TOWARDS GRAPH FOUNDATION MODEL: FROM NETWORK SCIENCE THEORY TO PRACTICE APPLICATION

By

Haitao Mao

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science—Doctor of Philosophy

2025

ABSTRACT

Graphs, which abstract complex systems of relations and interactions, can model transportation networks, trading networks, epidemic networks, the World Wide Web, and more. Graph Neural Networks (GNNs), aiming to adaptively integrate both feature and structure knowledge on networks, have emerged as a popular graph representation learning technique. The key component in GNNs is the non-parameterized message-passing operator, which updates node representation via aggregating neighbor node messages. Message-passing operators generally follow the algorithm alignment design principle, where the operator should align with the task principle or the corresponding graph algorithms, e.g., Common Neighbor for friend recommendation, PageRank for web Search, and Dijkstra algorithm for finding the shortest path.

Despite the initial success achieved by GNNs, most of them are end-to-end trained and evaluated on one single dataset. Consequently, an architectural overfitting phenomenon can be found where GNNs generally cannot perform well across graphs with various properties, as they are specifically designed to beat existing methods on the limited benchmark dataset. Such obstacle makes it difficult for practitioners to determine which models will generalize effectively to unseen datasets. Thereby, a new but necessary requirement for graph representation learning is to be expressive enough to capture diverse properties and adapt across different graphs. To achieve this ultimate goal, I first endeavor to elucidate all the underlying network patterns across diverse graphs with a properly designed random network model. Theoretical analysis is further conducted to principally characterize the relationship between different network patterns. Secondly, I conduct analysis on GNN's inner mechanism about which network patterns GNNs can capture and which not. With the help of the random network model, I can identify the inevitable but largely ignored limitation of GNNs. Finally, I develop Graph Foundation Models (GFMs) to address the incapability of GNNs. GFMs comprehensively consider the training data diversity, expressive architecture design, and suitable training objective, which moderates the over-reliance on the algorithm alignment and captures diverse network patterns on different graphs.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my family for their unconditional love and support. Their unwavering belief in me has been a constant source of strength throughout this journey. They have stood by me through every challenge and triumph. Their patience, sacrifices, and boundless kindness have shaped me in ways words cannot fully express. This achievement would not have been possible without their love, and for that, I am forever grateful.

I would like to express my gratitude to my supervisor, Dr. Jiliang Tang. Over the years, I have learned invaluable lessons from him—not only in constructing rigorous research but also in being a kind and principled person, as well as in mentoring students.

I am deeply appreciative of my dissertation committee members, Dr. Sijia Liu, Dr. Hui Liu, Dr. Mikhail Galkin, and Dr. Neil Shah, for their insightful feedback and invaluable suggestions, which have greatly enriched my work.

I would like to extend my special thanks to those who offered me inspiration during challenging moments, including Dr. Yao Ma, Guangliang Liu, Justin Ding, Dr. Qiang Fu, Yuanqi Du, Yupeng Hou, Haonan Wang, Dr. Mikhail Galkin, and Dr. Zhaocheng Zhu, Dr. Yuke Wang, Yanqiao Zhu, Jieyu Zhang, Yanbang Wang, and Xingyue Huang, Dr. Hong Gao. Their insights and encouragement were invaluable in helping me navigate obstacles along the way.

Additionally, I am incredibly grateful for the support and encouragement of my friends and colleagues throughout my Ph.D. journey. I feel fortunate to have shared this experience with Zhikai Chen, Jingzhe Liu, Wenzhuo Tang, Kaiqi Yang, Haoyu Han, Kai Guo, Hanbing Wang, Juanhui Li, Harry Shomer, Quang Truong, and all the other members of the Data Science and Engineering Lab.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Graphs, which abstract complex systems of relations and interactions, can model transportation networks, trading networks [329], epidemic networks, the World Wide Web, and more. Graph Neural Networks (GNNs) [134, 84], aiming to adaptively integrate both feature and structure knowledge on networks, have emerged as a popular graph machine learning technique. The key component in GNNs is the non-parameterized message-passing operator, which updates node representation via aggregating neighbor node messages. The message-passing operator requires task-specific designs to match the corresponding graph algorithms, e.g., Common Neighbor [5] for friend recommendation, PageRank for web Search, and Dijkstra algorithm for finding the shortest path.

Despite the initial success achieved by GNNs, the fixed, non-parameterized message-passing operators often fail to generalize on different networks with distribution shifts [190]. To break the inevitable but largely ignored "algorithmic glass ceiling" of GNNs, my research aims to develop a Graph Foundation Model (GFM) which can generalize across different graphs with diverse properties. Despite graphs can have different appearance, I propose a Graph Vocabulary Hypothesis which suppose the existence of shared patterns across graphs.

To achieve this ultimate goal, my research are majorly two-fold: **(i)** Where are the essential patterns? My research lies foundation in the network science [203]. I design the random network model to comprehensively describe essential network patterns. Theoretical analysis is then conducted to principally characterize the relationship between different network patterns. With suitable network model and well-defined network patterns, I can uncover the limitations of GNNs by identifying the patterns they fail to capture, revealing their intrinsic drawbacks. **(ii)** How to capture essential patterns? To address the incapability of GNNs, I design flexible GFM architectures, obtain their capability in a data-driven approach via large-scale pre-training.

Organization: The structure of this dissertation proposal is outlined as follows:

`Chapter 2` provides a comprehensive summary of the evolution of graph machine learning technique, spanning from the traditional graph heuristic, Graph Neural Network to recent GFM design.

The chapter offers a detailed architectural overview, illustrating the graph learning progression.

Chapter 3 finds the performance degradation of GNN under domain shift. To address this challenge, I propose a model-agnostic algorithm to adapt to the target graph structure and enhance discriminative ability.

Chapter 4 builds the CSBM-S model to study the feature homophily and heterophily pattern simultaneously. I challenge the conventional belief that GNNs struggle with heterophily by demonstrating their effectiveness under specific conditions, while also revealing their intrinsic limitations when dealing with both homophily and heterophily nodes together.

Chapter 5 build the latent space model capturing fundamental network proximities, feature proximity (FP), local structural proximity (LSP), and global structural proximity (GSP) crucial for link prediction and reveals that current GNN-based methods predominantly capture LSP while failing to effectively model GSP and FP.

Chapter 6 proposes the blueprint of Graph Foundation Models. I introduce a graph vocabulary perspective to identify fundamental transferable units in graphs, providing a principled framework for GFM development, discussing graph transferability principles, and outlining challenges such as scalability, model architecture, and the integration of large language models (LLMs) for graph-related tasks.

Chapter 7 showcases the feasiblity of GFM with a Graph Foundation model with data scaling behavior. The proposed GFM can both predict missing edges and generate effective data argumentation to enhance other related tasks.

Chapter 8 concludes this dissertation. The overarching goal of our research is to move from the GNN learning paradigm towards the new GFM learning paradigm grounded in network science insights.

CHAPTER 2

BACKGROUND

2.1 Heuristic & Network Patterns

Network patterns provide a conventional understanding of the network system by identifying fundamental network patterns, e.g., network motif [202] and establishing the key principles, e.g., triadic closure principle [107] and homophily principle, which are generally valid across different domains. Those principles have been generally utilized to guide the design of heuristic algorithms. Moreover, they are the most elementary principles across graphs from different domains, identifying essential invariances as the foundation to build GFMs.

Network patterns Over the years, various theories and expertise knowledge have emerged to understand and predict links in networks. We typically review the foundational theories shaping the landscape of link prediction and network analysis.

Homophily [127] is a long-standing principle in social network analysis, stemming from the shared beliefs and thoughts of individuals. It suggests that people with aligned perspectives are likely to connect with one another, despite potential differences in their social position. [211, 67, 127, 54, 201] provide further study illustrating how homophily induced different phenomenons in social networks. It serves as the principle guidance for methods ranging from conventional page-rank [51] and label propagation [37] to recent advanced GNNs. Existing GNN architectures, often crafted based on the homophily principle, demonstrate strong performance on diverse homophilous graphs across various domains.

While homophily predominates in network analysis, it is not a universal rule. In many real-world scenarios, “opposites attract", resulting in networks characterized by heterophily—where nodes are more likely to link with dissimilar nodes. GNNs built with the homophily principle often struggle with heterophilious networks, except in cases of “good heterophily” [183, 177], where GNNs can identify and leverage consistent patterns in connections between dissimilar nodes. However, most heterophilious networks are complex and varied, posing challenges for GNNs due to their irregular and intricate interaction patterns [178, 282, 187]. Consequently, GNNs’ transferability, more assured

in homophilous graphs, is facing significant challenges in heterophilous ones.

Tradic closure [107] is another fundamental concept in social network analysis, stemming from that friends of friends become friends themselves. It suggests that two individuals, who have a mutual friend, become connected themselves. [61, 237, 106, 217] provide further study illustrating how homophily induced different phenomenons in social networks. [245] theoretically proves the effectiveness of heuristics algorithms.

More recently, [211, 8, 1] focus on investigating the interplay between the role of the tradic closure and homophily. [8] first demonstrate that triadic closure intensifies the impacts of homophily. Nonetheless, [1] points out that [8] builds on the existence of sufficient tradic closure rather than considering the tradic closure and homophily simultaneously. With a modest modification, [1] finds that tradic closure can introduce individuals to those unlike themselves, consequently reducing segregation.

Heuristic algorithms Heuristic algorithms inspired network patterns serves as the primary methodology for graph tasks, due to their simplicity, interpretability, and scalability. Heuristic algorithms inspired by the network principles are based on the hypothesis that nodes with higher network similarity have an increased probability of connection. The proximity score between two nodes represents their potential connection probability. This score is tailored to the characteristics of specific node pairs, viewed from various perspectives.

Table 2.1 Popular heuristics for link prediction, where $\Gamma(i)$ denotes the neighbor set of vertex i.

Name	Formula	Factor
common neighbors (CN)	$\lvert\Gamma(i) \cap \Gamma(j)\rvert$	LSP
Adamic-Adar (AA)	$\sum_{k\in\Gamma(i)\cap\Gamma(j)} \frac{1}{\log\lvert\Gamma(k)\rvert}$	LSP
resource allocation (RA)	$\sum_{k\in\Gamma(j)\cap\Gamma(i)} \frac{1}{\lvert\Gamma(k)\rvert}$	LSP
Katz	$\sum_{l=1}^{\infty} \lambda^l \lvert\text{paths}^{\langle l\rangle}(i,j)\rvert$	GSP
Personal PageRank (PPR)	$[\pi_i]_j + [\pi_j]_i$	GSP
Feature Homophily (FH)	$\text{dis}(x_i, x_j)$	FP

Generally, there are three types of heuristic approaches to measure the proximity and the formal

definition of some representative ones are listed in Table 2.1. (1) **Local structural heuristics.** This category includes four prevalent algorithms: Common Neighbor (CN) [212], Jaccard [112], Adamic Adar (AA) [5] and Resource Allocation (RA) [339]. These methods are fundamentally based on the assumption that nodes sharing more common neighbors are more likely to connect. AA and RA are weighted variants of common Neighbors. Typically, they incorporate node degree information, emphasizing that nodes with a lower degree have a higher influence. (2) **Global structural heuristics.** This category includes algorihms Katz [126], SimRank [113], and Personalized PageRank (PPR) [26] in the graph. They consider the global structural patterns with the number of paths between two nodes, where more paths indicate high similarity and are more likely to connect. Specifically, Katz counts the number of all paths between two nodes, emphasizing shorter paths by penalizing longer ones with a factor. SimRank assumes that two nodes are similar if they are linked to similar nodes, and PPR produces a ranking personalized to a particular node based on the random walk. (3) **Feature proximity heuristics** is available when node attributes are available, incorporating the side information about individual nodes. [214, 332] combines graph structure with latent features and explicit features for better performance.

Network Analysis. Important network analysis principles [192] fall into three primary concepts including: (1) local structural proximity corresponding to the triadic closure principle [107], where friends of friends become friends themselves. It inspires well-known conventional methods including CN, RA, AA [5]. (2) global structural proximity corresponding to the decay factor principle, where two nodes with more short paths between them have a higher probability of being connected. It inspires well-known conventional methods, e.g. Simrank and Katz [126, 113]. (3) feature proximity corresponding to the homophily principle [211] where shared beliefs and thoughts can be found in connected individuals.

These principles guide the evolution of link prediction algorithms, from basic heuristics to sophisticated GNNs [35, 148]. GNNs, inspired by these principles, perform well across diverse graphs in multiple domains. Moreover, [337] provides empirical evidence supporting the beneficial transferability of these guiding principles.

2.2 Graph Neural Networks

Graph Neural Networks (GNNs) have emerged as a powerful technique in Deep Learning, specifically designed for graph-structured data. They address the limitations of traditional neural networks in dealing with irregular data structures. GNNs learn node representations by aggregating neighborhood and transforming features recursively. The node representation can then be successfully utilized to a wide range of graph-related downstream tasks [132, 325, 297, 89, 287, 346, 68, 304].

The aggregation mechanism in GNNs is often viewed as feature smoothing [150, 185, 342]. This perspective leads some recent studies [341, 50, 152, 83] claiming that GNN models are overly reliant on homophilic patterns and unsuited to capturing heterophilic patterns To accommodate heterophilic graphs, recent works propose carefully designed GNN architectures, e.g., CPGNN [340], GGNN [303], GPRGNN [50], GCNII [42] GBK-GNN [63], ACM-GNN [176], Bernnet [87]. More recent analyses on GNNs [14, 184, 176] indicate that even GCN [132], a vanilla GNN, can deliver strong performance on certain heterophilic graphs. According to these findings, new metrics and understandings [21, 223, 156, 176, 184, 178] have been proposed to further expose the remaining weaknesses of GNNs.

Advanced GNNs for the ink prediction task (GNN4LP) are built on the basic of vanilla GNNs [133, 135] which learn single node structural representation by aggregating neighborhood and transforming features recursively, equipped with pairwise decoders. GNN4LP models augment vanilla GNNs by incorporating more complicated pairwise structural information inspired by heuristic methods. For instance, NCNC [283] and NBFNet [346] generalize CN and Katz heuristics with neural functions to incorporate those pairwise information, thereby achieving efficiency and promising performance.

Despite satisfying results in many tasks via utilizing structural information, [325, 326, 155] find that vanilla GNNs are still sub-optimal suffering from disability on capturing important pairwise patterns for Link Prediction, e.g., Common Neighborhood. Graph Neural Networks for Link Prediction (GNN4LP) [325, 326, 317, 279, 283, 36, 346] are then proposed which incorporates different inductive bias to capturing more pairwise information. SEAL [325], Neo-GNN [317] and

NCNC [283] involve more neighbor-overlapping knowledge. BUDDY [36], and NBFNet [346] exploit the higher order structural information, e.g., number of paths between nodes. Nonetheless, advanced GNN4LP models with better effectiveness usually have more complicated model designs, leading to the efficiency issue. [310, 309, 291, 345, 140] are then proposed to improve efficiency via approximating methods, e.g., hashing algorithm, A* algorithm, and random walk approximations.

2.3 Graph Foundation Models

Graph Foundation Models (GFMs) have recently emerged as a promising approach for learning transferable graph representations that can generalize to previously unseen graphs [189]. A key challenge in this field is the wide variation in graph structures, where both connectivity and feature patterns differ significantly. GNNs are not regarded as GFMs because they perform optimally only on one single dataset. At present, there is no widely accepted universal approach to capture graph representation, resulting in a broad range of GFM designs. Based on how well models transfer across tasks and domains, GFMs can be divided into three main categories: task-specific GFMs, domain-specific GFMs, and primitive GFMs. Below, we define each category and provide relevant examples.

A *task-specific/domain-specific GFM* should be transferable across the specific task/domain and thus adapt to diverse downstream datasets and domain-specific tasks. A notable example of a task-specific GFM is ULTRA [72], achieving superior zero-shot knowledge graph completion performance across datasets from various domains. A task-specific GFM shows great practical benefits, as it can be trained on data-rich domains, e.g., Wikipedia knowledge graphs, and subsequently improve effectiveness in resource-limited domains, e.g., geography knowledge graph. A domain-specific GFM instance, DiG [336], learns universal representations across various chemical tasks by leveraging domain-specific knowledge. The domain-specific GFM is highly efficient, as one model can serve all tasks while also delivering improved effectiveness compared to single-task models.

A *primitive GFM* exhibits the capability to generalize towards a limited number of datasets and tasks. A notable example is OFA [164], which is co-trained on data ranging from citation networks and molecule graphs to knowledge graphs via a unified task formation on node, link, and graph level

tasks. The OFA model can achieve comparable or even better performance over the vanilla GNNs on each task. Nonetheless, the OFA requires transforming all node features into text for co-training, which may not be convenient for all types of data. This co-training paradigm may also limit its generalization to unseen tasks and domains.

CHAPTER 3

SOURCE FREE GRAPH DOMAIN ADAPTATION

3.1 Introduction

Node classification [134] is a crucial task on graph-structural data such as transaction network [329], citation networks [262, 256, 193], and so on. Recently, Graph Neural Networks [134, 188] have greatly advanced the performance of node classification. However, most existing studies only concentrate on how to classify well on one given graph of a specific domain, while ignoring its performance degradation when applying it to graphs from other domains due to the domain gap. For example, regarding two real-world citation networks with papers as nodes and edges representing their citations, papers published between 2000 - 2010 and papers published between 2010 - 2020 may have significant differences from the following two aspects: (1) Feature distribution shifts as the advanced research topics and high-frequency keywords change over time. (2) Discrepancy between graph structures: due to the great success of Deep Learning, papers on machine learning and neuro-science have been more frequently cited in recent years. More concretely, we select two datasets: ACM-D, and ACM-S, two subgraphs from ACM dataset [306] to study their graph structure discrepancy. Their degree distributions are shown in Fig. 3.1. It is obvious to see that the node degree distributions on different graphs are different. The above problems, i.e., feature distribution shift and graph structure discrepancy, lead to unsatisfactory performance when transferring the GNN model across graphs from different domains to handle the same task. The naive way to achieve good

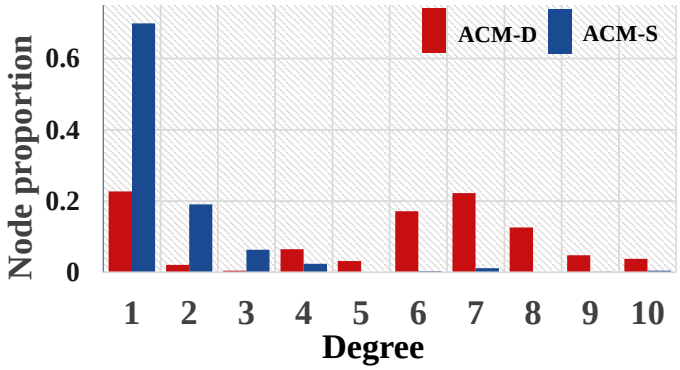


Figure 3.1 Discrepancy between degree distributions on ACM-D and ACM-S datasets.

results on the target graph from a different domain is to label the graph manually and retrain a new model from scratch, which is expensive and time-consuming. To solve this problem, Unsupervised Domain Adaptation (UDA), a transfer learning technique leveraging knowledge learned from a sufficiently labeled source domain to enhance the performance on an unlabeled target domain, has raised increasing attention recently.

UDA has shown great success on image data [267, 74, 243, 186] and text data [114, 55]. Recently, Unsupervised Graph Domain Adaptation (UGDA) has been proposed as a new application of UDA on graph data. It utilizes important properties of graphs, especially the structural information indicating the correlation between nodes. Generally speaking, most existing UGDA methods [307, 248, 330, 289] utilize a joint learning framework: (1) A feature encoder is trained to align the feature distributions between the source domain and target domain to mitigate the domain gap. (2) A classifier is trained on encoded features with cross-entropy loss, supervised by source labels. The model can achieve satisfying performance with strong discriminative ability on the aligned feature distribution.

However, a crucial requirement for these joint learning methods is access permission to the source data, which might be problematic for both accessibility and privacy issues [274]. In the real-world scenario, access to the source domain is not always available (e.g., domain adaptation between two different platforms). The usage of sensitive attributes on graphs may lead to potential data leakage and other severe privacy issues. Therefore, we propose a new scenario, **Source Free Unsupervised Graph Domain Adaptation** (SFUGDA), in which only the unlabeled target data and the GNN model trained from source data are available for adaptation. [1]

The key challenges in this scenario are two-fold: (1) How the model can adapt well to the shifted target data distribution without accessing the source graph for aligning the feature distributions. (2) How to enhance the discriminative ability of the source model without accessing source labels for supervision. In this paper, we propose SOGA, a model agnostic **SO**urce free domain **G**raph **A**daptation algorithm, which enables the GNN model trained on the labeled source graph to perform

[1] reuse [191] from the ACM Digital Library

well on the unlabeled target graph.

SOGA addresses these challenges by the following two components: (1) Structure Consistency (SC) optimization objective: inspired by the unsupervised graph embedding methods [267, 231], which learn node representations by preserving various graph properties using well-designed objective functions, we propose SC objective to tune the source model to reflect the target graph structure in the model output representation space. It can adapt the source model to the shifted target data distribution. (2) Information Maximization (IM) optimization objective is proposed to enhance the discriminative ability of the source model by maximizing the mutual information between the target graph and its corresponding output. We theoretically prove that IM can improve the confidence of prediction and raise the lower bound of the AUC metric.

Moreover, we usually can neither determine the source model architecture nor its training procedure in practice, other than no accessibility to the source data. Our algorithm is also model agnostic which be easily adopted to arbitrary GNN models. With this property, our SOGA can easily satisfy the above practical requirements.

In summary, the main contributions of our work are as follows:

- We first articulate a new scenario called SFUGDA when we have no access to the source graph and its labels. To the best of our knowledge, this is the first work in SFUGDA.
- We propose a model agnostic unsupervised algorithm called SOGA to tackle challenges in SFUGDA. It can both adapt the source model to the shifted target distribution and enhance its discriminative ability with a theoretical guarantee.
- Extensive experiments are conducted on real-world datasets. Our SOGA outperforms all the baselines on four cross-domain tasks. Moreover, experimental results verify the model agnostic property as SOGA can be applied with different representative GNN models successfully.

3.2 Related Work

3.2.1 Comparison with topics on Domain Adaptation

Unsupervised Graph Domain Adaptation (UGDA) aims to transfer the knowledge learned on a labeled graph from the source domain to an unlabeled graph from the target domain tackling the same task. Most existing UGDA methods aim to mitigate the domain gap by aligning the source feature distribution and the target one. According to different alignment approaches, they can be roughly divided into two categories. (1) Distance-based methods like [248, 307] incorporate maximum mean discrepancy (MMD) [25] as a domain distance loss to match the distribution statistical moments at different order. (2) Domain adversarial methods [330, 289] follow the guidance of adversarial training, which confuses generated features across the source domain and the target one to mitigate the domain discrepancy. DANE [330] adds an adversarial regularizer inspired by LSGAN [195], while UDAGCN [289] uses Gradient Reversal Layer [74] and domain adversarial loss to extract cross-domain node embedding.

However, all the above UGDA methods heavily rely on access to the source data, which leads to failure in the SFUGDA scenario where source data is not available anymore.

In computer vision, Source Free Unsupervised Domain Adaptation is a new research task with practical value. Most existing studies focus on different strategies to generate pseudo labels on images inspired by [243]. [154] utilizes the Deep Cluster algorithm to assign cleaner pseudo labels with a global view and an Information Maximization algorithm to minimize the prediction uncertainty PrDA [130] uses a set-to-set distance to filter confident pseudo labels. [151] focuses on how to adapt the feature distribution on the target domain by generating similar feature samples from a GAN-based model. [305] encourages label consistency on local affinity neighborhoods based on the key observation that the target data can still form clear data clusters. However, the above methods designed for image data are not suitable for graph-structural data. Since graph node samples are naturally structured by dependencies (i.e., edges) between nodes, strategies focusing on i.i.d. data like images, cannot be well adapted. For example, even when feature distribution stays the same, the graph may still suffer from domain gaps for various structure patterns. Thus, methods

for SFUGDA should handle structural dependencies well. Despite the graph structure leading to new challenges, various properties of the graph structure, such as structure proximity, could help the adaptation if modeled properly.

Moreover, our proposed SFUGDA method is more practical than methods for images as most of them need specific-designed source model architecture. They utilize different components like BatchNorm or WeightNorm to implicitly memorize the knowledge from source data. However, it seems not feasible in practice to retrain a specific source model for adaptation. Contrastively, our SOGA can combine with any GNN model with no need for a specific design.

3.2.2 Comparison between SFUGDA with other topics on graphs

Graph self-supervised learning [271, 344, 324, 104, 105, 226] is a new scenario on the graph which also utilizes the two-stage procedure similar with SFUGDA. Typically, those methods will first have an unsupervised learning procedure by creating graph-specific pretext tasks and training on the unlabeled data. This procedure aims to learn a good representation that could benefit different downstream tasks. Then a supervised fine-tuning procedure is employed with the labeled data for the specific downstream tasks. However, those methods are not applicable in the SFUGDA scenario as there is no label information in the target domain for the supervised fine-tuning.

More recently, self-supervised learning is also utilized as an unsupervised fine-tuning strategy [39, 210, 275]. They typically update the original model by minimizing a self-supervised loss on the target distribution. However, most are specifically designed for image data and may not be suitable for graph-structured data. [120] is the first to utilize self-supervised learning for finetuning in the graph domain. It focuses on learning an adaptive graph transformation from a data-centric perspective rather than learning a well-performed model.

Graph Federated learning is a distributed machine learning approach for privacy which has raised great interest in graph [145, 294]. They aggregate a server-side model from multiple decentralized edge devices without data leakage, offering a privacy-preserving mechanism. However, it requires each local data with labeled information. It fails to address the SFUGDA scenario where label information is only available in one single source domain.

3.3 Preliminary

Definition 1 (***Node Classification***)**.** *Node classification is a task to learn a conditional probability* $Q(\mathbf{y}|\mathbf{G};\Theta)$ *to distinguish the category of each unlabeled node on a* ***single*** *graph* $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X}, \mathbf{y})$*, where* Θ *is the model parameters.* $\mathbf{V} = \{v_1, \cdots v_n\}$ *is the node set with n nodes and* $\mathbf{E}$ *is the edge set.* $\mathbf{X} \in \mathbb{R}^{n\times d}$ *is the node feature matrix,* $\mathbf{y} \in \mathbb{R}^n$ *is the partially observed node label set of which each element satisfies* $y_i \in \{1, 2, \cdots k, -1\}$*.* $y_i = -1$ *indicates i-th node is unlabeled, d is the feature dimension, and k is the number of categories.*

Node classification differs from typical classification tasks since the latter usually assumes that different samples are i.i.d (independent identical distribution), whereas samples in the former case are correlated through edges. In deep learning, we use Graph Neural Networks (GNNs) [84] to model $Q(\mathbf{y}|\mathbf{G};\Theta)$, the conditional probability to distinguish the category of all nodes, for capturing such relationships. Based on the assumption of localization [57], the predicted conditional distribution is usually decomposed as follows, $Q(\mathbf{y}|\mathbf{G};\Theta) = \prod_{v_i\in\mathbf{V}} q(y_i|x_i, \mathcal{N}_\mathbf{i};\Theta)$, where $q(y_i|x_i, \mathcal{N}_\mathbf{i};\Theta)$ is the conditional probability to distinguish the category of one single node v_i. Notice that, $\mathbf{G}$ includes the feature x_i and the neighbor information $\mathcal{N}_\mathbf{i}$ for each node v_i. Following the GNN model, Cross Entropy is usually adopted as the loss function:

$$\mathbb{E}_{v_i\sim p(v)}\left[-\sum_{y=1}^{k} p(y|x_i, \mathcal{N}_\mathbf{i}) \log q(y|x_i, \mathcal{N}_\mathbf{i};\Theta)\right], \tag{3.1}$$

where $p(v)$ is the prior distribution and $p(y|x, \mathcal{N})$ is the oracle conditional distribution. As information in the node v_i contains its own feature x_i and neighborhood information $\mathcal{N}_\mathbf{i}$, we will simplify $p(x_i, \mathcal{N}_\mathbf{i})$ to $p(v_i)$ for brevity.

According to Def. 1, a typical node classification task is defined on a single graph with partial supervision. To better leverage the knowledge from a labeled graph (namely source graph) to tackle the same node classification task on another unlabeled graph (namely target graph), Unsupervised Graph Domain Adaptation (UGDA) [330] is proposed. We then give a clear definition of UGDA which has already been well recognized in [247, 330, 289, 117, 247].

