training data is no longer accessible for the training of the rest of the tasks.

Node Classification $\mathcal{T}_t$, identified by corresponding task id t, consists of a subgraph $\mathcal{G}_t = \{\mathcal{V}_t, \mathcal{E}_t, \boldsymbol{X}_t\}$ extracted from a complete graph $\mathcal{G}$, a label set $\boldsymbol{y}_t = \{y_1, y_2, \ldots, y_{c_t}\}$ corresponding to labled nodes $\mathcal{V}_t' \subset \mathcal{V}_t$. Given a sequence of tasks $\mathcal{T}$, a graph operator $f_\theta : \mathcal{G} \mapsto \mathbb{R}^{N \times C}$, parameterized by $\boldsymbol{\theta}$, is trained sequentially to

estimate the sample class, by solving the following optimization procedure:

$$\boldsymbol{\theta}^*_{\mathcal{T}_t}, \mathbf{m}^*_{\mathcal{T}_t} = \underset{\boldsymbol{\theta}, \mathbf{m}_{\mathcal{T}_t}}{\arg\min} \frac{1}{|\mathcal{V}'_t|} \sum_{v \in \mathcal{V}'_t} \mathcal{L}(f(\mathcal{G}_t; \mathbf{m}_{\mathcal{T}_t}(\boldsymbol{\theta}))[v], y_v) \tag{5.2}$$

Graph Classification $\mathcal{T}_t$ consists of a labeled graph set $G_t = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_n\}$, where each $\mathcal{G}_i$ in G_t is a complete graph, a label set $\boldsymbol{y}_t = \{y_1, y_2, \ldots, y_{c_t}\}$, and corresponding task id t. For a graph operator $f_\theta : \mathcal{G} \mapsto \mathbb{R}^{1\times C}$, the objective is to solve the following optimization problem:

$$\boldsymbol{\theta}^*_{\mathcal{T}_t}, \mathbf{m}^*_{\mathcal{T}_t} = \underset{\boldsymbol{\theta}, \mathbf{m}_{\mathcal{T}_t}}{\arg\min} \frac{1}{|G_t|} \sum_{\mathcal{G}_i \in G_t} \mathcal{L}(f(\mathcal{G}_i; \mathbf{m}_{\mathcal{T}_t}(\boldsymbol{\theta})), y_i) \tag{5.3}$$

5.3 Methodology

In this section, we present the proposed Capacity-Agnostic Graph Neural Network (CAGNN) and its efficiency-oriented design for continual graph learning. Simulating real neurons that perceive external synaptic stimulation and modulate somatic signals through spike activation via their distal dendrites, CAGNN adopts an over-parameterized Graph Convolutional Network (GCN) (Kipf & Welling, 2017) as the feedforward unit and employs the Context-aware Hierarchical Masking (CHM) that activates or inhibits the model's weights to construct task-specific subnetworks. As shown in Figure 5.2, given the task-specific context extracted when a new task $\mathcal{T}_t$ arrives, the CHM module sequentially performs masking from global to local to compress an over-parameterized parameter space into a dense subspace, learning the subnetwork's structure. For $t > 1$, a Fisher information-based regularization further refines the subnetwork structure for effective parameter reuse. After a brief warm-up, CHM is deactivated, and the selected subnetwork structure is frozen for the remaining training. Afterward, CAGNN continues training as a lightweight conventional GCN within the selected subspace. The overall pipeline of CAGNN is provided in Algorithm 2.

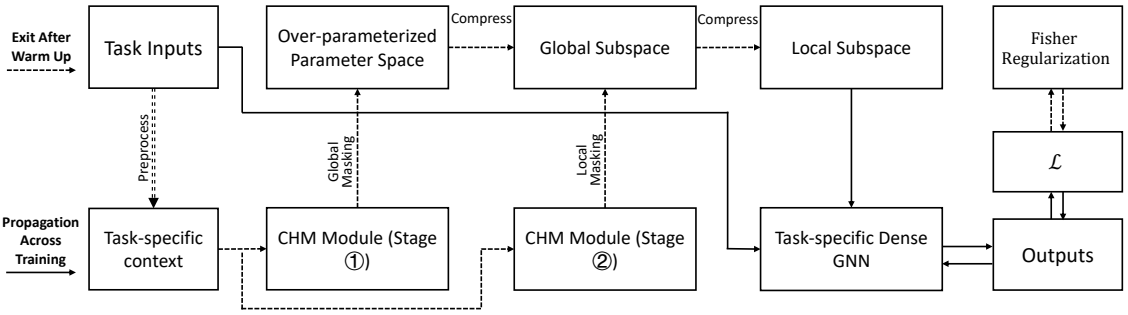


Figure 5.2: A brief demonstration of CAGNN, consisting of a graph convolutional feedforward part and a masking part. The graph convolutional feedforward part operates across the entire training to learn representations, while the masking part searches subnetwork structures during the warm-up phase. When the warm-up phase is completed, the subnetwork structure is fixed for the corresponding task.

5.3.1 Context-Aware Hierarchical Masking

The Context-aware Hierarchical Masking (CHM) module uses contextual information as prior knowledge to construct structured task-specific subnetworks through two stages, which are dendritic masking and low-rank masking. Together, they form a hierarchical scheme that enables fine-grained parameter selection.

5.3.1.1 Dendritic Masking (Stage 1)

Unlike dense activation in conventional deep neural networks, dendritic activation is characterized by sparse synaptic connectivity, where each dendritic unit randomly forms branch connections from the input without replacement. This branch-wise activation induces *intragroup sparsity* (X. Wu et al., 2018), which was originally introduced to improve data locality and inference efficiency compared to irregular element-wise sparsity. We further observe that intragroup sparsity offers an additional benefit for subnetwork construction, which is capable of capturing richer patterns from contextual inputs by uniformly involving all input channels. By further exploiting the dendritic unit's structural regularity, we propose dendritic masking, which naturally supports network compression that facilitates efficient training and inference.

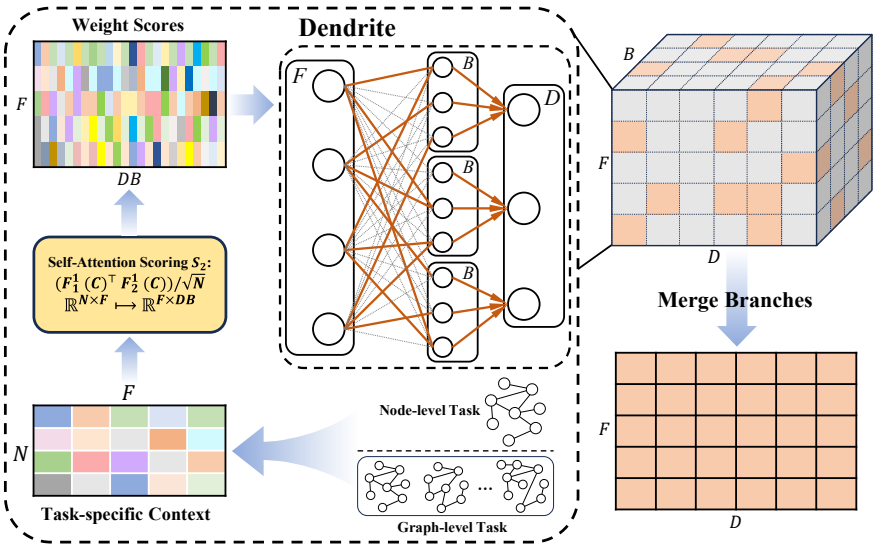


Figure 5.3: Illustration of Dendritic Masking.

The dendritic masking for task $\mathcal{T}_t$ is visually shown in Figure 5.3. To be specific, we initialize a weight $\boldsymbol{W}_0$ with a dendritic shape $F \times D \times B$, where F, D, and B, respectively, are the input dimension, the intermediate dimension, and the number of branches. Given $\mathcal{T}_t$'s context matrix $\boldsymbol{C}$ of shape $N \times F$ as input, where N is the number of samples, we compute the weight scores $\mathcal{S}_{global}$ by self-attention:

$$\mathcal{S}_w = \frac{F_1(\boldsymbol{C})^\top F_2(\boldsymbol{C})}{\sqrt{N}} \in \mathbb{R}^{F\times DB} \tag{5.4}$$

where $F_1(\cdot) : \boldsymbol{C} \mapsto \mathbb{R}^{N\times F}$ and $F_2(\cdot) : \boldsymbol{C} \mapsto \mathbb{R}^{N\times DB}$ are two linear projections. Afterward, $\mathcal{S}_{global}$ is reshaped into a 3D tensor of the branch connection scores as $\mathcal{S} = [\mathcal{S}(f,d,b)] \in \mathbb{R}^{F\times D\times B}$ where $\mathcal{S}(f,d,b)$ denotes the score at position (f,d,b) of $\boldsymbol{W}_0$ ($1 \le f \le F$, $1 \le d \le D$ and $1 \le b \le B$).

Given the 3D tensor of branch connection scores $\mathcal{S}$, we follow equation (5.1) and define our dendritic activation $\Phi(\cdot)$ as taking weights by the maximal score values along the third tensor dimension, i.e., gathering the most activated branch for each position (f,d), such that:

$$\Phi(\boldsymbol{W}_0|\mathcal{S}) = [\max_b \boldsymbol{W}_0|\mathcal{S}(f,d,b)] \in \mathbb{R}^{F\times D}. \tag{5.5}$$

This map $\Phi : \mathbb{R}^{F\times D\times B} \mapsto \mathbb{R}^{F\times D}$, however, is nondifferentiable due to the max-pooling. To enable end-to-end optimization, we decompose this activation into a masking operation and approximate it with the straight-through estimation trick (STE) (Bengio et al., 2013). For this purpose, we introduce the dendritic mask $\boldsymbol{M}_{dm} = [\boldsymbol{M}_{dm}(f, d, b)] \in \{0, 1\}^{F\times D\times B}$ for each input-intermediate pair (f, d) in the following way:

$$\boldsymbol{M}_{dm}(f, d, b) = \begin{cases} 1 & \text{if } b = \tilde{b} \\ 0 & \text{if } b \neq \tilde{b} \end{cases} \tag{5.6}$$

where $\tilde{b} = \arg\max_b\{\mathcal{S}(f, d, b)\}$. $\boldsymbol{M}_{dm}$ only activates the weight with the highest score learned from $\boldsymbol{C}$ among the B branch connections of each input, resembling sparse synaptic connectivity.

In practice, we sample the binary dendritic mask $\boldsymbol{M}_{dm}$ along the branch dimension by employing Gumbel-Softmax (Jang et al., 2016; Maddison et al., 2016) with STE, which approximates discrete masking in a differentiable manner. STE guarantees the gradient flow during propagation, thus rendering $\mathcal{S}_{global}$, together with the model parameter $\boldsymbol{W}_0$, optimizable. Specifically:

$$\boldsymbol{M}_{dm} = \text{Softmax}_b\left(\frac{\log(\mathcal{S}) - \log(1 - \mathcal{S}) + g}{\tau}\right) \tag{5.7}$$

where $g \in \mathbb{R}^{F\times D\times B}$ denotes Gumbel noise sampled from Gumbel$(0, 1)$, and τ controls the sharpness of Gumbel-Softmax that the output approaches one-hot vectors as $\tau \to 0$ and a uniform distribution as $\tau \to \infty$.

To compress model's parameters $\boldsymbol{W}_0$ into a dense subspace $\boldsymbol{W}_{global}$, we gather selected activated branches by applying $\boldsymbol{M}_{dm}$ on $\boldsymbol{W}_0$ using the Hadamard product, with a summation trick on the outputs across branches of each dendritic unit to preserve the gradient of each masking score. Ultimately, we have the following relation:

$$\boldsymbol{W}_{global} = \Phi\left(\boldsymbol{W}_0|\mathcal{S}\right) = \text{sum}_3(\boldsymbol{M}_{dm} \odot \boldsymbol{W}_0) \tag{5.8}$$

where $\text{sum}_3(\cdot)$ denotes sum pooling along the branch dimension B. By design, dendritic masking thereby partitions task-relevant parameters into branch-wise regions based on task-specific context, reducing cross-task interference while encouraging a global-level parameter reuse across tasks.

5.3.1.2 Low-Rank Masking (Stage 2)

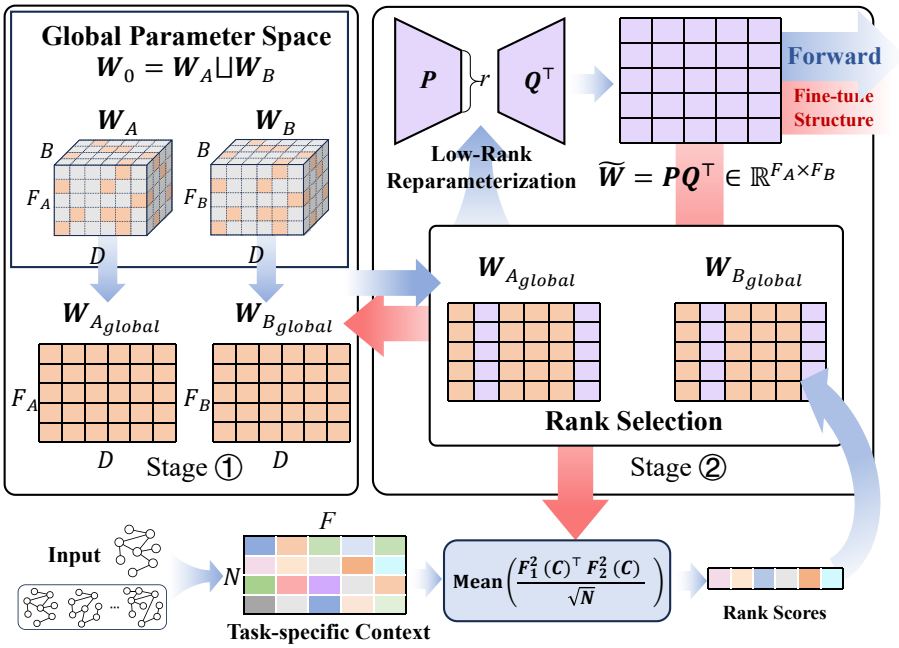


Figure 5.4: Illustration of Low-rank Masking.

To improve the model's scalability, task-specific subnetworks should be highly sparse within the global parameter space. Yet dendritic masking alone only yields $1/B\%$ sparsity, which constrains the shape over the branch dimension and subnetworks' flexibility. However, under the same model's capacity condition, simply increasing B narrows the hidden width and may incur an information bottleneck, especially in GNNs with the over-squashing problem (Alon & Yahav, 2021). Keeping in mind the objective of computational efficiency, to pursue a further high granularity-level of parameter masking and sharing, we therefore keep B moderate and introduce low-rank masking as the second stage for finer-grained sparsity, as shown in Figure 5.4.

Low-rank methods, such as the low-rank approximation and the low-rank adaptation, have been widely studied in large-scale models for efficiency, and have been proven that low intrinsic-rank weights usually suffice or even bring better generalization than full-rank counterparts do (Denton et al., 2014; E. J. Hu et al., 2022; C. Li et al., 2018; Aghajanyan et al., 2021; Gunasekar et al., 2017). Our low-rank masking begins with an over-parameterized parameter space $\boldsymbol{W}_0$ separated into two complementary spaces $\boldsymbol{W}_A \in \mathbb{R}^{F_A \times D \times B}$ and $\boldsymbol{W}_B \in \mathbb{R}^{F_B \times D \times B}$ and derive two dense subspaces $\boldsymbol{W}_{A_{global}} \in \mathbb{R}^{F_A \times D}$ and $\boldsymbol{W}_{B_{global}} \in \mathbb{R}^{F_B \times D}$ derived by the dendritic masking. Subsequently, we use self-attention scoring to obtain a score $\mathcal{S}_{local} \in \mathbb{R}^{F \times D}$, which is averaged to rank scores $s_{local} \in \mathbb{R}^{1 \times D}$ for rank selection. Different from neuron-wise masking, low-rank masking selects top-r columns of $\boldsymbol{W}_{A_{global}}$ and $\boldsymbol{W}_{B_{global}}$, and yields two low-rank factors $\boldsymbol{P} \in \mathbb{R}^{F_A \times r}$ and $\boldsymbol{Q} \in \mathbb{R}^{F_B \times r}$, forming a local subspace. Note that r is determined by the target sparsity. Specifically:

$$\boldsymbol{P} = \boldsymbol{W}_{A_{global};[:,\mathcal{I}]}, \quad \boldsymbol{Q} = \boldsymbol{W}_{B_{global};[:,\mathcal{I}]} \tag{5.9}$$

where $\mathcal{I} = \mathrm{Top}_r(s_{local})$ are selected columns. Ultimately, we form the forward pass weight $\tilde{\boldsymbol{W}}$ through low-rank reparameterization:

$$\tilde{\boldsymbol{W}} = \boldsymbol{P}\boldsymbol{Q}^\top \tag{5.10}$$

and express the forward pass of the layer l of CAGNN as:

$$\boldsymbol{H}^{(l+1)} = \sigma(\tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-1/2}\boldsymbol{H}^{(l)}\tilde{\boldsymbol{W}}^{(l)}) \tag{5.11}$$

where $\boldsymbol{H}$ denotes the node representation matrix, initialized as $\boldsymbol{X}$ in the first layer, and $\hat{\boldsymbol{A}} = \tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-1/2}$ denotes the normalized adjacency matrix.