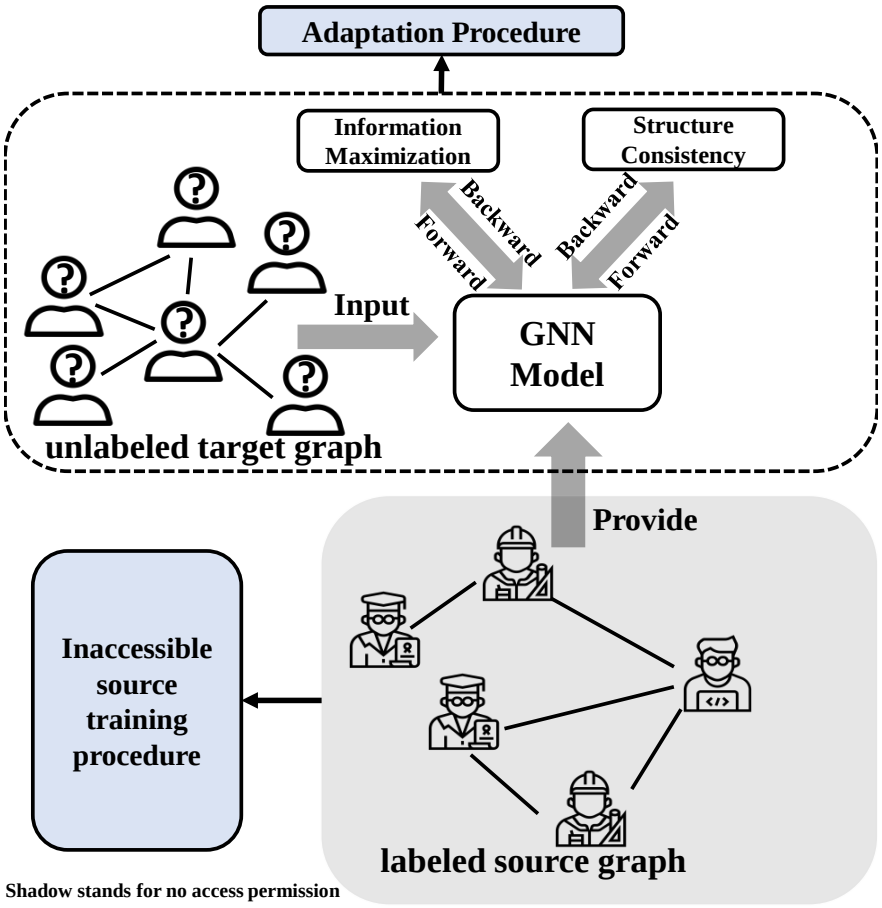


Figure 3.2 The detailed procedure of the SOGA algorithm. In the SFUGDA scenario, the source training procedure and labeled source graph in the shadow box are not accessible. Our algorithm only includes the upper dashed box describing the adaptation procedure. SOGA utilizes the output of the model on the unlabeled target graph to optimize two objectives: Information Maximization and Structure Consistency to adapt the model on the target domain.

Definition 2 (***Unsupervised Graph Domain Adaptation***)**.** *aims to learn a node classification model* $Q(\mathbf{y}|\mathbf{G};\Theta)$ *based on a source graph* $\mathbf{G_s} = (\mathbf{V_s}, \mathbf{E_s}, \mathbf{X_s}, \mathbf{y_s})$*, and the model performs well on a target graph* $\mathbf{G_t} = (\mathbf{V_t}, \mathbf{E_t}, \mathbf{X_t}, \mathbf{y_t})$ *where node features (i.e.,* $\mathbf{X_s}$ *and* $\mathbf{X_t}$*) and labels (i.e.,* $\mathbf{y_s}$ *and* $\mathbf{y_t}$*) express the same meanings* $\mathbf{y_t}$ ***is fully unknown which indicates an unsupervised problem in the target graph.*** *The oracle conditional distributions for the source and target graph are defined as* $p_s(y|x, \mathcal{N})$ *and* $p_t(y|x, \mathcal{N})$*, respectively.*

A general assumption in Unsupervised Graph Domain Adaptation is that prediction tasks are almost the same, i.e., $p_s(y|x, \mathcal{N})$ and $p_t(y|x, \mathcal{N})$ are similar [330, 289]. Thus, according to Eq. (3.1),

the main challenges for UGDA are the misalignment between prior distributions $p_s(v)$ of the source graph and $p_t(v)$ of the target graph. Consequently, the majority of current approaches attempt to align two distributions as part of their methodologies, which predominantly depends on access to the source data. Nonetheless, in practical situations, obtaining the training data of the source model is frequently challenging, primarily due to privacy concerns or data collection expenses. This gives rise to a novel problem named **Source-Free Unsupervised Graph Domain Adaptation (SFUGDA)**, which necessitates the absence of source data access in addition to the requirements of UGDA.

3.4 Problem Statement & Methodology

3.4.1 Problem Statement & Overview

Problem Statement: Source Free Unsupervised Graph Domain Adaptation aims to learn a well-performed node classification model $Q_{\mathbf{t}}(\mathbf{y}|\mathbf{G};\Theta_t)$ on the target graph $\mathbf{G_t}$, while the accessible information only contains two parts: (1) the well-trained source model $Q_{\mathbf{s}}(\mathbf{y}|\mathbf{G};\Theta_s)$ (i.e., well-performed in the source graph but not guaranteed to be well-performed in the target graph); (2) The unlabeled target graph $\mathbf{G_t} = (\mathbf{V}_t, \mathbf{E}_t, \mathbf{X}_t)$.

Overview of Framework: As the framework outlines shown in Fig. 3.2, the well-performed source model is provided by the first procedure, the inaccessible training procedure on the source graph. As we cannot interfere with the source training procedure, **the source model architecture could be an arbitrary GNN** as we could not determine. In our experiments, the well-trained source model is the model with the best performance on the validation set of the source graph. Parameters of the source model Θ_s with primary discriminative ability are utilized as the initialization of our model in the latter adaptation procedure.

The adaptation procedure is the key to solving this problem. In this procedure, we need to further adapt $Q_s(\mathbf{y}|\mathbf{G};\Theta_s)$ on the unlabeled target graph $\mathbf{G}_t$ without the information of GNN architecture and the prior distribution of the source data. This leads to three main challenges: (1) We are required to adapt the source model to the target distribution with no access to features in the source domain; (2) The adaption learning in the target domain is an entirely unsupervised learning procedure because

of no access to the label in both source and target domains. With only optimizing the unsupervised loss in the target domain, it could be easy to lose the initial discriminatory power of the source model; (3) Algorithm design that depends on the model structure will no longer be feasible because of no access to the source training procedure. In our work, we design unsupervised optimization objectives to solve the above challenges and achieve better performance on the target graph.

3.4.2 Overall Objectives

To adapt the given source model, we mainly design two optimization objectives. One is to leverage the information stored in the given model, namely the Information Maximization (IM) optimization objective, and the other is to utilize the target graph structure, namely the Structure Consistency (SC) optimization objective, to enhance the discriminative ability of the model on the target graph. The overall objective is defined as follows:

$$\max \mathcal{L} = \mathcal{L}_{IM} + \mathcal{L}_{SC} \tag{3.2}$$

Note that, both optimization objectives are designed on the model output space $\mathbb{R}^k$, where k is the number of classes. The objective can be easily applied to any GNN model. Details on the two objectives are presented in the following sections.

3.4.3 Information Maximization Optimization Objective

We define the IM objective as the mutual information between inputs and outputs of the model enhancing the discriminative ability:

$$\mathcal{L}_{IM} = \mathrm{MI}(\hat{\mathbf{y}}_\mathbf{t}, \mathbf{V}_\mathbf{t}) = -\mathrm{H}(\hat{\mathbf{y}}_\mathbf{t}|\mathbf{V}_\mathbf{t}) + \mathrm{H}(\hat{\mathbf{y}}_\mathbf{t}), \tag{3.3}$$

where $\hat{\mathbf{y}}_\mathbf{t}$ is the prediction on target domain and $\mathbf{V}_\mathbf{t}$ is the information of input nodes containing node feature $\mathbf{X}_\mathbf{t}$ and information from node neighbor $\mathcal{N}_\mathbf{t}$. $\mathrm{MI}(\cdot,\cdot)$ is the mutual information, and $\mathrm{H}(\cdot)$ and $\mathrm{H}(\cdot|\cdot)$ are entropy and conditional entropy, respectively. The objective can be divided into two parts, one is to minimize the conditional entropy and the other is to maximize the entropy of the marginal distribution of $\hat{\mathbf{y}}_\mathbf{t}$. We will introduce the implementation and the idea behind such a design for these two parts, respectively.

Conditional Entropy The conditional entropy can be implemented by the following equation:

$$H(\hat{\mathbf{y}}_\mathbf{t}|\mathbf{V_t}) = \mathbb{E}_{x_i \sim p_t(x)} \left[-\sum_{y=1}^{|C|} q(y|x_i, \mathcal{N}_\mathbf{i}; \Theta) \log q(y|x_i, \mathcal{N}_\mathbf{i}; \Theta) \right] \tag{3.4}$$

which can be easily optimized by sampling nodes from the prior distribution $p_t(x)$ on the target graph.

Intuitively, the goal of this objective is to enhance the certainty of predictions made on the target graph, which will lead to a substantial improvement in the lower bound of the model's effectiveness.

Theoretically, we present two key lemmas. The first lemma demonstrates the manner in which the objective bolsters the confidence of the model predictions. Meanwhile, the second lemma indicates that the lower bound of the Area Under Curve (AUC) will increase when the objective is applied.

Lemma 1. When the source model is optimized by the objective Eq. (3.4) with a gradient descent optimizer and the capacity of the source model is sufficiently large, for each node v_i on the target graph, the predicted conditional distribution $q(y|x_i, \mathcal{N}_i; \Theta)$ will converge to a vector $\mathbf{q}$, where the value of η elements will be $\frac{1}{\eta}$, and the other elements will be 0. η is determined by the number of categories with the maximum probability value predicted by the original source model $q(y|x_i, \mathcal{N}_i; \Theta_s)$. Similarly, the non-zero positions of $\mathbf{q}$ are the indices of categories with the maximum probability value.

In most cases, η equals one, and hence the prediction $\mathbf{q}$ will be a one-hot encoding vector. For further verifying the effectiveness of the objective, we theoretically analyze its effect on the Area Under Curve (AUC) metric on a binary classification problem:

Lemma 2. When the original source model is trained for a binary classification problem with the discriminative ability of r_p and r_n accuracy for positive samples and negative samples on the target graph respectively, the lower bound of AUC can be raised from $r_p \times r_n$ to $\frac{1}{2}(r_p + r_n)$ by using the conditional entropy objective Eq. (3.4).

Raising the lower bound of AUC from $r_p \times r_n$ to $\frac{1}{2}(r_p + r_n)$ is significant, for instance, if $r_p = r_n = 0.7$, then the absolute improvement will be 0.21.

3.4.3.1 Entropy of Marginal Distribution

The entropy of marginal distribution $\hat{\mathbf{y}}_\mathbf{t}$ can be calculated as:

$$\begin{aligned} H(\hat{\mathbf{y}}_\mathbf{t}) &= -\sum_y q(y) \log q(y), \\ q(y) &= \mathbb{E}_{v_i \sim p_t(v)} \left[q(y|x_i, \mathcal{N}_\mathbf{i}; \Theta) \right] \end{aligned} \tag{3.5}$$

This objective is designed to avoid the unsupervised objective easily stuck in a bad solution where predictions concentrate on the same category. Particularly, if we have additional knowledge about the prior distribution of labels $p_t(y)$ on the target graph, a KL-divergence objective can be a replacement for the Eq. (3.5):

$$KL(p_t(y)||q(y)) = \sum_y p_t(y) \log \frac{p_t(y)}{q(y)}. \tag{3.6}$$

To summarize, we adopt Eq. (3.5) by default to balance different categories, and we can use Eq.(3.6) to approximate the real distribution with additional information about the prior label distribution.

3.4.4 Structure Consistency Optimization Objective

To adapt the source model to the shifted target domain without source data, leveraging the structural information of the target graph becomes the key solution. Thus, we design a Structure Consistency (SC) objective based on two hypotheses, i.e., (1) the probability of sharing the same label for local neighbors is relatively high; (2) the probability of sharing the same label for the nodes with the same structural role is relatively high. These two hypotheses are commonly utilized in lots of Graph Embedding works [221, 78, 231] where several structure-preserving losses based on the above hypotheses are designed for learning node representations. To be specific, the SC objective is

designed as follows:

$$\begin{aligned}
\mathcal{L}_{SC} &= \sum_{v_i \in \mathbf{V_t}} \sum_{v_j \in \mathbf{V_t}} \lambda_1 \mathcal{J}_L^{i,j} + \lambda_2 \mathcal{J}_C^{i,j}, \\
\mathcal{J}_L^{i,j} &= p_l^{(i,j)} \log \sigma(\left\langle \hat{\mathbf{y}}_\mathbf{t}^{(\mathbf{i})}, \hat{\mathbf{y}}_\mathbf{t}^{(\mathbf{j})} \right\rangle) + (1 - p_l^{(i,j)}) \log \sigma(-\left\langle \hat{\mathbf{y}}_\mathbf{t}^{(\mathbf{i})}, \hat{\mathbf{y}}_\mathbf{t}^{(\mathbf{j})} \right\rangle), \\
\mathcal{J}_C^{i,j} &= p_c^{(i,j)} \log \sigma(\left\langle \hat{\mathbf{y}}_\mathbf{t}^{(\mathbf{i})}, \hat{\mathbf{y}}_\mathbf{t}^{(\mathbf{j})} \right\rangle) + (1 - p_c^{(i,j)}) \log \sigma(-\left\langle \hat{\mathbf{y}}_\mathbf{t}^{(\mathbf{i})}, \hat{\mathbf{y}}_\mathbf{t}^{(\mathbf{j})} \right\rangle),
\end{aligned} \tag{3.7}$$

where $\hat{y}_t^{(i)}$ is the predicted label vector for i-th node, $\sigma(x) = 1/(1 + e^{-x})$ is Sigmoid function, $\langle \cdot, \cdot \rangle$ is the inner product, λ_1 and λ_2 are hyperparameters to control the importance of two sub-objectives. Notice that both λ_1 and λ_2 are set to the default value 1 in the experiments. $p_l^{(i,j)}, p_s^{(i,j)} \in \{0, 1\}$ are defined by the local neighbor similarity and structural role similarity. Specifically, if $(v_i, v_j) \in \mathbf{E_t}$, then $p_l^{(i,j)} = 1$, otherwise $p_l^{(i,j)} = 0$. Similarly, if $(v_i, v_j) \in \mathcal{S}_\mathbf{t}$ then $p_c^{(i,j)} = 1$, otherwise $p_c^{(i,j)} = 0$, where $\mathcal{S}_\mathbf{t}$ is a set containing the top κ structurally similar node pairs. We follow struc2vec [231] to define the structural similarity that can be roughly understood as calculating the similarity of the sorted degree sequences around two given nodes. In order to reduce the number of hyperparameters, κ is set as the same size of the edge set $|\mathbf{E_t}|$ by default in all of our experiments while it can be adjusted as needed.

Intuitively speaking, the first and the second sub-objectives correspond to hypotheses (1) and (2), respectively.$\mathcal{J}_L^{i,j}$ and $\mathcal{J}_C^{i,j}$ are cross-entropy loss defined in the node pair level. The objective $\mathcal{J}_L^{i,j}$ enlarges the prediction similarity between the nodes with connection and distinguishing nodes without connection. Similarly, the objective $\mathcal{J}_C^{i,j}$ enlarges the similarity between the nodes with similar structural roles and distinguishing nodes with different structural roles. Finally, we use the negative sampling technique [221] to avoid calculating the objective function for each node pair for acceleration. Combining the objectives Eq. (3.2), Eq. (3.4), and Eq. (3.5), we obtain the overall differentiable objective of the model parameters Θ. We adopt the adaptive moment estimation method (i.e., Adam) [131] to optimize the overall objective.

3.5 Experiments

We conduct experiments on real-world datasets to study our proposed algorithm SOGA. We design a series of experiments to answer the following research questions:

Table 3.1 Average Macro-F1 and Macro-AUC scores on target graph in four unsupervised graph domain adaptation tasks on baseline methods and GCN-SOGA. GCN-SOGA-prior is the GCN-SOGA with additional label distribution knowledge. Notice that, For UGDA baselines including DANE and UDAGCN, we give them additional access to source data.

Methods	Group1				Group2			
	DBLP→ACM		ACM→DBLP		ACM-D→ACM-S		ACM-S→ACM-D	
	Macro-F1	Macro-AUC	Macro-F1	Macro-AUC	Macro-F1	Macro-AUC	Macro-F1	Macro-AUC
DeepWalk	0.135 ± 0.012	0.593 ± 0.010	0.112 ± 0.012	0.613 ± 0.008	0.183 ± 0.012	0.549 ± 0.006	0.237 ± 0.012	0.573 ± 0.007
Node2vec	0.128 ± 0.023	0.567 ± 0.011	0.080 ± 0.018	0.533 ± 0.004	0.134 ± 0.012	0.537 ± 0.005	0.219 ± 0.014	0.649 ± 0.003
GCN	0.583 ± 0.002	0.887 ± 0.004	0.668 ± 0.015	0.937 ± 0.003	0.685 ± 0.005	0.856 ± 0.008	0.796 ± 0.030	0.924 ± 0.002
GraphSAGE	0.418 ± 0.057	0.763 ± 0.054	0.752 ± 0.010	0.934 ± 0.003	0.407 ± 0.042	0.835 ± 0.013	0.743 ± 0.015	0.909 ± 0.003
GAT	0.227 ± 0.004	0.831 ± 0.004	0.745 ± 0.036	0.929 ± 0.011	0.681 ± 0.006	0.854 ± 0.005	0.804 ± 0.007	0.928 ± 0.002
GRACE	0.604 ± 0.014	0.908 ± 0.007	0.604 ± 0.014	0.806 ± 0.004	0.662 ± 0.003	0.876 ± 0.003	0.792 ± 0.007	0.907 ± 0.007
DGI	0.592 ± 0.010	0.894 ± 0.012	0.621 ± 0.005	0.872 ± 0.010	0.610 ± 0.003	0.842 ± 0.002	0.808 ± 0.006	0.919 ± 0.007
TENT	0.617 ± 0.007	0.912 ± 0.008	0.913 ± 0.011	0.957 ± 0.009	0.702 ± 0.015	0.893 ± 0.004	0.813 ± 0.007	0.922 ± 0.004
GTrans	0.610 ± 0.003	0.913 ± 0.006	0.911 ± 0.007	0.948 ± 0.004	0.723 ± 0.021	0.864 ± 0.003	0.753 ± 0.004	0.876 ± 0.018
SHOT	0.556 ± 0.004	0.858 ± 0.007	0.673 ± 0.079	0.938 ± 0.008	0.658 ± 0.011	0.864 ± 0.002	0.827 ± 0.016	0.916 ± 0.018
NRC	0.561 ± 0.009	0.858 ± 0.009	0.644 ± 0.010	0.897 ± 0.007	0.629 ± 0.007	0.823 ± 0.010	0.817 ± 0.003	0.921 ± 0.005
DANE	0.614 ± 0.017	0.906 ± 0.023	0.584 ± 0.008	0.937 ± 0.003	0.722 ± 0.004	0.888 ± 0.002	0.821 ± 0.004	0.923 ± 0.001
UDAGCN	0.626 ± 0.070	0.930 ± 0.006	0.696 ± 0.009	0.953 ± 0.008	0.665 ± 0.010	0.881 ± 0.004	0.822 ± 0.018	0.928 ± 0.002
GCN-SOGA	0.636 ± 0.003	0.931 ± 0.004	0.928 ± 0.018	0.988 ± 0.002	0.733 ± 0.005	0.907 ± 0.005	0.842 ± 0.008	0.951 ± 0.002
GCN-SOGA-prior	**0.650 ± 0.007**	**0.943 ± 0.008**	**0.935 ± 0.011**	**0.990 ± 0.001**	**0.737 ± 0.004**	**0.908 ± 0.005**	**0.843 ± 0.003**	**0.953 ± 0.005**

- **RQ1**: How does the GCN-SOGA compare with other state-of-the-art node classification methods? (GCN-SOGA represents SOGA applying on the default source domain model: GCN.)
- **RQ2**: How effective can SOGA be integrated with different GNN models?
- **RQ3**: How do different components in SOGA contribute to its effectiveness?
- **RQ4**: How do different choices of hyperparameters λ_1 and λ_2 affect the performance of SOGA?
- **RQ5**: Can GCN-SOGA learn more distinguishable node representations from visualization compared with other baselines?

3.5.1 Experiment Settings

Datasets. We use two groups of real-world graph datasets for our experiments. DBLPv8 and ACMv9 are the first group of citation networks collected by [289] from arnetMiner [262]. Their domain gap mainly comes from different origins (DBLP, ACM respectively) and different publication time periods, i.e. DBLPv8 (after 2010), ACMv9 (between years 2000 and 2010). ACM-D (Dense)

Table 3.2 Statistics of the experimental datasets.

Datasets	# Nodes	# Edges	# Features	# Labels
DBLPv8	5578	7341	7537	6
ACMv9	7410	11135	7537	6
ACM-D	1500	4960	300	4
ACM-S	1500	759	300	4

and ACM-S (Sparse) are the second group of citation networks from the ACM dataset collected by [306]. The detailed statistics of these datasets are illustrated in Tab. 3.2.

Baselines. We select some state-of-the-art methods as baselines to verify the effectiveness of our proposed algorithm. They are (1) Graph embedding methods including DeepWalk, and Node2vec [221, 78]. (2) Graph Neural Network (GNN) methods including GCN, GraphSAGE, and GAT [134, 84, 269]. (3) SFUDA methods on the image including SHOT [154] and NRC [305] (4) self-supervised learning methods including DGI [271], GRACE [344], TenT [275], and Gtrans [120]. DGI and GRACE are two graph self-supervised learning baselines that focus on learning good representation for the downstream task. However, they require labeled data on the target domain for finetuning, which is not available in our scenario. To make a fair comparison, we employ the proposed self-supervised learning algorithm for unsupervised fine-tuning, which shows a similar train manner with our proposed SOGA. TenT and Gtrans are the selected self-supervised learning algorithms for unsupervised finetuning from image and graph domains, respectively. (5) UGDA methods including DANE and UDAGCN [330, 289]. **Notice that we give them UGDA methods additional access to the source data.** For all baseline methods with GNN, we utilize a two-layer GCN in which the hidden dimensions are 256, and 128 respectively as the default encoder. All baseline methods are included with a careful hyperparameter check..

Reproducibility Settings. To ensure the validity of experiments, each source dataset is randomly split into training and validation sets with a ratio 4:1. All the experimental results are averaged over 5 runs with different random seeds [1, 3, 5, 7, 9].

Considering the hyperparameter search setting, **we do not have any hyperparameter search in SOGA**. We set the hyperparameters λ_1 and λ_2 to the default value: of 1 in all experiments except for the hyperparameter sensitivity analysis. For baseline methods, we apply large-scale hyperparameter

Table 3.3 The performance comparison on the Macro-F1 score of different GNN models with or without applying SOGA.

Methods	Group1		Group2	
	DBLPv8→ACMv9	ACMv9→DBLPv8	ACM-D→ACM-S	ACM-S→ACM-D
GCN	0.583 ± 0.002	0.668 ± 0.015	0.685 ± 0.005	0.796 ± 0.030
GCN-SOGA	**0.636 ± 0.003**	**0.928 ± 0.018**	**0.736 ± 0.007**	**0.838 ± 0.008**
GraphSAGE	0.418 ± 0.057	0.752 ± 0.010	0.407 ± 0.042	0.743 ± 0.015
GraphSAGE-SOGA	**0.594 ± 0.086**	**0.947 ± 0.002**	**0.734 ± 0.006**	**0.820 ± 0.020**
GAT	0.227 ± 0.004	0.745 ± 0.036	0.681 ± 0.006	0.804 ± 0.007
GAT-SOGA	**0.592 ± 0.086**	**0.946 ± 0.001**	**0.736 ± 0.006**	**0.824 ± 0.027**

grid search on all other baselines including GNN, SFUDA, self-supervised learning, and UGDA methods to ensure those baseline methods reach their best performance. Our reproducing settings are different from some baseline methods reported in their papers like five different random seeds and the validation partition. After checking with their authors, we consider that this may induce a different experimental result from the one in the original paper. We also release our code and the experimental details in the repository here

Evaluation Methods. We conduct the stability evaluation to verify the stability of algorithms. The main reason is that models trained with different epochs may have large performance differences. Generally speaking, there should be an additional validation set used to select the best training epoch. However, the validation set is not available on the unlabeled target domain. Therefore, it is of great importance to evaluate the stability of model performance on the target domain. With good stability, it is easy for models to achieve satisfying performance for stability prevents significant performance fluctuations after convergence. Contrastively, the performance of the unstable method will drop quickly after reaching the peak or fluctuate continuously. Concretely speaking, we (1) plot the line chart describing the Macro-F1 score on the target domain in each training epoch. (2) calculate the mean and standard deviation of Macro-F1 scores on the target domain across the epochs. To avoid the initial fluctuation before convergence, we choose the epochs after the first n ones for calculation (set to 20 by default). A flat curve with little fluctuation indicates good stability, corresponding to results with large expectations and small standard deviations.

3.5.2 Overall Results

The experimental results of all baseline methods, GCN-SOGA (applying SOGA on GCN), and GCN-SOGA-prior on Macro-F1 score and Macro-AUC score are illustrated on Tab. 3.1.

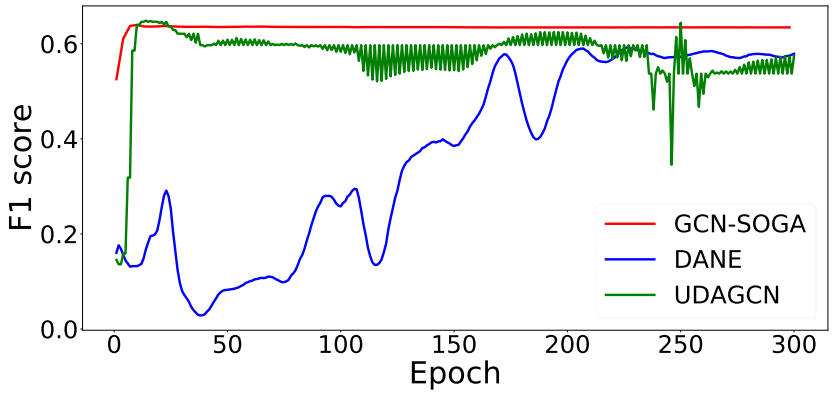


Figure 3.3 The comparison of learning curves of different UGDA methods on the DBLPv8 → ACMv9 task. The x axis denotes the training epoch, whereas the y axis denotes the Macro-F1 score on the target graph.

GCN-SOGA-prior is a variant of GCN-SOGA which we give SOGA algorithm with additional knowledge about the prior distribution of labels $p_t(y)$. Therefore, we replace the entropy of marginal distribution in eq. (3.5) by default with the KL-divergence objective in eq. (3.5). Notice that UGDA methods other than our proposed GCN-SOGA require additional information, i.e., access permission to the source graph in the adaptation procedure. Therefore, these methods are not feasible in the SFUGDA scenario. When reproducing these methods, **we give them additional access to the source data.**

Experimental results of GCN-SOGA on four cross-domain tasks show consistent improvements in Macro-F1 score and Macro-AUC score, with a maximum gain of 2.1% and 4%, respectively. Additionally, GCN-SOGA-prior can have greater gain than GCN-SOGA on the first group. It indicates that SOGA can work better with prior label distribution awareness. The reason for the performance GCN-SOGA-prior is similar to GCN-SOGA is that both ACM-D and ACM-S datasets have an almost uniform label distribution $p(y)$. The additional prior knowledge happens to be close to the default assumption. Therefore, it is no wonder that the performance of GCN-SOGA-prior is

similar to the one of GCN-SOGA. From the perspective of different cross-domain tasks, we can find the performance on the DBLPv8 → ACMv9 and ACM-D → ACM-S is much lower than the other two tasks which indicates its difficulties. We will mainly focus on these difficult tasks and conduct further experiments on them in later sections.