Rank selection truncates the gradient flow and introduces differentiability issues. Directly applying STE still encounters convergence difficulty on the subnetwork structure because the selection operation is inherently non-smooth. To preserve

the adaptivity of low-rank masking, we reformulate rank selection as dendritic sparse synaptic connectivity and create a binary dendritic mask $\boldsymbol{M}_{\mathcal{I}}$ as a differentiable proxy for $\mathcal{I}$. Specifically, given r columns to be selected, we reshape $s_{local} \in \mathbb{R}^{1\times D}$ into $\mathbb{R}^{1\times r\times k}$, where $k = D/r$. Following Equation (5.7), we obtain $\boldsymbol{M}_{\mathcal{I}} \in \mathbb{R}^{1\times r\times k}$ via Gumbel sampling, which partitions s_{local} into k groups and performs group-wise column activation:

$$\boldsymbol{M}_{\mathcal{I}} = \mathrm{Softmax}_k\left(\frac{\log(s_{local}) - \log(1 - s_{local}) + g}{\tau}\right) \tag{5.12}$$

Accordingly, we reshape $\boldsymbol{W}_{A_{global}} \in \mathbb{R}^{F_A\times D}$ and $\boldsymbol{W}_{B_{global}} \in \mathbb{R}^{F_B\times D}$ into $\mathbb{R}^{F_A\times r\times k}$ and $\mathbb{R}^{F_B\times r\times k}$, respectively, and formulate $\boldsymbol{P}_{soft}$ and $\boldsymbol{Q}_{soft}$ via the summation trick as Equation (5.8), preserving the gradient flow to each masking score:

$$\begin{aligned}\boldsymbol{P}_{soft} &= \mathrm{sum}_3(\boldsymbol{M}_{\mathcal{I}} \odot \boldsymbol{W}_{A_{global}}) \in \mathbb{R}^{F_A\times r}\\ \boldsymbol{Q}_{soft} &= \mathrm{sum}_3(\boldsymbol{M}_{\mathcal{I}} \odot \boldsymbol{W}_{B_{global}}) \in \mathbb{R}^{F_B\times r}\end{aligned} \tag{5.13}$$

Note that in practice, we temporarily increase r until $r \mid D$, and prune the selected columns back to the target rank after warm-up. During warm-up, the forward pass weight is consistently constructed by $\tilde{\boldsymbol{W}}_{soft} = \boldsymbol{P}_{soft}\boldsymbol{Q}_{soft}^{\top}$, which acts as a differentiable proxy of $\tilde{\boldsymbol{W}}$ for subnetwork structure learning. The mask $\boldsymbol{M}_{\mathcal{I}}$ is then mapped back to $\boldsymbol{W}_0$, also in a differentiable manner, to obtain the global task-specific mask $\boldsymbol{M}_{\mathrm{CHM}}$ for Fisher regularization.

Integrating the above steps into the CHM operation $\mathbb{M}_{\mathrm{CHM}}$, the warm-up forward pass of CAGNN incorporating dendritic activation into the feedforward unit is formulated as:

$$\boldsymbol{H}^{(l+1)} = \sigma(\hat{\boldsymbol{A}}\boldsymbol{H}^{(l)}\mathbb{M}_{\mathrm{CHM}}(\boldsymbol{W}_0, \boldsymbol{C})) \tag{5.14}$$

When the warm-up is completed, CHM fixes the subnetwork by recording $\boldsymbol{M}_{dm}$ and $\mathcal{I}$ to reconstruct $\tilde{\boldsymbol{W}}$ from $\boldsymbol{W}_0$, forming a general depiction of the subnetwork's structure. Due to the hierarchical design, these masks can be stored as indices and

used for reconstruction via indexing, requiring much less memory than dense masks. Afterward, $\mathbb{M}_{\text{CHM}}(\boldsymbol{W}_0, \boldsymbol{C})$ is fixed as $\tilde{\boldsymbol{W}}$ following Equations (5.8)–(5.10), allowing CAGNN to train and infer equivalently on a dense, lightweight conventional GCN as in Equation (5.11). Throughout the training, an accumulated mask $\boldsymbol{M}_{accum}$ freezes the parameters used by previous tasks and is updated with the final $\boldsymbol{M}_{\text{CHM}}$ after training each task.

5.3.2 Task-specific Context

Task-specific context is crucial for constructing task-specific subnetworks and is expected to encapsulate compact yet informative representations that capture both the distinguishability and transferability of knowledge across tasks, thereby supporting the model to maintain sufficient plasticity and scalability for new tasks.

Node Classification We generate task-specific context $\boldsymbol{C}_{node}$ of task $\mathcal{T}_t$ using one-hop aggregation of subgraph $\mathcal{G}_t$:

$$\boldsymbol{C}_{node} = \hat{\boldsymbol{A}}_t \boldsymbol{X}_t \tag{5.15}$$

Equation (5.15) essentially corresponds to a low-pass filter that smooths the node features while relatively depicting $\mathcal{G}_t$ in raw (Nt & Maehara, 2019; F. Wu et al., 2019).

Graph Classification We consider that the Rayleigh Quotient distribution (Chan et al., 2011) reveals class-consistent patterns across graphs with varying sample sizes (Dong et al., 2024), making it appropriate for graph-level continual learning scenarios. Therefore, we construct graph-level context $\boldsymbol{C}_{graph}$ by evaluating the Rayleigh Quotient $\mathcal{R}_i$ for each graph $\mathcal{G}_i \in G_t$:

$$\boldsymbol{C}_{graph} = \{\mathcal{R}_i\}_{i=1}^{n} \in \mathbb{R}^{n \times F} \tag{5.16}$$

where $\mathcal{R}_i = \text{diag}\left((\boldsymbol{X}_i^\top \boldsymbol{L}_i \boldsymbol{X}_i)/(\boldsymbol{X}_i^\top \boldsymbol{X}_i)\right)$, and $\boldsymbol{L}_i = \boldsymbol{I} - \hat{\boldsymbol{A}}_i$ is the Laplacian matrix of $\mathcal{G}_i$.

5.3.3 Theoretical Analysis

We provide theoretical justification for the robustness of our proposed low-rank masking to cross-task bias brought by parameter reuse, and propose a Fisher regularization term to promote effective parameter reuse.

5.3.3.1 Effective Parameter Reuse

Notation. Considering a certain level of parameter reuse in the continual learning problem, we write the masked forward-pass weight as $\boldsymbol{W} = \boldsymbol{W}^f + \boldsymbol{W}^t \in \mathbb{R}^{F\times D}$, where $\boldsymbol{W}^f$ denotes the frozen reused weights from previous tasks and $\boldsymbol{W}^t$ denotes the trainable weights for the current task. Given the unknown task-optimal weight $\boldsymbol{W}^*$, we define the reuse-induced bias as $\Delta := \boldsymbol{W}^* - \boldsymbol{W}^f$.

Proposition 5.3.1 (Reuse Bias under Neuron-wise Selection/Masking)**.** Let columns of the weight be partitioned into a frozen set $\mathcal{I}_f$ and a trainable set $\mathcal{I}_t = \mathcal{I}\backslash\mathcal{I}_f$. The minimum parameter bias from the optimal weight $\boldsymbol{W}^*$ brought by neuron-wise masking with pure column-wise reuse is limited by an irreducible bias from the not learnable portion $\boldsymbol{W}^f = \boldsymbol{W}_{:,\mathcal{I}_f}$, such that:

$$\inf_{\boldsymbol{W}_{:,\mathcal{I}_t}} \|\boldsymbol{W}^* - \boldsymbol{W}\|_F = \left\|\Delta_{:,\mathcal{I}_f}\right\|_F \tag{5.17}$$

Proof. For neuron-wise masking, we have $\boldsymbol{W}^t_{:,\mathcal{I}_f} = 0$, and thus:

$$\begin{aligned} ||\boldsymbol{W}^* - \boldsymbol{W}||_F &= ||\Delta - \boldsymbol{W}^t||_F \\ &= ||[\Delta_{:,\mathcal{I}_t} - \boldsymbol{W}^t_{:,\mathcal{I}_t} \quad \Delta_{:,\mathcal{I}_f} - 0]||_F \\ &= \sqrt{||\Delta_{:,\mathcal{I}_t} - \boldsymbol{W}^t_{:,\mathcal{I}_t}||_F^2 + ||\Delta_{:,\mathcal{I}_f}||_F^2} \end{aligned} \tag{5.18}$$

Assuming the trainable set perfectly fits such that $\Delta_{:,\mathcal{I}_t} - \boldsymbol{W}^t_{:,\mathcal{I}_t} = 0$, the neuron-wise

parameter reuse yields an irreducible bias $||\Delta_{:,\mathcal{I}_f}||_F$. This finishes the proof. □

Theorem 5.3.1 (Reuse Bias under Low-rank Masking)**.** Let $\boldsymbol{W} = \boldsymbol{P}\boldsymbol{Q}^\top$ be with $\text{rank}(\boldsymbol{W}) \leq r$. The columns of $\boldsymbol{P}$ and $\boldsymbol{Q}$ are partitioned into a frozen set $\mathcal{I}_f$ and a trainable set $\mathcal{I}_t = \mathcal{I} \backslash \mathcal{I}_f$, where $r_t = |\mathcal{I}_t| < r$ denotes the effective trainable rank. Under low-rank masking with column-wise reuse, the minimum reuse-induced bias depends only on the effective trainable rank r_t:

$$\inf_{\text{rank}(\boldsymbol{W}^t)} \|\boldsymbol{W}^* - \boldsymbol{W}\|_F = \left\|\Delta - \Delta_{(r_t)}\right\|_F = \left(\sum_{i>r_t}^{p} \sigma_i^2\right)^{\frac{1}{2}} \tag{5.19}$$

where the infimum is taken over the trainable component $\boldsymbol{W}^t$ with rank at most r_t, $\Delta = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ is the SVD of the reuse bias with singular values $\boldsymbol{\Sigma} = \text{diag}(\sigma_1 \geq \cdots \geq \sigma_p \geq 0)$, and $\Delta_{(r_t)} := \boldsymbol{U}\boldsymbol{\Sigma}_{(r_t)}\boldsymbol{V}^\top$ is its best rank-r_t approximation.

To prove the above theorem, we introduce two well-known facts as Lemmas without proof.

Lemma 5.3.1 (Unitary Invariance of the Frobenius Norm)**.** For any orthogonal $\boldsymbol{U}$, $\boldsymbol{V}$ and a matrix $\boldsymbol{P}$: $||\boldsymbol{U}\boldsymbol{P}\boldsymbol{V}||_F = ||\boldsymbol{P}||_F$.

Lemma 5.3.2 (von Neumann's Trace Inequality)**.** For matrices $\boldsymbol{P}$ and $\boldsymbol{P}$ with singular values $\sigma_{\boldsymbol{P}}$ and $\sigma_{\boldsymbol{Q}}$ sorted in decreasing order: $\text{trace}(\boldsymbol{P}^\top\boldsymbol{Q}) \leq \sum_i \sigma_{\boldsymbol{P},i}\sigma_{\boldsymbol{Q},i}$, with equality if the singular vectors of $\boldsymbol{P}$ and $\boldsymbol{Q}$ are aligned.

Proof of Theorem 5.3.1. Without loss of generality, let $\boldsymbol{P} = [\boldsymbol{P}_f \ \boldsymbol{P}_t]$ and $\boldsymbol{Q} = [\boldsymbol{Q}_f \ \boldsymbol{Q}_t]$, where $\boldsymbol{P}_f = \boldsymbol{P}_{:,\mathcal{I}_f}$, $\boldsymbol{Q}_f = \boldsymbol{Q}_{:,\mathcal{I}_f}$, $\boldsymbol{P}_t = \boldsymbol{P}_{:,\mathcal{I}_t}$, and $\boldsymbol{Q}_t = \boldsymbol{Q}_{:,\mathcal{I}_t}$. We can rewrite $\boldsymbol{W} = \boldsymbol{P}\boldsymbol{Q}^\top = \boldsymbol{P}_f\boldsymbol{Q}_f^\top + \boldsymbol{P}_t\boldsymbol{Q}_t^\top$, where $\boldsymbol{P}_f\boldsymbol{Q}_f^\top = \boldsymbol{W}^f$ and $\boldsymbol{P}_t\boldsymbol{Q}_t^\top = \boldsymbol{W}^t$ with

rank r_t. Denoting $\Delta = U\Sigma V^\top$, we define $\mathcal{W}^t := U^\top \boldsymbol{W}^t V$. By Lemma 5.3.1:

$$\begin{aligned} ||\boldsymbol{W}^* - \boldsymbol{W}||_F &= ||\Delta - \boldsymbol{W}^t||_F \\ &= ||U\Sigma V^\top - \boldsymbol{W}^t||_F \\ &= ||U^\top (U\Sigma V^\top - \boldsymbol{W}^t)V||_F \\ &= ||\Sigma - \mathcal{W}^t||_F \end{aligned} \tag{5.20}$$

Let $\sigma_{\mathcal{W}^t}$ be the singular values of $\mathcal{W}_t$, we have $||\Sigma||_F^2 = \text{trace}(\Sigma^\top \Sigma) = \sum_{i=1}^r \sigma_i^2$, and $||\mathcal{W}^t||_F^2 = \sum_{i=1}^r \sigma_{\mathcal{W}^t,i}^2$. Then, by Lemma 5.3.2:

$$\text{trace}(\Sigma^\top \mathcal{W}^t) \leq \sum_{i=1}^{p} \sigma_i \sigma_{\mathcal{W}^t,i} \tag{5.21}$$

we can derive:

$$\begin{aligned} ||\Sigma - \mathcal{W}^t||_F^2 &= ||\Sigma||_F^2 + ||\mathcal{W}^t||_F^2 - 2\ \text{trace}(\Sigma^\top \mathcal{W}^t) \\ &\geq \sum_{i=1}^{p} (\sigma_i^2 + \sigma_{\mathcal{W}^t,i}^2 - 2\sigma_i \sigma_{\mathcal{W}^t,i}) = \sum_{i=1}^{p} (\sigma_i - \sigma_{\mathcal{W}^t,i})^2 \end{aligned} \tag{5.22}$$

Because rank($\mathcal{W}^t$)= r_t, and $\sigma_{\mathcal{W}^t,i} = 0$ for $i > r_t$, hence:

$$||\Sigma - \mathcal{W}^t||_F^2 \geq \sum_{i=1}^{r_t} (\sigma_i - \sigma_{\mathcal{W}^t,i})^2 + \sum_{i>r_t}^{p} \sigma_i^2 \tag{5.23}$$

Assuming the trainable set perfectly fits, and for each $i \leq r_t$, the former term is minimized at $\sigma_{\mathcal{W}^t,i} = \sigma_i$, we then have a infimum:

$$\inf\ ||\Sigma - \mathcal{W}^t||_F^2 = ||\Delta - \boldsymbol{W}^t||_F^2 = \sum_{i>r_t}^{p} \sigma_i^2 \tag{5.24}$$

Combining the optimal $\boldsymbol{W}^{t*} = U\Sigma_{(r_t)}V^\top = \Delta_{(r_t)}$. This completes the proof. □

With the same number of reused and trainable parameters, low-rank masking achieves a smaller minimum reuse bias than neuron-wise masking when Δ is well

captured by a rank-r_t subspace. In particular, if $\text{rank}(\Delta) \leq r_t$ and $\Delta_{:,\mathcal{I}_f} \neq 0$, the error, which is contributed by tail singular values of Δ, vanishes (i.e., $\sum_{i>r_t}^{p} \sigma_i^2 \approx 0$), whereas neuron-wise masking still retains the irreducible bias $\|\Delta_{:,\mathcal{I}_f}\|_F$. Thus, between two masking schemes designed for efficiency, low-rank masking offers greater flexibility in absorbing cross-task bias under the same parameter budget.

Remark 5.3.1. Although the upper-level dendritic masking may induce element-wise reuse within $\boldsymbol{P}$ and $\boldsymbol{Q}$, the resulting weight can still be decomposed as $\boldsymbol{W} = \boldsymbol{P}\boldsymbol{Q}^\top = \sum_{i=1}^{r} \boldsymbol{p}_i \boldsymbol{q}_i^\top$. In this view, each column pair $(\boldsymbol{p}_i, \boldsymbol{q}_i)$ defines a rank-one component that implicitly absorbs the effect of element-wise frozen parameters. These frozen entries can be interpreted as linear constraints imposed on each component $\boldsymbol{p}_i \boldsymbol{q}_i^\top$, where the fully non-trainable contributions are incorporated into the frozen part $\boldsymbol{W}^f$, while the remaining trainable entries provide a constrained correction. Hence, our theoretical analysis serves as a column-wise abstraction of the two-stage masking scheme.