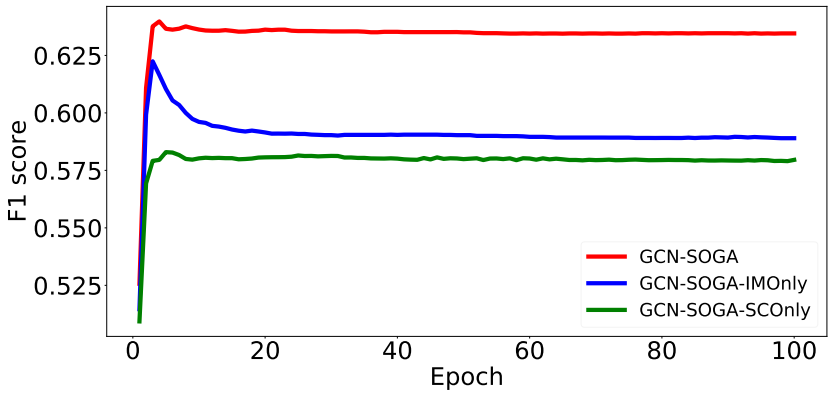


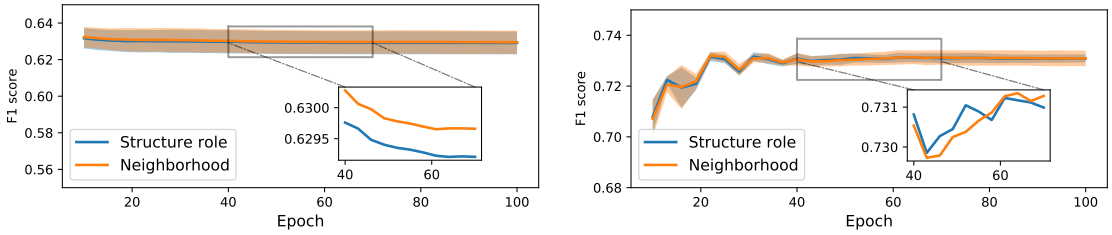
Figure 3.4 The comparison of learning curves of GCN-SOGA and its variants on the DBLPv8 → ACMv9 task. The x axis denotes the training epoch, whereas the y axis denotes the Macro-F1 score in the target graph.

From the perspective of different baseline methods, we can see that the graph embedding methods perform poorly, probably due to the lack of preserving cross-graph similarity. Though the relative position between nodes is preserved by structural proximity, similar nodes may have entirely different absolute positions in different graphs. GNNs perform better for the message passing mechanism like graph convolution and can preserve the similarity of nodes if their local sub-graphs are similar as proven by [62]. Self-supervised learning methods DGI and GRACE do not perform well since they are designed to learn a good representation but not the discriminative ability on the specific downstream task. TENT also does not work so well since it does not take the complex relationship on the graph into consideration. GTrans shows unsatisfying performance on ACM-D→ACM-S task since it transforms the target graph structure into a more sparse one. However, this sparsity assumption does not always come true in reality. SFUDA methods on the image domain including SHOT and NRC do not show satisfying performance since they do not have the specific design on the graph. Two UGDA methods, UDAGCN and DANE, perform best among all baselines for

they implicitly mitigate the distribution gap. To give a more careful comparison between UGDA baselines and our proposed GCN-SOGA, we conduct stability evaluation in detail. From Fig. 3.3, we can see that GCN-SOGA in red illustrates stability with a high Macro-F1 score, while DANE in blue reveals a slower convergence. The performance of UDAGCN in green illustrates a violent fluctuation. The fluctuation majorly contributes to the conflict in optimizing the domain alignment loss and cross-entropy loss. Such fluctuation will cause great difficulty in deciding which epoch to stop the adaptation procedure. Thus, the result further indicates the strength of our GCN-SOGA. Moreover, such fluctuation indicates that the alignment loss may have a strong conflict with the main cross-entropy loss during the optimization. It could be the evidence of why those UGDA methods with additional access to the source data perform worse than our SOGA.

Table 3.4 Statistical results of stability evaluation on GCN-SOGA and its variants. The reported results are the expectation and standard derivation of Macro-F1 scores on the training procedure after 20 epochs.

Methods	Group1		Group2	
	DBLPv8→ACMv9	ACMv9→DBLPv8	ACM-D→ACM-S	ACM-S→ACM-D
GCN	0.5832 ± 0.0000	0.6683 ± 0.0000	0.6857 ± 0.0000	0.7961 ± 0.0000
GCN-SOGA	0.6151 ± 0.0005	0.9382 ± 0.0002	0.7323 ± 0.0017	0.8244 ± 0.0056
GCN-SOGA-IMOnly	0.5823 ± 0.0007	0.9406 ± 0.3640	0.7227 ± 0.0170	0.8263 ± 0.0060
GCN-SOGA-SCOnly	0.5576 ± 0.0004	0.6491 ± 0.0653	0.7160 ± 0.0026	0.3182 ± 0.0013



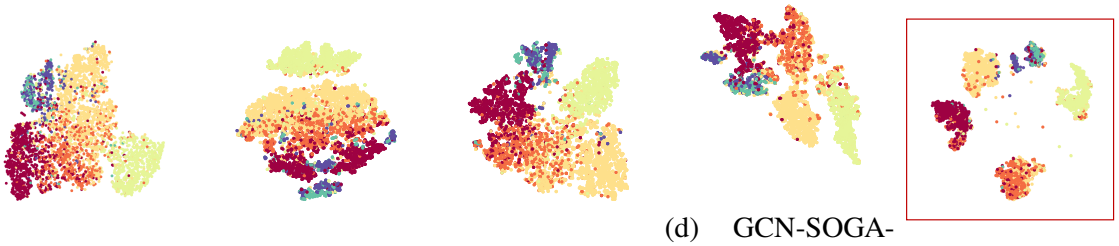
(a) The performance on the DBLPv8 → ACMv9 task

(b) The performance on the ACM-D → ACM-S task

Figure 3.5 Performances on the target domain with different choices of λ_1 and λ_2. The orange solid line and the corresponding shadow indicate the mean value and the standard deviation of the results, respectively, when $\lambda_2 > \lambda_1$, which means more attention on the neighborhood. The blue ones are similar except $\lambda_1 > \lambda_2$.

3.5.3 Effectiveness of SOGA on different GNN models

To demonstrate the efficacy and the model agnostic property of our proposed algorithm: SOGA, we evaluate SOGA with different representative GNN models. Specifically, we combine SOGA with GCN, GraphSAGE and GAT, named GCN-SOGA, SAGE-SOGA, and GAT-SOGA, respectively. The results are shown in Tab. 3.3. One observation is that SOGA can bring consistent improvement on different GNN models, which verifies that SOGA is model-agnostic. Meanwhile, the poor performance of GAT and GraphSAGE on some tasks (e.g., that in ACM-D → ACM-S) with the careful hyperparameter grid search, indicates the potential overfitting problem of these expressive models. Nonetheless, on the poor performance like GraphSAGE on the ACM-D → ACM-S task, the original Macro-F1 performance is merely 0.407 while performance with SOGA is 0.734, almost the same with the best result: 0.736. It indicates that SOGA can help to achieve a comparable good performance regardless of the poor origin GNN model.



(a) GCN (b) DANE (c) UDAGCN (d) GCN-SOGA-IMOnly (e) GCN-SOGA

Figure 3.6 Target domain network representation visualization of the source GCN and UGDA methods using t-SNE on the task DBLPv8 → ACMv9. Our proposed GCN-SOGA is marked with the red box. GCN-SOGA-IMOnly is a variant of GCN-SOGA mentioned in 3.5.4 without SC optimization objective.

3.5.4 Ablation Study

We conduct ablation experiments to investigate the contribution of each component. The study can be divided into two parts including the necessity of the well-trained source model, and the roles of two optimization objectives. All experiments in this section utilize the stable evaluation for a more detailed and fair comparison.

First, we verify the necessity of the indispensable component: the well-trained source model. Experiments show that only applying two unsupervised objectives with a randomly initialized model

leads to failure where the highest Macro-F1 score among all tasks is no more than 0.20, similar to random guess. Therefore, it is of great significance to utilize the primary discriminative ability of the well-trained source model.

Then we further explore the different roles of two unsupervised optimization objectives by stability evaluation. We propose the variants of our proposed SOGA called SOGA-IMOnly and SOGA-SCOnly corresponding to the algorithm trained with only IM or SC objective, respectively. The curves on DBLPv8 $\rightarrow$ ACMv9 are shown in Fig. 3.4. We can find the following observations: (1) For our GCN-SOGA in red, shows significant performance gain and strong stability with a flat curve after the first few epochs. (2) For GCN-SOGA-IMOnly in blue, there has been an evident drop after reaching the peak around 10 epochs. This unstable phenomenon reveals the difficulty and uncertainty of achieving a good result. (3) For GCN-SOGA-SCOnly in green, the curve is smooth after reaching the peak with only a little gain. Overall speaking, IM and SC objectives show a complementary effect. IM enhances the discriminative ability to achieve better results while SC takes charge of maintaining stability to maintain good performance consistently in the training procedure.

The detailed statistical results of stability evaluation on all tasks are illustrated in Tab. 3.4. We can find that on easy tasks like ACMv9 $\rightarrow$ DBLPv8, GCN-SOGA-IMOnly can achieve similar results with GCN-SOGA. However, on more difficult tasks, which are DBLPv8 $\rightarrow$ ACMv9 and ACM-D $\rightarrow$ ACM-S, GCN-SOGA could achieve more significant and stable improvement than GCN-SOGA-IMOnly. It further indicates the necessity of SC especially on more difficult tasks. Moreover, we notice that GCN-SOGA-SCONLY may perform even worse than the original GCN trained on the source domain. The key reason is that we could not utilize any supervised signal in the SFUGDA scenario. Therefore, both SC and IM objectives are unsupervised objectives. It could be possible that we lose the original discriminative ability with unsupervised finetuning.

3.5.5 Hyperparameter sensitivity analysis

Though we have achieved impressive results with the default hyperparameter setting as $\lambda_1 = \lambda_2 = 1$, it is still noteworthy to examine how the different choices of λ_1 and λ_2 affect the performance

of SOGA. Specifically, we focus on the Macro-F1 score performance of GCN-SOGA as well as its stability in the training procedure with different hyperparameter settings. Revolving around this goal, we conduct neighborhood evaluation and structure role evaluation which spare larger weight to λ_1 and λ_2, respectively.

For neighborhood evaluation, we run experiments 10 times with different choices where $\lambda_1 > \lambda_2$. Then we plot the line chart describing the Macro-F1 score on the target domain after the first 10 training epochs, skipping initial fluctuations for brevity. Structure role evaluation is similar except $\lambda_2 > \lambda_1$. The experimental results on DBLPv8 $\rightarrow$ ACMv9 and ACM-D $\rightarrow$ ACM-S are illustrated in Fig. 3.5. The solid line is the average result of 10 experiments while the shadow one represents the corresponding standard deviation. We can see that curves stay at a high level and the shadow area is narrow, which indicates good performance with stability in the training procedure. We can conclude that our performance is robust to different choices of λ_1 and λ_2.

3.5.6 Visualization

We visualize node representations of the target domain generated by the baseline methods: GCN, DANE, UDAGCN, our proposed GCN-SOGA and its variant GCN-SOGA-IMOnly (without the Structure Consistency optimization objective). The aim is to further prove the performance of SOGA even without explicit domain adaptation component to mitigate the domain discrepancy. One thing we want to point out is that the key to achieving good classification performance on the target domain is good class separability (the learned target node representations with the same label are close and the target node representations with different labels are far). Mitigating the domain discrepancy with an explicit domain adaptation component, which has been adopted by many UGDA methods, is just one of the effective approaches to enhance class separability on the target domain. However, it may not be necessary. The method like SOGA can also achieve good separation by both fully exploring the potential of the source model and utilizing the target graph structure.

For simplicity, we choose the most difficult task, DBLPv8 $\rightarrow$ ACMv9, to visualize the node representation. The node representation is the hidden representation closest to the final linear full-connected layer, which is of the same dimension size in all methods: 128. The dimensionality

reduction method for visualization is the T-distributed Stochastic Neighbor Embedding (t-SNE) [268]. The visualization results of different methods are shown in Fig. 3.6. The color of each node represents its label.

We can observe that the well-trained source model GCN shows low-class separability on the target graph, where clusters have many overlaps. UGDA methods, DANE and UDAGCN, are somehow better with better clustering. However, the boundaries are still difficult to find. For our proposed GCN-SOGA marked in the red box, the large boundary can be seen though there are some nodes clustered mistakenly due to the limitation of the totally unsupervised adaptation procedure. We further compare the GCN-SOGA and its variant GCN-SOGA-IMOnly without SC optimization object to see how the structural information benefits the class separation. This phenomenon further reveals that with the graph structure to learn structure proximity, the source model can better adapt to the target data distribution.

3.6 Conclusion

In this work, we articulate a new scenario called Source Free Unsupervised Graph Domain Adaptation with no access to the source graph because of practical reasons like privacy policies. Existing methods cannot work well as it is impossible for feature alignment. Facing challenges in SFUGDA, we propose our algorithm SOGA, which could be applied to arbitrary GNNs by adapting to the target domain distribution and enhancing the discriminative ability of the source model. Extensive experiments indicate its effectiveness.

CHAPTER 4

DEMYSTIFYING STRUCTURAL DISPARITY IN GRAPH NEURAL NETWORKS: CAN ONE SIZE FIT ALL?

4.1 Introduction

How to infer the missing node feature can be an new essential for understanding a graph with additional node features. It is also an essential network problem with extensive usage in many real-world applications, including the recommender system to infer the user portrait, citation networks to predict the missing paper topic, and biological networks to identify the protein roles. This problem can be formalized as the node classification or regression task, where the goal is to predict the missing feature utilizing the inputs with graph structure and existing features. When GNNs are applied to node classification, it is generally believed to work well under the homophily assumption, where neighboring nodes are likely to share the same label and similar features. An example of the homophilic pattern is depicted in the upper part of Figure 4.1, where node features and node labels are denoted by colors (i.e., blue and red) and numbers (i.e., 0 and 1), respectively. We can observe that all connected nodes exhibit homophilic patterns and share the same label 0. The message passing can then be viewed as a denoising process towards a smooth node representation with its neighborhood. Recently, several studies have demonstrated that GNNs can also perform well on certain heterophilic graphs [14, 184, 176]. In heterophilic graphs, connected nodes tend to have different labels, which we refer to as *heterophilic patterns*. The example in the lower part of Figure 4.1 shows the heterophilic patterns. Based on this example, we intuitively illustrate how GNNs can work on such heterophilic patterns (lower right): after averaging features over all neighboring nodes, nodes with label 0 completely switch from their initial blue color to red, and vice versa; despite this feature alteration, the two classes remain easily distinguishable since nodes with the same label (number) share the same color (features).

However, existing studies on the effectiveness of GNNs [13, 14, 184, 176] only focus on either homophilic or heterophilic patterns solely and overlook the fact that real-world graphs typically exhibit a mixture of homophilic and heterophilic patterns. Recent studies [152, 156] reveal that

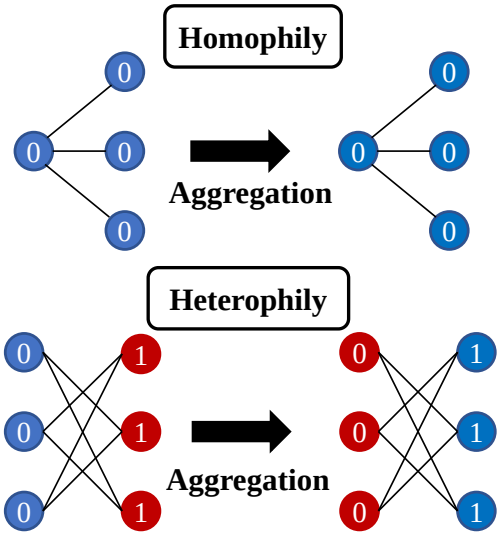


Figure 4.1 Examples of homophilic and heterophilic patterns. Colors/numbers indicate node features/labels.

many heterophilic graphs, e.g., `Squirrel` and `Chameleon` [239], contain over 20% homophilic nodes. Similarly, our preliminary study depicted in Figure 4.2 demonstrates that heterophilic nodes are consistently present in many homophilic graphs, e.g., `PubMed` [246] and `Ogbn-arxiv` [101]. Hence, real-world homophilic graphs predominantly consist of homophilic nodes as the majority structural pattern and heterophilic nodes in the minority one, while heterophilic graphs exhibit an opposite phenomenon with heterophilic nodes in the majority and homophilic ones in the minority.

To provide insights aligning the real-world scenario with structural disparity, we revisit the toy example in Figure 4.1, considering both homophilic and heterophilic patterns together. Specifically, for nodes labeled 0, both homophilic and heterophilic node features appear in blue before aggregation. However, after aggregation, homophilic and heterophilic nodes in label 0 exhibit different features, appearing blue and red, respectively. Such differences may lead to performance disparity between nodes in majority and minority patterns. For instance, in a homophilic graph with the majority pattern being homophilic, GNNs are more likely to learn the association between blue features and class 0 on account of more supervised signals in majority. Consequently, nodes in the majority structural pattern can perform well, while nodes in the minority structural pattern may exhibit poor performance, indicating an over-reliance on the majority structural pattern. Observations indicate that GCN [132], a vanilla GNN, often underperforms MLP-based models on nodes with the minority

pattern while outperforming them on the majority nodes.

To develop a model that simultaneously captures both homophily and heterophily patterns, it is essential to move beyond the **fixed message passing operators** with pre-defined neighborhood treatment. To address this issue, I adapt transformers [255] with their self-attention mechanisms, offering a powerful alternative due to their ability to consider more neighboring nodes and dynamically weigh the importance of each node. This adaptability gives the possibility to capture diverse network patterns together. Moreover, the large parameter scale of transformers further enhances their ability to model complex network patterns with extensive network data. To meet the transformer's sequential input requirement, I convert the original node neighborhood into sequences via a random walk algorithm. This algorithm traverses connected nodes in a random manner, thereby revealing structural proximity through the sequence of visited nodes. To learn how to select neighbor nodes adaptively, I propose a masked node modeling pre-text task, aiming to recover the masked neighbor nodes in the random-walk sequence. The large-scale pre-trained transformer, trained over 100 million instances, can automatically select informative neighbor nodes with high attention scores while ignoring noisy neighbor nodes. Consequently, the pre-trained model can achieve satisfying few-shot performance across networks, comparable to a fully supervised GNN. In contrast, a transformer with supervised training cannot generalize well and tends to over-fit shortcut solutions.

4.2 Prelimaries

Semi-Supervised Node classification (SSNC). Let $G = (V, E)$ be an undirected graph, where $V = \{v_1, \cdots, v_n\}$ is the set of n nodes and $E \subseteq V \times V$ is the edge set. Nodes are associated with node features $\mathbf{X} \in \mathbb{R}^{n \times d}$, where d is the feature dimension. The number of class is denoted as K. The adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$ represents graph connectivity where $\mathbf{A}[i, j] = 1$ indicates an edge between nodes i and j. $\mathbf{D}$ is a degree matrix and $\mathbf{D}[i, i] = d_i$ with d_i denoting degree of node v_i. Given a small set of labeled nodes, $V_{\text{tr}} \subseteq V$, SSNC task is to predict on unlabeled nodes $V \setminus V_{\text{tr}}$.

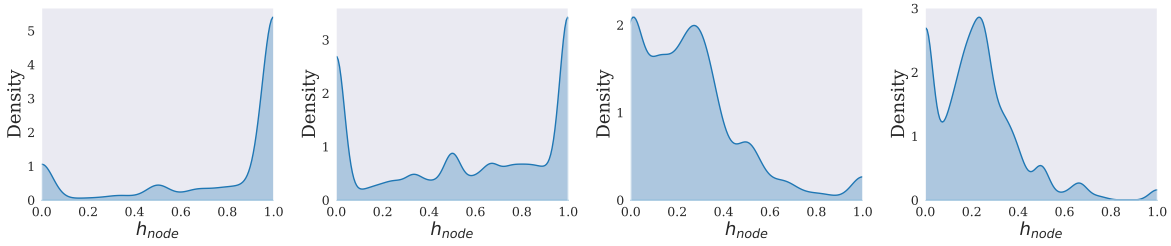
Node homophily ratio is a common metric to quantify homophilic and heterophilic patterns. It is calculated as the proportion of a node's neighbors sharing the same label as the node [220, 341, 152].

It is formally defined as $h_i = \frac{|\{u \in \mathcal{N}(v_i) : y_u = y_v\}|}{d_i}$, where $\mathcal{N}(v_i)$ denotes the neighbor node set of v_i and $d_i = |\mathcal{N}(v_i)|$ is the cardinality of this set. Following [152, 63, 220], node i is considered to be homophilic when more neighbor nodes share the same label as the center node with $h_i > 0.5$. We define the graph homophily ratio h as the average of node homophily ratios $h = \frac{\sum_{i \in V} h_i}{|V|}$. Moreover, this ratio can be easily extended to higher-order cases $h_i^{(k)}$ by considering k-order neighbors $\mathcal{N}_k(v_i)$.

Node subgroup refers to a subset of nodes in the graph sharing similar properties, typically homophilic and heterophilic patterns measured with node homophily ratio. Training nodes are denoted as V_{tr}. Test nodes V_{te} can be categorized into M node subgroups, $V_{\text{te}} = \bigcup_{m=1}^{M} V_m$, where nodes in the same subgroup V_m share similar structural pattern.

4.3 Effectiveness of GNN on nodes with different structural properties

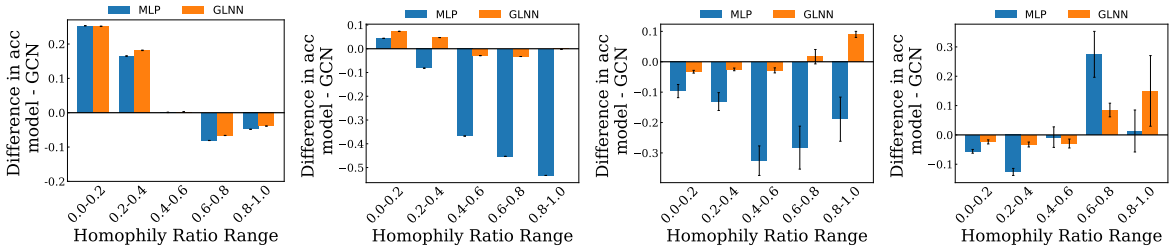
In this section, we explore the effectiveness of GNNs on different node subgroups exhibiting distinct structural patterns, specifically, homophilic and heterophilic patterns. It is different from previous studies [184, 13, 14, 178, 176] that primarily conduct analysis on the whole graph and demonstrate effectiveness with an overall performance gain. These studies, while useful, do not provide insights into the effectiveness of GNNs on different node subgroups, and may even obscure scenarios where GNNs fail on specific subgroups despite an overall performance gain. To accurately gauge the effectiveness of GNNs, we take a closer examination on node subgroups with distinct structural patterns. The following experiments are conducted on two common homophilic graphs, Ogbn-arxiv [101] and Pubmed [246], and two heterophilic graphs, Chameleon and Squirrel [239]. These datasets are chosen since GNNs can achieve better overall performance than MLP.



(a) PubMed (h=0.79) (b) Arxiv (h=0.63) (c) Cham.(h=0.22) (d) Squirrel (h=0.25)

Figure 4.2 Node homophily ratio distributions. All graphs exhibit a mixture of homophilic and heterophilic nodes despite various graph homophily ratio h.

Existence of structural pattern disparity within a graph is to recognize real-world graphs exhibiting different node subgroups with diverse structural patterns, before investigating the GNN effectiveness on them. We demonstrate node homophily ratio distributions on the aforementioned datasets in Figure 4.2. We can have the following observations. **Obs.1:** All four graphs exhibit a mixture of both homophilic and heterophilic patterns, rather than a uniform structural patterns. **Obs.2:** In homophilic graphs, the majority of nodes exhibit a homophilic pattern with h_i>0.5, while in heterophilic graphs, the majority of nodes exhibit the heterophilic pattern with $h_i \leq 0.5$. We define nodes in majority structural pattern as majority nodes, e.g., homophilic nodes in a homophilic graph.



(a) PubMed (h=0.79) (b) Arxiv(h=0.63) (c) Cham.(h=0.22) (d) Squirrel (h=0.25)

Figure 4.3 Performance comparison between GCN and MLP-based models. Each bar represents the accuracy gap on a specific node subgroup exhibiting a homophily ratio within the range specified on the x-axis. MLP-based models often outperform GCN on heterophilic nodes in homophilic graphs and homophilic nodes in heterophilic graphs with a positive value.

Examining GCN performance on different structural patterns. To examine the effectiveness of GNNs on different structural patterns, we compare the performance of GCN [132] a vanilla GNN, with two MLP-based models, vanilla MLP and Graphless Neural Network (GLNN) [328], on testing nodes with different homophily ratios. It is evident that the vanilla MLP could have a large performance gap compared to GCN (i.e., 20% in accuracy) [184, 328, 132]. Consequently, an under-trained vanilla MLP comparing with a well-trained GNN leads to an unfair comparison without rigorous conclusion. Therefore, we also include an advanced MLP model GLNN. It is trained in an advanced manner via distilling GNN predictions and exhibits performance on par with GNNs. Notably, only GCN has the ability to leverage structural information during the inference phase while both vanilla MLP and GLNN models solely rely on node features as input. This comparison ensures a fair study on the effectiveness of GNNs in capturing different structural

patterns with mitigating the effects of node features. Experimental results on four datasets are presented in Figure 4.3. In the figure, y-axis corresponds to the accuracy differences between GCN and MLP-based models where positive indicates MLP models can outperform GCN; while x-axis represents different node subgroups with nodes in the subgroup satisfying homophily ratios in the given range, e.g., [0.0-0.2]. Based on experimental results, the following key observations can be made: **Obs.1:** In homophilic graphs, both GLNN and MLP demonstrate superior performance on the heterophilic nodes with homophily ratios in [0-0.4] while GCN outperforms them on homophilic nodes. **Obs.2:** In heterophilic graphs, MLP models often outperform on homophilic nodes yet underperform on heterophilic nodes. Notably, vanilla MLP performance on Chameleon is worse than that of GCN across different subgroups. This can be attributed to the training difficulties encountered on Chameleon, where an unexpected similarity in node features from different classes is observed [175]. Our observations indicate that despite the effectiveness of GCN suggested by [184, 176, 13], GCN exhibits limitations with performance disparity across homophilic and heterophilic graphs. It motivates investigation why GCN benefits majority nodes, e.g., homophilic nodes in homophilic graphs, while struggling with minority nodes.

Organization. In light of the above observations, we endeavor to understand the underlying causes of this phenomenon in the following sections by answering the following research questions. Section 4.3.1 focuses on how aggregation, the fundamental mechanism in GNNs, affects nodes with distinct structural patterns differently. Building on these observations, Section 4.3.2 recognizes the key factors driving performance disparities on different structural patterns with a non-i.i.d. PAC-Bayes bound. Section 4.3.3 empirically corroborates the validity of our theoretical analysis with real-world datasets.

4.3.1 How does aggregation affect nodes with structural disparity differently

In this subsection, we examine how aggregation reveals different effects on nodes with structural disparity, serving as a precondition for performance disparity. Specifically, we focus on the discrepancy between nodes from the same class but with different structural patterns.

For a controlled study on graphs, we adopt the contextual stochastic block model (CSBM) with

two classes. It is widely used for graph analysis, including generalization [184, 13, 276, 14, 277, 71], clustering [70], fairness [116, 115], and GNN architecture design [285, 52, 292]. Typically, nodes in CSBM model are generated into two disjoint sets C_1 and C_2 corresponding to two classes, c_1 and c_2, respectively. Each node with c_i is associated with features $x \in \mathbb{R}^d$ sampling from $N(\mu_i, I)$, where μ_i is the feature mean of class c_i with $i \in \{1, 2\}$. The distance between feature means in different classes $\rho = |\mu_1 - \mu_2|$, indicating the classification difficulty on node features. Edges are then generated based on intra-class probability p and inter-class probability q. For instance, nodes with class c_1 have probabilities p and q of connecting with another node in class c_1 and c_2, respectively. The CSBM model, denoted as CSBM(μ_1, μ_2, p, q), presumes that all nodes follow either homophilic with $p > q$ or heterophilic patterns $p < q$ exclusively. However, this assumption conflicts with real-world scenarios, where graphs often exhibit both patterns simultaneously, as shown in Figure 4.2. To mirror such scenarios, we propose a variant of CSBM, referred to as CSBM-Structure (CSBM-S), allowing for the simultaneous description of homophilic and heterophilic nodes.

Definition 1 (CSBM-S$(\mu_1, \mu_2, (p^{(1)}, q^{(1)}), (p^{(2)}, q^{(2)}), \Pr(\text{homo}))$)**.** *The generated nodes consist of two disjoint sets C_1 and C_2. Each node feature x is sampled from $N(\mu_i, I)$ with $i \in \{1, 2\}$. Each set C_i consists of two subgroups: $C_i^{(1)}$ for nodes in homophilic pattern with intra-class and inter-class edge probability $p^{(1)} > q^{(1)}$ and $C_i^{(2)}$ for nodes in heterophilic pattern with $p^{(2)} < q^{(2)}$.* $\Pr(homo)$ *denotes the probability that the node is in homophilic pattern. $C_i^{(j)}$ denotes node in class i and subgroup j with $(p^{(j)}, q^{(j)})$. We assume nodes follow the same degree distribution with $p^{(1)} + q^{(1)} = p^{(2)} + q^{(2)}$.*

Based on the neighborhood distributions, the mean aggregated features $\mathbf{F} = \mathbf{D}^{-1}\mathbf{A}\mathbf{X}$ obtained follow Gaussian distributions on both homophilic and heterophilic subgroups.

$$\mathbf{f}_i^{(j)} \sim N\left(\frac{p^{(j)}\boldsymbol{\mu}_1 + q^{(j)}\boldsymbol{\mu}_2}{p^{(j)} + q^{(j)}}, \frac{\mathbf{I}}{\sqrt{d_i}}\right), \text{for } i \in C_1^{(j)}; \mathbf{f}_i^{(j)} \sim N\left(\frac{q^{(j)}\boldsymbol{\mu}_1 + p^{(j)}\boldsymbol{\mu}_2}{p^{(j)} + q^{(j)}}, \frac{\mathbf{I}}{\sqrt{d_i}}\right), \text{for } i \in C_2^{(j)} \tag{4.1}$$

Where $C_i^{(j)}$ is the node subgroups with structural pattern with $(p^{(j)}, q^{(j)})$ in label i. Our initial examination of different effects on aggregation focuses on the aggregated feature distance between

homophilic and heterophilic node subgroups within class c_1.

Proposition 1. *The aggregated feature mean distance between homophilic and heterophilic node subgroups within class* c_1 *is* $\left|\frac{p^{(1)}\mu_1+q^{(1)}\mu_2}{p^{(1)}+q^{(1)}} - \frac{p^{(2)}\mu_1+q^{(2)}\mu_2}{p^{(2)}+q^{(2)}}\right| > 0$*, indicating the aggregated feature of homophilic and heterophilic subgroups are from different feature distributions, with a mean distance larger than 0 distance before aggregation, since original node features draw from the same distribution, regardless of different structural patterns.*

Notably, the distance between original features is regardless of the structural pattern. This proposition suggests that aggregation results in a distance gap between different patterns within the same class.

In addition to node feature differences with the same class, we further examine the discrepancy between nodes u and v with the same aggregated feature $\mathbf{f}_u = \mathbf{f}_v$ but different structural patterns. We examine the discrepancy with the probability difference of nodes u and v in class c_1, denoted as $|\mathbf{P}_1(y_u = c_1|\mathbf{f}_u) - \mathbf{P}_2(y_v = c_1|\mathbf{f}_v)|$. $\mathbf{P}_1$ and $\mathbf{P}_2$ are the conditional probability of $y = c_1$ given the feature $\mathbf{f}$ on structural patterns $(p^{(1)}, q^{(1)})$ and $(p^{(2)}, q^{(2)})$, respectively.

Lemma 1. *With assumptions (1) A balance class distribution with* $\mathbf{P}(Y = 1) = \mathbf{P}(Y = 0)$ *and (2) aggregated feature distribution shares the same variance* σ*. When nodes u and v have the same aggregated features* $\mathbf{f}_u = \mathbf{f}_v$ *but different structural patterns,* $(p^{(1)}, q^{(1)})$ *and* $(p^{(2)}, q^{(2)})$*, we have:*

$$|\mathbf{P}_1(y_u = c_1|\mathbf{f}_u) - \mathbf{P}_2(y_v = c_1|\mathbf{f}_v)| \leq \frac{\rho^2}{\sqrt{2\pi}\sigma}|h_u - h_v| \tag{4.2}$$

Notably, above assumptions are not strictly necessary but employed for elegant expression. Lemma 1 implies that nodes with a small homophily ratio difference $|h_1 - h_2|$ are likely to share the same class, and vice versa.

4.3.2 Why does Performance Disparity Happen? Subgroup Generalization Bound for GNNs

In this subsection, we conduct a rigorous analysis elucidating primary causes for performance disparity across different node subgroups with distinct structural patterns. We identify two key factors for satisfying test performance: (1) test node u should have a close feature distance $\min_{v\in V_{\text{tr}}} |\mathbf{f}_u - \mathbf{f}_v|$

to training nodes V_{tr}, indicating that test nodes can be greatly influenced by training nodes. (2) With identifying the closest training node v, nodes u and v should be more likely to share the same class, where $\sum_{c_i \in C} |\mathbf{P}(y_u = c_i|\mathbf{f}_u) - \mathbf{P}(y_v = c_i|\mathbf{f}_v)|$ is required to be small. The second factor, focusing on whether two close nodes are in the same class, is dependent on the homophily ratio difference $|h_u - h_v|$, as shown in Lemma 1. Notably, since training nodes are randomly sampled, their structural patterns are likely to be the majority one. Therefore, training nodes will show a smaller homophily ratio difference with majority test nodes sharing the same majority pattern than minority test nodes, resulting in the performance disparity in distinct structural patterns.

To rigorously examine the role of aggregated feature distance and homophily ratio difference in performance disparity, we derive a non-i.i.d. PAC-Bayesian GNN generalization bound, based on the Subgroup Generalization bound of Deterministic Classifier [181]. We begin by stating key assumptions on graph data and GNN model to clearly delineate the scope of our theoretical analysis.

Definition 2 (Generalized CSBM-S model)**.** *Each node subgroup V_m follows the CSBM distribution $V_m \sim CSBM(\mu_1, \mu_2, p^{(i)}, q^{(i)})$, where different subgroups share the same class mean but different intra-class and inter-class probabilities $p^{(i)}$ and $q^{(i)}$. Moreover, node subgroups also share the same degree distribution as $p^{(i)} + q^{(i)} = p^{(j)} + q^{(j)}$.*

Instead of CSBM-S model with one homophilic and heterophilic pattern, we take the generalized CSBM-S model assumption, allowing more structural patterns with different levels of homophily.

Assumption 1 (GNN model)**.** *We focus on SGC [288] with the following components: (1) a one-hop mean aggregation function g with $g(X, G)$ denoting the output. (2) MLP feature transformation $f(g_i(X, G); W_1, W_2, \cdots, W_L)$, where f is a ReLU-activated L-layer MLP with $W_1, \cdots, W_L$ as parameters for each layer. The largest width of all the hidden layers is denoted as b.*

Notably, despite analyzing simple GNN architecture theoretically, similar with [181, 184, 77], our theory analysis could be easily extended to the higher-order case with empirical success across different GNN architectures shown in Section 4.3.3.

Our main theorem is based on the PAC-Bayes analysis which typically aims to bound the generalization gap between the expected margin loss $\mathcal{L}_m^0$ on test subgroup V_m for a margin 0 and the empirical margin loss $\widehat{\mathcal{L}}_{\text{tr}}^{\gamma}$ on train subgroup V_{tr} for a margin γ. Those losses are generally utilized in PAC-Bayes analysis[198, 197, 53, 65]. The formulation is shown as follows:

Theorem 1 (Subgroup Generalization Bound for GNNs)**.** *Let $\tilde{h}$ be any classifier in the classifier family $\mathcal{H}$ with parameters $\{\widetilde{W}_l\}_{l=1}^{L}$. for any $0 < m \leq M$, $\gamma \geq 0$, and large enough number of the training nodes $N_{tr} = |V_{tr}|$, there exist $0 < \alpha < \frac{1}{4}$ with probability at least $1 - \delta$ over the sample of $y^{tr} := \{y_i\}_{i \in V_{tr}}$, we have:*

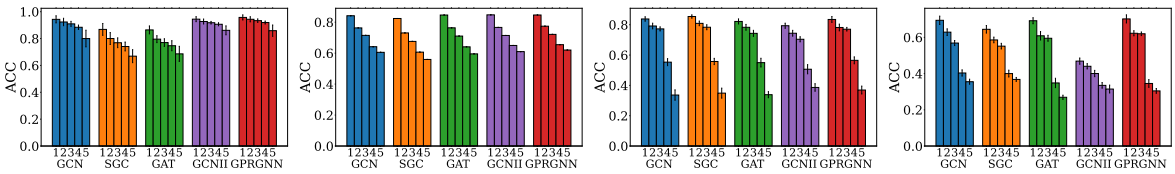
$$\mathcal{L}_m^0(\tilde{h}) \leq \widehat{\mathcal{L}}_{tr}^{\gamma}(\tilde{h}) + O\left(\underbrace{\frac{K\rho}{\sqrt{2\pi}\sigma}(\epsilon_m + |h_{tr} - h_m| \cdot \rho)}_{(a)} + \underbrace{\frac{b\sum_{l=1}^{L}|\widetilde{W}_l|_F^2}{(\gamma/8)^{2/L}N_{tr}^{\alpha}}(\epsilon_m)^{2/L}}_{(b)} + \mathbf{R} \right) \tag{4.3}$$

The bound is related to three terms: **(a)** describes both large homophily ratio difference $|h_{\text{tr}} - h_m|$ and large aggregated feature distance $\epsilon = \max_{j \in bV_m} \min_{i \in V_{\text{tr}}} |g_i(X, G) - g_j(X, G)|_2$ between test node subgroup V_m and training nodes V_{tr} lead to large generalization error. $\rho = |\mu_1 - \mu_2|$ denotes the original feature separability, independent of structure. K is the number of classes. **(b)** further strengthens the effect of nodes with the aggregated feature distance ϵ, leads to a large generalization error. **(c)** $\mathbf{R}$ is a term independent with aggregated feature distance and homophily ratio difference, depicted as $\frac{1}{N_{\text{tr}}^{1-2\alpha}} + \frac{1}{N_{\text{tr}}^{2\alpha}} \ln \frac{LC(2B_m)^{1/L}}{\gamma^{1/L}\delta}$, where $B_m = \max_{i \in V_{\text{tr}} \cup V_m} |g_i(X, G)|_2$ is the maximum feature norm. $\mathbf{R}$ vanishes as training size N_0 grows.

Our theory suggests that both homophily ratio difference and aggregated feature distance to training nodes are key factors contributing to the performance disparity. Typically, nodes with large homophily ratio difference and aggregated feature distance to training nodes lead to performance degradation.

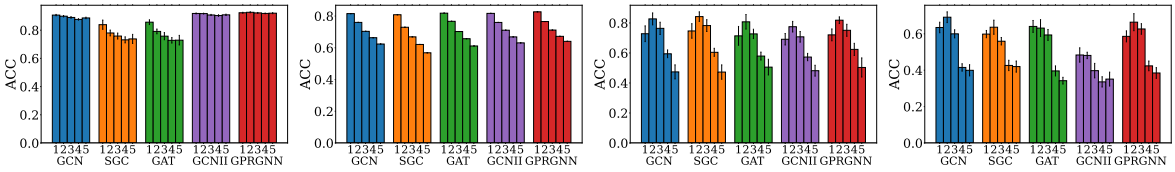
4.3.3 Performance Disparity Across Node Subgroups on Real-World Datasets

To empirically examine the effects of theoretical analysis, we compare the performance on different node subgroups divided with both homophily ratio difference and aggregated feature



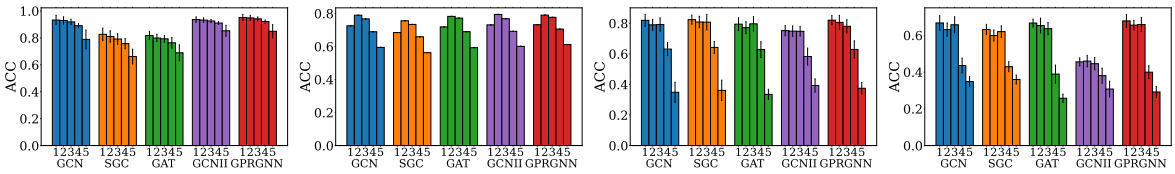
(a) PubMed (h=0.79) (b) Arxiv (h=0.63) (c) Cham. (h=0.22) (d) Squirrel (h=0.25)

Figure 4.4 Test accuracy disparity across node subgroups by **aggregated-feature distance and homophily ratio difference** to training nodes. Each figure corresponds to a dataset, and each bar cluster corresponds to a GNN model. A clear performance decrease tendency can be found from subgroups 1 to 5 with increasing differences to training nodes.



(a) PubMed (h=0.79) (b) Arxiv (h=0.63) (c) Cham. (h=0.22) (d) Squirrel (h=0.25)

Figure 4.5 Test accuracy disparity across node subgroups by **aggregated-feature distance** to train nodes. Each figure corresponds to a dataset, and each bar cluster corresponds to a GNN model. A clear performance decrease tendency can be found from subgroups 1 to 5 with increasing differences to training nodes.



(a) PubMed (h=0.79) (b) Arxiv (h=0.63) (c) Cham. (h=0.22) (d) Squirrel (h=0.25)

Figure 4.6 Test accuracy disparity across node subgroups by **homophily ratio difference** to train nodes. Each figure corresponds to a dataset, and each bar cluster corresponds to a GNN model. A clear performance decrease tendency can be found from subgroups 1 to 5 with increasing differences to training nodes.

distance to training nodes with popular GNN models including GCN [132], SGC [288], GAT [270], GCNII [42], and GPRGNN [50]. Typically, test nodes are partitioned into subgroups based on their disparity scores to the training set in terms of both 2-hop homophily ratio $h_i^{(2)}$ and 2-hop aggregated features $\mathbf{F}^{(2)}$ obtained by $\mathbf{F}^{(2)} = (\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}})^2\mathbf{X}$, where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}$. For a test node i, we measure the node disparity by (1) selecting the closest training node $v = \arg\min_{v \in V_0} ||\mathbf{F}_u^{(2)} - \mathbf{F}_v^{(2)}||$ (2) then calculating the disparity score $s_u = ||\mathbf{F}_u^{(2)} - \mathbf{F}_v^{(2)}||_2 + |h_u^{(2)} - h_v^{(2)}|$, where the first and the second terms correspond to the aggregated-feature distance and homophily ratio differences,

respectively. We then sort test nodes in terms of the disparity score and divide them into 5 equal-binned subgroups accordingly. Performance on different node subgroups is presented in Figure 4.4 with the following observations. **Obs.1:** We note a clear test accuracy degradation with respect to the increasing differences in aggregated features and homophily ratios. Furthermore, we investigate on the individual effect of aggregated feature distance and homophily ratio difference in Figure 4.5 and 4.6, respectively. An overall trend of performance decline with increasing disparity score is evident though some exceptions are present. **Obs.2:** When only considering the aggregated feature distance, there is no clear trend among groups 1, 2, and 3 on GCN, SGC, and GAT on heterophilic datasets. **Obs.3:** When only considering the homophily ratio difference, there is no clear trend among groups 1, 2, and 3 across four datasets. These observations underscore the importance of both aggregated-feature distance and homophily ratio differences in shaping GNN performance disparity. Combining these factors together provides a more comprehensive and accurate understanding of the reason for GNN performance disparity.

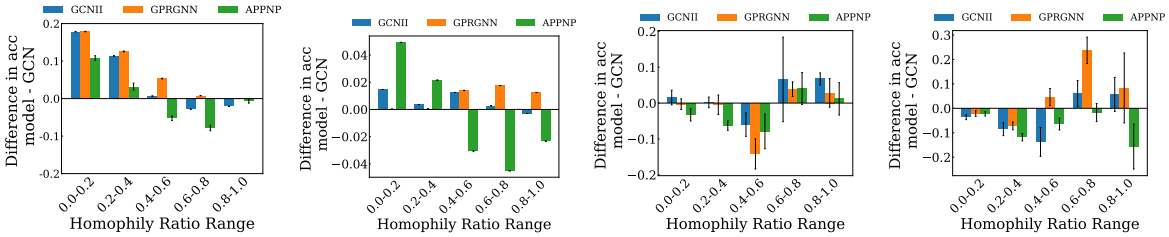
Summary. In this section, we study GNN performance disparity on nodes with distinct structural patterns and uncover its underlying causes. We primarily investigate the impact of aggregation, the key component in GNNs, on nodes with different structural patterns in Sections 4.3.1. We observe that aggregation effects vary across nodes with different structural patterns, notably enhancing the discriminative ability on majority nodes. These observed performance disparities inspire us to identify crucial factors contributing to GNN performance disparities across nodes with a non-i.i.d PAC-Bayes bound in Section 4.3.2. The theoretical analysis indicates that test nodes with larger aggregated feature distances and homophily ratio differences with training nodes experience performance degradation. We substantiate our findings on real-world datasets in Section 4.3.3.

4.4 Implications of graph structural disparity

In this section, we illustrate the significance of our findings on structural disparity via (1) elucidating the effectiveness of existing deeper GNNs (2) unveiling an over-looked aspect of distribution shift on graph out-of-distribution (OOD) problem, and introducing a new OOD scenario accordingly.

4.4.1 Elucidate the effectiveness of Deeper GNNs

Deeper GNNs [50, 42, 138, 3, 168, 147, 300] enable each node to capture more complex higher-order graph structure than vanilla GCN, via reducing the over-smoothing problem [216, 242, 38, 168, 150] Nonetheless, which structural patterns deeper GNNs can exceed and the reason for its effectiveness remain unclear. To investigate this problem, we compare vanilla GCN with different deeper GNNs, including GPRGNN[50], APPNP[138], and GCNII[42], on node subgroups with varying homophily ratios, adhering the same setting with Figure 4.3. Experimental results are shown in Figure 4.7. We can observe that deeper GNNs primarily surpass GCN on minority node subgroups with slight performance trade-offs on the majority node subgroups. We conclude that the effectiveness of deeper GNNs majorly contributes to improved discriminative ability on minority nodes.



(a) PubMed (h=0.79) (b) Ogbn-arxiv (h=0.63) (c) Chameleon (h=0.22) (d) Squirrel (h=0.25)

Figure 4.7 Performance comparison between GCN and deeper GNNs. Each bar represents the accuracy gap on a specific node subgroup exhibiting homophily ratio within range specified on x-axis.

4.4.2 A new graph out-of-distribution scenario

The graph out-of-distribution (OOD) problem refers to the underperformance of GNN due to distribution shifts on graphs. Many Graph OOD scenarios [290, 343, 330, 166, 101, 81, 181, 190], e.g., biased training labels, time shift, and popularity shift, have been extensively studied. These OOD scenarios can be typically categorized into covariate shift with $\mathbf{P}^{\text{train}}(\mathbf{X}) \neq \mathbf{P}^{\text{test}}(\mathbf{X})$ and concept shift [227, 205, 286] with $\mathbf{P}^{\text{train}}(\mathbf{Y}|\mathbf{X}) \neq \mathbf{P}^{\text{test}}(\mathbf{Y}|\mathbf{X})$. $\mathbf{P}^{\text{train}}(\cdot)$ and $\mathbf{P}^{\text{test}}(\cdot)$ denote train and test distributions, respectively. Existing graph concept shift scenarios [290, 81] introduce different environment variables e resulting in $\mathbf{P}(\mathbf{Y}|\mathbf{X}, e_{\text{train}}) \neq \mathbf{P}(\mathbf{Y}|\mathbf{X}, e_{\text{test}})$ leading to spurious

correlations. To address existing concept shifts, algorithms [290, 327] have been developed to capture the environment-invariant relationship $\mathbf{P}(\mathbf{Y}|\mathbf{X})$. Nonetheless, existing concept shift settings overlook the scenario where there is not a unique environment-invariant relationship $\mathbf{P}(\mathbf{Y}|\mathbf{X})$. For instance, $\mathbf{P}(\mathbf{Y}|\mathbf{X}_{\text{homo}})$ and $\mathbf{P}(\mathbf{Y}|\mathbf{X}_{\text{hete}})$ can be different, indicated in Section 4.3.1. $\mathbf{X}_{\text{homo}}$ and $\mathbf{X}_{\text{hete}}$ correspond to features of nodes in homophilic and heterophilic patterns. Notably, homophilic and heterophilic patterns are crucial task knowledge that cannot be recognized as the irrelevant environmental variable. Consequently, we find that homophily ratio difference between train and test sets could be an important factor leading to an overlook concept shift, namely, graph structural shift. Notably, structural patterns cannot be considered as environment variables given their integral role in node classification task. The practical implications of this concept shift are substantiated by the following scenarios: **(1)** graph structural shift frequently occurs in most graphs, with a performance degradation in minority nodes, as depicted in Figure 4.3. **(2)** graph structural shift hides secretly in existing graph OOD scenarios. For instance, the FaceBook-100 dataset [290] reveals a substantial homophily ratio difference between train and test sets, averaging 0.36. This discrepancy could be the primary cause of OOD performance deterioration since the exist OOD algorithms [290, 121] that neglect such a concept shift can only attain a minimal average performance gain of 0.12%. **(3)** graph structural shift is a recurrent phenomenon in numerous real-world applications where new nodes in graphs may exhibit distinct structural patterns. For example, in a recommendation system, existing users with rich data can receive well-personalized recommendations in the exploitation stage (homophily), while new users with less data may receive diverse recommendations during the exploration stage (heterophily).

Given the prevalence and importance of the graph structural shift, we propose a new graph OOD scenario emphasizing this concept shift. Specifically, we introduce a new data split on existing datasets, namely Cora, CiteSeer, PubMed, Ogbn-Arxiv, Chameleon, and Squirrel, where majority nodes are selected for train and validation, and minority ones for test. This data split strategy highlights the homophily ratio difference and the corresponding concept shift. To better illustrate the challenges posed by our new scenario, we conduct experiments on models including GCN,

Table 4.1 Performance (Accuracy) on the proposed OOD split.

	Pubmed	Ogbn-Arxiv	Squirrel	Chameleon
GCN(i.i.d)	89.18±0.15	72.99±0.14	58.09±0.71	75.09±0.79
GCN	51.04±0.16	34.94±0.07	32.13±4.93	43.35±3.47
MLP	68.38±0.43	33.17±0.37	24.57±0.77	34.78±4.97
GLNN	67.51±0.25	35.89±0.14	31.51±0.70	47.01±1.09
GCNII	67.76±0.36	36.81±0.14	37.15±1.39	41.25±2.03
GPRGNN	57.24±0.18	34.95±0.43	42.43±7.71	35.27±7.67
SRGNN	57.91±0.10	40.37±1.65	37.62±1.74	42.09±0.43
EERM	65.37±1.35	34.23±0.46	40.93±0.57	45.84±1.05
EERM(II)	67.59±0.91	40.28±0.84	44.31±0.40	48.59±0.78

MLP, GLNN, GPRGNN, and GCNII. We also include graph OOD algorithms, SRGNN [343] and EERM [290], with GCN encoders. EERM(II) is a variant of EERM with a GCNII encoder For a fair comparison, we show GCN performance on an i.i.d. random split, GCN(i.i.d.), sharing the same node sizes for train, validation, and test. Results are shown in Table 4.1. Following observations can be made: **Obs.1:** The performance degradation can be found by comparing OOD setting with i.i.d. one across four datasets, confirming OOD issue existence. **Obs.2:** MLP-based models and deeper GNNs generally outperform vanilla GCN, demonstrating the superiority on minority nodes. **Obs.3:** Graph OOD algorithms with GCN encoders struggle to yield good performance across datasets, indicating a unique challenge over other Graph OOD scenarios. This primarily stems from the difficulty in learning both accurate relationships on homophilic and heterophilic nodes with distinct $\mathbf{P}(\mathbf{Y}|\mathbf{X})$. Nonetheless, it can be alleviated by selecting a deeper GNN encoder, as the homophily ratio difference may vanish in higher-order structure information, with reduced concept shift. **Obs.4:** EERM(II), EERM with GCNII, outperforms the one with GCN. Observations suggest that GNN architecture plays an indispensable role in addressing graph OOD issues, highlighting the new direction.

4.5 Conclusion

In conclusion, this work provides crucial insights into GNN performance meeting structural disparity, common in real-world scenarios. We recognize that aggregation exhibits different effects on nodes with structural disparity, leading to better performance on majority nodes than those in

minority. The understanding also serves as a stepping stone for multiple graph applications.

Our exploration majorly focuses on common datasets with clear majority structural patterns while real-world scenarios, offering more complicated datasets, posing new challenges Additional experiments are conducted on Actor, IGB-tiny, Twitch-gamer, and Amazon-ratings.

CHAPTER 5

REVISITING LINK PREDICTION: A DATA PERSPECTIVE

5.1 Introduction

Graphs are essential data structures that use links to describe relationships between objects. Link prediction, which aims to find missing links within a graph, is a fundamental task in the graph domain. Link prediction methods aim to estimate proximity between node pairs, often under the assumption that similar nodes are inclined to establish connections. Originally, heuristic methods [339, 126] were proposed to predict link existence by employing handcrafted proximity features to extract important ***data factors***, e.g., local structural proximity and feature proximity. For example, Common Neighbors(CN) algorithm [339] assumes that node pairs with more overlapping between one-hop neighbor nodes are more likely to be connected. To mitigate the necessity for handcrafted features, Deep Neural Networks are utilized to automatically extract high-quality proximity features. In particular, Graph Neural Networks (GNNs) [135, 133, 84] become increasingly popular owing to their excellence in modeling graph data. Nonetheless, vanilla GNNs fall short in capturing pairwise structural information [326, 155], e.g., neighborhood-overlapping features, achieving modest performance in link prediction. To address these shortcomings, Graph Neural Networks for Link Prediction(GNN4LP)[325, 279, 36] are proposed to incorporate different inductive biases revolving on pairwise structural information.

New designs on GNN4LP models strive to improve vanilla GNN to capture diverse pairwise data patterns, e.g., local structural patterns [317, 283], the number of paths [346], and structural position [325]. These models have found wide applicability across a myriad of real-world graph problems from multiple domains, e.g., paper recommendation, drug interaction prediction, and protein analysis [141, 102]. A recent benchmark [149] evaluates the performance of GNN4LP models on datasets from diverse domains, and finds performance disparity as there is no universally best-performing GNN4LP model, observing that even vanilla GCN can achieve best performance on certain datasets. [4, 34] reveal similar phenomena across heuristic algorithms. We conjecture the main reasons for such phenomena are that **(i)** From a model perspective, different models often

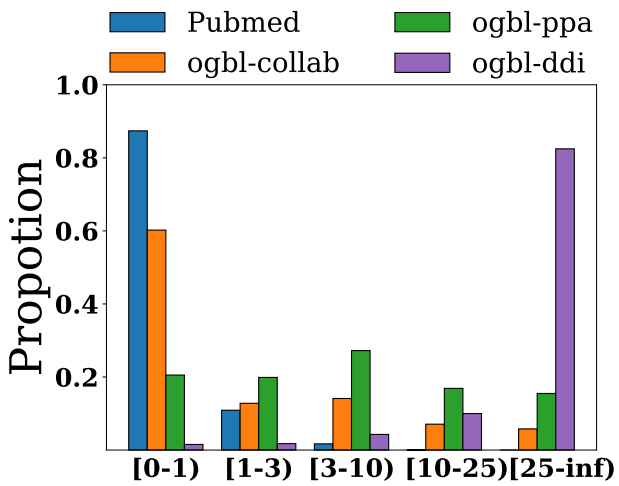


Figure 5.1 Distribution disparity of Common Neighbors across datasets.

have preferred data patterns due to their distinct capabilities and inductive biases. **(ii)** From a data perspective, graphs from different domains could originate from distinct underlying mechanisms of link formation. Figure 5.1 illustrates this disparity in the number of CNs on multiple benchmark datasets. Notably, edges in the OGBL-PPA and OGBL-DDI datasets tend to have many CNs. Considering both model and data perspectives, performance disparity becomes evident where certain models perform well when their preferred data patterns align with particular data mechanisms on particular datasets, but others do not. This suggests that both model and data perspectives are significant to the success of link prediction. While mainstream research focuses on designing better models [325, 326], we opt to investigate a data-centric perspective on the development of link prediction. Such a perspective can provide essential guidance on model design and benchmark dataset selection for comprehensive evaluation.