5.3.3.2 Fisher Regularization

Because the task-optimal solution is unknown a priori, quantifying whether reused parameters facilitate knowledge transfer is usually infeasible without a fully flexible reference run. We therefore establish the following relationship between parameter reuse and model loss:

Proposition 5.3.2. Given the optimal parameter θ^* for task $\mathcal{T}_t$, the excess loss $\Delta\mathcal{L} = \mathcal{L}(\theta) - \mathcal{L}(\theta^*)$ can be expressed by:

$$\Delta(\mathcal{L}) \approx \frac{1}{2} \Delta\theta^\top \boldsymbol{H}(\theta^*) \Delta\theta + o(||\Delta\theta||^2) \tag{5.25}$$

where $\boldsymbol{H}(\theta^*)$ denotes the Hessian matrix of the loss at θ^*.

Proof. Denoting $\Delta\theta = \theta - \theta^*$, Equation (5.25) can be derived using the second-order Taylor expansion on $\mathcal{L}(\theta^* + \Delta\theta)$ at the point θ^*, with $\nabla\mathcal{L}(\theta^*) = 0$. □

Corollary 5.3.1. The variation is governed by $\boldsymbol{\delta}$ since $\boldsymbol{H}(\theta^*)$ is a fixed constant. If r_t is sufficient, $\Delta\theta = 0$, within which the reused part θ_f itself is already optimum. Otherwise, any unavoidable deviation $\boldsymbol{\delta}_f = \theta_f - \theta_f^*$ leads to $\Delta\theta \neq 0$.

To mitigate the impact of reuse when r_t is insufficient, we utilize the diagonal of the Fisher information matrix $\mathcal{F}$ (Aich, 2021; Kirkpatrick et al., 2017) as a proxy of δ_f and introduce a regularization term by minimizing:

$$\boldsymbol{M}^*_{\text{CHM}} = \arg\min_{\boldsymbol{M}_{\text{CHM}}} \frac{1}{|\theta_f|} \sum_{i=1}^{|\theta|} F_{\theta_{f,i}} \tag{5.26}$$

where $\mathcal{F}_{\theta_f} = (\boldsymbol{M}_{\text{CHM}} \odot \boldsymbol{M}_{accum;\mathcal{T}_{t-1}}) \odot \mathcal{F}_\theta$, and $\boldsymbol{M}_{\text{CHM}} \odot \boldsymbol{M}_{accum;\mathcal{T}_{t-1}}$ indicates resued parameters. Minimizing this term encourages CHM to reuse parameters with lower Fisher values, i.e., parameters which are close to optimum θ_f^* or to which the current-task loss is less sensitive. Equivalently, it forces $\theta_f \to \theta_f^*$ and thereby reduces the deviation from θ_f^*.

5.3.4 Complexity Analysis

We analyze the time complexity of CAGNN by comparing it with WSN (Kang et al., 2024) under the same parameter budget, which provides a fair and transparent assessment of computational efficiency. Let N, $|\mathcal{E}|$, F, and B denote the number of nodes, number of edges, hidden dimension, and number of branches, respectively. In this context, CAGNN has total capacity $2BF^2$, and the selected rank is given by $r = (1-p)BF$, where p is a hyperparameter controlling the parameter sparsity for each subnetwork.

WSN Under the matched capacity setting, the hidden dimension of WSN becomes $F_w = \sqrt{2B}F$. Assuming WSN employs two-phase training (Note that in WSN's original setting, it optimizes masks throughout training), during warm-up, generating a subnetwork requires top-k selection over F_w^2 mask scores, leading to an additional complexity of approximately $\mathcal{O}(F_w^2)$. Together with the graph convolution

Algorithm 2: CAGNN-CHM

Input: Tasks $\{\mathcal{T}_t\}_{t=1}^{T}$, branch number B, model parameters $\boldsymbol{W}_0 \in \mathbb{R}^{(F_A+F_B)\times DB}$, subnetwork sparsity p, empty accumulated mask $\boldsymbol{M}_{accum}$, Fisher factor λ, training iterations E, warm-up iterations E_w ($E_w \ll E$)

Initialize $\boldsymbol{W}_0$ into two complementary spaces $\boldsymbol{W}_A \in \mathbb{R}^{F_A\times D\times B}$ and $\boldsymbol{W}_B \in \mathbb{R}^{F_B\times D\times B}$;
for task $t = 1, \ldots, T$ **do**
    Construct task-specific context $\boldsymbol{C}$ by (5.15) or (5.16), and initialize mask learner to calculate $\mathcal{S}_{global}$ and $\mathcal{S}_{local}$;
    for $e = 1$ **to** E **do**
        while $e = 1$ **to** E_w **do**
            Stage 1
            Calculate branch connection scores $\mathcal{S}$ using self-attention scoring $\mathcal{S}_w$;
            Generate dendritic mask $\boldsymbol{M}_{dm}$ following (5.7);
            Apply (5.8) to extract global subspaces: $\boldsymbol{W}_{A_{global}}$ and $\boldsymbol{W}_{B_{global}}$;
            Stage 2
            Calculate rank scores s_{local} using self-attention scoring $\mathcal{S}_{local}$;
            Calculate r according to p;
            Reshape $s_{local} \in \mathbb{R}^{1\times D} \to \mathbb{R}^{1\times r\times k}$;
            Perform rank selection following (5.12) to obtain $\boldsymbol{M}_{\mathcal{I}}$ and $\boldsymbol{M}_{\text{CHM}}$;
            Apply (5.13) to extract local subspaces: $\boldsymbol{P}_{soft} \in \mathbb{R}^{F_A\times\tilde{r}}$ and $\boldsymbol{Q}_{soft} \in \mathbb{R}^{F_B\times\tilde{r}}$;
            Reparameterize forward weights $\tilde{\boldsymbol{W}}_{soft} \leftarrow \boldsymbol{P}_{soft}\boldsymbol{Q}_{soft}^{\top}$;
            Forward propagation following (5.14);
            Compute forward loss $\mathcal{L}$;
            for $t \neq 1$ **do**
                Compute Fisher regularization term $\mathcal{L}(\mathcal{F})$;
                $\mathcal{L} \leftarrow \mathcal{L} + \lambda\mathcal{L}(\mathcal{F})$;
            Backward propagation;
            for $t \neq 1$ **do**
                Freeze $\boldsymbol{W}_0$ according to $\boldsymbol{M}_{accum;\mathcal{T}_{t-1}}$;
            Update $\boldsymbol{W}_0$, $\mathcal{S}_{global}$, and $\mathcal{S}_{local}$;
            while $e = E_w$ **do**
                Record $\boldsymbol{M}_{dm}$ and $\mathcal{I}$ for $\mathcal{T}_t$;
                Accumulate $\boldsymbol{M}_{accum;\mathcal{T}_t} \leftarrow \boldsymbol{M}_{accum;\mathcal{T}_{t-1}} \vee \boldsymbol{M}_{\text{CHM}}$;
            continue;
        Reconstruct $\tilde{\boldsymbol{W}}$ using task-specific $\boldsymbol{M}_{dm}$ and $\mathcal{I}$;
        Forward propagation following (5.11);
        Compute forward loss $\mathcal{L}$;
        Backward propagation;
        Update $\boldsymbol{W}_0$;
        continue;

cost $\mathcal{O}(NF_w^2 + |\mathcal{E}|F_w)$, the warm-up complexity of WSN is:

$$\mathcal{O}(2BF^2 + 2BNF^2 + \sqrt{2B}|\mathcal{E}|F) \tag{5.27}$$

After the warm-up, the mask is fixed, and the complexity reduces to:

$$\mathcal{O}(2BNF^2 + \sqrt{2B}|\mathcal{E}|F) \tag{5.28}$$

CAGNN During warm-up, the mask-learning operations of CAGNN require approximately $\mathcal{O}(NF^2)$ complexity. The low-rank reparameterized transformation $(\boldsymbol{XP})\boldsymbol{Q}^\top$ costs $\mathcal{O}(NFr)$, and the message-passing costs $\mathcal{O}(|\mathcal{E}|F)$. Therefore, the overall warm-up complexity of CAGNN is:

$$\mathcal{O}(NF^2 + NFr + |\mathcal{E}|F) \tag{5.29}$$

After the warm-up, the mask is fixed, and the complexity reduces to:

$$\mathcal{O}(NFr + |\mathcal{E}|F) \tag{5.30}$$

Finally, since the selected rank r is typically much smaller than WSN's hidden dimension, i.e., $r \ll F_w$, CAGNN achieves lower complexity than WSN under a reasonable branch number B. This advantage becomes more pronounced as the model capacity grows, i.e., as F increases. With the introduction of contextual information, we typically make the warm-up phase only account for a small fraction of the training process, making the overall complexity of CAGNN substantially lower than WSN. Moreover, we can further reduce the mask-learning cost by pooling the task-specific context into a vector $\boldsymbol{c} \in \mathbb{R}^{1\times F}$. In this case, the warm-up complexity of CAGNN becomes:

$$\mathcal{O}(F^2 + NFr + |\mathcal{E}|F) \tag{5.31}$$

which is consistently smaller than that of WSN.

5.4 Experimental Evaluation

To assess the effectiveness of our proposed CAGNN framework, in this section, we conduct extensive experiments across eight real-world datasets in the task-incremental setting, targeting not only catastrophic forgetting but also the efficiency bottleneck that existing parameter isolation-based methods may encounter. This empirical study answers the following questions:

- **Question 1**: How is CAGNN's robustness to the catastrophic forgetting problem, and how is its plasticity in continual learning?
- **Question 2**: How is the efficiency of CAGNN, and how much faster is it than the other parameter isolation-based baselines?
- **Question 3**: How is CAGNN's parameter utilization under a long-term task stream, and can the proposed Fisher regularization term facilitate the model's scalability with less interference?
- **Question 4**: Is CAGNN's performance stable under a long-term task stream, and is it sensitive to hyperparameters?

5.4.1 Experimental Setup

5.4.1.1 Datasets

Recognizing that parameter isolation-based methods naturally accommodate different levels of graph tasks, we evaluate CAGNN on both node-level and graph-level classification across a total of eight real-world datasets spanning diverse domains. Specific summary statistics of these datasets are provided in Table 5.1.

Table 5.1: Summary of Dataset Statistics and Characteristics

	Datasets	#Nodes(Graphs)	# Edges	# Attributes	# Tasks	# Classes/task	# Classes
Node Classification	Cora	2,708	10,556	1,433	3	2	7
	Citeseer	3,327	9,104	3,703	3	2	6
	CoraFull	19,793	126,842	8,710	35	2	70
	Computers	13,752	491,722	767	5	2	10
	OGB-Arxiv	169,343	1,166,243	128	8	5	40
	Syn-CoraFull	∼19,793	∼126,842	8,710	140	2	280
Graph Classification	MNIST	∼70.6(55,000)	∼564.5	3	5	2	10
	CIFAR10	∼117.6(45,000)	∼941.2	5	5	2	10
	Aromaticity	∼29.7(3,868)	∼65.4	2	10	3	30

Node Classification We conduct node classification tasks on 5 datasets, including Cora (Sen et al., 2008), Citeseer (Sen et al., 2008), CoraFull (Bojchevski & Günnemann, 2018), OGB-Arxiv (W. Hu et al., 2020), and Amazon Computers (McAuley et al., 2015). Cora, Citeseer, and CoraFull are citation networks, where each node represents a scientific publication and its label corresponds to the research field of the publication. Following the task-incremental setting, they are split into binary classification tasks with 3, 3, and 35 tasks, respectively. OGB-Arxiv is also a citation network, where each node denotes a research paper associated with a subject area. It is divided into 8 tasks, with each task containing 5 classes. Amazon Computers is a co-purchase network, where nodes represent products and edges connect items that are frequently bought together. It is organized into 5 tasks with 2 classes per task. We also construct a synthetic CoraFull dataset by repeating the original CoraFull task sequence four times with minor random node dropping, resulting in 140 tasks while preserving the basic properties of the original CoraFull dataset. Note that although tasks are repeated in synthetic CoraFull, each repeated occurrence is regarded as a new task when it arrives in the continual learning stream.

Graph Classification We conduct graph classification tasks on 3 datasets, including MNIST (Achanta et al., 2012), CIFAR10 (Dwivedi et al., 2023), and Aromaticity (Z. Wu et al., 2018). MNIST and CIFAR10 are super-pixel graph datasets derived from images, and both are organized into 5 tasks with 2 classes per task.

Aromaticity consists of molecular graphs, where nodes represent atoms and edges represent chemical bonds. Its labels indicate the number of aromatic atoms in each molecule. We partition Aromaticity into 10 tasks, with each task containing 3 classes.

5.4.1.2 Baselines

Our evaluation of CAGNN focuses on the performance within the category of parameter isolation-based methods, including PackNet (Mallya & Lazebnik, 2018), Piggyback (Mallya et al., 2018), HAT (Serra et al., 2018), TAAM (J. Liu & Zhang, 2026), and WSN (Kang et al., 2024). In particular, we compare its computational efficiency and scalability with WSN, a closely related state-of-the-art element-wise masking method. To assess forgetting mitigation more broadly, we also include TWP (H. Liu et al., 2021) and ERGNN (F. Zhou & Cao, 2021) as graph-specific regularization- and replay-based baselines, together with LwF (Z. Li & Hoiem, 2017) and EWC (Kirkpatrick et al., 2017). All methods adopt GCN (Kipf & Welling, 2017) as the backbone, except TAAM, which follows its original setting and uses SGC (F. Wu et al., 2019). We consider a bare GCN without continual learning mechanisms as the lower bound, and joint training (Caruana, 1997), where the model can access all data, as the upper bound in our continual learning setting.

5.4.1.3 Evaluation Metrics

For the evaluation of the catastrophic forgetting, we use two commonly adopted metrics: Average Performance (AP) and Average Forgetting (AF) (Lopez-Paz & Ranzato, 2017). AP measures the model's plasticity by calculating the average test performance across all T tasks, and AF measures the model's stability by calculating the average performance change of each task $\mathcal{T}_t$ after learning T tasks. The following

two formulations give them:

$$\text{AP} = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{P}_{tt}, \quad \text{AF} = \frac{1}{T-1}\sum_{t=1}^{T-1}(\boldsymbol{P}_{Tt} - \boldsymbol{P}_{tt}) \tag{5.32}$$

where $\boldsymbol{P}_{i,j}$ records the performance of jth task $\mathcal{T}_j$ after learning ith task $\mathcal{T}_i$ ($i \geq j$). Note that AF > 0 indicates forgetting, and each task's performance on the datasets used in our experiments is measured by accuracy.

For the evaluation of the computational efficiency, we measure the average runtime per epoch after completing all tasks, reported in milliseconds, as well as per-task GPU memory usage, reported in megabytes. Regarding the model's scalability for future tasks, we evaluate it by the percentage of parameters used after learning all T tasks. To ensure a fair playground for parameter isolation-based methods, we evaluate them under the same budget of the total model's capacity.

5.4.1.4 Hardware and Experimental Framework Setup

We ran all of our experiments on a server equipped with an AMD EPYC 7J13 64-core CPU, 256 GB of RAM, and four NVIDIA GeForce RTX 4090 GPUs, and this configuration was used for all baselines. For all experiments, we run them based on a unified continual graph learning framework, BeGin[1] (Ko et al., 2025), and set all common hyperparameters (e.g., learning rate, number of layers, hidden dimensions, dropout rate) the same across all models during hyperparameter tuning. For hyperparameter tuning, we utilize Optuna[2] (Akiba et al., 2019), which is a hyperparameter optimization framework, to search for the best model.

5.4.2 Hyperparameter Search Space

We summarize the hyperparameter search spaces used for CAGNN, WSN, TAAM, the upper-bound model, and the BeGin baselines. All hyperparameters are selected

[1] https://github.com/ShinhwanKang/BeGin
[2] https://optuna.org/

based on the validation performance of each dataset.

Table 5.2: Hyperparameter search spaces and fixed settings.