To analyze link prediction from a data-centric perspective, we must first understand the underlying data factors across different datasets. To achieve these goals, our study proceeds as follows: **(i)** Drawing inspiration from well-established literature [107, 201] in network analysis, we pinpoint three key data factors for link prediction: local structural proximity, global structural proximity, and feature proximity. Comprehensive empirical analyses confirm the importance of these three factors. **(ii)** In line with empirical analysis, we present a latent space model for link prediction, providing theoretical guarantee on the effectiveness of the empirically identified data factors. **(iii)** We conduct

an in-depth analysis of relationships among data factors on the latent space model. Our analysis reveals the presence of incompatibility between feature proximity and local structural proximity. This suggests that the occurrence of both high feature similarity and high local structural similarity within a single edge rarely happens. Such incompatibility sheds light on an overlooked vulnerability in GNN4LP models: they typically fall short in predicting links that primarily arise from feature proximity. **(iv)** Building upon the systematic understandings, we provide guidance for model design and benchmark dataset selection, with opportunities for link prediction.

5.2 Main Analysis

In this section, we conduct analyses to uncover the key data factors for link prediction and the underlying relationships among those data factors. Since underlying data factors contributing to link formation are difficult to directly examine from datasets, we employ heuristic algorithms as a lens to reflect their relevance. Heuristic algorithms calculate similarity scores derived from different data factors to examine the probability of whether two nodes should be connected. They are well-suited for this analysis as they are simple and interpretable, rooted in principles from network analysis [211, 127]. Leveraging proper-selected heuristic algorithms and well-established literature in network analysis, we endeavor to elucidate the underlying data factors for link prediction.

Organization. Revolving on the data perspective for link prediction, the following subsections are organized as follows. Section 5.2.1 focuses on identifying and empirically validating the key data factors for link prediction using corresponding heuristics. In line with the empirical significance of those factors, Section 5.2.2 introduces a theoretical model for link prediction, associating data factors with node distances within a latent space. Links are more likely to be established between nodes with a small latent distance. Section 5.2.3 unveils the relationship among data factors building upon the theoretical model. We then clearly identify an incompatibility between local structural proximity and feature proximity factors. Specifically, incompatibility indicates it is unlikely that the occurrence of both large feature proximity and large local structural proximity within a single edge. Section 5.2.4 highlights an overlooked limitation of GNN4LP models stemming from this incompatibility.

Preliminaries & Experimental Setup. $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an undirected graph where $\mathcal{V}$ and $\mathcal{E}$ are the set of N nodes and $|\mathcal{E}|$ edges, respectively. Nodes can be associated with features $\mathbf{X} \in \mathbb{R}^{n \times d}$, where d is the feature dimension. We conduct analysis on CORA, CITESEER, PUBMED, OGBL-COLLAB, OGBL-PPA, and OGBL-DDI datasets [102, 199] with the same model setting as recent benchmark [149].

5.2.1 Underlying data factors on link prediction

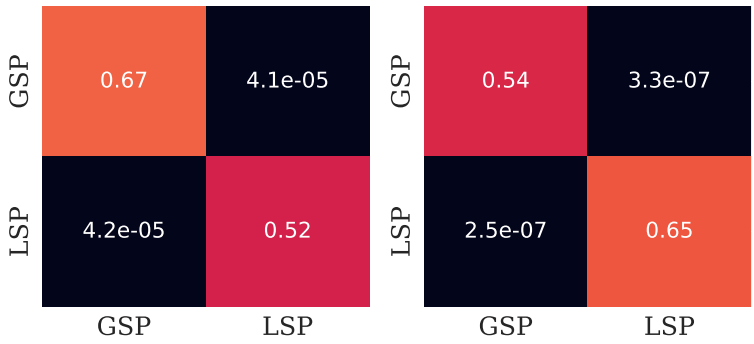
Motivated by well-established understandings in network analysis [56, 281, 144] and heuristic designs [5, 126], we conjecture that there are three key data factors for link prediction.

(1) `Local structural proximity(LSP)`[212] corresponds to the similarity of immediate neighborhoods between two nodes. The rationale behind LSP is rooted in the principle of triadic closure [107], which posits that two nodes with more common neighbors have a higher probability of being connected. Heuristic algorithms derived from the LSP perspective include CN, RA, and AA [5], which quantify overlap between neighborhood node sets. We mainly focus on common neighbors (CN) in the following discussion. The CN score for nodes i and j is calculated as $|\Gamma(i) \cap \Gamma(j)|$, where $\Gamma(\cdot)$ denotes the neighborhood set.

(2) `Global structural proximity(GSP)`[126, 113] goes beyond immediate neighborhoods between two nodes by considering their global connectivity. The rationale behind GSP is that two nodes with more paths between them have a higher probability of being connected. Heuristic algorithms derived from GSP include SimRank, Katz, and PPR [26], to extract the ensemble of paths information. We particularly focus on the Katz heuristic in the following discussion. The Katz score for nodes i and j is calculated as $\sum_{l=1}^{\infty} \lambda^l |\text{paths}^{\langle l \rangle}(i, j)|$, where $\lambda < 1$ is a damping factor, indicating the importance of the higher-order information. $|\text{paths}^{\langle l \rangle}(i, j)|$ counts the number of length-l paths between i and j.

To understand the importance of those data factors, we aim to answer the following questions: **(i)** Does each data factor indeed play a key role in link prediction? **(ii)** Does each factor provide unique information instead of overlapping information?

We further investigate the relationship between heuristics from the same and different data factors. This includes whether heuristics from the same data factor provide similar information for



(a) OGBL-DDI (b) OGBL-COLLAB

Figure 5.2 Overlapping ratio between top-ranked edges on different heuristic algorithms. **Diagonals are the comparison between two heuristics within the same factor**, while others compare heuristics from different factors. MRR is selected as the metric.

link prediction and whether those from different data factors could offer unique perspectives. To this end, we examine disparities in predictions among different heuristics. Similar predictions imply that they provide similar information, while divergent predictions indicate that each factor could provide unique information. Predictions for node pairs can be arranged in descending order according to the predicted likelihood of them being connected. We primarily focus on top-ranked node pairs since they are likely to be predicted as links. Thus, they can largely determine the efficacy of the corresponding algorithm. If two algorithms produce similar predictions, high-likelihood edges should have a high overlap. Else, their overlap should be low.

Experimental results are shown in Figure 5.2. It focuses on the overlapping ratio between the top ranking (25%) node pairs of two different heuristics either from the same data factor or different data factors. We make the following observations: **(i)** Comparing two heuristics from the same factor, i.e., the diagonal cells, we observe that high-likelihood edges for one heuristic are top-ranked in the other. This indicates heuristics from the same data factor capture similar information. **(ii)** Comparing two heuristics derived from different factors, We can observe that the overlapping of top-ranked edges is much low. These observations suggest that **(i)** selecting one representative heuristic for one data factor could be sufficient as heuristics from the same factors share similar predictions, and **(ii)** different factors are unique as there is little overlap in predictions.

5.2.2 Theoretical model for Link Prediction based on underlying data factors

In this subsection, we rigorously propose a theoretical model for link prediction based on the important data factors empirically analyzed above. We first introduce a latent space model and then theoretically demonstrate that the model reflects the effectiveness of LSP, GSP, and FP factors.

The latent space model [96] has been widely utilized in many domains, e.g., sociology and statistics. It is typically utilized to describe proximity in the latent space, where nodes with a close in the latent space are likely to share particular characteristics. In our paper, we propose a latent space model for link prediction, describing a graph with N nodes incorporating both feature and structure proximity, where each node is associated with a location in a D-dimensional latent space. Intuitions on modeling feature and structure perspectives are shown as follows. *Structural perspective* is of primarily significance in link prediction. In line with this, the latent space model connects the link prediction problem with the latent node pairwise distance d, where d is strongly correlated with structural proximity. A small d_{ij} indicates two nodes i and j sharing similar structural characteristics, with a high probability of being connected. Considering feature and structural perspectives together, we develop an undirected graph model inspired by [245]. Detailed formulation is as follows:

$$P(i \sim j|d_{ij}) = \begin{cases} \dfrac{1}{1+e^{\alpha(d_{ij}-\max\{r_i,r_j\})}} & d_{ij} \leq \max\{r_i, r_j\} \\ 0 & d_{ij} > \max\{r_i, r_j\} \end{cases} \tag{5.1}$$

where $P(i \sim j|d_{ij})$ depicts the probability of forming an undirected link between i and j ($i \sim j$), predicated on both the features and structure. The latent distance d_{ij} indicates the structural likelihood of link formation between i and j. The feature proximity parameter $\beta_{ij} \in [0, 1]$ additionally introduces the influence from the feature perspective. Moreover, the model has two parameters α and r. $\alpha > 0$ controls the sharpness of the function. To ease the analysis, we set $\alpha = +\infty$. r_i is a connecting threshold parameter corresponding to node i. With $\alpha = +\infty$, $\frac{1}{1+e^{\alpha(d_{ij}-\max\{r_i,r_j\})}} = 0$ if $d_{ij} > \max\{r_i, r_j\}$, otherwise it equals to 1. Therefore, a large r_i indicates node i is more likely to form edges, leading to a potentially larger degree. Nodes in the graph can be associated with different r values, allowing us to model graphs with various degree distributions. Such flexibility enables our theoretical model to be applicable to more real-world graphs. We

identify how the model can reveal different important data factors in link prediction. Therefore, we **(i)** derive heuristic scores revolving around each factor in the latent space and **(ii)** provide a theoretical foundation suggesting that each score can offer a suitable bound for the probability of link formation. Theoretical results underscore the effectiveness of each factor.

Effectiveness of Local Structural Proximity (LSP). We first derive the common neighbor (CN) score on the latent space model. Notably, since we focus on the local structural proximity, the effect of the features is ignored. We therefore set the FP parameter $\beta_{ij} = 0$, for ease of analysis. Considering two nodes i and j, a common neighbor node k can be described as a node connected to both nodes i and j. In the latent space, it should satisfy both $d_{ik} < \max\{r_i, r_k\}$ and $d_{kj} < \max\{r_k, r_j\}$, which lies in the intersection between two balls, $V(\max\{r_i, r_k\})$ and $V(\max\{r_k, r_j\})$. Notably, $V(r) = V(1)r^D$ is the volume of a radius r, where $V(1)$ is the volume of a unit radius hypersphere. Therefore, the expected number of common neighbor nodes is proportional to the volume of the intersection between two balls. With the volume in the latent space, we then derive how CN provides a meaningful bound on the structural distance d_{ij}.

Proposition 1 (latent space distance bound with CNs)**.** *For any $\delta > 0$, with probability at least $1 - 2\delta$, we have $d_{ij} \leq 2\sqrt{r_{ij}^{max} - \left(\frac{\eta_{ij}/N - \epsilon}{V(1)}\right)^{2/D}}$, where η_{ij} is the number of common neighbors between nodes i and j, $r_{ij}^{max} = max\{r_i, r_j\}$, and $V(1)$ is the volume of a unit radius hypersphere in D dimensional space. ϵ is a term independent of η_{ij}. It vanishes as the number of nodes N grows.*

Proposition 1 indicates that a large number of common neighbors η_{ij} results in a smaller latent distance d_{ij}, leading to a high probability for an edge connection. We then extend the above analysis on local structure to global structure with more complicated structural patterns.

Effectiveness of Global Structural Proximity (GSP). We first derive the number of paths between node i and j on the latent space. Notably, most heuristics on the GSP factor can be viewed as a weighted number of paths. The key idea is to view each common neighbor node as a path with a length $\ell = 2$, serving as the basic element for paths with a length $\ell > 2$. We denote that the nodes i, j are linked through path of length ℓ, i.e., $i = k_0 \sim k_1 \sim \ldots \sim k_{\ell-1} \sim k_\ell = j$. As we assume each node is only associated with its neighborhood, the probability that the

path $P(k_0 \sim k_1 \sim \ldots \sim k_{\ell-1} \sim k_\ell)$ exists can be easily bounded by a decomposition of $P(k_0 \sim k_1 \sim k_2) \cdot P(k_1 \sim k_2 \sim k_3) \cdots P(k_{\ell-2} \sim k_{\ell-1} \sim k_\ell) = \prod_{l=1}^{\ell-1} P(k_{\ell-1}, k_\ell, k_{\ell+1})$. Notably, each element is the common neighbor probability discussed in Proposition 1, equivalent to the path with $\ell = 2$. We then calculate the volume of the number of paths and derive how it bound the latent distance d_{ij}.

Proposition 2 (latent space distance bound with the number of paths)**.** *For any $\delta > 0$, with probability at least $1 - 2\delta$, we have $d_{ij} \leq \sum_{n=0}^{M-2} r_n + 2\sqrt{r_M^{\max} - \left(\frac{\eta_\ell(i,j) - b(N,\delta)}{c(N,\delta,\ell)}\right)^{\frac{2}{D(\ell-1)}}}$, where $\eta_\ell(i,j)$ is the number of paths of length ℓ between i and j in D dimensional Euclidean space. $M \in \{1, \cdots, \ell-1\}$ is the set of intermediate nodes.*

Proposition 2 indicates that a large number of paths $\eta_\ell(i,j)$ results in a smaller latent distance d_{ij}, leading to a high probability for an edge connection. It demonstrates the effectiveness of GSP factor.

5.2.3 Intrinsic relationship among underlying data factors

In this subsection, we conduct a rigorous analysis elucidating the intrinsic relationship among different factors, upon the theoretical model. Our analyses are two-fold: **(i)** the relationship between structural factors, i.e., LSP and GSP; and **(ii)** the relationship between factors focusing on feature and structure, i.e., FP and LSP, FP and GSP.

The relationship between local and global structural proximity. To consider both local and global structural factors, we treat the CN algorithm as the number of paths $\eta_\ell(i,j)$ with length $\ell = 2$. Therefore, analysis between local and global structural factors can be regarded as the influence of $\eta(i,j)$ on different lengths ℓ. The key for the proof is to identify the effect of ℓ by bounding other terms related with ℓ in Proposition 2, i.e., $\eta_\ell(i,j)$ and $c(N,\delta,\ell)$. We also ignore the feature effect to ease the structural analysis.

Lemma 2 (latent space distance bound with local and global structural proximity)**.** *For any $\delta > 0$, with probability at least $1 - 2\delta$, we have $d_{ij} \leq \sum_{n=0}^{M-2} r_n + 2\sqrt{r_M^{max} - \left(\sqrt{\frac{N\ln(1/\delta)}{2}} - 1\right)^{\frac{2}{D(\ell-1)}}}$, where $\sum_{n=0}^{M-2} r_n$, r_M^{max} serve as independent variables that do not change with ℓ.*

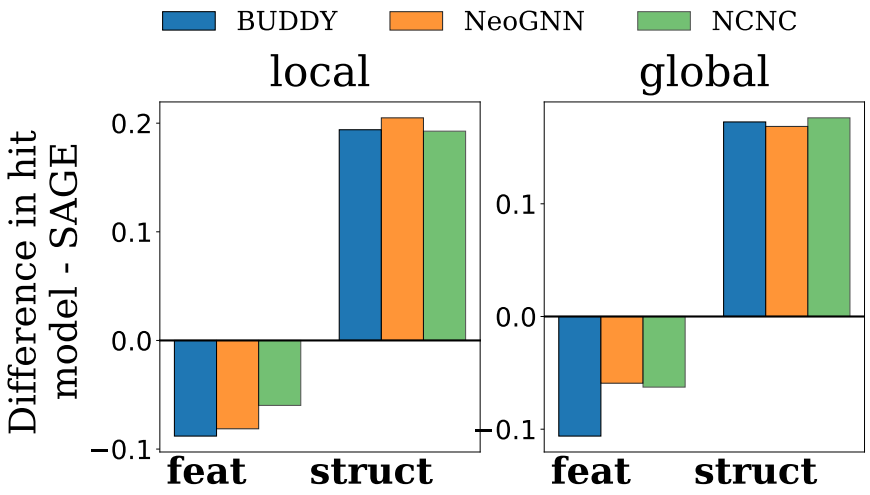


Figure 5.3 Performance comparison between GNN4LP models and SAGE on the OGBL-COLLAB dataset. Bars represent the performance gap on node pairs dominated by feature and structural proximity, respectively. Figures correspond to compare FP with GSP and LSP, respectively.

Given the same number of paths η_ℓ with different lengths ℓ, a small ℓ provides a much tighter bound with close distance in the latent space. The bound becomes exponentially loose with the increase of ℓ as the hop ℓ in $\left(\sqrt{\frac{N\ln(1/\delta)}{2}}-1\right)^{\frac{2}{D(\ell-1)}}$ acts as an exponential coefficient. This indicates that **(i)** When both LSP and GSP are sufficient, LSP can provide a tighter bound, indicating a more important role. **(ii)** When LSP is deficient, e.g., the graph is sparse with not many common neighborhoods, GSP can be more significant.

5.2.4 An overlooked vulnerability in GNN4LP models inspired from data factors

In this subsection, we delve into how the incompatibility between structural proximity and feature proximity affects the effectiveness of GNN4LP models. These models are inherently designed to learn pairwise structural representation, encompassing both feature and structural proximity. Despite their strong capability, the incompatibility between structural and feature factors leads to potentially conflicting training signals. For example, while structural proximity patterns may imply a likely link between two nodes, feature proximity patterns might suggest the opposite. Therefore, it seems challenging for a single model to benefit both node pairs with feature proximity factor and those with the structural ones. Despite most research primarily emphasizing the capability of GNN4LP models on structural proximity, the influence of incompatibility remains under-explored.

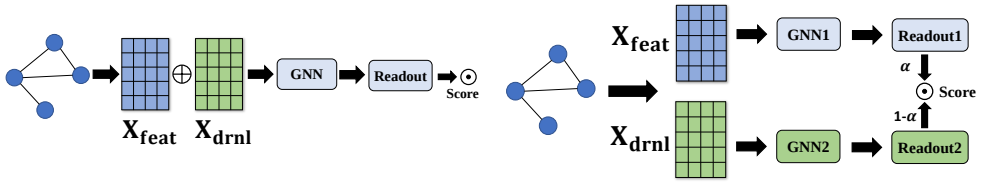
To validate our statement, we conduct experiments to compare the performance of vanilla GNNs, e.g., SAGE and GCN, with the advanced GNN4LP models including Buddy, NeoGNN, and NCNC. The fundamental difference between GNN4LP models and vanilla GNNs is that vanilla GNNs only learn single-node structural representation with limitations in capturing pairwise structural factors while GNN4LP models go beyond. Such comparison sheds light on examining how the key capacity of GNN4LP, i.e., capturing pairwise structural factor, behaves along the incompatibility. Comparisons are conducted on node pairs dominated by different factors, represented as node pairs $\mathcal{E}_s \setminus \mathcal{E}_f$ and $\mathcal{E}_f \setminus \mathcal{E}_s$ with only structural proximity and only feature proximity accurately predicted, respectively. $\mathcal{E}_s$ and $\mathcal{E}_f$ denote node pairs accurately predicted with structural proximity and feature proximity, respectively. Experimental results are presented in Figure 5.3, where the x-axis indicates node pairs dominated by different underlying factors. The y-axis indicates the performance differences between GNN4LP models and vanilla GraphSAGE. A notable trend is that GNN4LP models generally outperform vanilla GNNs on edges governed by LSP and GSP while falling short in those on feature proximity. This underlines the potential vulnerability of GNN4LP models, especially when addressing edges primarily influenced by feature proximity. This underlines the overlooked vulnerability of GNN4LP models on node pairs dominated by the FP factor due to the incompatibility between feature and structural proximity.

5.3 Guidance for practitioners on Link Prediction

In this section, we provide guidance for the new model design and how to select benchmark datasets for comprehensive evaluation, based on the above understandings from a data perspective.

5.3.1 Guidance for the model design

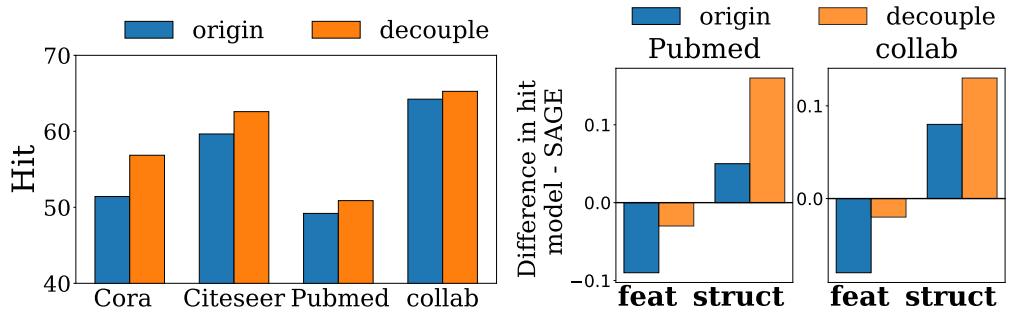
In Section 5.2, we highlight the incompatibility between structural and feature proximity factors in influencing GNN4LP models. When both structural and feature factors come into play simultaneously, there is a potential for them to provide conflicting supervision to the model. Such understanding suggests that the model design should learn the feature proximity factors and pairwise structural ones independently before integrating their outputs, in order to mitigate such incompatibility. In particular, we apply such a strategy to the SEAL [325], a representative GNN4LP



(a) Original coupling SEAL model architecture

(b) Proposed decoupled SEAL model architecture

Figure 5.4 The original SEAL and the proposed decoupled SEAL architectures. $\mathbf{X}_{\text{feat}}$ and $\mathbf{X}_{\text{drnl}}$ are the original node feature and the structural embedding via Double-Radius Node Labeling.



(a) Performance on original and decoupled SEAL.

(b) Comparison between SEAL models and SAGE.

Figure 5.5 Effectiveness of proposed decoupled SEAL.

model. Different from the vanilla GNNs only utilizing original node features $\mathbf{X}_{feat}$ as feature input, it additionally employs local structural features $\mathbf{X}_{drnl}$ by double-radius node labeling (DRNL) based on their structural roles. $\mathbf{X}_{feat}$ and $\mathbf{X}_{drnl}$ are concatenated and then forwarded to one single GNN, as depicted in Figure 5.5a. Therefore, the GNN must wrestle with the incompatibility between FP and structural factors. Guided by the above understanding, we propose the decoupled SEAL, which separates the original node features $\mathbf{X}_{feat}$ and local structural features $\mathbf{X}_{drnl}$ into different GNNs. Each dedicated GNN could learn either feature patterns or pairwise structural patterns separately. The decoupled model architecture is depicted in Figure 5.4b. Experimental results comparing the original SEAL and our proposed decoupled SEAL are illustrated in Figure 5.5a. Notably, our decoupled SEAL consistently outperforms, with gains reaching up to 1.03% on the large ogbl-collab dataset. Furthermore, Figure 5.5b shows comparisons with GraphSAGE, following the same setting with Figure 5.3. The decoupled SEAL demonstrates a reduced performance drop on node pairs dominated by the FP factor with a larger gain on those by structural factors.

Table 5.1 The Hit@10 performance on the newly selected datasets.

	CN	Katz	FH	MLP	SAGE	BUDDY
POWER	12.88	29.85	NA	5.03 ± 0.88	6.99 ± 1.16	19.88 ± 1.37
PHOTO	18.34	7.07	13.78	12.37 ± 4.13	18.61 ± 5.97	18.09 ± 2.52

5.3.2 Guidance for Benchmark Dataset selection

With the recognized data factors and their relationships, we enumerate all potential combinations among different data factors, illuminating the complete dataset landscape. It allows us to categorize prevalent datasets and pinpoint missing scenarios not covered by those datasets. Consequently, we introduce new datasets addressing those identified gaps and offer guidance for practitioners on more comprehensive benchmark dataset selection. In particular, we recognize datasets into four categories considering two main aspects: **(i)** From the feature perspective, we verify whether FP dominates, indicated with decent performance on FH. **(ii)** From the structural perspective, we verify whether GSP dominates, indicated by whether a GSP heuristic can provide additional improvement over LSP (if not, then LSP dominates). Section 5.2.3 demonstrates that such scenario happens when LSP is inadequate. Therefore, there are four categories including **category 1**: both LSP and FP factors dominate. **Category 2**: Only LSP factor dominates. **Category 3**: both GSP and FP factors dominate. **Category 4**: Only GSP factor dominates. The prevalent datasets like CORA, CITESEER , and PUBMED are in category 3 with both GSP and FP factors dominating, while datasets like OGBL-DDI and OGBL-PPA primarily are in category 2, focusing on the LSP factor. We can then clearly identify that two significant dataset categories, 1 and 4, are not covered on existing datasets.

To broaden for a more comprehensive evaluation beyond existing benchmark datasets, we introduce more datasets to cover these categories . This includes the unfeatured POWER dataset in category 4 and the PHOTO dataset in category 1. The categorizations of these datasets are confirmed through experimental results illustrated in Table 5.1. We observe: **(i)** For the POWER dataset with only GSP matters, the Katz significantly outperforms other algorithms, even the GNN4LP model, BUDDY. **(ii)** Deep models do not show superior performance on both datasets, indicating that success focusing on existing datasets cannot extend to the new ones, suggesting potential room

for improvement. We can then provide the following guidance for benchmarking dataset selection for practitioners: **(i)** selecting algorithms that perform best on the datasets belonging to the same category as the proposed one. **(ii)** selecting datasets from their own domain rather than datasets from other domains. To help with that, we collect most of the existing datasets for link prediction covering most domains including biology, transportation, web, academia, and social science, assisting in a more comprehensive evaluation aligning with real-world scenarios. Details on all datasets are in the repository.

5.4 Conclusion

In this work, I explore how to infer the missing edge from a data perspective, elucidating two pivotal factors: LSP and GSP. Theoretical analyses uncover the underlying incompatibility. Inspired by incompatibility, our paper shows a positive broader impact as we identify the overlooked biased prediction in GNN4LP models and show the potential solution to address this issue. Thereby, I propose a GFM model named *UniAug* to leverage the increasing scale of graph data from different domains. I collect thousands of graphs from various domains and pre-train a self-conditioned discrete diffusion model on them. In the downstream stage, I augment the graphs by preserving the original node features and generating synthetic structures. *UniAug* can achieve satisfying performance on the link prediction task while achieving consistent performance gain as the data argumentation for node and graph level tasks.

CHAPTER 6

GRAPH FOUNDATION MODEL ARE ALREADY HERE

6.1 Introduction

Foundation models [23], which are pre-trained on massive data and can be adapted to tackle a wide range of downstream tasks, have achieved inimitable success in various domains, e.g., computer vision (CV) [228] and natural language processing (NLP) [29, 264]. Typically, foundation models can effectively utilize both the prior knowledge obtained from the pre-training stage and the data from downstream tasks to achieve better performance [86] and even deliver promising efficacy with few-shot task demonstrations [60, 194].

Meanwhile, graphs are vital and distinctive data structures that encapsulate non-Euclidean and intricate object relationships. Since various graphs embody unique relations, most graph learning approaches are tailored to train from scratch for a single task on a particular graph. This approach necessitates separate data collection and deployment for each individual graph and task. Consequently, an intriguing question emerges: *Is it possible to devise a Graph Foundation Model (GFM) that benefits from large-scale training with better generalization across different domains and tasks?*

Despite advanced foundation models in other domains, the development of GFMs remains in the infant stage. Recent research has demonstrated initial successes of GFMs in specialized areas, such as knowledge graphs [72, 73] and molecules [18]. Notably, most of these models are built on principles specific to their domains. For instance, ULTRA for knowledge graph completion [72] draws inspiration from double equivariance for inductive link prediction [75]. However, there is still a lack of general guidance on how to build GFMs that can effectively cater to a broad spectrum of graph-based applications.

The key difficulty in designing GFMs lies in finding the invariance across diverse graph data, ranging from social networks to molecular graphs with countless structural patterns, into the same representation space to achieve positive transfer. The answer from the CV and NLP domains is a shared vocabulary. In the NLP foundation models, the text is first broken down into smaller units

based on the vocabulary, which can be words, phrases, or symbols. In the CV foundation models, the image is mapped to a series of discrete image tokens [316, 11] based on the vision token vocabulary. The vocabulary defines the basic units in the particular domain, transferable across different tasks and datasets. Therefore, the key challenges in achieving the GFMs narrow down to how we can find the **graph vocabulary**, the basic transferable units underlying graphs to encode the invariance on graphs.