Model	Search Space
CAgnn	lr $\in \{10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}\}$; nhid $\in \{16, 32, 64, 128, 256, 512\}$; branch $\in \{1, 2, 3, 4, 5, 6\}$; dropout $\in \{0.3, 0.4, 0.5, 0.6, 0.7\}$; $\alpha \in \{0.1, 0.3, 0.5, 0.8, 1, 2, 5, 10\}$; sparsity $\in \{0.85, 0.90, 0.92, 0.95, 0.97, 0.99\}$; warm-up epochs $\in \{1, 25, 50, 100\}$.
Baseline	
WSN	lr $\in \{10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}\}$; nhid $\in \{16, 32, 64, 128, 256, 512\}$; dropout $\in \{0.3, 0.4, 0.5, 0.6, 0.7\}$; sparsity $\in \{0.80, 0.85, 0.90, 0.92, 0.95, 0.97, 0.99\}$.
TAAM	lr $\in \{10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}\}$; nhid $\in \{64, 128, 256\}$; dropout $\in \{0.2, 0.4, 0.5, 0.6\}$; task embedding dimension $\in \{16, 32, 64, 128\}$; TAAM heads $\in \{2, 4, 8\}$; TAAM rank $\in \{4, 8, 16, 32\}$.
PackNet	lr $\in \{10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}\}$; dropout $\in \{0.0, 0.25, 0.5\}$; weight decay $\in \{0.0, 5\times10^{-4}\}$.
Piggyback	lr $\in \{10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}\}$; dropout $\in \{0.0, 0.25, 0.5\}$; weight decay $\in \{0.0, 5\times10^{-4}\}$; threshold $\in \{10^{-1}, 10^{-2}\}$.
HAT	lr $\in \{10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}\}$; dropout $\in \{0.0, 0.25, 0.5\}$; weight decay $\in \{0.0, 5\times10^{-4}\}$; fixed settings: $\lambda = 0.75$ and $s_{\max} = 400.0$.
LwF	lr $\in \{10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}\}$; dropout $\in \{0.0, 0.25, 0.5\}$; weight decay $\in \{0.0, 5\times10^{-4}\}$; $\lambda = 1.0$; temperature $T = 2.0$.
EWC	lr $\in \{10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}\}$; dropout $\in \{0.0, 0.25, 0.5\}$; weight decay $\in \{0.0, 5\times10^{-4}\}$; $\lambda = 10000.0$.
TWP	lr $\in \{10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}\}$; dropout $\in \{0.0, 0.25, 0.5\}$; weight decay $\in \{0.0, 5\times10^{-4}\}$; $\lambda_t \in \{100.0, 1000.0\}$; fixed settings: $\lambda_l = 10000.0$, $\beta = 0.01$.
ERGNN	lr $\in \{10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}\}$; dropout $\in \{0.0, 0.25, 0.5\}$; weight decay $\in \{0.0, 5\times10^{-4}\}$; fixed settings: the number of experience nodes follows the official benchmark configuration, sampler name is CM, and distance threshold is 0.5.
Joint	lr $\in \{5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}\}$; nhid $\in \{16, 32, 64, 128, 256, 512\}$; dropout $\in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$; weight decay $\in \{0, 10^{-5}, 5\times10^{-5}, 10^{-4}, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}\}$.
Bare	lr $\in \{10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}\}$; dropout $\in \{0.0, 0.25, 0.5\}$; weight decay $\in \{0.0, 5\times10^{-4}\}$.

For BeGin baselines, we follow the official benchmark settings. The common search space is: lr $\in \{10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}\}$, dropout $\in \{0.0, 0.25, 0.5, 0.7\}$, and weight decay $\in \{0.0, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}\}$. For the default backbone, node-level baselines use 3 layers with hidden dimension 256, while graph-level baselines use 4 layers with hidden dimension 146.

Table 5.3: Performance of CAGNN on GCN (CAGCN) and GAT (CAGAT), and baselines on 8 datasets. Results are reported as the Average Performance (AP) and Average Forgetting (AF) of 5 iterations. Each dataset's best and second-best results are highlighted with gray and underline, respectively.

Dataset (# Task)	Node Classification										Graph Classification					
	Cora (3)		Citeseer (3)		CoraFull (35)		Computers (5)		OGB-Arxiv (8)		Mnist (5)		Cifar10 (5)		Aromaticity (10)	
	AP	AF	AP	AF	AP	AF	AP	AF	AP	AF	AP	AF	AP	AF	AP	AF
Bare	90.28 ± 2.2	2.1 ± 1.7	80.43 ± 3.6	5.3 ± 1.9	79.79 ± 3.5	19.8 ± 4.2	96.18 ± 1.9	4.2 ± 1.1	70.62 ± 6.7	21.7 ± 6.6	56.24 ± 12.8	51.1 ± 9.7	55.30 ± 3.7	35.5 ± 6.2	43.26 ± 3.1	23.6 ± 2.7
LwF	92.17 ± 1.1	1.5 ± 1.0	86.63 ± 0.5	1.8 ± 0.7	OOM	OOM	95.92 ± 1.7	6.9 ± 2.0	80.21 ± 4.1	12.9 ± 4.8	72.25 ± 7.7	31.5 ± 8.2	72.18 ± 4.2	6.7 ± 3.9	49.33 ± 2.1	16.5 ± 2.0
EWC	90.39 ± 1.9	2.0 ± 1.2	85.43 ± 1.1	2.7 ± 1.2	90.68 ± 2.9	5.7 ± 2.1	97.22 ± 0.8	3.6 ± 1.2	88.52 ± 2.0	3.6 ± 1.7	72.30 ± 5.4	23.9 ± 5.1	71.02 ± 2.9	6.0 ± 2.7	$\underline{55.47 \pm 1.4}$	6.4 ± 1.7
TWP	90.52 ± 1.3	3.3 ± 2.1	86.07 ± 0.7	2.2 ± 0.9	89.35 ± 3.3	8.9 ± 3.5	97.90 ± 0.6	2.0 ± 0.8	87.45 ± 2.4	3.8 ± 2.1	76.20 ± 7.8	18.2 ± 7.0	72.18 ± 3.7	6.7 ± 3.3	57.25 ± 2.0	6.4 ± 2.2
ERGNN	86.92 ± 4.2	3.7 ± 3.1	86.52 ± 0.7	3.6 ± 1.1	94.39 ± 2.3	3.8 ± 2.7	97.83 ± 0.6	0.5 ± 0.2	89.96 ± 1.3	2.6 ± 1.5	N/A	N/A	N/A	N/A	N/A	N/A
PackNet	93.79 ± 0.7	0.0 ± 0.0	85.66 ± 1.8	0.0 ± 0.0	97.14 ± 0.7	0.0 ± 0.0	98.17 ± 0.2	0.0 ± 0.0	90.93 ± 1.1	0.0 ± 0.0	96.05 ± 1.1	0.0 ± 0.0	$\underline{80.17 \pm 0.7}$	0.0 ± 0.0	41.08 ± 3.3	0.0 ± 0.0
Piggyback	93.87 ± 0.9	0.0 ± 0.0	87.33 ± 0.4	0.0 ± 0.0	96.39 ± 0.7	0.0 ± 0.0	98.23 ± 0.3	0.0 ± 0.0	92.07 ± 0.5	0.0 ± 0.0	83.12 ± 3.0	0.0 ± 0.0	76.17 ± 1.9	0.0 ± 0.0	41.64 ± 2.7	0.0 ± 0.0
HAT	94.01 ± 1.2	2.4 ± 1.5	85.58 ± 1.2	0.6 ± 0.3	91.79 ± 2.1	7.0 ± 1.4	97.92 ± 0.5	0.4 ± 0.0	90.07 ± 0.4	1.2 ± 0.5	59.91 ± 6.9	46.4 ± 6.8	54.87 ± 4.6	36.3 ± 4.4	54.28 ± 1.9	11.9 ± 2.5
TAAM	94.20 ± 1.1	0.0 ± 0.0	$\underline{87.99 \pm 0.4}$	0.0 ± 0.0	97.21 ± 1.9	0.0 ± 0.0	98.64 ± 1.1	0.0 ± 0.0	86.27 ± 1.2	0.0 ± 0.0	95.26 ± 1.4	0.0 ± 0.0	70.99 ± 4.2	0.0 ± 0.0	53.51 ± 1.9	0.0 ± 0.0
WSN	94.06 ± 1.1	0.0 ± 0.0	87.38 ± 0.3	0.0 ± 0.0	97.28 ± 0.8	0.0 ± 0.0	98.24 ± 0.2	0.0 ± 0.0	90.28 ± 0.9	0.0 ± 0.0	95.92 ± 1.6	0.0 ± 0.0	80.02 ± 1.7	0.0 ± 0.0	47.15 ± 2.0	0.0 ± 0.0
CAGCN	$\underline{94.68 \pm 0.7}$	0.0 ± 0.0	87.63 ± 0.3	0.0 ± 0.0	$\underline{97.32 \pm 1.2}$	0.0 ± 0.0	98.01 ± 0.6	0.0 ± 0.0	$\underline{91.55 \pm 0.7}$	0.0 ± 0.0	96.50 ± 1.6	0.0 ± 0.0	81.13 ± 1.5	0.0 ± 0.0	49.70 ± 1.1	0.0 ± 0.0
CAGAT	95.28 ± 1.6	0.0 ± 0.0	88.28 ± 0.7	0.0 ± 0.0	97.80 ± 0.9	0.0 ± 0.0	$\underline{98.32 \pm 0.2}$	0.0 ± 0.0	87.91 ± 1.7	0.0 ± 0.0	$\underline{96.41 \pm 1.1}$	0.0 ± 0.0	79.81 ± 1.8	0.0 ± 0.0	49.96 ± 1.5	0.0 ± 0.0
Joint	96.02 ± 1.6	0.0 ± 0.0	89.06 ± 2.6	0.0 ± 0.0	98.45 ± 2.3	0.0 ± 0.0	99.09 ± 0.4	0.0 ± 0.0	91.62 ± 2.6	0.0 ± 0.0	97.58 ± 0.7	0.0 ± 0.0	81.65 ± 1.5	0.0 ± 0.0	63.49 ± 3.4	0.0 ± 0.0

[1] OOM indicates out-of-memory during training.

[2] N/A indicates not applicable.

5.4.3 Performance Analysis

For continual learning, the method's robustness to catastrophic forgetting is always worth investigating initially. Table 5.3 provides the average performance (AP) and average forgetting (AF) of methods of different paradigms on both node classification and graph classification tasks. Figure 5.5 demonstrates the per-task performance after finishing all tasks on Computers, OGB-Arxiv, MNIST, and CIFAR10 datasets. Parameter isolation-based methods generally outperform regularization- and replay-based methods, which is expected in the task-incremental setting due to their forget-free characteristic (AF = 0). However, it is noticeable that those graph-specific regularization- and replay-based methods (i.e., TWP and ERGNN) do not show significant advantages over the general continual learning methods (i.e., LwF and EWC) on two node-level datasets, which are Cora and Citeseer. This situation is probably because these two datasets may introduce common patterns across tasks, such that some advances in accommodating graph properties become less significant. As can be seen, ERGNN regains its advantage on CoraFull and OGB-Arxiv datasets, while TWP remains comparable with EWC. Note that TWP incurred OOM during training, and ERGNN does not apply to the graph-level task, so their relevant results are not reported.

Within parameter isolation methods, although the Piggyback performs masking on a fixed backbone, whose parameters are not learnable, it still shows outstanding performance, particularly for OBG-Arxiv. However, this is largely attributed to the experimental environments, where tasks within a dataset usually share relevant or similar patterns. Otherwise, Piggyback may degrade because its backbone's zero plasiticity may not be sufficient. TAAM also performs strongly, but it further possesses task-specific modulators beyond the fixed backbone, which offer greater plasticity than Piggyback. Nevertheless, their fixed backbone may still limit plasticity on some tasks with substantially different patterns. All parameter isolation-based methods exhibit similar performance, while HAT suffers from forgetting because

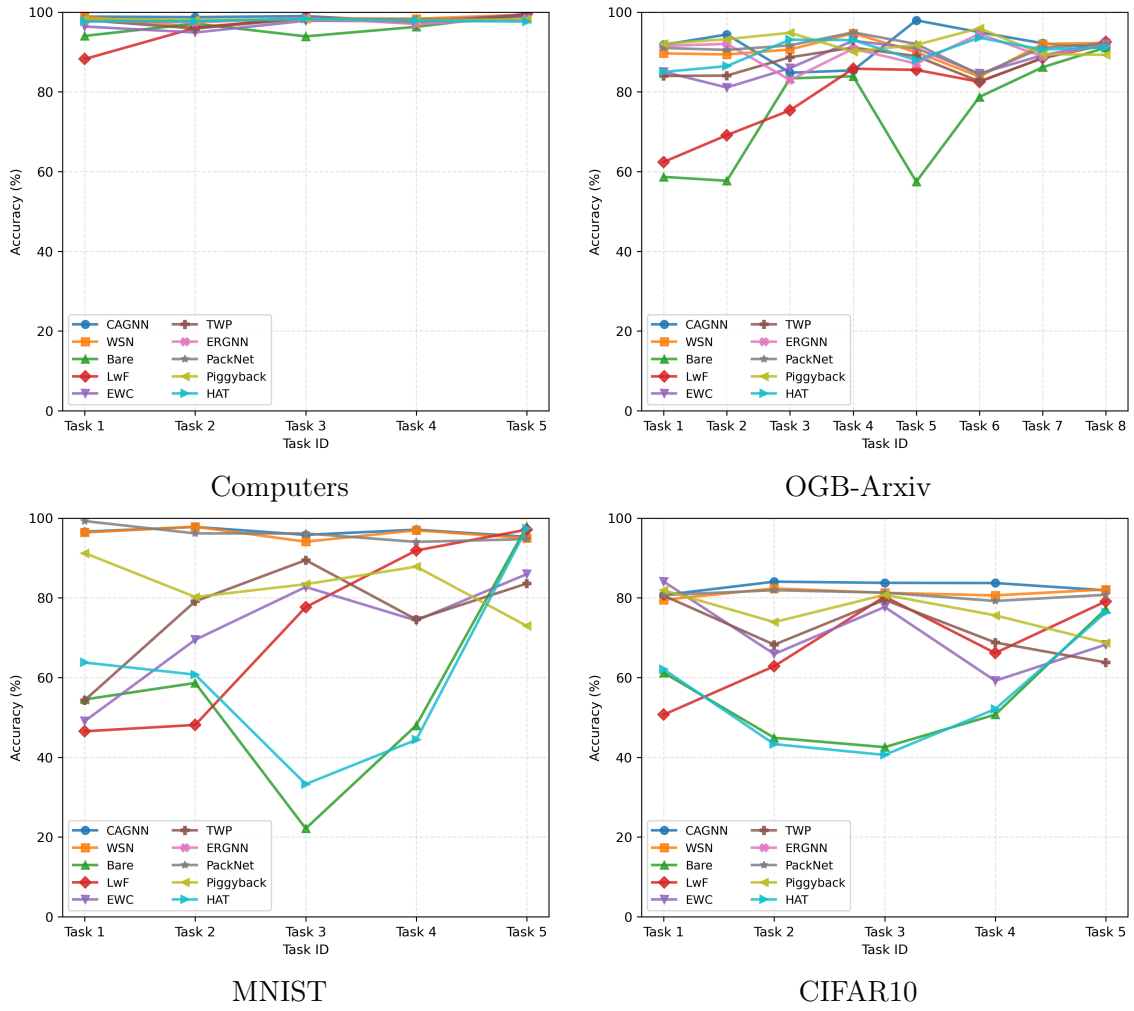


Figure 5.5: Per-Task Performance of 10 Models on Citeseer, OGB-Arxiv, MNIST, and CIFAR10 datasets

its used parameters are not fully frozen, leading to a relatively large performance degradation on the CoraFull dataset, which has more tasks than other datasets. From the design perspective, WSN and our CAGNN both learn subnetwork structures and network parameters, resulting in comparable performance across datasets. For graph-level datasets, some parameter isolation-based methods, including Piggyback and HAT, perform worse than they do in node-level datasets. Only PackNet, WSN, and CAGNN, which possess higher plasticity, showcase consistent high performance across node-level and graph-level datasets. Notably, for the aromaticity dataset, parameter isolation-based methods, except for HAT, yield unsatisfactory performance, and the reason needs further investigation. In general, methods with

higher plasticity, including PackNet, WSN, and CAGNN, achieve more consistent performance. Overall, CAGNN's stable results across datasets highlight its effectiveness in mitigating catastrophic forgetting. We further examine the transferability of CAGNN by applying it to Graph Attention Network (GAT) (Veličković, Cucurull, et al., 2018). The results show consistent performance across two different GNN backbones, highlighting the generality of CAGNN.