However, finding a suitable graph vocabulary that works across diverse graphs is challenging, which is the primary focus of this paper.

Our contributions: In this paper, we present a vocabulary perspective to clearly state the position of the GFM. A comprehensive review of the graph transferability principles and corresponding actionable steps is illustrated in Section 6.2, serving us the principle for future vocabulary construction and the GFM design. In Section 6.3, we discuss the potential for building the GFM following neural scaling laws from several perspectives (1) building and training vocabulary from scratch, and (2) leveraging existing LLM. Finally, we introduce more insights and open questions to inspire constructive discussions on the GFMs in Section 6.4.

6.2 Graph Transferability Principles with Actionable Steps

In the last section, we investigate the key to building an effective GFM, which lies in constructing a suitable graph vocabulary to keep the essential invariance across datasets and tasks. Despite existing successes, more graph transferability principles, identifying different invariances, can serve as guidance for constructing new suitable graph vocabulary for future GFMs. We present a few actionable next steps inspired by these principles, highlighting their potential benefits.

The following discussions are organized as follows: We first provide a general introduction to the graph transferability principles in Section 6.2.1. Detailed task-specific principles on node classification, link prediction, and graph classification tasks can be found in Section 6.2.2, 6.2.3, and 6.2.4, respectively. We finally discuss the principles for task transferability in Section 6.2.5. Notably, the following discussions majorly concentrate on the transferability of the graph structure.

6.2.1 An overview on Graph transferability principles

In this subsection, we introduce principles that enable transferability on graphs, focusing on three key aspects: network analysis, expressiveness, and stability.

Network analysis provides a conventional understanding of the network system by identifying fundamental graph patterns, e.g., network motif [202] and establishing the key principles, e.g., triadic closure principle [107] and homophily principle, which are generally valid across different domains. Those principles have been generally utilized to guide the design of advanced GNNs. For example, the state-of-the-art GNN for link prediction [283] is a Neural Common Neighbor, inspired by the triadic closure principle. Despite its effectiveness, network analysis heavily relies on expert knowledge without a provable guarantee.

Expressiveness provides a theoretical background as to which functions graph neural architectures can model in general, e.g., a well-known connection that graph-level performance of GNNs is bounded by Weisfeiler-Leman tests [299, 208, 207]. The most-expressive structural representation [257] is the key concept describing that the representation of two node sets should be invariant if and only if the node sets are symmetric with a permutation equivalence. Such most-expressive structural representation serves as an important principle to design a suitable graph vocabulary that perfectly distinguishes all non-isomorphic structural patterns in multi-ary prediction tasks.

Stability [241] assesses the representation sensitivity to graph perturbations. It aims to maintain a bounded gap in predictions for pairs under minor perturbations, rather than the expressiveness only distinguishing between isomorphic and non-isomorphic cases. The stability imposes a stricter constraint leading to better generalization. It can be an analogy to the constraint on the graph vocabulary where similar structure patterns should have similar representation.

6.2.2 Transferability Principles in Node Classification

Network analysis. *Homophily* [127], which describes the phenomenon of linked nodes often sharing similar features ("birds of a feather flock together"), is a longstanding principle in social science. It serves as the principle guidance for methods ranging from conventional page-rank [51] and label propagation [37] to recent advanced GNNs. Existing GNN architectures, often crafted

based on the homophily principle, demonstrate strong performance on diverse homophilous graphs across various domains. This adherence to homophily not only enhances model effectiveness but also facilitates model transferability among homophilous graph datasets. Notably, successful transfers among such graphs are evidenced in [311].

While homophily predominates in network analysis, it is not a universal rule. In many real-world scenarios, “opposites attract", resulting in networks characterized by heterophily—where nodes are more likely to link with dissimilar nodes. GNNs built with the homophily principle often struggle with heterophilious networks, except in cases of “good heterophily” [183, 177], where GNNs can identify and leverage consistent patterns in connections between dissimilar nodes. However, most heterophilious networks are complex and varied, posing challenges for GNNs due to their irregular and intricate interaction patterns [178, 282, 187]. Consequently, GNNs’ transferability, more assured in homophilous graphs, is facing significant challenges in heterophilous ones.

Stability. [318] theoretically establishes the relationship between transferability and network stability, demonstrating that graph filters with enhanced spectral smoothness and a smaller maximum frequency response exhibit improved transferability in terms of node features and structure, respectively. In particular, spectral smoothness, characterized by the Lipschitz constant of the graph filter function of the corresponding GNN indicates stability against edge perturbations. The maximum frequency response, reflecting the highest spectral frequency after applying a graph filter (essentially the largest eigenvalue of the Laplacian matrix), describes stability against feature perturbations.

Actionable steps inspired by principles. [187] illustrates the network analysis principle that a single GNN can perform well on either homophily patterns or heterophily patterns, but not both. This principle provides actionable insight for GFM design, suggesting that the graph vocabulary for homophily patterns and heterophily patterns should be modeled separately. Consequently, the model backbone for GFMs in node classification should not rely on a single GNN, which only excels on either homophilic graphs or heterophilic graphs. A better architecture design choice could be (1) an adaptive GNN with different aggregation filters for homophilic and heterophilic graphs or (2) a

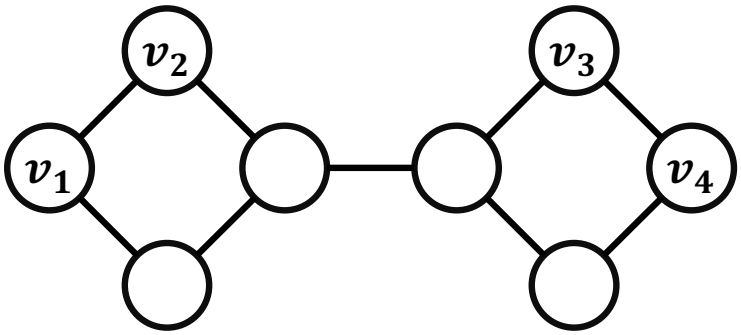


Figure 6.1 In this graph, nodes v_1 and v_4 are isomorphic; links (v_1, v_2) and (v_2, v_4) are not isomorphic. However, vanilla GNN with the same node representations v_1 and v_4 gives the same prediction to links (v_1, v_2) and (v_2, v_4).

graph transformer without a fixed aggregation process [255, 253].

[318] designs a spectral regularization term inspired by the network stability to address the out-of-distribution problem. Adapting spectral regularization for GFMs could be a potential next step.

6.2.3 Transferability Principles in Link Prediction

Network Analysis. Important network analysis principles [192] fall into three primary concepts including: (1) local structural proximity corresponding to the triadic closure principle [107], where friends of friends become friends themselves. It inspires well-known conventional methods including CN, RA, AA [5]. (2) global structural proximity corresponding to the decay factor principle, where two nodes with more short paths between them have a higher probability of being connected. It inspires well-known conventional methods, e.g. Simrank and Katz [126, 113]. (3) feature proximity corresponding to the homophily principle [211] where shared beliefs and thoughts can be found in connected individuals.

These principles guide the evolution of link prediction algorithms, from basic heuristics to sophisticated GNNs [35, 148]. GNNs, inspired by these principles, perform well across diverse graphs in multiple domains. Moreover, [337] provides empirical evidence supporting the beneficial transferability of these guiding principles.

Expressiveness. A vanilla GNN, equipped with only single-node permutation equivalence, cannot achieve transferability for the link prediction task due to its lack of expressiveness. An

example of such failure is shown in Figure 6.1 with a featureless graph. v_1 and v_4 are represented identically by the vanilla GNN, as they possess identical neighborhood structures.

Therefore, the similarity between v_1 and v_2 will be the same as the one between v_4 and v_2, leading to identical representations and predictions for both links (v_1, v_2) and (v_2, v_4) However, according to the global structural proximity, (v_1, v_2), with a shorter distance of 1, should be more likely to be connected. The vanilla GNN, which computes the representation of v_1' solely from its neighborhood, overlooks the structural dependence with v_2. As a result, this potentially leads to negative transfer, where the GNN might erroneously predict both or neither link to exist, whereas it is more likely that only (v_1, v_2) has a link.

To consider all possible dependencies between node pairs, we aim for the most expressive structural representation for the link prediction. This representation should be invariant if and only if links are symmetric. [325] achieves such structural representation by incorporating node labeling features that depend on both the source and the target nodes in a link. [326] further highlights the key aspects of node labeling design, including: (1) target-nodes-distinguishing, where the source and target nodes have distinct labels compared to other nodes; and (2) permutation equivariance. Node labeling methods that meet these criteria, such as double radius node labeling (DRNL) and zero-one (ZO) labeling, can produce the most expressive structural representations. Many other GNNs [312, 313, 278] can achieve similar expressiveness, serving as the potential backbone for GFMs on the link prediction task. The expressiveness representation can find the complete set of distinct relations to differentiate all nonisomorphic node pairs, thereby mitigating the risk of negative transfer in standard GNNs. [109] extends the relational Weisfeiler-Leman framework [15] to link prediction and incorporate the concept of labeling tricks to multirelational graphs.

Stability. For those equally expressive structural representations, there may still be a gap in terms of their stability. For example, empirical evidence [326] shows that GNNs with DRNL labeling outperform those with ZO labeling. From the perspective of stability, it is crucial to maintain a bounded gap in predictions for pairs under minor perturbations. [278] provides a theoretical analysis identifying key properties of stable positional encoding (GNNs should be rotation and permutation

equivariant to positional encodings) that enhance generalization. The stable positional encoding may be directly applied towards better GFMs.

Actionable step inspired by principles. [192] illustrates the network analysis principle concerning the incompatibility between structural proximity and feature proximity. Node pairs with high feature proximity are likely to be with low local structural proximity and vice versa. This incompatibility leads to over-emphasis on node pairs with high structural proximity while neglecting those with high feature proximity. This principle provides actionable insight for the GFM design, suggesting that the graph vocabulary for feature proximity patterns and structural proximity patterns should be modeled separately. Consequently, the model backbone for GFMs in link prediction should separately encode the pairwise structural proximity and the feature proximity.

A GNN following the expressiveness principles could include all the important structural information relevant to the link prediction [326]. [59] utilizes in-context learning to effectively transfer expressive GNN representations to new, unseen graphs. Satisfying performance can be found across graphs from biology, transport, web, and social domains. An actionable next step could be to better utilize expressive representations for downstream graphs from specific domains.

6.2.4 Transferability Principles in Graph Classification

Network Analysis. Network motifs, typically composed of small and recurrent sub-graphs, are often considered the building blocks of a graph [203, 20]. A proper selection of the motif set can cover most essential knowledge on the specific datasets. Graph kernels [273] are proposed to quantify motif counts or other pre-defined graph structural features and then utilize the extracted features to build a classifier such as SVM. Despite the essential motif sets from different domains being generally different, there could exist a uniform set of motifs shared across different domains. In such cases, the positive transfer can be found on the uniform sets, where [17] shows the positive transfer across neuronal connectivity networks, food webs, and electronic circuits. Therefore, we conjecture that the network motif could be the base unit for the vocabulary (a set of invariant elements) for the graph classification as it is both explainable and potentially shared across graphs.

Expressiveness. [322] proposes a unified framework to understand the ability of different GNNs

to detect and count graph substructures (motif). More expressive GNN which could detect more diverse motifs and construct a richer graph vocabulary. In analogy with the uniform motif sets, we conjecture that it is more possible for expressive GNNs to find the uniform motif sets and achieve better transferability.

Stability. [110] proposes a provably stable position encoding that surpasses the expressive sign and invariant encoding [142] and modeling [158], enabling minimal changes to positional encodings on the minor modifications to the Laplacian. The key innovation is to apply a weighted sum of eigenvectors instead of treating each eigen-subspace independently. Satisfying performance can be observed on the out-of-distribution molecular graph prediction. Such stable positional encoding may be directly applied towards better GFMs.

Actionable step inspired by principles. Inspired by the network analysis with graph kernels, one concrete next step towards GFMs could revolve around how to identify frequent network motif [95, 232] which should be transferable across all graphs. Expressive GNNs with better network motif model capability could be a suitable architecture for GFMs.

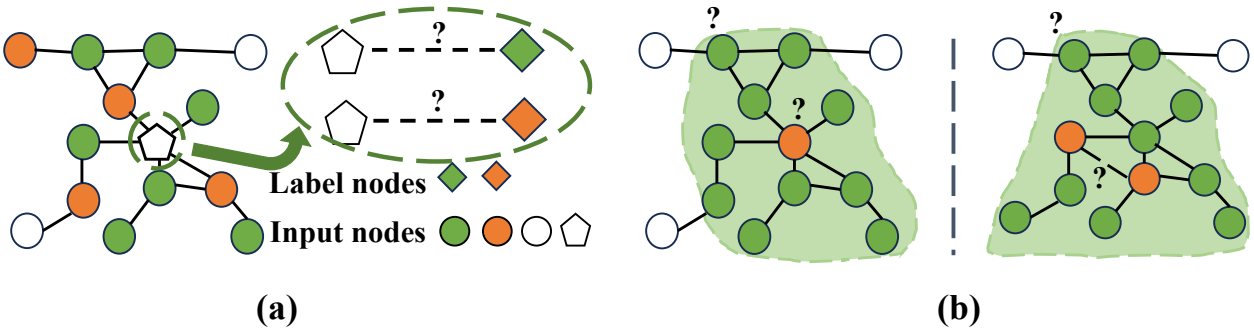


Figure 6.2 Unifying different task formulations: (a) Link view: Given the target node, node classification is converted to the link prediction between the target node and corresponding label nodes. (b) Subgraph view: Node classification (orange node) is converted to the (green) ego-graph classification. Link prediction (orange nodes) is converted to the (green) induced-subgraph classification.

6.2.5 Transferability Principles across Tasks

A unified task formulation is generally employed to facilitate transferability across various tasks. The unified task formulation enables (1) enlarging the dataset size via converting datasets for

different downstream tasks as one and (2) utilizing one pre-training model to serve different tasks. The significance of aligning task formulations is evident in the following example: [118] shows that using link prediction directly as a pretext task leads to negative transfer for node classification. However, by reformulating node classification into a link prediction problem [258, 108], where a node's class membership is treated as the link likelihood between the node and label nodes, positive transfer is achieved. [172, 259] further propose a sub-graph view to adapt the node classification as an ego graph classification, and link prediction as a binary classification on the induced sub-graph of the target node pair. Figure 6.2 provides illustrative examples for these two unified views. More recently, [164] unifies node-level, link-level, and graph-level tasks via (1) adding a virtual prompt node and (2) connecting the virtual nodes to nodes of interests, i.e., the center node for node classification, source and target nodes for link prediction, and all nodes for graph classification.

A unified formulation provides the possibility for co-training all tasks together while it remains unknown whether this can be done without negative transfer. Moreover, the unified task formulation may not be necessary to achieve transfer across tasks. It is generally utilized for supervised co-training and prompt-based prediction as discussed above. A GFM can be (1) pre-trained with self-supervised tasks and (2) adapted to downstream tasks via fine-tuning without requiring specific task formulations. The success is due to the transferability principles across different tasks. However, there remains limited study in this direction. We list a few existing principles as follows. (1) Node classification and link prediction tasks share the feature homophily as an important principle. (2) [165] indicates that the global structural proximity principle on the link prediction can improve the node classification performance on the non-homophilous graph. (3) The triadic closure in the link prediction is a particular network motif utilized in the graph classification. There are more shared motifs [92, 61, 4, 143] on both graph classification and link prediction tasks. We emphasize the importance of cross-task transferability principles as an important future direction.

6.3 Neural Scaling Law on GFMs

The success of the foundation model can be attributed to the validity of the neural scaling law [124] which shows performance enhancement with increasing model scale and data scale. In

this section, we first discuss when the neural scaling law happens in Section 6.3.1 from a graph vocabulary perspective. We then discuss techniques towards successful data scaling and model scaling in Section 6.3.2 and 6.3.3, respectively. We finally discuss the potential of leveraging large-scale LM on the graph domain in Section 6.3.4.

6.3.1 When Neural Scaling Law Happens

In section 6.2, we discuss the underlying transferable principles guiding future vocabulary construction. This principle guidance has led to the successful scaling behavior in the material science domain [252, 323, 16] with the help of the geometric prior. Nevertheless, we are still cautious about whether the existing success can be extended to the graph domain. The key concern is whether graphs can strictly follow those principles. Uncertainty can be found on the human-defined graph construction criteria [28]. For instance, the construction knowledge relying on expert knowledge may lead to uncertainty in edges [308]. [45, 153] observe that mislabeled samples exist widely in all datasets, where the popular CiteSeer dataset has more than 15% wrongly labeled data. Despite the above uncertainty, different graph constructions with manual design can follow opposite principles. For instance, Ogbn-arxiv [103] and Arxiv-year [157] are two node classification datasets with identical graph information. The only difference lies in the label where Ogbn-arxiv employs paper categories and Arxiv-year uses publication years as labels, resulting in conflicting homophily and heterophily properties [187]. Therefore, when uncertainties and opposite graph constructions exist, the scaling behavior may not happen as the data do not obey the graph transferable principles.

6.3.2 Data Scaling

Data scaling refers to the phenomenon that the performance consistently improves with the increasing data scale. [40, 108] initially validate that GNNs trained in both supervised and self-supervised manners follow data scaling law on molecular property predictions, and node classification on text-attributed graphs. [31] further exhibits that the similarity between pre-training data and downstream task data serves as a prerequisite for the data scaling on graphs. Specifically, [31] provides concrete guidance on how to select the pre-training data via the graphon signal analysis and the essential network property, i.e. network entropy, respectively. Notably, all principles

mentioned in Section 6.2 can be applied to facilitate positive transfer with data scaling phenomena. [263] indicates the possibility to achieve scaling behavior across different domains.

A limitation in current research on data scaling is graph data insufficiency, in contrast to the readily available trillion-level real-world data in CV and NLP domains. The key reasons are two-fold: (1) constructing graphs requires expert intervention e.g., defining relationships (2) intellectual property issues.

Synthetic graph generation can be utilized to alleviate the data insufficiency issue, enabling more comprehensive training. Traditional graph generative models [7, 233, 6, 146] are capable of generating graphs satisfying some certain statistical properties, which still plays an important role on node-level and link-level tasks. Deep generative models on graph [122, 180, 123, 272, 161] have shown great success in generating high-quality synthetic graphs which help graph-level tasks by providing a more comprehensive description of the graph distributions space. With successful evidence of pre-training on synthetic data from other domains [204, 266], we anticipate the potential benefits of high-quality synthetic graphs.

6.3.3 Model Scaling

Model scaling refers to the phenomenon that the performance consistently improves with the increasing model scale. Previous research in NLP indicates that apart from data, the backbone model constitutes a fundamental for scaling [124]. [167] primarily validates the neural scaling law on various graph tasks and model architectures under the supervised setting.

However, [129] demonstrates that the GAT [269] with a larger number of parameters underperforms on the graph regression tasks compared to the smaller-sized counterparts. As a comparison, geometric GNNs scale well to predict atomic potentials in material science [252, 323, 16]. Observations indicate that geometric GNNs with a good geometric-prior vocabulary design can help achieve model scaling over the vanilla GNN.

Graph transformer is another popular choice for the model architecture, where geometric-prior graph vocabulary design is explicitly modeled through either a GNN encoder or positional encoding [209]. [196, 174] show that graph transformers show positive scaling capabilities for

molecular data under a supervised setting. More recently, [334] demonstrates vanilla transformer's effectiveness in protein and molecular property prediction. Particularly, it views the graph as a sequence of tokens forming an Eulerian path [66], which ensures the lossless serialization, and then adopts next-token prediction to pre-train transformers. After fine-tuning, it achieves promising results on the protein association prediction and molecular property prediction and shows that vanilla transformers also follow the model scaling law [124]. Nonetheless, the effectiveness of transformers on other tasks remains unclear.

6.3.4 Leveraging Large-scale LMs for Graphs

LLMs with successful scaling behavior have achieved tremendous success in the NLP domain. Surprisingly, well-trained LLMs can be applied to other domains with satisfying performance such as time series forecast [79], recommendation [119], and search engine [347]. Larger-scale LLMs can even capture key symmetries of crystal structures, suggesting that LLMs may possess a strong simplicity bias [218] across domains by implementing Bayesian model averaging algorithm [331].

A recent line of research on GFMs focuses on leveraging the strong capabilities of LLMs on graph tasks [46, 48, 47]. Our discussions can be roughly categorized on LLM applications (i) conventional graph tasks (such as node, edge, and graph classification), and (ii) language-driven tasks like Graph Question Answer (GQA).

LLMs on conventional graph tasks. One natural way to utilize LLMs is as textual feature encoders [45]. Despite the original node features not being text, [164] manually converts them into knowledge-enhanced text descriptions and then encodes features into textual embedding. This LLM embedding approach offers the following benefits. (i) High-quality features help to achieve satisfactory performance with vanilla GCN [45]. (ii) LLMs encode diverse original features in an aligned feature space, enabling training and inference across graphs from different domains without the feature heterogeneity problem. Notably, when LLMs are utilized as feature encoders for textural understanding, the scaling law does not happen [19], meaning that a larger model does not necessarily lead to better performance.

Another approach is to utilize LLMs as predictors which first fine-tune LLMs and then generate

predictions in a natural language form. [45, 90] treats node classification as text classification on the target node feature, illustrating promising results in the zero-shot setting. However, simply flattening graph structures into prompts does not yield additional improvement, leaving a large performance gap compared to well-trained GNNs [45]. To better encode the graph structure knowledge, methods such as GNNs [261], graph transformers [33], and non-parametric aggregations [43] are utilized as structure encoders. The encoded structural embeddings are then linearly mapped into text space as prompt tokens. LLMs generate predictions based on a concatenation of the prompt token and the textual instruction. Instead of additional graph modeling, [333] employs a novel tree-based prompt design that transforms the graph into sequence while retaining important structural semantics. This approach indicates the potential for LLMs to understand particular graph structures. Overall, proper LLM fine-tuning can achieve satisfying graph performance while efficiency may be a potential issue.

LLMs on language-driven graph tasks. Instead of adapting LLMs for conventional graph tasks, LLMs can also be applied to language-driven tasks they are originally skilled in, for example, Graph Question Answer (GQA). [69, 280] apply LLMs on various GQA tasks, e.g., cycle check, and maximum flow, by describing graph structure with natural language. More recently, [222] incorporates an external GNN tokenizer to encode graph information, achieving satisfying out-of-domain generalization to unseen graph tasks. Interestingly, [222] illustrates that equivariance is not necessary when equipped with LLMs. [91] proposes new real-world challenging GQA tasks and a corresponding LLM-based conversational framework. This framework integrates GNNs and retrieval-augmented generation (RAG) to improve graph understanding and mitigate issues like hallucination, demonstrating effectiveness across multiple domains. Until now, most GQA challenges have focused on abstract graphs without concrete descriptions for each node, creating to obstacle to o leveraging the extensive internal knowledge in LLMs. We call for more real-world GQA challenges enabling better leverage of LLM capabilities.

Overall, a consensus can be found that LLMs can serve as a better textual feature encoder with superior performance. Nonetheless, it remains unclear whether LLMs should play a key role in building GFMs with concerns about the LLM's capability to understand essential graph structures.

[244, 64] theoretically observe that LLMs are required to tackle problems sequentially greedily [200], leading to a shortcut solution rather than a formal analysis on the graph structure.

6.4 Insights & Open Questions

In this section, we explore key insights gained from recent advancements in GFMs and highlight open questions that remain to be addressed in this evolving field.

6.4.1 Potential Redundancy on Pretext Task and Architecture Design

There are mainly two approaches to achieving transferability: (1) designing GNNs with specific geometric properties for transfer, e.g., ULTRA [72], and (2) creating pretext tasks to automatically learn these properties. [118] suggests an overlap between these approaches, indicating that pretext tasks targeting local structural information might be unnecessary, given that GNNs often inherently encode this information. Investigating the strengths and limitations of these techniques, along with providing practical guidance for their selection, could be a valuable research direction. A hypothesis might be that model design methods are more suitable for data that strictly adheres to geometric priors, while pretext task designs are more effective in the opposite scenario.

6.4.2 The Feasibility of GFMs

Graphs can be defined in different ways based on different criteria like similarity or influence between node pairs [28]. We can then categorize graphs based on the observability of the criteria. The observable graph is unambiguously known, e.g., whether one paper cites another paper in a citation graph. Text and images can also be viewed as a specific case of observable graphs. In contrast, the observable ones are manually conducted with ambiguous descriptions of the relationship, e.g., whether one gene regulates the expression of another in a gene-regulate graph. These graphs may not naturally exist in the world, leading to uncertainty with a lack of invariant principle. It remains unknown whether GFMs can learn shared knowledge while avoiding manually introduced noisy patterns.

There are concerns about the benefit of training a GFM on graphs that are neither from the same domain nor share the same downstream task. On the one hand, it seems that training on them simultaneously shows no positive transfer benefit while increasing the risk of the negative

transfer. On the other hand, there may be potential undiscovered transferable patterns that could lead to success. Therefore, we pose an open question of whether there exists a universal structural representation space that can benefit all the graph tasks.

6.4.3 Broader Usage of GFM

In this paper, we majorly focus on building GFMs for conventional graph-focused tasks. Notably, graph formulation provides the universal representation ability, which has a broader usage in other domains, e.g., scene graphs for Computer Vision (CV) [319, 338], bipartite graphs for linear programming [49], and physical graphs for understanding physical mechanisms [250]. To emphasize more broader usage of GFMs, we illustrate the potential advantaged usage of GFMs over existing foundation models in reasoning, computer vision, and code intelligence domains domains. Details can be found as follows.

Reasoning. [111] proposes a task-specific GFM, focusing on neural algorithmic reasoning tasks. A strong reasoning capability can be found with effectiveness across sorting, searching, and dynamic programming tasks. We argue that this GFM following the theory of algorithmic alignment [296] may achieve better reasoning capability than the LLM merely relying on the textual inputs via retrieving concepts co-occur frequently in training data [225].

Computer Vision. Scene graph is a data structure representing objects, their attributes, and the relationships between them within an image, facilitating CV tasks such as image understanding and visual reasoning. However, current research remains a naive scene graph modeling with vanilla GNNs with more emphasis on image modeling. We argue that the GFM on the scene graph may help to preserve global and local scene-object relationships [320], avoiding the potential conflict or redundancy between multiple objectives which frequently appears on the recent popular Sora model [27].

Code Intelligence. Graphs, e.g., code property graph [169], control flow graph, and program dependency graph, play an important role in code-relevant tasks, e.g., vulnerability detection [169], fault localization [230], and code search [160]. Compared to sequence-based modeling with LLMs, graphs can provide a complementary perspective on the overlooked essential code attributes such

as syntax, control flow, and data dependencies. However, the graph modeling remains naive with unknown transferability across different program languages.

Overall, GFMs demonstrate unique value compared to foundation models in other domains. However, they are limited to applications involving graph structure data. An exciting future topic is how to adaptively combine GFMs with other foundation models across different modalities toward a powerful Artificial General Intelligence (AGI).

6.5 Conclusion

From the transferability principles of graphs, we review existing GFMs and ground their effectiveness from a vocabulary view to find a set of basic transferrable units across graphs and tasks. Our key perspectives can be summarized as follows: (1) Constructing a universal GFM is challenging, but domain/task-specific GFMs are approachable with the usual availability of a specific vocabulary. (2) One challenge is developing GFMs following the neural scaling law, which requires more data collection, suitable architecture design, and properly leveraging LLMs. This paper summarizes the current position of GFMs and challenges toward the next step, which may be a blueprint for GFMs to inspire relevant research.

CHAPTER 7

CROSS-DOMAIN GRAPH DATA SCALING: A SHOWCASE WITH DIFFUSION MODELS

7.1 Introduction

The effectiveness of existing foundation models [229, 264, 137] heavily relies on the availability of substantial amounts of data, where the relationship manifests as a scaling behavior between model performance and data scale [125]. Consistent performance gain has been observed with the increasing scale of pre-training data in both Natural Language Processing [125, 97] and Computer Vision [2, 321] domains. This data scaling phenomenon facilitates the development of general models endowed with extensive knowledge and effective data pattern recognition capabilities. In downstream applications, these models are capable of adaptively achieving performance gains across different tasks.