5.4.4 Efficiency Analysis

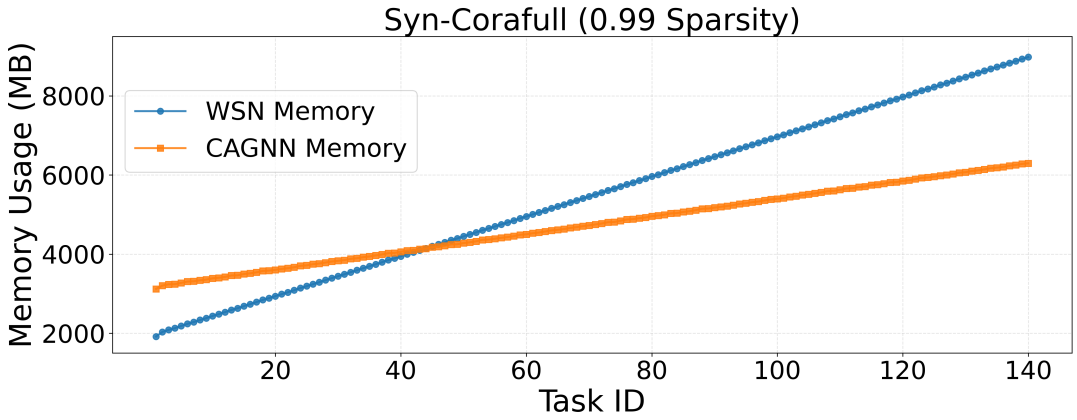
Table 5.4: Average runtime per epoch and runtime multiple on three different datasets

MODEL	OGB-ARXIV		MNIST		CIFAR10	
	Time (ms)	Ratio	Time (ms)	Ratio	Time (ms)	Ratio
WSN	787	1.00×	9,611	1.00×	10,325	1.00×
PackNet	335	0.43×	6,419	0.67×	6,922	0.67×
Piggyback	299	0.38×	6,831	0.71×	7,114	0.69×
HAT	343	0.44×	6,590	0.68×	7,068	0.68×
CAGNN (0.95)	215	0.27×	3,345	0.35×	2,961	0.29×
CAGNN (0.99)	197	0.25×	3,101	0.32×	2,783	0.27×

We evaluate the efficiency of CAGNN by measuring the average training time (in milliseconds) per epoch. Table 5.4 provides the average runtime and the runtime multiple of four parameter isolation-based baselines, including PackNet, Piggyback, HAT, and WSN, on OGB-Arxiv, MNIST, and CIFAR10 datasets, which are usually more time-consuming for training than others. In this experiment, we assume a capacity expansion scenario and thus over-parametrized models. To be specific, we set the hidden dimension to 2048 for all baselines, introducing a total parameter count of $8,650,752$ for OGB-Arxiv and $12,582,912$ for both MNIST and CIFAR10. Equivalently, CAGNN with the same capacity is set with hidden dimension 512 and

Table 5.5: Memory usage (in MB) of CAGNN and WSN across 140 tasks on the synthetic CoraFull dataset.

MODEL	$\mathcal{T}_0$	$\mathcal{T}_{35}$	$\mathcal{T}_{70}$	$\mathcal{T}_{105}$	$\mathcal{T}_{140}$
WSN	1918.5	3744.9	5508.2	7270.7	8980.4
CAGNN	3121.5	3966.6	4750.2	5558.7	6296.2



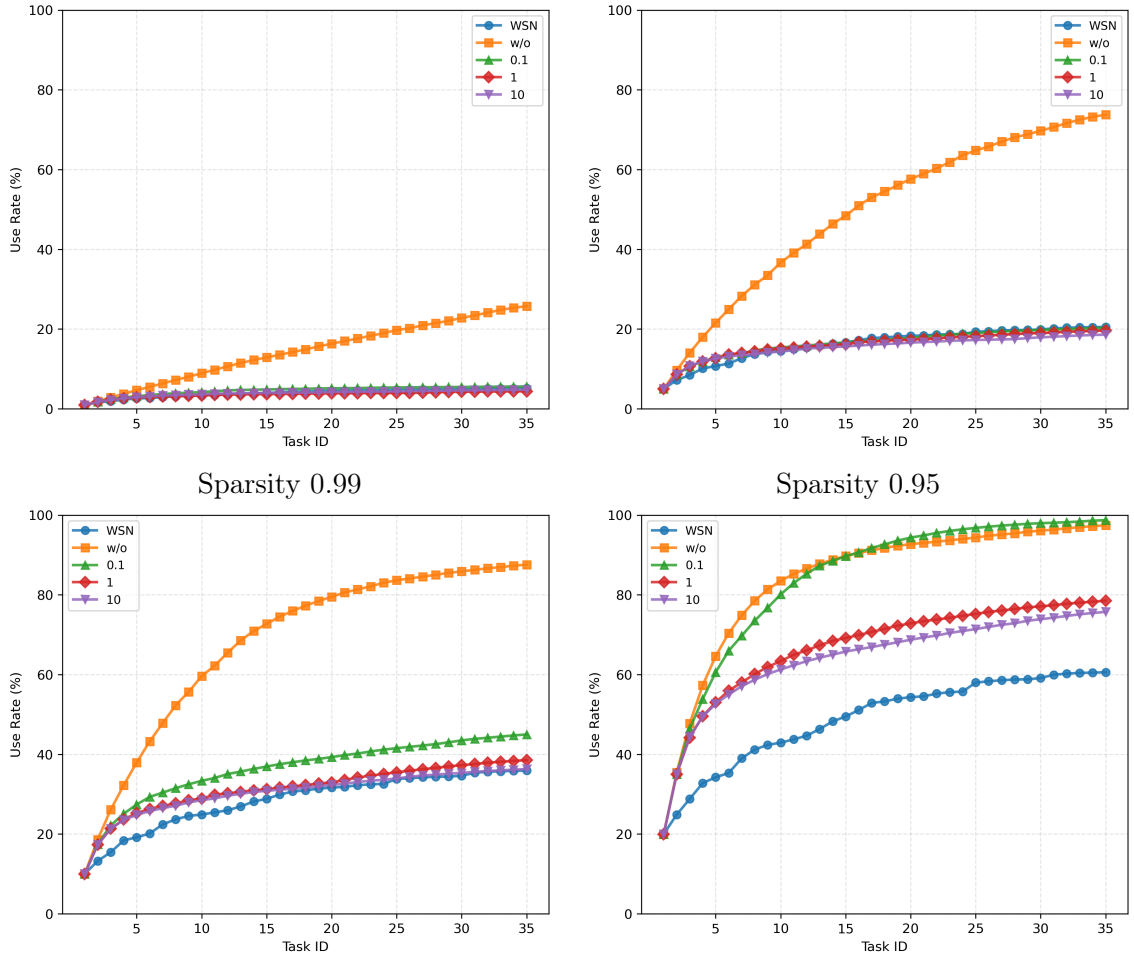
branch number 10. For the runtime of CAGNN, it is calculated by the average of 10% warm-up epochs and 90% after-warm-up epochs, following the setting for evaluating average performance. Due to the decoupled mechanism of CAGNN, its runtime may also vary with the size of subnetworks. To investigate it, we set WSN as the base ($1\times$) and compare a different sparsity $p = 0.95$ ($r = 256$). Note that this sparsity is not large enough for the current capacity, and we usually expect a larger one for a larger capacity. This setting is just for a clear comparison. From the result, we observe that WSN has a larger complexity overhead due to its flexibility in learning subnetworks, rendering it the slowest among the 5 models. Because of relatively simpler masking mechanisms, PackNet, Piggyback, and HAT are much faster than WSN and are at the same level of efficiency. CAGNN exhibits outstanding efficiency among 5 parameter isolation-based models, and is approximately 3.6 times faster than WSN and 2.1 times faster than PackNet, Piggyback, and HAT, while remaining strong plasticity. Moreover, a higher sparsity level yields smaller subnetworks, which in turn enables faster training for CAGNN. The average runtime per warm-up

epoch of CAGNN is 563 ms on OGB-Arxiv, 8,849 ms on MNIST, and 8,468 ms on CIFAR10, and it still remains faster than WSN. Overall, as the training batch size increases or the model's total capacity grows, the efficiency advantage of CAGNN will become more pronounced.

Table 5.5 further provides the memory usage of WSN and CAGNN on synthetic CoraFull with 140 tasks under 0.99 sparsity, without any additional compression of the saved masks. Although CAGNN starts with a higher initial memory cost due to its hierarchical masking components, its memory usage grows much more slowly as the task stream becomes longer. In contrast, WSN shows a nearly linear and steeper memory increase, since it needs to maintain dense task-specific masks for an expanding set of tasks. As the number of tasks increases, the memory usage of WSN quickly exceeds that of CAGNN, and the gap becomes increasingly large. This demonstrates that CAGNN's index-form mask storage provides better memory scalability under continual capacity expansion, making it more suitable for long-horizon continual learning.

5.4.5 Scalability Analysis

To evaluate the scalability of the proposed CAGNN, we analyze from the perspectives of the Fisher regularization and the length of the warm-up phase on the CoraFull dataset, which comprises 35 tasks and is sufficiently large to stress-test long-horizon continual learning. Note that model capacities of both CAGNN and WSN are set to the same, and the mask generators of CAGNN are initialized whenever a new task arrives. Figure 5.6 demonstrates the impact of different λ of fish regularization on the total utilization of parameters at 4 different levels of sparsity (0.99, 0.95, 0.90, 0.80). Our proposed Fisher regularization contributes significantly to CAGNN's scalability, so that only 10% epochs of warm-up with task-specific contextual information can achieve almost the same level of parameter utilization as WSN, which tunes sub-networks during the entire training phase. Most importantly, it imposes almost no

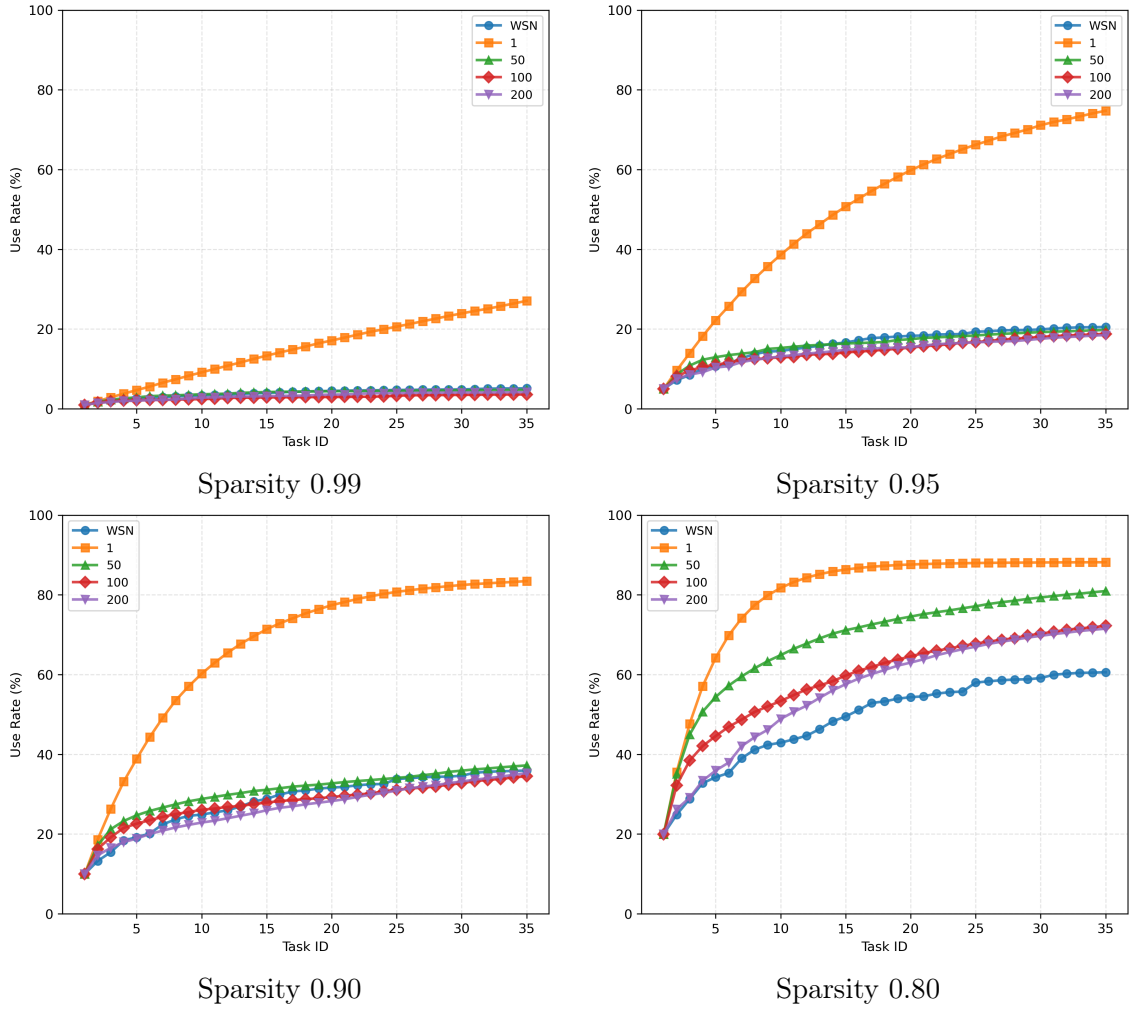


Sparsity 0.99 Sparsity 0.95

Sparsity 0.90 Sparsity 0.80

Figure 5.6: The Impact of Fisher Regularization on Parameter Utilization (50 Warm-Up in 500 Epochs)

performance loss, but at a price of slightly higher time complexity during warm-up (It was applied in the previous efficiency analysis). As can be seen, CAGNN without Fisher regularization (orange line) consumes parameters rapidly and is easily saturated. Although not significant on sparsity 0.99 and 0.95, a larger λ tends to force the model to reuse parameters across tasks as much as possible. When the subnetwork sparsity is set to 0.8, CAGNN consumes more parameters than WSN. We observe that increasing λ can further reduce parameter utilization. However, such a sparsity level is usually impractical and accelerates the model's saturation of capabilities. Figure 5.7 shows the impact of the duration of the warm-up phase on the utilization of parameters. Obviously, different lengths of the warm-up phase



Sparsity 0.99 Sparsity 0.95

Sparsity 0.90 Sparsity 0.80

Figure 5.7: The Impact of Warm-Up Epoch on Parameter Utilization ($\lambda = 1$)

introduce similar parameter utilization for sparsity levels of 0.99, 0.95, and 0.90. This observation leads us to set the warm-up phase account for only 10% of the total training phase, considering the model's efficiency. Although a longer warm-up phase can further promote parameter reuse, its contribution becomes marginal at the expense of increased training time, which is undesirable. It is noticeable that Fisher regularization will not apply when the warm-up epoch is only 1, yielding a high parameter utilization and separation between subnetwork structures. Such a high utilization and separation indicate how task-specific context initially shapes different subnetworks and is crucial for effective and efficient subnetwork constructions.

We further evaluate CAGNN on synthetic CoraFull, which contains 140 tasks

Table 5.6: Ablation studies on the effect of contextual information on the synthetic CoraFull dataset, using sparsity 0.99.

Context Type	Syn-CoraFull	Use Rate
Parameterized Score	96.21±0.7	55.50%
Random Context	95.96±1.7	57.80%
All 1's Context	96.51±1.1	61.30%
Task-specific Context	97.05±0.3	22.45%

Table 5.7: Ablation studies on the effect of differentiable proxy on the synthetic CoraFull dataset, using sparsity 0.90.

Dataset (# Task)	w/o Diff. Proxy	Use Rate
Cora (3)	93.97±1.2	27.10%
CiteSeer (3)	86.64±2.7	25.20%
CoraFull (35)	95.20±0.9	83.70%
Computers (5)	97.72±0.3	61.50%
OGB-Arxiv (8)	89.81±1.4	55.10%
Mnist (5)	96.22±1.2	47.80%
Cifar10 (5)	77.52±1.5	48.40%
Aromaticity (10)	45.43±2.1	75.30%

for long-horizon continual learning, and study contextual information. Table 5.6 shows that CAGNN maintains robust performance with effective parameter reuse in long-task settings. Furthermore, context mainly affects parameter usage, and Fisher regularization becomes less effective when informative task-specific context is absent. Figure 5.8 provides a visualization that further illustrates the effect of contextual information on subnetwork construction. With task-specific context, the activated weights exhibit a more structured and compact pattern, indicating that CAGNN can selectively allocate task-relevant parameters. In contrast, random context leads to more scattered and redundant activations across branches, suggesting less effective parameter reuse. The observation on the accumulated mask $\boldsymbol{M}_{accum}$ (Figure 5.8 Left) is consistent with Table 5.6, where task-specific context achieves comparable or better performance while substantially facilitating the parameter reusability compared with random context.

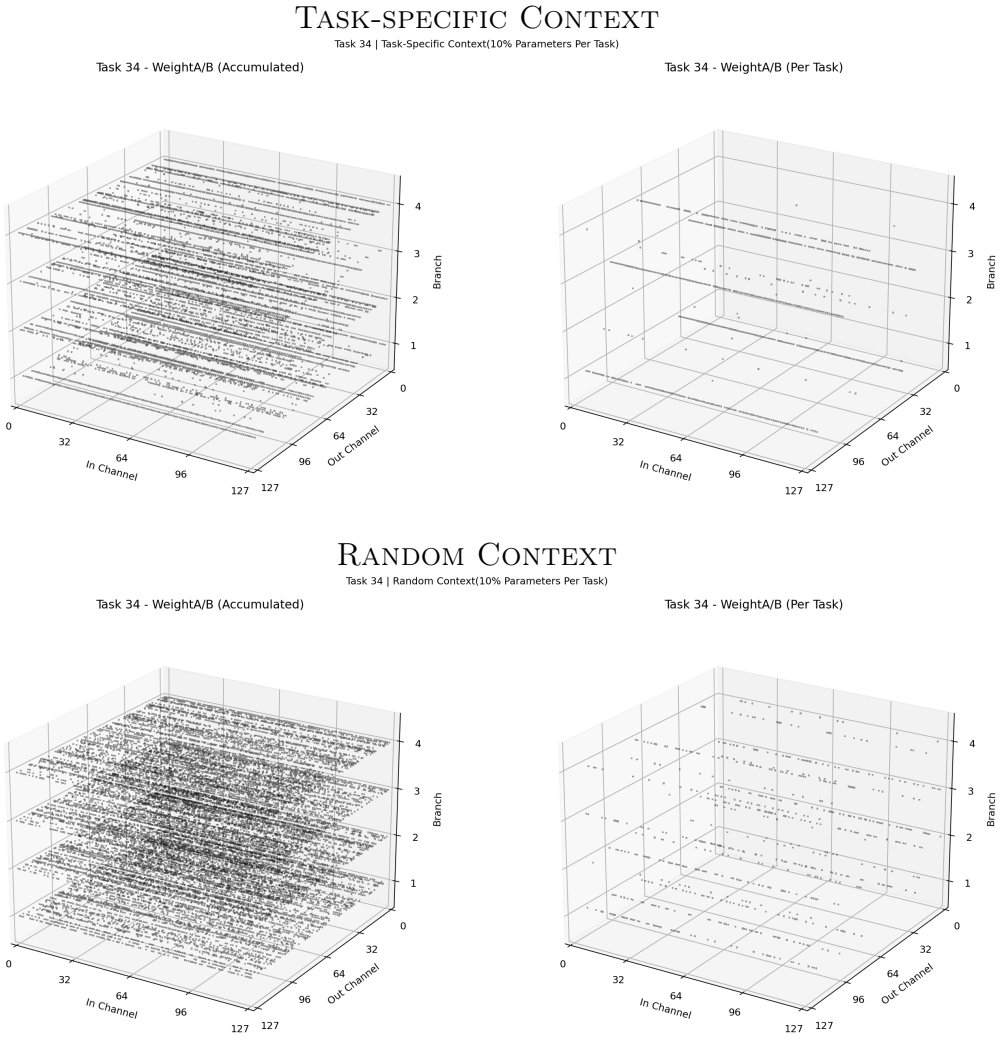


Figure 5.8: Visualization of subnetwork structures using task-specific context and random context on Task 34 in the first convolutional layer on CoraFull.

Table 5.7 highlights the necessity of the differentiable proxy for mask learning. Although average performance is only slightly affected, gradient truncation makes subnetworks harder to optimize and results in higher parameter consumption. Also, we study a variant where all tasks share the same mask generator, which means that mask scores are not re-initialized for new tasks. Under this setting, contextual information still helps reduce performance degradation while maintaining low parameter utilization, even without Fisher regularization. Nevertheless, this introduces a trade-off between reuse bias and plasticity. Specifically, the shared generator encourages parameter reuse but restricts task-specific adaptation. Consequently, it

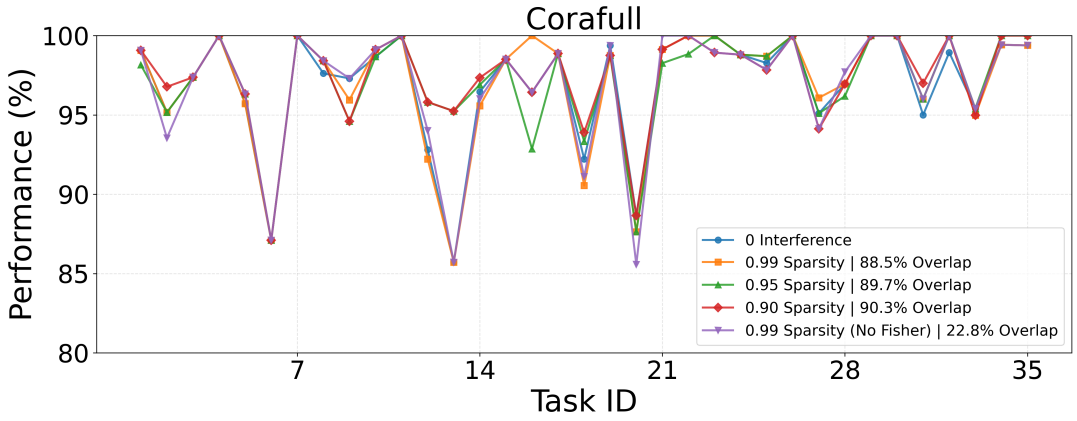
requires a longer warm-up to reach performance comparable to the original setting, where each task-specific subnetwork is initialized in a more flexible state guided by contextual information. Lastly, we have attempted WSN's strategy that does not rely on contextual information or Fisher regularization, where masks are instead initialized as parameterized scores. When the parameterized scores are not re-initialized for each task, the resulting subnetworks become highly similar, leading to low parameter utilization after 50 warm-up epochs. However, such low utilization is largely caused by a nearly fixed subnetwork configuration, where many parameters selected for later tasks have already been frozen and cannot adapt to the current task. This reduces the average performance by approximately 8%. Re-initializing the parameterized scores (Table 5.6 Parameterized Score) for each task alleviates this issue by improving task-specific adaptability, but at the cost of higher parameter consumption. Recovering performance typically requires a longer warm-up phase, during which the model searches for more suitable subnetworks and mitigates the bias inherited from previous tasks. With Fisher regularization, CAGNN can simultaneously achieve higher performance and lower parameter utilization with only a brief warm-up phase.

5.4.6 Inteference Analysis

We also conduct a case study on CoraFull to validate our theory by evaluating interference under different overlap levels. We use the fully learnable setting as the zero-interference baseline, where each task is trained without constraints from frozen parameters of previous tasks. As shown in Table 5.8, different sparsity levels with Fisher regularization, i.e., 0.99, 0.95, and 0.90, achieve highly similar task-wise performance to the zero-interference baseline, even though they induce high parameter overlap ratios around 88.5%–90.3%. This indicates that the proposed low-rank masking can tolerate substantial cross-task parameter reuse without introducing notable negative interference. The performance drops observed on several tasks are

Table 5.8: Performance comparison under different overlap ratios on the CoraFull dataset.

Overlap (Sparsity)	$\mathcal{T}_5$	$\mathcal{T}_{10}$	$\mathcal{T}_{15}$	$\mathcal{T}_{20}$	$\mathcal{T}_{25}$	$\mathcal{T}_{30}$	$\mathcal{T}_{35}$
0 Interference	95.71	98.67	98.51	88.66	98.27	100.00	99.38
22.8% (0.99 w/o Fisher)	96.32	99.11	98.51	85.57	97.84	100.00	99.38
88.5% (0.99)	95.71	99.11	98.51	87.63	98.70	100.00	99.38
89.7% (0.95)	96.32	98.67	98.51	87.63	98.70	100.00	100.00
90.3% (0.90)	96.32	99.11	98.51	88.66	97.84	100.00	100.00



also shared by the zero-interference baseline, suggesting that they mainly come from task difficulty rather than reuse-induced bias. In contrast, removing Fisher regularization greatly reduces the overlap ratio to 22.8%, showing that Fisher regularization effectively encourages reusable subnetwork structures. Overall, these results support our theoretical claim that the proposed two-stage masking scheme is robust to parameter reuse interference while enabling effective parameter sharing across tasks.

We also compare with neuron-wise masking, an efficiency-oriented scheme corresponding to Proposition 5.3.1, in Figure 5.9. Although neuron-wise masking maintains high parameter reuse on the synthetic CoraFull, such reuse does not always lead to beneficial knowledge transfer. When the task sequence becomes longer, strong interference from frozen parameters causes the model to fail to converge on some tasks.

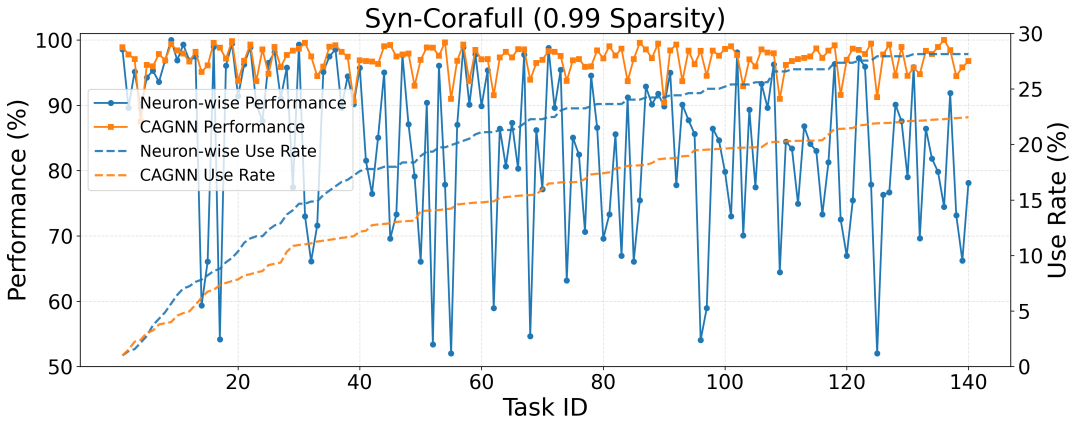


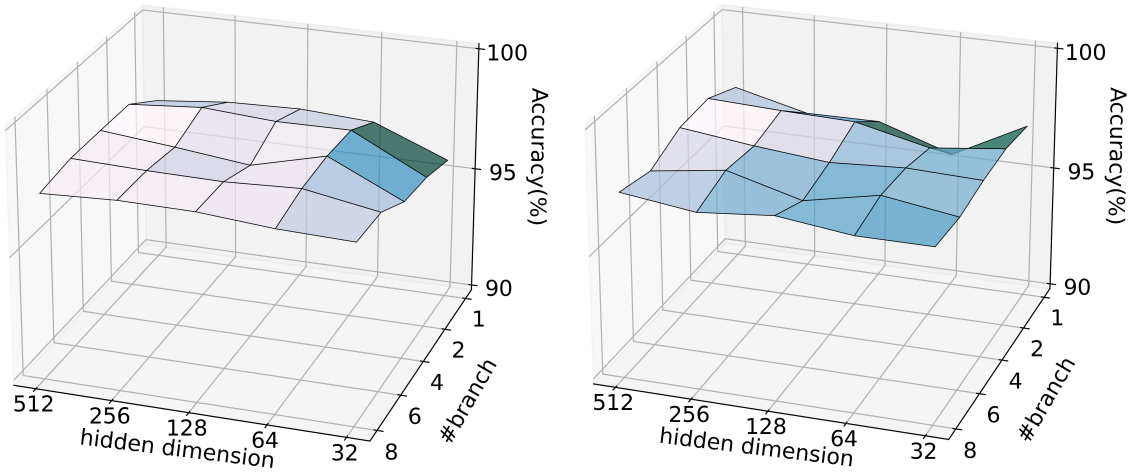
Figure 5.9: Performance comparison against neuron-wise masking.

5.4.7 Ablation and Sensitivity Analysis

Table 5.9: Ablation study of two-stage masking. For each dataset, the same sparsity is used across different stage variants.

Dataset	Stage 1		Stage 2		Stage 1&2	
	Ap	Use Rate	Ap	Use Rate	Ap	Use Rate
Cora (3)	93.15±1.3	29.10%	93.05±1.9	12.70%	94.41±1.6	13.90%
CiteSeer (3)	82.63±4.2	80.56%	83.57±3.2	60.50%	85.64±3.5	62.70%
CoraFull (35)	97.31±0.3	72.20%	97.18±0.4	96.10%	97.13±0.4	55.30%
Computers (5)	95.55±1.2	44.50%	95.92±1.9	34.70%	97.81±0.4	30.20%
OGB-Arxiv (8)	87.37±1.1	58.30%	89.67±1.4	69.60%	90.97±1.1	48.10%
MNIST (5)	96.89±1.2	62.80%	94.46±1.4	21.20%	96.04±1.7	22.10%
CIFAR10 (5)	82.21±1.3	56.10%	78.45±1.9	21.50%	81.29±1.1	19.90%
Aromaticity (10)	45.61±2.3	94.20%	43.38±2.0	84.90%	49.28±1.2	60.85%

The two-stage design is motivated by the limitations of using either stage alone. It aims to achieve fine-grained parameter masking while reducing interference and improving efficiency. Although two stages are designed as a unified masking scheme, we provide an ablation study in Table 5.9. Removing either stage only slightly affects performance, as each stage can independently function as a parameter-isolation structure. However, parameter reuse drops significantly, demonstrating the necessity of the two-stage design for scalable reuse.



Sparsity 0.99 Sparsity 0.95

Figure 5.10: Sensitivity to the Number of Branches and Hidden Dimension (Model Capacity) on the CoraFull Dataset

Continual learning is a long-term task that imposes stringent requirements on model stability. We also conduct sensitivity analysis on the CoraFull dataset to evaluate CAGNN's sensitivity to the hidden dimension and the number of branches, which collectively determine the model's capacity. As shown in Figure 5.10, CAGNN performs consistently and steadily with various model capacities. The sparsity levels do not have a significant influence on its performance on the CoraFull dataset. However, we observe that lower model capacity, such as various hidden dimensions with only 1 branch, tends to cause slightly lower performance and larger fluctuations, which indicates the necessity of over-parameterization to guarantee the stability as well as the scalability of the model. Generally, the maximum performance gap among various model capacities is only 2.17%, underscoring CAGNN's robustness and insensitivity to hyperparameters.

5.5 Conclusion

In this chapter, we investigate and identify the efficiency bottleneck of the parameter-isolation paradigm, where many methods still rely on full-network propagation and

their masked parameters remain in the forward pass, leading to increasing computational cost as capacity expands. Therefore, we formalize this catastrophic computational complexity issue and explore how data can be leveraged in the architecture design to address this problem. Observing that a real neuron possesses distal dendrites, which can perceive contextual information that enables the neuron to adapt to varying environments, we ultimately propose the novel dendrite-architectural CAGNN that is agnostic to the model capacity, enabling efficient training under long task streams while simultaneously maintaining high plasticity and scalability. By design, CAGNN adopts a two-stage architecture with hierarchical maskings, which can perceive task-specific context to guide the construction of task-specific subnetworks. Its dendritic structure design can further decouple each subnetwork from the full network for both training and inference to sustain efficiency under continual capacity expansion. We also introduce a Fisher regularization to enable more flexible cross-task parameter reuse. Comprehensive experiments on the model's plasticity, efficiency, scalability, and stability have elucidated that CAGNN consistently outperforms strong parameter-isolation baselines not only in catastrophic forgetting, but also in catastrophic computational complexity.

Chapter 6

Conclusion

This thesis investigates data-aware solutions for enhancing graph representation learning with Graph Neural Networks (GNNs) in complex, real-world settings, where simply increasing architectural sophistication is not always suitable. Revolving around the theme of data as the primary driver of downstream task success, the main contributions of this thesis follow a unified data-aware principle across three key levels of the representation learning pipeline. Each level targets a distinct problem setting, together reflecting the increasing complexity of real-world scenarios. Specifically, the objective level for conventional multi-label graph classification, the data level for graph-level anomaly detection, and the architecture level for continual graph learning. Ultimately, the thesis provides a systematic data-aware advance that enhances graph representation learning with GNNs to deliver more reliable and practical performance. In this chapter, we briefly recap the key contributions of each of the three works and then outline potential future research directions that are relevant to this thesis's theme.

6.1 Summary

Chapter 3 works on **Research Question 1** by revisiting the learning objective of GNN-based graph-level classification, and proposes Laplacian Eigenmaps Enhanced

Loss (LEELoss), a regularizer that augments the standard cross-entropy with graph-level priors. Conventional message-passing GNNs mainly operate information exchange and aggregation within individual graphs, providing limited channels for shaping the relative relationships among graph representations. To better exploit inter-graph information, LEELoss integrates it into a contrastive setting at the node level and graph similarity matrices at the graph level, explicitly pulling together representations of related instances while pushing apart irrelevant ones. Rather than keeping increasing focus on the message-passing and graph pooling layers to infer graph-level relationships, it highlights the potential of the objective function incorporating suitable graph priors to cope with various complex problems. As an objective-level advance, LEELoss is also plug-and-play, so that suitable GNNs and downstream heads can be readily paired with it without modifying the underlying architecture.

Chapter 4 tackles **Research Question 2** from a data-centric view by studying supervised graph-level anomaly detection, and develops Graph Augmentation-based Reference Alignment (GARA), an effective training framework that tends to make full utilization of limited but precise anomalies to enhance GNNs' generalization capability for future unseen cases. Although recent efforts have shown that a small number of labeled anomalies can substantially improve detection performance, most existing methods still overemphasize the detailed patterns of anomalies. Such a manner is inherently fragile because anomalous graphs are often incomplete, biased, and heterogeneous, and thus fitting to their detailed patterns poses the model at risk. Under these circumstances, GARA leverages graph mixup techniques with two different anomaly-specific strategies to enrich the sample pool of anomalies and construct balanced learning tasks to simulate an appropriate training environment in conventional supervised learning. It also emphasizes the role of normal instances as unsupervised learning does, and thus leverages abundant normal samples as solid references to refine decision boundaries between normal and anomalous graphs. The data-level advance enables GARA to be more adaptive to various more challenging

graph anomaly detection scenarios when compared with methods that specifically model existing but limited anomalies.