In the context of graphs, the availability of large-scale graph databases [236] enables possible data scaling across datasets and domains. Existing works have demonstrated graph data scaling following two limited settings: in-domain pre-training [293, 173] and task-specific selection for pre-training data [32]. During the pre-training process, each graph in the pre-training pool must be validated as in-domain or relevant to the downstream dataset. Given a specific domain or task, the crucial discriminative data patterns are likely confined to a fixed set [193], leaving other potential patterns in diverse graph data distribution as noisy input. In terms of structure, graphs from different domains are particularly composed of varied patterns [203], making it hard to transfer across domains [182]. For example, considering the building blocks of the graphs, the motifs shared by the World Wide Web hyperlinks only partially align with those shared by genetic networks [203]. Therefore, closely aligning the characteristics of the pre-training graphs and the downstream data both in feature and structure is essential for facilitating positive transfer [32]. As a consequence, the necessity of such meticulous data filtering restricts these methods from scaling up graphs effectively, as they can only utilize a small part of the available data. Given the limitation of the graph pre-training methods, a pertinent question emerges: *How can we effectively leverage the increasing scale of graph data across domains?*

Rather than focusing solely on data patterns specific to particular domains, we aim to develop a model that has a comprehensive understanding of data patterns inherent across various types of graphs. In line with the principles of data scaling, we hypothesize that incorporating a broader range of training datasets can help the model build an effective and universal graph pattern library, avoiding an overemphasis on major data patterns specific to any single dataset [188]. To construct such a general-purpose model, we propose to utilize a diffusion model operating only on the structure as the backbone, for the following key reasons. (1) Unlike features, graph structures follow a uniform construction principle, namely, the connections between nodes. This allows for positive transfer across domains when the upstream and downstream data exhibit similar topological patterns [32]. In particular, while the graph representations of neurons and forward electronic circuits are derived from distinct domains, they still share common motifs [203]. (2) Current supervised and self-supervised methods tend to capture only specific patterns of graph data, with models designed for particular inductive biases [188, 193, 298]. For instance, graph convolutional networks (GCNs) excel in node-level representation learning by emphasizing homophily, whereas graph-level representation learning benefits from expressive GNNs capable of distinguishing complex graph structures. (3) We opt for a structure-only model due to the heterogeneous feature spaces across graphs, which often include missing features or mismatched semantics [189]. For instance, node features yield completely different interpretations in citation networks, where they represent keywords of documents, compared to molecular networks, where they denote properties of atoms. To this end, we pre-train a structure-only diffusion model on thousands of graphs, which serves as the upstream component of our framework.

In the downstream stage, we employ the pre-trained diffusion model as a **Uni**versal graph structure **Aug**mentor (**UniAug**) to enhance the dataset, where diffusion guidance [94, 58, 80] is employed to align the generated structure with the downstream requirements. Specifically, we generate synthetic structures with various guidance objectives, and the resulting graphs consist of *generated structures* and *original node features*. This data augmentation paradigm strategically circumvents feature heterogeneity and fully utilizes downstream inductive biases by applying carefully designed

downstream models to the augmented graphs in a plug-and-play manner. Empirically, we apply *UniAug* to graphs from diverse domains and consistently observe performance improvement in node classification, link prediction, and graph property prediction. To the best of our knowledge, this study represents the first demonstration of a data-scaling graph structure augmentor on graphs across domains.

7.2 Preliminary and Related Work

Learning from unlabeled graphs Graph self-supervised learning (SSL) methods provide examples of pre-training and fine-tuning paradigm [104, 99, 128, 314, 301]. However, these methods benefit from limited data scaling due to feature heterogeneity, structural pattern differences across domains, and varying downstream inductive biases. It is worth mentioning that DCT [162] presents a pre-training and then data augmentation pipeline on molecules. Despite its impressive performance improvement on graph-level tasks, DCT is bounded with molecules and thus the use cases are limited.

Graph data augmentation There have been many published works exploring graph data augmentation (GDA) since the introduction of graph neural networks (GNNs), with a focus on node-level [219, 170, 10], link-level [335, 213], and graph-level [85, 159, 179, 163, 139]. These GDA methods have been generally designed for specific tasks or particular aspects of graph data. In addition, they are often tailored for a single dataset and struggle to transfer to unseen patterns, which limits their generalizability to a broader class of applications.

Diffusion models on graphs Diffusion models [93, 254, 234] are latent variable models that learn data distribution by gradually adding noise into the data and then recovering the clean input. Existing diffusion models on graphs can be classified into two main categories depending on the type of noise injected, i.e. Gaussian or discrete. Previous works employed Gaussian diffusion models both on general graphs [215, 123] and molecules [249, 302]. However, adding Gaussian noise into the adjacency matrix will destroy the sparsity of the graph, which hinders the scalability of the diffusion models [82]. Recent works adapted discrete diffusion models to graphs with categorical transition kernels [272, 44, 41]. We denote the adjacency matrix of a graph as $\mathbf{A}^0 \in \{0,1\}^{n\times n}$ with

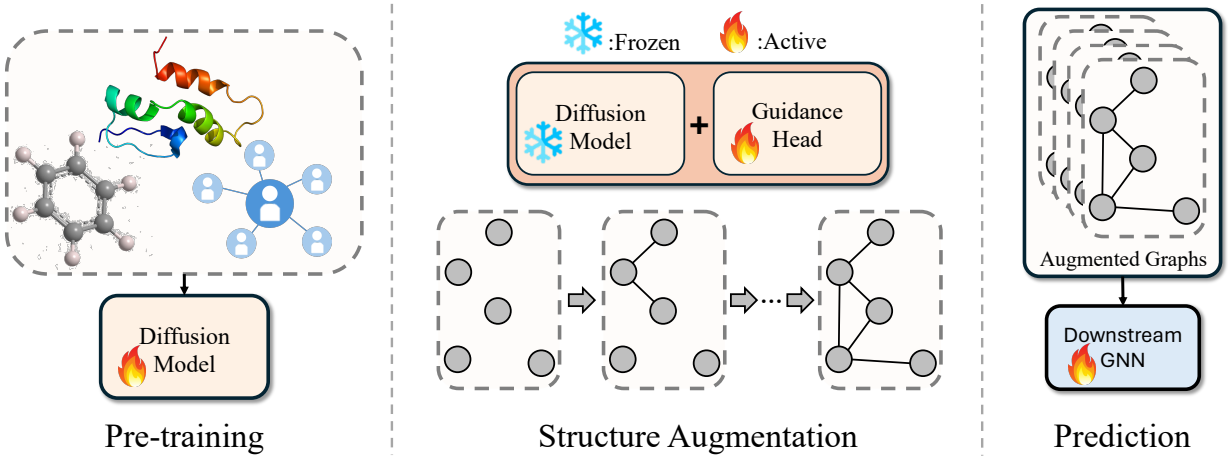


Figure 7.1 The pipeline of *UniAug*. We pre-train a diffusion model across domains and perform structure augmentation on the downstream graphs. The augmented graphs consist of *generated structures* and *original node features* and are then processed by a downstream GNN.

n nodes. We write the *forward process* to corrupt the adjacency matrix into a sequence of latent variables as Bernoulli distribution

$$\begin{aligned} q\left(\mathbf{A}^t \mid \mathbf{A}^{t-1}\right) &= \text{Bernoulli}\left(\mathbf{A}^t; \alpha^t \mathbf{A}^{t-1} + \left(1-\alpha^t\right)\pi\right), \\ q\left(\mathbf{A}^{t-1} \mid \mathbf{A}^t, \mathbf{A}^0\right) &= \frac{q\left(\mathbf{A}^t \mid \mathbf{A}^{t-1}\right) q\left(\mathbf{A}^{t-1} \mid \mathbf{A}^0\right)}{q\left(\mathbf{A}^t \mid \mathbf{A}^0\right)}, \end{aligned} \tag{7.1}$$

where π is the converging non-zero probability, α^t is the noise scale, and $\bar{\alpha}^t = \prod_{i=1}^{t} \alpha^i$. Under predict-$\mathbf{A}^0$ parameterization, the *reverse process* denoise the adjacency matrix with a Markov chain

$$p_\theta\left(\mathbf{A}^{t-1} \mid \mathbf{A}^t\right) \propto \sum_{\widetilde{\mathbf{A}}_0} q\left(\mathbf{A}^{t-1} \mid \mathbf{A}^t, \widetilde{\mathbf{A}}_0\right) \tilde{p}_\theta\left(\widetilde{\mathbf{A}}_0 \mid \mathbf{A}^t\right), \tag{7.2}$$

where $\tilde{p}_\theta(\widetilde{\mathbf{A}}_0 \mid \mathbf{A}^t)$ represents the denoising network that predicts the original adjacency matrix from the noisy adjacency matrix. The parameters are estimated by optimizing the variational lower bound on the negative log-likelihood [9]

$$\begin{aligned} L_{\text{vb}} = &\sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{A}^t \mid \mathbf{A}^0)}\left[D_{\text{KL}}\left(q\left(\mathbf{A}^{t-1} \mid \mathbf{A}^t, \mathbf{A}^0\right) \| \, p_\theta\left(\mathbf{A}^{t-1} \mid \mathbf{A}^t\right)\right)\right] \\ &- \mathbb{E}_{q(\mathbf{A}^1 \mid \mathbf{A}^0)}\left[\log p_\theta\left(\mathbf{A}^0 \mid \mathbf{A}^1\right)\right] + \mathbb{E}_{q(\mathbf{A}^0)}\left[D_{\text{KL}}\left(q\left(\mathbf{A}^t \mid \mathbf{A}^0\right) \| \, p\left(\mathbf{A}^t\right)\right)\right]. \end{aligned} \tag{7.3}$$

7.2.1 Method

In this section, our goal is to build *UniAug* to understand the diverse structure patterns of graphs and perform data augmentation with a range of objectives. As illustrated in Fig.7.1, *UniAug*

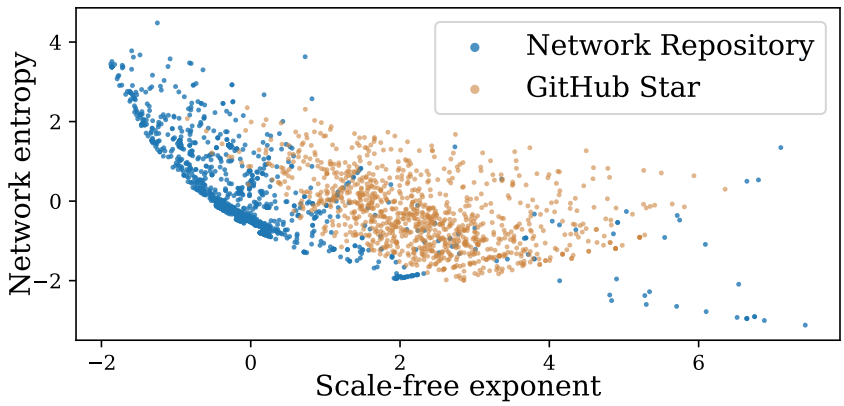


Figure 7.2 Normalized structural properties.

consists of two main components: a pre-trained diffusion model and the downstream adaptation through structure augmentation. We first collect thousands of graphs from varied domains with diverse patterns. To construct a general model free of downstream inductive biases, we train a self-conditioned discrete diffusion model on graph structures. In the downstream stage, we train an *MLP guidance head* on top of the diffusion model with objectives across different levels of granularity. We then augment the downstream dataset by generating synthetic structures through guided generation, where the augmented graph is composed of *generated structures* and *original node features*. Subsequently, we apply the augmented data to train a task-specific model for performing downstream tasks. Below, we elaborate on the data collection process, the architecture of the discrete diffusion model, and the guidance objectives employed.

7.2.2 Pre-training data collection

In light of the data scaling spirit, we expect our pre-training data to contain diverse data patterns with sufficient volume. As graphs from different domains exhibit different patterns [203], we wish to build a collection of graphs from numerous domains to enable a universal graph pattern library with pre-training. Within the publicly available graph databases, Network Repository [236] provides a comprehensive collection of graphs with varied scales from different domains, such as biological networks, chemical networks, social networks, and many more. Among the thousands of graphs in the Network Repository, some of them contain irregular patterns, including multiple levels of edges, extremely high density, et cetera. To ensure the quality of the graphs, we analyze the graph

properties following Xu et al. [295] and filter out the outliers. In addition, we observe that the coverage of graphs in the Network Repository is incomplete according to the network entropy and scale-free exponent, as we observe a relatively scattered space in the middle of Fig. 7.2. To fill in the gap, we include a 1000 graphs subset of the GitHub Star dataset [240]. The selected graphs are utilized to train a discrete diffusion model.

7.2.3 Pre-training through diffusion model

Diffusion models have demonstrated the ability to facilitate transferability from a data augmentation perspective on the images [265, 315, 88]. Unlike the traditional hand-crafted data augmentation methods, diffusion models are capable of producing more diverse patterns with high quality [265]. With the aid of diffusion guidance [94, 58], these methods can achieve domain customization tailored to specific semantic spaces [315, 88]. Despite the success of data augmentation through diffusion models on images, the non-Euclidean nature of graph structures poses challenges for data-centric learning on graphs. In addition, the fact that most graphs in the Network Repository are unlabeled exacerbates the challenges, as the absence of labeled data results in substantially lower generation quality for diffusion models [58, 12].

To address the aforementioned challenges, we propose to construct a self-conditioned discrete diffusion model on graph structures. Unlike Gaussian-based diffusion models, discrete diffusion models [98, 9, 30, 272] operate with discrete transition kernels between latent variables, as shown in Section 7.2. The key reason we opt for the discrete diffusion models lies in the sparse nature of graphs, where adding Gaussian noise into the adjacency matrix will result in a dense graph [82]. On the contrary, discrete diffusion models effectively preserve the sparse structure of graphs during the diffusion process, thus maintaining the efficiency of the models on graphs.

To accommodate for unlabeled graphs, we adopt a self-supervised labeling strategy as an auxiliary conditioning procedure [76, 100]. By leveraging the self-labeling technique, we are able to upscale the diffusion model to data with more diverse patterns [76]. The self-labeling technique requires two components, i.e., a feature extractor and a self-supervised annotator.

Feature extractor. We extract graph-level features by calculating graph properties, including

the number of nodes, density, network entropy, average degree, degree variance, and scale-free exponent following Xu et al. [295]. The first two represent the scale of the graph corresponding to nodes and edges, and the rest indicate the amount of information contained within a graph [295]. We compute the properties of one graph and concatenate them to get a graph-level representation.

Self-supervised annotator. To assign labels to graphs in a self-supervised manner, we employ clustering algorithms on the graph-level representations. The number of clusters is determined jointly by the silhouette score [238] and the separation of the graphs. The candidates of the number of clusters are chosen to ensure different clusters are well separated. Among the candidates, we select the final number of clusters by maximizing the mean Silhouette Coefficient of all samples.

Next we detail the parameterization of the denoising model $\tilde{p}_\theta(\widetilde{\mathbf{A}}^0 \mid \mathbf{A}^t)$ with the self-assigned graph-level labels $\mathbf{k}$. The denoising model recovers the edges of the original adjacency matrix by predicting the connectivity of the upper triangle, which can be formulated as a link prediction problem [325, 144]. Following the link prediction setup, the denoising model is composed of a graph transformer (GT) [251] and an MLP link predictor. Denote the hidden dimension as d, we treat the node degrees as node features and utilize a linear mapping $f_d : \mathbb{R} \mapsto \mathbb{R}^d$ to match the dimension. Similarly, we utilize another linear mapping $f_t : \mathbb{R} \mapsto \mathbb{R}^d$ for timestep t and learnable embeddings $f_k : \{0, \ldots, K\} \mapsto \mathbb{R}^d$ for labels $\mathbf{k}$, where K is the number of clusters. The outputs are summed together and then fed into the GT. Mathematically, we have

$$\begin{aligned} \mathbf{h}^t &= \mathrm{GT}\left(f_d\left(\mathrm{degree}\left(\mathbf{A}^t\right)\right) + f_t(t) + f_k(\mathbf{k}), \mathbf{A}^t\right), \\ \tilde{p}_\theta(\widetilde{\mathbf{A}}^0_{ij} \mid \mathbf{A}^t; t, \mathbf{k}) &\coloneqq \tilde{p}_\theta(\widetilde{\mathbf{A}}^0_{ij} \mid \mathbf{h}^t) = \mathrm{MLP}\left([\mathbf{h}^t_i, \mathbf{h}^t_j]\right). \end{aligned} \tag{7.4}$$

With the above denoising network, our diffusion model is trained on the collected graphs by optimizing the variational lower bound in (7.3). After the pre-training process, we perform adaptive downstream enhancement through graph structure augmentation.

7.2.4 Downstream adaptation through data augmentation

The downstream phase of *UniAug* is to augment the graph topology through guided generation. This guidance process serves to provide downstream semantics for the diffusion model, thus bridging the gap between the pre-training distribution and the downstream datasets. Among the techniques

for diffusion guidance, gradient-based methods [58, 80] offer versatile approaches by incorporating external conditions that are not present during training. For the discrete diffusion process, we opt for the gradient-based NOS method [80] due to its flexibility and efficiency. Specifically, we build an *MLP regression head* $g_\theta : \mathbb{R}^d \mapsto \mathbb{R}^r$ that takes the hidden representations $\mathbf{h}^t$ as the input and outputs the guidance objective of dimension r. Denote τ as the temperature, γ as the step-size, λ as the regularization strength, and ε drawn from $\mathcal{N}(0, I)$, we sample from $\tilde{p}'(\widetilde{\mathbf{A}}^0 \mid \mathbf{h}^t) \propto \tilde{p}_\theta(\widetilde{\mathbf{A}}^0 \mid \mathbf{h}^t) \exp\left(g_\theta\left(\mathbf{h}^t\right)\right)$ via Langevin dynamics

$$\mathbf{h}^{t,\prime} \leftarrow \mathbf{h}^{t,\prime} - \gamma \nabla_{\mathbf{h}^{t,\prime}} \left[\lambda \mathrm{KL}\left(\tilde{p}'\left(\widetilde{\mathbf{A}}^0 \mid \mathbf{h}^{t,\prime}\right) \| \, \tilde{p}'\left(\widetilde{\mathbf{A}}^0 \mid \mathbf{h}^t\right)\right) - g_\theta\left(\mathbf{h}^{t,\prime}\right)\right] + \sqrt{2\gamma\tau}\varepsilon. \tag{7.5}$$

One key question to answer is how to choose the proper guidance objectives. Our goal is to find numerical characteristics that can best describe the structural properties of a graph. This includes supervision signal and self-supervised information on the level of node, edge, and graph.

Node level. Node labels provide the supervision signal for node classification tasks. Beyond node labels, node degrees are a fundamental factor in the evolutionary process of a graph [171]. From the perspective of network analysis, centrality measures indicate the importance of nodes from various viewpoints [24]. Empirically, we observe that utilizing different node-level heuristics as guidance targets tends to yield similar outcomes. Therefore, we focus on node labels and node degrees.

Edge level. Edge-level heuristics can be broadly classified into two categories: local structural heuristics, such as Common Neighbor and Adamic Adar [5], and global structural heuristics, such as Katz [126] and SimRank [113]. Similar to node-level heuristics, empirical observations suggest that different edge-level heuristics tend to yield comparable guidance effects. In this work, we focus on the Common Neighbors (CN) heuristic due to its efficiency. Another edge-level guidance objective is to recover the adjacency matrix from the node representations in a link prediction way, similar to how we parameterize the denoising network. We anticipate that such link prediction objective helps to align the generated graph with the downstream data on the granularity of edges.

Graph level. Graph labels offer the supervision signal for graph classes or regression targets. In addition, we incorporate graph-level properties [295] as quantitative measures to bridge the gap

Table 7.1 Comparison between GDA methods, pre-training methods, and *UniAug*. By cross-domain transfer, we emphasize the ability of the method to train on vastly different domains and benefit all of them.

	GDA methods				Pre-training methods			*UniAug*
	GraphAug	CFLP	Half-Hop	FLAG	AttrMask	D-SLA	GraphMAE	
Effective on graph-level task	✓	–	–	✓	✓	✓	✓	✓
Effective on edge-level task	–	✓	–	✓	–	✓	–	✓
Effective on node-level task	–	–	✓	✓	–	–	✓	✓
In-domain transfer	–	–	–	–	✓	✓	✓	✓
Cross-domain transfer	–	–	–	–	–	–	–	✓

between the pre-training distribution and the downstream dataset. We empirically observe that graph label guidance offers significantly higher performance boosts compared to properties on graph-level tasks. Therefore, we focus on graph labels in our experiments.

We note that all the above objectives are natural choices inspired by heuristics and downstream tasks. There exist many other self-supervised objectives to be explored, such as community-level spectral change [260] and motif occurrence prediction [235]. We leave the study of these objectives as one future work. With the diffusion guidance, we assemble the augmented graphs with *generated structures* and *original node features*. The augmented graphs are then fed into downstream-specific GNNs.

7.2.5 Comparison to existing methods

The data augmentation paradigm of *UniAug* allows us to disentangle the upstream and downstream. We construct a diffusion model as the upstream component to comprehend the structural patterns of graphs across various domains. In addition, we leverage downstream inductive biases with downstream-specific models in a plug-and-play manner. This allows *UniAug* to facilitate cross-domain transfer, offering a unified method that benefits graphs across different domains for various downstream tasks. On the contrary, existing GDA methods are typically designed for specific tasks and hard to transfer to unseen patterns. In the meantime, existing pre-training methods fail to transfer across domains due to heterogeneity in features and structures. This comparison highlights the success of *UniAug* as a data-scaling graph structure augmentor across domains. We summarize the comparison between methods in Table 7.1.

7.3 Results

In this section, we conduct experiments to validate the effectiveness of *UniAug*. We first pre-train our discrete diffusion model on thousands of graphs collected from diverse domains. For each downstream task, we train an *MLP guidance head* with corresponding objectives on top of the diffusion model. We then perform structure augmentation using *UniAug* and subsequently train a task-specific GNN on augmented data for prediction. Through the experiments, we aim to answer the following research questions:

- RQ1: Can *UniAug* benefit graphs from various domains across different downstream tasks?
- RQ2: What is the scaling behavior of *UniAug* corresponding to data scale and amount of compute?
- RQ3: Which components of *UniAug* are effective in preventing negative transfer?

7.3.1 Main results

To get a comprehensive understanding of *UniAug*, we evaluate it on 25 downstream datasets from 7 domains for graph property prediction, link prediction, and node classification.

Baselines. We evaluate our model against three main groups of baselines. (1) Task-specific GNNs: For graph property prediction, we use GIN [298]; for link prediction, we use GCN [136] and NCN [284]; and for node classification, we use GCN [136]. (2) Graph pre-training methods: These include AttrMask, CtxtPred, EdgePred, and InfoMax [104], JOAO [314], D-SLA [128], and GraphMAE [99]. For each of these methods, we pre-train it on the same pre-training set as *UniAug*. While most of the pre-training graphs lack node features, we calculate the node degrees as the input. Each method consists of three pre-trained variants with different backbone GNNs, including GIN, GCN, and GAT. We note that all these methods require the downstream graphs to have the same node feature space as the pre-training data. Therefore, in the fine-tuning stage, we replace the node features of the downstream datasets with node degrees, evaluate all three variants, and report the highest performance for each method in each task. (3) Graph data augmentation (GDA) methods: For graph property prediction, we include S-Mixup [159], GraphAug [179], FLAG [139],

Table 7.2 Mean and standard deviation of accuracy (%) with 10-fold cross-validation on graph classification. The best result is **bold**. The last column is the average rank of each method.

	DD	Enzymes	Proteins	NCI1	IMDB-B	IMDB-M	Reddit-B	Reddit-12K	Collab	A.R.
GIN	75.81 ± 6.11	66.00 ± 7.52	73.32 ± 4.03	78.30 ± 3.20	71.10 ± 2.90	49.07 ± 2.81	90.85 ± 1.30	48.63 ± 1.62	74.54 ± 2.41	5.56
AttrMask	72.93 ± 3.09	23.66 ± 6.09	73.10 ± 3.90	77.67 ± 2.53	71.20 ± 2.40	48.00 ± 3.14	87.50 ± 3.31	48.00 ± 1.60	75.64 ± 1.52	8.00
CtxtPred	75.14 ± 2.67	21.67 ± 3.87	72.21 ± 4.60	78.99 ± 1.29	70.70 ± 1.55	48.20 ± 2.23	90.35 ± 2.31	47.62 ± 2.50	75.60 ± 1.49	7.67
EdgePred	75.64 ± 2.77	22.00 ± 3.32	71.22 ± 3.53	77.82 ± 2.95	70.20 ± 2.23	47.80 ± 2.42	90.80 ± 1.69	48.35 ± 1.44	74.64 ± 2.24	8.56
InfoMax	75.23 ± 3.43	22.50 ± 6.76	71.30 ± 5.18	76.94 ± 1.48	71.60 ± 2.06	46.70 ± 2.46	89.15 ± 2.84	48.98 ± 1.83	75.44 ± 1.12	8.00
JOAO	75.98 ± 2.86	22.17 ± 3.67	71.57 ± 5.31	76.87 ± 2.27	71.02 ± 1.81	48.85 ± 2.06	90.17 ± 2.13	49.01 ± 1.90	74.77 ± 1.71	7.11
D-SLA	74.66 ± 3.30	22.67 ± 4.21	71.97 ± 4.17	77.95 ± 2.11	71.92 ± 2.75	47.28 ± 1.88	89.77 ± 1.87	48.50 ± 1.33	75.99 ± 2.08	7.00
GraphMAE	76.07 ± 3.25	23.00 ± 3.64	70.45 ± 4.19	79.08 ± 2.72	71.50 ± 2.01	47.93 ± 3.03	86.10 ± 3.63	47.67 ± 1.16	74.84 ± 1.36	7.67
S-Mixup	73.12 ± 3.27	66.85 ± 7.04	74.61 ± 5.08	78.91 ± 1.61	69.61 ± 4.43	48.33 ± 5.36	88.65 ± 3.12	48.30 ± 2.50	75.89 ± 3.26	6.67
GraphAug	75.21 ± 2.63	68.14 ± 7.92	74.21 ± 3.70	79.53 ± 3.21	**74.00 ± 3.41**	48.11 ± 1.85	90.50 ± 3.17	49.00 ± 1.99	76.02 ± 2.67	3.67
FLAG	76.87 ± 7.21	68.35 ± 7.45	74.31 ± 4.21	79.03 ± 3.75	68.83 ± 4.67	47.21 ± 3.45	89.11 ± 2.40	47.48 ± 3.01	75.32 ± 3.13	7.00
UniAug	**78.13 ± 2.61**	**71.50 ± 5.85**	**75.47 ± 2.50**	**80.54 ± 1.77**	73.50 ± 2.48	**50.13 ± 2.05**	**92.28 ± 1.59**	**49.48 ± 0.71**	**77.00 ± 2.02**	1.11

GREA [163], and DCT [162]; for link prediction, we include CFLP [335]; and for node classification on heterophilic graphs, we include Half-Hop [10]. The GDA methods are implemented based on chosen task-specific GNNs.

Graph property prediction. We employ graph label guidance for *UniAug* throughout the graph-level tasks by training a 2-layer MLP as the guidance head on the graph labels in the training set. In the augmentation stage, we generate multiple graphs per training sample, and the generated graphs are then fed into the baseline GIN. We present the results of graph classification in Table 7.3. Three key observations emerge from the analysis: (1) Existing pre-training methods show negative transfer compared to GIN. Some special cases are the Enzymes and molecule regression datasets, where all pre-training methods fail to yield satisfactory results. In these datasets, the features are one of the driving components for graph property prediction, while the pre-training methods fail to encode such information due to incompatibility with the feature dimension. This reveals one critical drawback of the pre-training methods: their inability to handle feature heterogeneity. (2) GDA methods yield inconsistent results across different datasets. While these methods enhance performance in some datasets, they cause performance declines in others. This variability is directly reflected in the average rank, where some of them even fall behind the GIN. (3) Unlike the pre-training methods and GDA methods, *UniAug* shows consistent performance improvements against GIN with a large margin. In the molecule regression tasks, *UniAug* effectively compensates for the absence of bond features and achieves performance comparable to DCT, which is a data augmentation method pre-trained on in-domain molecule graphs. These findings affirm that the

Table 7.3 Mean and standard deviation of accuracy (%) with 10-fold cross-validation on graph classification. The best result is **bold**. The last column is the average rank of each method. *Results are taken from DCT [162].

	ogbg-Lipo	ogbg-ESOL	ogbg-FreeSolv	A.R.
GINE*	0.545 ± 0.019	0.766 ± 0.016	1.639 ± 0.146	5.00
GIN	0.543 ± 0.021	0.729 ± 0.018	1.613 ± 0.155	3.67
JOAO	0.859 ± 0.007	1.458 ± 0.040	3.292 ± 0.117	7.00
FLAG*	0.528 ± 0.012	0.755 ± 0.039	1.565 ± 0.098	3.00
GREA*	0.586 ± 0.036	0.805 ± 0.135	1.829 ± 0.368	6.00
DCT*	0.516 ± 0.071	0.717 ± 0.020	1.339 ± 0.075	1.33
UniAug	0.528 ± 0.006	0.677 ± 0.026	1.448 ± 0.049	1.67

pre-training and structure augmentation paradigm of *UniAug* effectively benefits the downstream datasets at the graph level.