Chapter 5 is dedicated to addressing **Research Question 3** by investigating the computational efficiency challenge of parameter isolation-based graph representation learning under long task streams. While parameter isolation effectively mitigates catastrophic forgetting, many existing methods still perform full-network propagation, where masked parameters continue to participate in computation, and the overhead accumulates as model capacity expands to accommodate future tasks. This chapter identifies and formalizes this problem as severe as catastrophic forgetting, and proposes a Capacity-Agnostic GNN framework CAGNN, which possesses a two-stage biological neuron-inspired architecture that forms a hierarchical masking scheme. By perceiving task contextual information, the proposed framework guides efficient construction of task-specific subnetworks and then decouples them from the holistic network during both training and inference, thereby sustaining efficiency even under continual capacity expansion. In this chapter, we propose a gradient proxy to ensure effective subnetwork constructions and provide theoretical justification for the effectiveness of such dendritic architecture. Moreover, we introduce a Fisher information-based regularization that enables flexible cross-task parameter reuse for greater scalability. Ultimately, this chapter bridges data and architectural design in a natural and coherent manner, leading to a data-aware advance at the architecture level.

6.2 Future Outlook

We realize this thesis's efforts to provide a comprehensive insight into practical utility. However, real-world applications usually involve a wider range of complexities than any single benchmark can fully reflect, and still rely on mature, lightweight, and easy-to-deploy models. Considering the ongoing wave of data-centric research, future work can further extend the data-aware view toward a more thoroughly

data-centric paradigm, where the emphasis shifts from "designing a sophisticated end-to-end mechanism to fit an idealized setting" to "designing data, supervision, and suitable training schemes that make representation learning more reliable under challenging conditions." Such conditions include graphs with high heterophily levels, distribution shift of temporal graphs across time, and time series represented by graphs, and so on.

Another promising direction is to generalize the data-aware principle from conventional GNNs to emerging graph foundation models and graph-oriented large models. As pretraining and instruction-tuning become increasingly prevalent for representation learning in these areas, this principle suggests that performance gains may be driven less by architectural novelty alone but more by how graph corpora are constructed, how weak or implicit supervisory signals can be extracted and leveraged from large-scale graph data, and how pretraining objectives align with downstream task semantics. Concretely, one may explore adaptation objectives that incorporate graph priors to calibrate pretraining signals (objective level), or task-tailored augmentation and training strategies that enrich adaptation corpora (data level), or efficient task-aware adapters that enable scalable and continual adaptation (architecture level). Together, these directions can form a pathway to apply the data-aware principle across different levels of graph representation learning, while leveraging the rise of graph foundation models to support a wider range of complex and evolving real-world graph tasks.

References

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süsstrunk, S. (2012). Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, *34*(11), 2274–2282.

Aggarwal, C. C., & Aggarwal, C. C. (2017). *An introduction to outlier analysis*. Springer.

Aghajanyan, A., Gupta, S., & Zettlemoyer, L. (2021). Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)* (pp. 7319–7328).

Aich, A. (2021). Elastic weight consolidation (ewc): Nuts and bolts. *arXiv preprint arXiv:2105.04093*.

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the acm international conference on knowledge discovery and data mining* (pp. 2623–2631).

Akoglu, L., Tong, H., & Koutra, D. (2015). Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery*, *29*(3), 626–688.

Alon, U., & Yahav, E. (2021). On the bottleneck of graph neural networks and its practical implications. In *International conference on learning representations (iclr)*.

Belkin, M., & Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, *14*.

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

Bengio, Y., Léonard, N., & Courville, A. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Bojchevski, A., & Günnemann, S. (2018). Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International conference on learning representations (iclr)*.

Borgwardt, K. M., & Kriegel, H.-P. (2005). Shortest-path kernels on graphs. In *Icdm* (pp. 8–pp).

Bouritsas, G., Frasca, F., Zafeiriou, S., & Bronstein, M. M. (2022). Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(1), 657–668.

Bronstein, M. M., Bruna, J., Cohen, T., & Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*.

Brun, A., Park, H.-J., Knutsson, H., & Westin, C.-F. (2003). Coloring of DT-MRI fiber traces using Laplacian eigenmaps. In *Proceedings of the international conference on computer aided systems theory* (pp. 518–529).

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

Cai, H., Zheng, V. W., & Chang, K. C.-C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE transactions on knowledge and data engineering*, *30*(9), 1616–1637.

Caruana, R. (1997). Multitask learning. *Machine learning*, *28*(1), 41–75.

Chan, K., Stephen, N., & Young, K. (2011). Perturbation theory and the rayleigh quotient. *Journal of sound and vibration*, *330*(9), 2073–2078.

Chen, F., Wang, Y.-C., Wang, B., & Kuo, C.-C. J. (2020). Graph representation learning: a survey. *APSIPA Transactions on Signal and Information Processing*, *9*, e15.

Chen, J., Ma, T., & Xiao, C. (2018). Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*.

Chen, T., & Wong, R. C.-W. (2020). Handling information loss of graph neural networks for session-based recommendation. In *Proceedings of the 26th acm sigkdd international conference on knowledge discovery & data mining* (pp. 1172–1180).

Chen, Z., Chen, C., Zhang, Z., Zheng, Z., & Zou, Q. (2019). Variational graph embedding and clustering with laplacian eigenmaps. In *Ijcai* (pp. 2144–2150).

Cheng, Y., Wang, D., Zhou, P., & Zhang, T. (2017). A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*.

Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., & Hsieh, C.-J. (2019). Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining* (pp. 257–266).

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

Coifman, R. R., Lafon, S., Lee, A. B., Maggioni, M., Nadler, B., Warner, F., & Zucker, S. W. (2005). Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *PNAS*, *102*(21), 7426–7431.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., & Veličković, P. (2020). Principal neighbourhood aggregation for graph nets. *Advances in neural information processing systems*, *33*, 13260–13271.

Cui, Y., Wang, Y., Sun, Z., Liu, W., Jiang, Y., Han, K., & Hu, W. (2023). Lifelong embedding learning and transfer for growing knowledge graphs. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 37, pp. 4217–4224).

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

De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., ... Tuytelaars, T. (2021). A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, *44*(7), 3366–3385.

Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., & Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, *27*.

Diaconis, P., & Janson, S. (2007). Graph limits and exchangeable random graphs. *arXiv preprint arXiv:0712.2749*.

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

Ding, K., Zhou, Q., Tong, H., & Liu, H. (2021). Few-shot network anomaly detection via cross-network meta-learning. In *Proceedings of the international world wide web conference* (pp. 2448–2456).

Dong, X., Zhang, X., & Wang, S. (2024). Rayleigh quotient graph neural networks for graph-level anomaly detection. In *International conference on learning representations (iclr)* (Vol. 2024, pp. 53937–53955).

Dwivedi, V. P., Joshi, C. K., Luu, A. T., Laurent, T., Bengio, Y., & Bresson, X. (2023). Benchmarking graph neural networks. *Journal of Machine Learning Research*, *24*(43), 1–48.

Erdős, P., & Rényi, A. (1960). On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, *5*(1), 17–60.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., & Yin, D. (2019). Graph neural networks for social recommendation. In *The world wide web conference* (pp. 417–426).

Febrinanto, F. G., Xia, F., Moore, K., Thapa, C., & Aggarwal, C. (2023). Graph lifelong learning: A survey. *IEEE Computational Intelligence Magazine*, *18*(1), 32–51.

Fiala, J. C., & Harris, K. M. (1999). Dendrite structure. *Dendrites*, *2*, 1–11.

Frankle, J., & Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International conference on learning representations (iclr)*.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, *3*(4), 128–135.

Gao, S., Zhou, M., Wang, Y., Cheng, J., Yachi, H., & Wang, J. (2018). Dendritic neuron model with effective learning algorithms for classification, approximation,

and prediction. *IEEE transactions on neural networks and learning systems*, *30*(2), 601–614.

Gao, S., Zhou, M., Wang, Z., Sugiyama, D., Cheng, J., Wang, J., & Todo, Y. (2021). Fully complex-valued dendritic neuron model. *IEEE transactions on neural networks and learning systems*, *34*(4), 2105–2118.

Gao, Y., Wang, X., He, X., Liu, Z., Feng, H., & Zhang, Y. (2023). Alleviating structural distribution shift in graph anomaly detection. In *Proceedings of the acm international conference on web search and data mining* (pp. 357–365).

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

Golkar, S., Kagan, M., & Cho, K. (2019). Continual learning via neural pruning. *arXiv preprint arXiv:1903.04476*.

Gori, M., Monfardini, G., & Scarselli, F. (2005). A new model for learning in graph domains. In *Proceedings. 2005 ieee international joint conference on neural networks, 2005.* (Vol. 2, pp. 729–734).

Goyal, P., & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, *151*, 78–94.

Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 ieee international conference on acoustics, speech and signal processing* (pp. 6645–6649).

Grewal, K., Forest, J., Cohen, B. P., & Ahmad, S. (2021). Going beyond the point neuron: Active dendrites and sparse representations for continual learning. *bioRxiv*.

Gunasekar, S., Woodworth, B. E., Bhojanapalli, S., Neyshabur, B., & Srebro, N. (2017). Implicit regularization in matrix factorization. *Advances in neural information processing systems*, *30*.

Guo, H., & Mao, Y. (2023a). Interpolating graph pair to regularize graph classification. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 37, pp. 7766–7774).

Guo, H., & Mao, Y. (2023b). Interpolating graph pair to regularize graph classification. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 37, p. 7766-7774).

Guo, Y., Bo, D., Yang, C., Lu, Z., Zhang, Z., Liu, J., ... Shi, C. (2024). Data-centric graph learning: A survey. *IEEE Transactions on Big Data*.

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in neural information processing systems*, *30*.

Hamilton, W. L. (2020). *Graph representation learning*. Morgan & Claypool Publishers.

Han, X., Jiang, Z., Liu, N., & Hu, X. (2022). G-mixup: Graph data augmentation for graph classification. In *International conference on machine learning* (pp. 8230–8248).

Harnett, M. T., Makara, J. K., Spruston, N., Kath, W. L., & Magee, J. C. (2012). Synaptic amplification by dendritic spines enhances input cooperativity. *Nature*, *491*(7425), 599–602.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., & Wang, M. (2020). Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd international acm sigir conference on research and development in information retrieval* (pp. 639–648).

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... others (2022). Lora: Low-rank adaptation of large language models. In *International conference on learning representations (iclr)* (Vol. 1, p. 3).

Hu, S., Zou, G., Yang, S., Lin, S., Gan, Y., & Zhang, B. (2025). Dynamic graph representation learning via edge temporal states modeling and structure-reinforced transformer. *Knowledge-Based Systems*, 113661.

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

Huang, R., Sakamuru, S., Martin, M. T., Reif, D. M., Judson, R. S., Houck, K. A., ... others (2014). Profiling of the tox21 10k compound library for agonists and antagonists of the estrogen receptor alpha signaling pathway. *Scientific reports*, *4*(1), 5664.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Icml* (pp. 448–456).

Iyer, A., Grewal, K., Velu, A., Souza, L. O., Forest, J., & Ahmad, S. (2022). Avoiding catastrophe: Active dendrites enable multi-task learning in dynamic environments. *Frontiers in neurorobotics*, *16*, 846219.

Jang, E., Gu, S., & Poole, B. (2016). Categorical reparameterization with gumbel-softmax. In *International conference on learning representations (iclr).*

Jia, T., Li, H., Yang, C., Tao, T., & Shi, C. (2024). Graph invariant learning with subgraph co-mixup for out-of-distribution generalization. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 38, pp. 8562–8570).

Jiang, Q., & Ma, J. (2018). A novel graph kernel on chemical compound classification. *Journal of Bioinformatics and Computational Biology*, *16*(06), 1850026.

Kaler, T., Stathas, N., Ouyang, A., Iliopoulos, A.-S., Schardl, T., Leiserson, C. E., & Chen, J. (2022). Accelerating training and inference of graph neural networks with fast sampling and pipelining. *Proceedings of Machine Learning and Systems*, *4*, 172–189.

Kang, H., Mina, R. J. L., Madjid, S. R. H., Yoon, J., Hasegawa-Johnson, M., Hwang, S. J., & Yoo, C. D. (2022). Forget-free continual learning with winning subnetworks. In *International conference on machine learning* (pp. 10734–10750).

Kang, H., Yoon, J., Hwang, S. J., & Yoo, C. D. (2024). Continual learning: Forget-free winning subnetworks for video representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Khoshraftar, S., & An, A. (2024). A survey on graph representation learning methods. *ACM Transactions on Intelligent Systems and Technology*, *15*(1), 1–55.

Kim, H., Lee, B. S., Shin, W.-Y., & Lim, S. (2022). Graph anomaly detection with graph neural networks: Current status and challenges. *IEEE Access*, *10*, 111820–111829.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv:1412.6980*.

Kipf, T. N., & Welling, M. (2016). Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International conference on learning representations (iclr)*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., ... others (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, *114*(13), 3521–3526.

Ko, J., Kang, S., Kwon, T., Moon, H., & Shin, K. (2025). Begin: Extensive benchmark scenarios and an easy-to-use framework for graph continual learning. *ACM Transactions on Intelligent Systems and Technology*, *16*(1), 1–22.

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

Kriege, N. M., Giscard, P.-L., & Wilson, R. (2016). On valid optimal assignment kernels and applications to graph classification. *NeurIPS*, *29*.

Kriege, N. M., Johansson, F. D., & Morris, C. (2020). A survey on graph kernels. *Applied Network Science*, *5*(1), 6.

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., ... others (2017). Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, *123*(1), 32–73.

Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, *8*(1), 98–113.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

Lee, J., Lee, I., & Kang, J. (2019). Self-attention graph pooling. In *International conference on machine learning* (pp. 3734–3743).

Lee, J. B., Rossi, R., & Kong, X. (2018). Graph classification using structural attention. In *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining* (pp. 1666–1674).

Levie, R., Monti, F., Bresson, X., & Bronstein, M. M. (2018). Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Trans. Signal Process.*, *67*(1), 97–109.

Li, B., Li, Y.-R., & Zhang, X.-L. (2019). A survey on Laplacian eigenmaps based manifold learning methods. *Neurocomputing*, *335*, 336–351.

Li, C., Farkhoor, H., Liu, R., & Yosinski, J. (2018). Measuring the intrinsic dimension of objective landscapes. In *International conference on learning representations (iclr).*

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

Li, P., Chen, X., & Shen, S. (2019). Stereo r-cnn based 3d object detection for autonomous driving. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 7644–7652).

Li, R., Xiao, Y., Ma, X., Vasnev, A., & Gao, J. (2025). Gdendrite: On heterophilous graph contexts mining with versatile neural dendrites framework. In *Proceedings of the 31st acm sigkdd conference on knowledge discovery and data mining v. 2* (pp. 1505–1516).

Li, Y., Gu, C., Dullien, T., Vinyals, O., & Kohli, P. (2019). Graph matching networks for learning the similarity of graph structured objects. In *Proceedings of the international conference on machine learning* (pp. 3835–3845).

Li, Z., & Hoiem, D. (2017). Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, *40*(12), 2935–2947.

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, *33*(12), 6999–7019.

Li, Z.-P., Su, H.-L., Zhu, X.-B., Wei, X.-M., Jiang, X.-S., Gribova, V., ... Huang, D.-S. (2021). Hierarchical graph pooling with self-adaptive cluster aggregation. *IEEE Transactions on Cognitive and Developmental Systems*, *14*(3), 1198–1207.

Lim, S. K., Loo, Y., Tran, N.-T., Cheung, N.-M., Roig, G., & Elovici, Y. (2018). Doping: Generative data augmentation for unsupervised anomaly detection with gan. In *Proceedings of the ieee international conference on data mining* (pp. 1122–1127).

Lin, T., Wang, Y., Liu, X., & Qiu, X. (2022). A survey of transformers. *AI open*, *3*, 111–132.

Ling, H., Jiang, Z., Liu, M., Ji, S., & Zou, N. (2023). Graph mixup with soft alignments. In *International conference on machine learning* (pp. 21335–21349).

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., ... Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical image analysis*, *42*, 60–88.

Liu, C., Zhan, Y., Wu, J., Li, C., Du, B., Hu, W., ... Tao, D. (2022). Graph pooling for graph neural networks: Progress, challenges, and opportunities. *arXiv preprint arXiv:2204.07321*.

Liu, F., Ma, X., Wu, J., Yang, J., Xue, S., Beheshti, A., ... Aggarwal, C. C. (2022). Dagad: Data augmentation for graph anomaly detection. In *Proceedings of the ieee international conference on data mining* (pp. 259–268).

Liu, H., Yang, Y., & Wang, X. (2021). Overcoming catastrophic forgetting in graph neural networks. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 35, pp. 8653–8661).

Liu, J., Xia, F., Wang, L., Xu, B., Kong, X., Tong, H., & King, I. (2019). Shifu2: A network representation learning based model for advisor-advisee relationship

mining. *IEEE Transactions on Knowledge and Data Engineering*, *33*(4), 1763–1777.

Liu, J., & Zhang, X. (2026). Taam: Inductive graph-class incremental learning with task-aware adaptive modulation. In *Proceedings of the 25th international conference on autonomous agents and multiagent systems* (pp. 3078–3086).

Liu, Y., Ding, K., Lu, Q., Li, F., Zhang, L. Y., & Pan, S. (2024). Towards self-interpretable graph-level anomaly detection. In *Proceedings of the advances in neural information processing systems* (Vol. 36).

Liu, Y., Jin, M., Pan, S., Zhou, C., Zheng, Y., Xia, F., & Yu, P. S. (2022). Graph self-supervised learning: A survey. *IEEE transactions on knowledge and data engineering*, *35*(6), 5879–5900.

Liu, Z., Chen, C., Yang, X., Zhou, J., Li, X., & Song, L. (2018). Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th acm international conference on information and knowledge management* (pp. 2077–2085).

Lopez-Paz, D., & Ranzato, M. (2017). Gradient episodic memory for continual learning. *Advances in neural information processing systems*, *30*.

Luo, W. (2011). Face recognition based on Laplacian eigenmaps. In *Csss* (pp. 416–419).

Luo, X., Wu, J., Yang, J., Xue, S., Peng, H., Zhou, C., ... Sheng, Q. Z. (2022). Deep graph level anomaly detection with contrastive learning. *Scientific Reports*, *12*(1), 19867.

Ma, R., Pang, G., Chen, L., & van den Hengel, A. (2022). Deep graph-level anomaly detection by glocal knowledge distillation. In *Proceedings of the acm international conference on web search and data mining* (pp. 704–714).

Ma, X., Wu, J., Xue, S., Yang, J., Zhou, C., Sheng, Q. Z., ... Akoglu, L. (2021). A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering*, *35*(12), 12012–12038.

Ma, X., Wu, J., Yang, J., & Sheng, Q. Z. (2023). Towards graph-level anomaly detection via deep evolutionary mapping. In *Proceedings of the 29th acm sigkdd conference on knowledge discovery and data mining* (pp. 1631–1642).

Ma, X., Zhang, X., Tang, X., Zhou, H., & Jiao, L. (2020). Hyperspectral anomaly detection based on low-rank representation with data-driven projection and dictionary construction. *Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *13*, 2226–2239.

Maddison, C. J., Mnih, A., & Teh, Y. W. (2016). The concrete distribution: A continuous relaxation of discrete random variables. In *International conference on learning representations.*

Mallya, A., Davis, D., & Lazebnik, S. (2018). Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the european conference on computer vision (eccv)* (pp. 67–82).

Mallya, A., & Lazebnik, S. (2018). Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 7765–7773).

Martinkus, K., Loukas, A., Perraudin, N., & Wattenhofer, R. (2022). Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *International conference on machine learning* (pp. 15159–15179).

McAuley, J., Targett, C., Shi, Q., & Van Den Hengel, A. (2015). Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international acm sigir conference on research and development in information retrieval* (pp. 43–52).

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

Merkwirth, C., & Lengauer, T. (2005). Automatic generation of complementary descriptors with molecular graph networks. *Journal of chemical information and modeling*, *45*(5), 1159–1168.

Mi, J.-X., Huang, D.-S., Wang, B., & Zhu, X. (2013). The nearest-farthest subspace classification for face recognition. *Neurocomputing*, *113*, 241–250.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., & Neumann, M. (2020). Tudataset: A collection of benchmark datasets for learning with graphs. *ArXiv:2007.08663*.

Newman, M. E. (2012). Communities, modules and large-scale structure in networks. *Nature physics*, *8*(1), 25–31.

Niepert, M., Ahmed, M., & Kutzkov, K. (2016). Learning convolutional neural networks for graphs. In *Icml* (pp. 2014–2023).

Noumir, Z., Honeine, P., & Richard, C. (2012). On simple one-class classification methods. In *2012 ieee international symposium on information theory proceedings* (pp. 2022–2026).

Nt, H., & Maehara, T. (2019). Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The pagerank citation ranking: Bringing order to the web.* (Tech. Rep.). Stanford infolab.

Pan, S., Wu, J., Zhu, X., Long, G., & Zhang, C. (2016). Task sensitive feature exploration and learning for multitask graph classification. *IEEE Transactions on Cybernetics*, *47*(3), 744–758.

Pan, S., Wu, J., Zhu, X., & Zhang, C. (2014). Graph ensemble boosting for imbalanced noisy graph stream classification. *IEEE transactions on cybernetics*, *45*(5), 954–968.

Pang, G., Shen, C., Cao, L., & Hengel, A. V. D. (2021). Deep learning for anomaly detection: A review. *ACM computing surveys (CSUR)*, *54*(2), 1–38.

Pang, G., Shen, C., Jin, H., & van den Hengel, A. (2023). Deep weakly-supervised anomaly detection. In *Proceedings of the acm conference on knowledge discovery and data mining* (pp. 1795–1807).

Pham, T.-C., Doucet, A., Luong, C.-M., Tran, C.-T., & Hoang, V.-D. (2020). Improving skin-disease classification based on customized loss function combined with balanced mini-batch logic and real-time image augmentation. *IEEE Access*, *8*, 150725–150737.

Qiao, H., Tong, H., An, B., King, I., Aggarwal, C., & Pang, G. (2025). Deep graph anomaly detection: A survey and new perspectives. *IEEE Transactions on Knowledge and Data Engineering*.

Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., & Cottrell, G. (2017). A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*.

Qiu, C., Kloft, M., Mandt, S., & Rudolph, M. (2022). Raising the bar in graph-level anomaly detection. In *Proceedings of the international joint conference on artificial intelligence*.

Rall, W. (1962). Electrophysiology of a dendritic neuron model. *Biophysical journal*, *2*(2 Pt 2), 145.

Ring, M. B. (1994). *Continual learning in reinforcement environments*. The University of Texas at Austin.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*(5500), 2323–2326.

Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2017). Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*.

Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN computer science*, *2*(3), 160.

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

Schachter, M. J., Oesch, N., Smith, R. G., & Taylor, W. R. (2010). Dendritic spikes amplify the synaptic signal to enhance detection of motion in a simulation of the direction-selective ganglion cell. *PLoS computational biology*, *6*(8), e1000899.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., ... Dennison, D. (2015). Hidden technical debt in machine learning systems. *Advances in neural information processing systems*, *28*.

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

Serra, J., Suris, D., Miron, M., & Karatzoglou, A. (2018). Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning* (pp. 4548–4557).

Shi, Z., Wang, J., Zhuang, Z., Liang, X., Li, B., & Wu, F. (2025). Accurate and scalable graph neural networks via message invariance. *arXiv preprint arXiv:2502.19693*.

Shimizu, R., Asako, K., Ojima, H., Morinaga, S., Hamada, M., & Kuroda, T. (2018). Balanced mini-batch training for imbalanced image data classification with neural

network. In *Proceedings of the international conference on artificial intelligence for industries (ai4i)* (pp. 27–30).

Siglidis, G., Nikolentzos, G., Limnios, S., Giatsidis, C., Skianis, K., & Vazirgiannis, M. (2020). GraKeL: a graph kernel library in python. *JMLR*, *21*(54), 1–5.

Song, J., Qu, X., Hu, Z., Li, Z., Gao, J., & Zhang, J. (2021). A subgraph-based knowledge reasoning method for collective fraud detection in e-commerce. *Neurocomputing*, *461*, 587–597.

Sun, X., Zhang, C., Li, G., Sun, D., Ren, F., Zomaya, A., & Ranjan, R. (2018). Detecting users' anomalous emotion using social media for business intelligence. *Journal of Computational Science*, *25*, 193–200.

Sun, Z.-L., Huang, D.-S., Cheung, Y.-M., Liu, J., & Huang, G.-B. (2005). Using fcmc, fvs, and pca techniques for feature extraction of multispectral images. *IEEE GRSL*, *2*(2), 108–112.

Tenenbaum, J. B., Silva, V. d., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*(5500), 2319–2323.

Todo, Y., Tang, Z., Todo, H., Ji, J., & Yamashita, K. (2019). Neurons with multiplicative interactions of nonlinear synapses. *International journal of neural systems*, *29*(08), 1950012.

Trivedi, P., Lubana, E. S., Yan, Y., Yang, Y., & Koutra, D. (2022). Augmentations in graph contrastive learning: Current methodological flaws & towards better practices. In *Proceedings of the acm web conference 2022* (pp. 1538–1549).

Veličković, P. (2023). Everything is connected: Graph neural networks. *Current Opinion in Structural Biology*, *79*, 102538.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2017). Graph attention networks. In *International conference on learning representations (iclr).*

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *International conference on learning representations.*

Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., & Hjelm, R. D. (2018). Deep graph infomax. *arXiv preprint arXiv:1809.10341*.

Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., & Bengio, Y. (2019). Manifold mixup: Better representations by interpolating hidden states. In *Proceedings of the international conference on machine learning* (pp. 6438–6447).

Wang, D., Lin, J., Cui, P., Jia, Q., Wang, Z., Fang, Y., ... Qi, Y. (2019). A semi-supervised graph attentive network for financial fraud detection. In *2019 ieee international conference on data mining (icdm)* (pp. 598–607).

Wang, H., Feng, J., Peng, L., Pan, S., Zhao, S., & Jin, H. (2022). Anomaly detection of multivariate time series based on metric learning. In *Proceedings of the international conference of pioneering computer scientists, engineers and educators* (pp. 94–110).

Wang, K., Shen, Z., Huang, C., Wu, C.-H., Dong, Y., & Kanakia, A. (2020). Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, *1*(1), 396–413.

Wang, L., Zhang, X., Su, H., & Zhu, J. (2024). A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, *46*(8), 5362–5383.

Wang, X., Jin, B., Du, Y., Cui, P., Tan, Y., & Yang, Y. (2021). One-class graph neural networks for anomaly detection in attributed networks. *Neural Computing and Applications*, *33*, 12073–12085.

Wang, Y., Liu, Y., Shen, X., Li, C., Ding, K., Miao, R., ... Wang, X. (2024). Unifying unsupervised graph-level anomaly detection and out-of-distribution detection: A benchmark. *ArXiv:2406.15523*.

Wang, Z., & Ji, S. (2020). Second-order pooling for graph neural networks. *TPAMI*(01), 1–1.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. (2019). Simplifying graph convolutional networks. In *International conference on machine learning* (pp. 6861–6871).

Wu, J., Chen, X., Xu, K., & Li, S. (2022). Structural entropy guided graph hierarchical pooling. In *International conference on machine learning* (pp. 24017–24030).

Wu, S., Sun, F., Zhang, W., Xie, X., & Cui, B. (2022). Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, *55*(5), 1–37.

Wu, X., Liu, X., Li, W., & Wu, Q. (2018). Improved expressivity through dendritic neural networks. *Advances in neural information processing systems*, *31*.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, *32*(1), 4–24.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., ... Pande, V. (2018). Moleculenet: a benchmark for molecular machine learning. *Chemical science*, *9*(2), 513–530.

Xiao, S., Wang, S., Dai, Y., & Guo, W. (2022). Graph neural networks in node classification: survey and evaluation. *Machine Vision and Applications*, *33*(1), 4.

Xie, H., Ma, J., Xiong, L., & Yang, C. (2021). Federated graph classification over non-iid graphs. *Advances in neural information processing systems*, *34*, 18839–18852.

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Xu, Y., Zhang, Y., Guo, W., Guo, H., Tang, R., & Coates, M. (2020). Graphsail: Graph structure aware incremental learning for recommender systems. In *Proceedings of the 29th acm international conference on information & knowledge management* (pp. 2861–2868).

Xu, Z., Huang, X., Zhao, Y., Dong, Y., & Li, J. (2022). Contrastive attributed network anomaly detection with data augmentation. In *Proceedings of the pacific-asia conference on knowledge discovery and data mining* (pp. 444–457).

Yanardag, P., & Vishwanathan, S. (2015). Deep graph kernels. In *Acm sigkdd* (pp. 1365–1374).

Yang, G. R., Murray, J. D., & Wang, X.-J. (2016). A dendritic disinhibitory circuit mechanism for pathway-specific gating. *Nature communications*, *7*(1), 12815.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining* (pp. 974–983).

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., & Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, *31*.

You, J., Ying, R., Ren, X., Hamilton, W., & Leskovec, J. (2018). Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning* (pp. 5708–5717).

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., & Shen, Y. (2020). Graph contrastive learning with augmentations. *Advances in neural information processing systems*, *33*, 5812–5823.

Yuan, Q., Guan, S.-U., Ni, P., Luo, T., Wong, P., Chang, V., & Man, K. L. (2026). Continual graph learning: A survey. *Pattern Recognition*, 113600.

Zeng, D., Liu, W., Chen, W., Zhou, L., Zhang, M., & Qu, H. (2023). Substructure aware graph neural networks. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 37, pp. 11129–11137).

Zeng, J., & Xie, P. (2021). Contrastive self-supervised learning for graph classification. In *Aaai* (Vol. 35, pp. 10824–10832).

Zha, D., Bhat, Z. P., Lai, K.-H., Yang, F., Jiang, Z., Zhong, S., & Hu, X. (2025). Data-centric artificial intelligence: A survey. *ACM Computing Surveys*, *57*(5), 1–42.

Zhang, G., Li, Z., Huang, J., Wu, J., Zhou, C., Yang, J., & Gao, J. (2022). efraudcom: An e-commerce fraud detection system via competitive graph neural networks. *ACM Transactions on Information Systems*, *40*(3), 1–29.

Zhang, G., Yang, Z., Wu, J., Yang, J., Xue, S., Peng, H., ... others (2022). Dual-discriminative graph neural network for imbalanced graph-level anomaly detection. *Advances in Neural Information Processing Systems*, *35*, 24144–24157.

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. In *Proceedings of the international conference on learning representations*.

Zhang, M., & Chen, Y. (2018). Link prediction based on graph neural networks. *NeurIPS*, *31*.

Zhang, M., Cui, Z., Neumann, M., & Chen, Y. (2018). An end-to-end deep learning architecture for graph classification. In *Aaai* (Vol. 32).

Zhang, P., Yan, Y., Li, C., Wang, S., Xie, X., Song, G., & Kim, S. (2023). Continual learning on dynamic graphs via parameter isolation. In *Proceedings of the 46th international acm sigir conference on research and development in information retrieval* (pp. 601–611).

Zhang, X., Song, D., Chen, Y., & Tao, D. (2024). Topology-aware embedding memory for continual learning on expanding networks. In *Proceedings of the 30th acm sigkdd conference on knowledge discovery and data mining* (pp. 4326–4337).

Zhang, X., Song, D., & Tao, D. (2022). Hierarchical prototype networks for continual graph representation learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(4), 4622–4636.

Zhang, X., Song, D., & Tao, D. (2024). Continual learning on graphs: Challenges, solutions, and opportunities. *arXiv preprint arXiv:2402.11565*.

Zhao, L., & Akoglu, L. (2023). On using classification datasets to evaluate graph outlier detection: Peculiar observations and new insights. *Big Data*, *11*(3), 151–180.

Zhao, L., Sawlani, S., Srinivasan, A., & Akoglu, L. (2022). Graph anomaly detection with unsupervised gnns. *arXiv:2210.09535*.

Zhou, F., & Cao, C. (2021). Overcoming catastrophic forgetting in graph neural networks with experience replay. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 35, pp. 4714–4722).

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., ... Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI open*, *1*, 57–81.

Zhou, J., Xie, C., Gong, S., Wen, Z., Zhao, X., Xuan, Q., & Yang, X. (2025). Data augmentation on graphs: A technical survey. *ACM Computing Surveys*, *57*(11), 1–34.

Zhu, H., Sun, K., & Koniusz, P. (2021). Contrastive Laplacian eigenmaps. *NeurIPS*, *34*, 5682–5695.

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., & Koutra, D. (2020). Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, *33*, 7793–7804.

Zhu, Q., Zhang, C., Park, C., Yang, C., & Han, J. (2022). Shift-robust node classification via graph clustering co-training. In *Neurips 2022 workshop: New frontiers in graph learning.*

Zhu, X., Ghahramani, Z., & Lafferty, J. D. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th international conference on machine learning (icml-03)* (pp. 912–919).

Zou, D., Hu, Z., Wang, Y., Jiang, S., Sun, Y., & Gu, Q. (2019). Layer-dependent importance sampling for training deep and large graph convolutional networks. *Advances in neural information processing systems*, *32*.