Link prediction. We choose three guidance objectives for *UniAug*, including node degree, CN, and link prediction objective, as described in Section 7.2.4. For each objective, we train an MLP to provide guidance information. We then augment the graph structure by generating a synthetic graph and preserving the original training edges, ensuring that the augmented graph does not remove any existing edges. The augmented graph is then fed into a GCN for link prediction. We summarize the results in Table 7.4, which show similar patterns to those observed in graph property prediction: (1) Existing pre-training methods provide negative transfer, especially on datasets with node features. (2) GDA method CFLP leads to performance drops on the datasets without features and also suffers from high computation complexity during preprocessing. (3) *UniAug* enhances performance across all tested datasets. In addition, we employ *UniAug* to NCN [284], one of the state-of-the-art methods for link prediction. The results demonstrate consistent performance boosts from *UniAug* when we apply NCN as the backbone. The structure augmentation paradigm of *UniAug* allows plug-and-play applications to any downstream-specific models, showcasing its adaptability and effectiveness.

Node classification. To demonstrate the effectiveness of *UniAug* in node-level tasks, we transform the node classification into subgraph classification. Specifically, we extract the aggregation tree of each node, i.e., 2-hop subgraph for a 2-layer GCN, and label the subgraph with the center

Table 7.4 Mean and standard deviation across 10 runs on link prediction. Results are scaled ×100. The last two methods are based on NCN, while the rest are GCN-based. The best result is **bold** for two backbones, respectively. The last column is the average rank of each GCN-based method.

	Cora MRR	Citeseer MRR	Pubmed MRR	Power Hits@10	Yeast Hits@10	Erdos Hits@10	Flickr Hits@10	A.R.
GCN	30.26 ± 4.80	50.57 ± 7.91	16.38 ± 1.30	30.61 ± 4.07	24.71 ± 4.92	35.71 ± 2.65	8.10 ± 2.58	4.14
AttrMask	13.43 ± 1.93	20.23 ± 1.29	16.39 ± 3.62	29.92 ± 2.61	25.10 ± 4.77	30.85 ± 3.13	8.77 ± 1.65	6.43
CtxtPred	15.68 ± 2.91	22.31 ± 1.31	13.10 ± 3.70	29.30 ± 3.55	22.96 ± 4.28	34.82 ± 2.55	3.61 ± 1.01	7.86
EdgePred	15.31 ± 3.54	22.91 ± 1.87	17.85 ± 4.45	29.54 ± 3.78	25.78 ± 4.51	34.65 ± 3.84	6.86 ± 3.24	5.43
InfoMax	16.35 ± 2.57	22.90 ± 1.30	15.91 ± 2.71	29.29 ± 4.72	26.33 ± 4.12	35.82 ± 4.12	3.23 ± 0.38	6.00
JOAO	17.21 ± 3.66	23.10 ± 1.41	15.33 ± 3.70	28.98 ± 4.01	26.47 ± 4.65	33.77 ± 3.05	6.01 ± 1.57	6.00
D-SLA	15.55 ± 3.12	23.05 ± 1.54	16.10 ± 3.96	29.37 ± 2.88	26.15 ± 3.32	36.02 ± 4.58	6.70 ± 2.03	5.29
GraphMAE	15.94 ± 1.73	20.35 ± 1.52	13.80 ± 1.36	27.69 ± 1.99	26.51 ± 2.92	35.63 ± 3.61	8.41 ± 2.44	6.14
CFLP	33.62 ± 6.44	**55.20 ± 4.16**	17.01 ± 2.75	16.02 ± 8.31	24.23 ± 5.23	28.74 ± 2.38	OOM	6.43
UniAug-GCN	**35.36 ± 7.88**	54.66 ± 4.55	**17.28 ± 1.89**	**34.36 ± 1.68**	**27.52 ± 4.80**	**39.67 ± 4.51**	**9.46 ± 1.18**	1.29
NCN	31.72 ± 4.48	58.03 ± 3.45	38.26 ± 2.56	27.36 ± 5.00	39.85 ± 5.07	36.81 ± 3.29	8.33 ± 0.92	–
UniAug-NCN	**35.92 ± 7.85**	**61.69 ± 3.21**	**40.30 ± 2.53**	**30.20 ± 1.46**	**42.11 ± 5.74**	**39.26 ± 2.84**	**8.85 ± 0.90**	–

Table 7.5 Mean and standard deviation of accuracy (%) across 10 splits on node classification of heterophilic graphs. The best result is **bold**. The last column is the average rank of each method.

	Cornell	Wisconsin	Texas	Actor	Chameleon*	Squirrel*	A.R.
GCN	59.41 ± 6.03	51.68 ± 4.34	63.78 ± 4.80	30.58 ± 1.29	40.94 ± 3.91	39.11 ± 1.74	3.83
AttrMask	44.86 ± 5.43	53.73 ± 4.31	60.54 ± 5.82	25.31 ± 1.03	35.81 ± 2.88	30.63 ± 1.68	5.83
CtxtPred	40.81 ± 7.78	36.67 ± 17.23	58.92 ± 4.32	23.97 ± 2.63	24.36 ± 4.13	26.26 ± 7.50	9.50
EdgePred	42.70 ± 5.51	48.04 ± 6.63	59.37 ± 5.11	22.99 ± 6.22	21.02 ± 5.06	27.94 ± 8.41	8.83
InfoMax	39.19 ± 12.75	39.80 ± 16.38	58.87 ± 4.06	23.30 ± 4.37	22.59 ± 4.91	27.52 ± 9.09	10.17
JOAO	40.13 ± 8.60	44.70 ± 7.45	57.06 ± 3.43	24.17 ± 5.02	25.81 ± 3.79	31.72 ± 7.03	8.33
D-SLA	41.05 ± 6.88	42.13 ± 9.58	59.93 ± 4.29	23.74 ± 4.06	26.49 ± 4.27	28.50 ± 6.90	8.00
GraphMAE	47.05 ± 4.37	57.06 ± 4.59	63.70 ± 5.51	24.69 ± 0.68	37.18 ± 3.08	31.94 ± 1.65	5.00
Half-Hop	62.46 ± 7.58	76.47 ± 2.61	72.35 ± 4.27	33.95 ± 0.68	38.59 ± 2.89	37.34 ± 2.18	3.00
UniAug	68.11 ± 6.72	69.02 ± 4.96	73.51 ± 5.06	33.11 ± 1.57	**43.84 ± 3.39**	**41.90 ± 1.90**	2.00
UniAug + Half-Hop	**72.43 ± 5.81**	**79.61 ± 5.56**	**77.03 ± 4.27**	**34.97 ± 0.55**	41.94 ± 2.77	38.79 ± 2.61	1.50

*Chameleon and Squirrel are filtered to remove duplicated nodes [224].

Table 7.6 Results of node classification on homophily graphs. Results are scaled ×100.

		Cora	Citeseer	Pubmed
ACC ↑	GCN	81.75 ± 0.73	70.71 ± 0.76	79.53 ± 0.25
	UniAug	81.78 ± 0.60	71.17 ± 0.58	79.54 ± 0.35
SD ↓	GCN	24.51 ± 1.06	22.57 ± 0.80	27.02 ± 0.56
	UniAug	**23.45 ± 0.90**	**19.90 ± 0.81**	**26.50 ± 0.55**

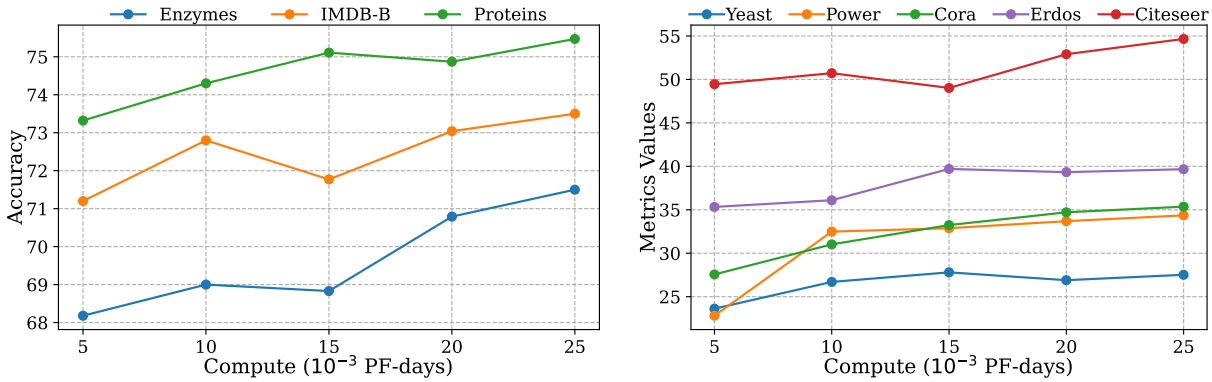


Figure 7.3 Effects of pre-training amount of compute on graph classification (left) and link prediction (right), where one PF-days = $10^{15} \times 24 \times 3600 = 8.64 \times 10^{19}$ floating point operations.

node. We then adopt a strategy similar to graph classification and train a 2-layer classifier as a guidance head. Inspired by the success of structure augmentation on heterophilic graphs [22, 10], we evaluate *UniAug* on 6 heterophilic datasets. We observe phenomena similar to those seen in graph property prediction and link prediction in Table 7.5. One thing to mention is the combination of *UniAug* and Half-Hop. Half-Hop offers performance improvements in four out of six datasets via data augmentation, and combining it with *UniAug* yields even higher results. This highlights the flexibility of *UniAug* and opens up possibilities for further exploration of its use cases. Given the impressive results of *UniAug* on heterophilic graphs, we anticipate it will also help to balance the performance disparities among nodes with different homophily ratios on homophilic graphs [188]. We split the nodes into five groups according to their homophily ratios and calculate the standard deviation (SD) across groups. As shown in Table 7.6, *UniAug* matches the performance of vanilla GCN and also reduces the performance discrepancies corresponding to SD.

7.3.2 Scaling behavior of *UniAug*

In light of the neural scaling law [125, 97, 2, 321, 167], we expect *UniAug* to benefit from an increased volume of data and more compute budget. In this subsection, we investigate the scaling behavior of *UniAug* in terms of data scale and amount of compute for pre-training.

Data scale During the data collection process, we prepare three versions of the training data with increasing magnitude. We first sample 10 graphs per category from the Network Repository [236]

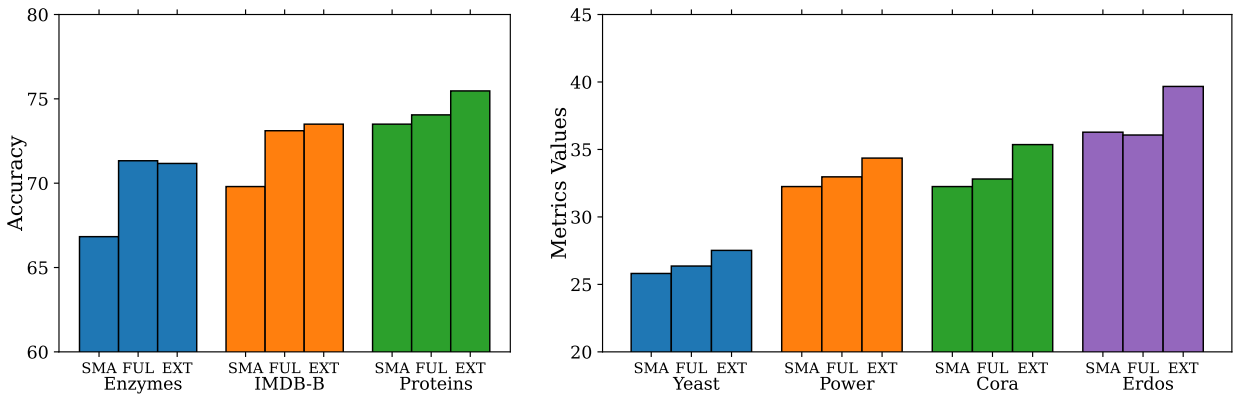


Figure 7.4 Effects of pre-training data scale on graph classification (left) and link prediction (right). The groups SMA, FUL, and EXT represent SMALL, FULL, and EXTRA data collection.

to build a SMALL collection. Next, we gather all the graphs from the Network Repository and filter out large-scale graphs and outliers for a FULL collection. In addition, we add a 1000 graphs subset of the GitHub Star dataset from TUDataset [206] to enlarge the coverage of diverse patterns and form an EXTRA collection. We pre-train three versions of *UniAug* respectively on the three collections and evaluate them on graph classification and link prediction. As shown in Fig. 7.4, we observe a clear trend of increase in performance as we enlarge the scale of pre-training data. This paves the way to scale up *UniAug* to even more pre-training graphs.

Amount of compute We sought to understand how effectively our diffusion model can learn data patterns as we continue to train it. To this end, we checkpointed *UniAug* every 2,000 epochs (5×10^{-3} PF-days) while training on the EXTRA collection, and then applied it to graph classification and link prediction tasks. The results are illustrated in Fig. 7.3. We observe that downstream performance generally improves with prolonged training, while the trend slows down for some datasets when we reach 8,000 epochs. We take the checkpoint at the 10,000th epoch for evaluations. Given the scaling behavior observed, we anticipate *UniAug* to become even more effective with additional resources.

7.3.3 Preventing negative transfer

In the previous parts of the experiments, we showcase the positive transfer of *UniAug* across different tasks. We now investigate which aspects of the design prevent negative transfer. *UniAug* consists of two main components: a pre-trained diffusion model and the structure augmentation

Table 7.7 Demonstration of negative transfer on graph classification (up) and link prediction (down).

	Enzymes	Proteins	IMDB-B	IMDB-M
GIN	66.00 ± 7.52	73.32 ± 4.03	71.10 ± 2.90	49.07 ± 2.81
UniAug	71.50 ± 5.85	75.47 ± 2.50	73.50 ± 2.48	50.13 ± 2.05
w/o self-cond	71.11 ± 7.50	73.31 ± 4.63	71.50 ± 2.27	49.00 ± 2.74
w/o guidance	62.17 ± 3.93	71.15 ± 4.56	53.80 ± 3.29	35.33 ± 3.17
w/ cross-guide	51.50 ± 7.64	72.46 ± 4.35	71.10 ± 2.38	49.20 ± 2.59

through guided generation. In the pre-training process, we inject self-supervised graph labels into the diffusion model and we wonder about the performance of its unconditioned counterpart. Regarding the augmentation process, we examine the impact of diffusion guidance by exploring outcomes when the guidance is either removed or applied using another dataset from a different domain (cross-guide). We summarize the results in Table 7.7 for graph classification and link prediction. All modifications investigated lead to performance declines in both tasks. We observe that removing guidance results in significant negative transfers for graph classification, while the effects of self-conditioning are more pronounced for link prediction. We conclude that both the self-conditioning strategy and diffusion guidance are crucial in preventing negative transfer, underscoring their importance in the design of *UniAug*.

7.4 Conclusion

In this work, we propose a graph structure augmentation pipeline *UniAug* to leverage the increasing scale of graph data. We collect thousands of graphs from various domains and pre-train a self-conditioned discrete diffusion model on them. In the downstream stage, we augment the graphs by preserving the original node features and generating synthetic structures. We apply *UniAug* to node-, link-, and graph-level tasks and achieve consistent performance gain. We have successfully developed a showcase that benefits from cross-domain graph data scaling using diffusion models.

One limitation of the current analysis is the absence of an investigation into the effects of model parameters due to limited resources. Given the scaling behavior of *UniAug* in terms of data scale and amount of compute, we anticipate that a large-scale model will provide significant performance improvements. One future direction is to investigate the adaptation of fast sampling methods to the

discrete diffusion models on graphs. This will lead to lower time complexity and enable broader application scenarios.

CHAPTER 8

CONCLUSION

This dissertation advances graph machine learning by transitioning from Graph Neural Networks (GNNs) to Graph Foundation Models (GFMs), aiming for generalization across diverse graphs. By identifying essential network patterns and addressing GNN limitations under domain shifts, homophily-heterophily trade-offs, and link prediction challenges, this work lays the foundation for scalable, transferable graph learning. The proposed Graph Vocabulary Hypothesis and GFM blueprint pave the way for robust, adaptable models, marking a shift from task-specific GNNs to universally applicable GFMs. Ultimately, this research bridges network science insights with modern graph learning, setting the stage for future advancements in graph Machine learning.

BIBLIOGRAPHY

[1] Rediet Abebe, Nicole Immorlica, Jon Kleinberg, Brendan Lucier, and Ali Shirali. On the effect of triadic closure on network segregation. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, pages 249–284, 2022.

[2] Samira Abnar, Mostafa Dehghani, Behnam Neyshabur, and Hanie Sedghi. Exploring the limits of large scale pre-training. In *International Conference on Learning Representations*, 2022.

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

[4] Ghadeer AbuOda, Gianmarco De Francisci Morales, and Ashraf Aboulnaga. Link prediction via higher-order motif features. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, pages 412–429. Springer, 2020.

[5] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

[6] Edo M Airoldi, David Blei, Stephen Fienberg, and Eric Xing. Mixed membership stochastic blockmodels. *Advances in neural information processing systems*, 21, 2008.

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

[8] Aili Asikainen, Gerardo Iñiguez, Javier Ureña-Carrión, Kimmo Kaski, and Mikko Kivelä. Cumulative effects of triadic closure and homophily in social networks. *Science Advances*, 6(19):eaax7310, 2020.

[9] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

[10] Mehdi Azabou, Venkataramana Ganesh, Shantanu Thakoor, Chi-Heng Lin, Lakshmi Sathidevi, Ran Liu, Michal Valko, Petar Veličković, and Eva L Dyer. Half-hop: A graph upsampling approach for slowing down message passing. In *International Conference on Machine Learning*, pages 1341–1360. PMLR, 2023.

[11] Yutong Bai, Xinyang Geng, Karttikeya Mangalam, Amir Bar, Alan Yuille, Trevor Darrell, Jitendra Malik, and Alexei A Efros. Sequential modeling enables scalable learning for large vision models. *arXiv preprint arXiv:2312.00785*, 2023.

[12] Fan Bao, Chongxuan Li, Jiacheng Sun, and Jun Zhu. Why are conditional generative models better than unconditional ones? *arXiv preprint arXiv:2212.00362*, 2022.

[13] Aseem Baranwal, Kimon Fountoulakis, and Aukosh Jagannath. Graph convolution for semi-supervised classification: Improved linear separability and out-of-distribution generalization. *arXiv preprint arXiv:2102.06966*, 2021.

[14] Aseem Baranwal, Kimon Fountoulakis, and Aukosh Jagannath. Effects of graph convolutions in multi-layer networks. In *The Eleventh International Conference on Learning Representations*, 2023.

[15] Pablo Barcelo, Mikhail Galkin, Christopher Morris, and Miguel Romero Orth. Weisfeiler and leman go relational. In *Learning on Graphs Conference*, pages 46–1. PMLR, 2022.

[16] Ilyes Batatia, Philipp Benner, Yuan Chiang, Alin M. Elena, Dávid P. Kovács, Janosh Riebesell, Xavier R. Advincula, Mark Asta, William J. Baldwin, Noam Bernstein, Arghya Bhowmik, Samuel M. Blau, Vlad Cărare, James P. Darby, Sandip De, Flaviano Della Pia, Volker L. Deringer, Rokas Elijošius, Zakariya El-Machachi, Edvin Fako, Andrea C. Ferrari, Annalena Genreith-Schriever, Janine George, Rhys E. A. Goodall, Clare P. Grey, Shuang Han, Will Handley, Hendrik H. Heenen, Kersti Hermansson, Christian Holm, Jad Jaafar, Stephan Hofmann, Konstantin S. Jakob, Hyunwook Jung, Venkat Kapil, Aaron D. Kaplan, Nima Karimitari, Namu Kroupa, Jolla Kullgren, Matthew C. Kuner, Domantas Kuryla, Guoda Liepuoniute, Johannes T. Margraf, Ioan-Bogdan Magdău, Angelos Michaelides, J. Harry Moore, Aakash A. Naik, Samuel P. Niblett, Sam Walton Norwood, Niamh O'Neill, Christoph Ortner, Kristin A. Persson, Karsten Reuter, Andrew S. Rosen, Lars L. Schaaf, Christoph Schran, Eric Sivonxay, Tamás K. Stenczel, Viktor Svahn, Christopher Sutton, Cas van der Oord, Eszter Varga-Umbrich, Tejs Vegge, Martin Vondrák, Yangshuai Wang, William C. Witt, Fabian Zills, and Gábor Csányi. A foundation model for atomistic materials chemistry, 2023.

[17] Federico Battiston, Giulia Cencetti, Iacopo Iacopini, Vito Latora, Maxime Lucas, Alice Patania, Jean-Gabriel Young, and Giovanni Petri. Networks beyond pairwise interactions: Structure and dynamics. *Physics Reports*, 874:1–92, 2020.

[18] Dominique Beaini, Shenyang Huang, Joao Alex Cunha, Gabriela Moisescu-Pareja, Oleksandr Dymov, Samuel Maddrell-Mander, Callum McLean, Frederik Wenkel, Luis Müller, Jama Hussein Mohamud, et al. Towards foundational models for molecular learning on large-scale multi-task datasets. *arXiv preprint arXiv:2310.04292*, 2023.

[19] Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961*, 2024.

[20] Austin R Benson, David F Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016.

[21] Sannat Singh Bhasin, Vaibhav Holani, and Divij Sanjanwala. What do graph convolutional neural networks learn? *arXiv preprint arXiv:2207.01839*, 2022.

[22] Wendong Bi, Lun Du, Qiang Fu, Yanlin Wang, Shi Han, and Dongmei Zhang. Make heterophily graphs better fit gnn: A graph rewiring approach. *arXiv preprint arXiv:2209.08264*, 2022.

[23] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[24] Stephen P Borgatti. Centrality and network flow. *Social networks*, 27(1):55–71, 2005.

[25] Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.

[26] Sergey Brin and Lawrence Page. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, 56(18):3825–3833, 2012.

[27] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024.

[28] Ivan Brugere, Brian Gallagher, and Tanya Y Berger-Wolf. Network structure inference, a survey: Motivations, methods, and applications. *ACM Computing Surveys (CSUR)*, 51(2):1–39, 2018.

[29] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[30] Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

[31] Yuxuan Cao, Jiarong Xu, Carl Yang, Jiaan Wang, Yunchao Zhang, Chunping Wang, Lei Chen, and Yang Yang. When to pre-train graph neural networks? an answer from data generation perspective! *arXiv preprint arXiv:2303.16458*, 2023.

[32] Yuxuan Cao, Jiarong Xu, Carl Yang, Jiaan Wang, Yunchao Zhang, Chunping Wang, Lei Chen, and Yang Yang. When to pre-train graph neural networks? an answer from data generation perspective! *arXiv preprint arXiv:2303.16458*, 2023.

[33] Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang

Yang. Graphllm: Boosting graph reasoning ability of large language model. *arXiv preprint arXiv:2310.05845*, 2023.

[34] Deepayan Chakrabarti. Avoiding biases due to similarity assumptions in node embeddings. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 56–65, 2022.

[35] Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Hammerla, Michael M Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching. *arXiv preprint arXiv:2209.15486*, 2022.

[36] Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Hammerla, Michael M Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching. In *ICLR*, 2023.

[37] Nitesh V Chawla and Grigoris Karakoulas. Learning from labeled and unlabeled data: An empirical study across techniques and domains. *Journal of Artificial Intelligence Research*, 23:331–366, 2005.

[38] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3438–3445, 2020.

[39] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 295–305, 2022.

[40] Dingshuo Chen, Yanqiao Zhu, Jieyu Zhang, Yuanqi Du, Zhixun Li, Qiang Liu, Shu Wu, and Liang Wang. Uncovering neural scaling laws in molecular representation learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[41] Jialin Chen, Shirley Wu, Abhijit Gupta, and Zhitao Ying. D4explainer: In-distribution explanations of graph neural network via discrete denoising diffusion. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[42] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International conference on machine learning*, pages 1725–1735. PMLR, 2020.

[43] Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. Llaga: Large language and graph assistant. *arXiv preprint arXiv:2402.08170*, 2024.

[44] Xiaohui Chen, Jiaxing He, Xu Han, and Liping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. In *International Conference on Machine Learning*,

pages 4585–4610. PMLR, 2023.

[45] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Haifang Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. Exploring the potential of large language models (llms) in learning on graphs. *ArXiv*, abs/2307.03393, 2023.

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

[47] Zhikai Chen, Haitao Mao, Jingzhe Liu, Yu Song, Bingheng Li, Wei Jin, Bahare Fatemi, Anton Tsitsulin, Bryan Perozzi, Hui Liu, et al. Text-space graph foundation models: Comprehensive benchmarks and new insights. *arXiv preprint arXiv:2406.10727*, 2024.

[48] Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. Label-free node classification on graphs with large language models (llms). *arXiv preprint arXiv:2310.04668*, 2023.

[49] Ziang Chen, Jialin Liu, Xinshang Wang, and Wotao Yin. On representing linear programs by graph neural networks. In *The Eleventh International Conference on Learning Representations*, 2022.

[50] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*.

[51] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021.

[52] Yoonhyuk Choi, Jiho Choi, Taewook Ko, and Chong-Kwon Kim. Is signed message essential for graph neural networks? *arXiv preprint arXiv:2301.08918*, 2023.

[53] Eugenio Clerico, George Deligiannidis, and Arnaud Doucet. Wide stochastic networks: Gaussian limit and pac-bayesian training. In *International Conference on Algorithmic Learning Theory*, pages 447–470. PMLR, 2023.

[54] Sergio Currarini, Matthew O Jackson, and Paolo Pin. An economic model of friendship: Homophily, minorities, and segregation. *Econometrica*, 77(4):1003–1045, 2009.

[55] Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Co-clustering based classification for out-of-domain documents. In Pavel Berkhin, Rich Caruana, and Xindong Wu, editors, *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, pages 210–219. ACM, 2007.

[56] Nur Nasuha Daud, Siti Hafizah Ab Hamid, Muntadher Saadoon, Firdaus Sahran, and Nor Badrul Anuar. Applications of link prediction in social networks: A review. *Journal of Network and Computer Applications*, 166:102716, 2020.

[57] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[58] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[59] Kaiwen Dong, Haitao Mao, Zhichun Guo, and Nitesh V Chawla. Universal link predictor by in-context learning. *arXiv preprint arXiv:2402.07738*, 2024.

[60] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

[61] Yuxiao Dong, Reid A Johnson, Jian Xu, and Nitesh V Chawla. Structural diversity and homophily: A study across more than one hundred big networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 807–816, 2017.

[62] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Learning structural node embeddings via diffusion wavelets. In *Association for Computing Machinery Special Interest Group on Knowledge Discovery and Data Mining*, pages 1320–1329. ACM, 2018.

[63] Lun Du, Xiaozhou Shi, Qiang Fu, Xiaojun Ma, Hengyu Liu, Shi Han, and Dongmei Zhang. Gbk-gnn: Gated bi-kernel graph neural networks for modeling both homophily and heterophily. In *Proceedings of the ACM Web Conference 2022*, pages 1550–1558, 2022.

[64] Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jian, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D Hwang, et al. Faith and fate: Limits of transformers on compositionality. *arXiv preprint arXiv:2305.18654*, 2023.

[65] Gintare Karolina Dziugaite, Kyle Hsu, Waseem Gharbieh, Gabriel Arpino, and Daniel Roy. On the role of data in pac-bayes bounds. In *International Conference on Artificial Intelligence and Statistics*, pages 604–612. PMLR, 2021.

[66] Jack Edmonds and Ellis L. Johnson. Matching, euler tours and the chinese postman. *Mathematical Programming*, 5:88–124, 1973.

[67] Anna Evtushenko and Jon Kleinberg. The paradox of second-order homophily in networks. *Scientific Reports*, 11(1):13360, 2021.

[68] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.

[69] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. *arXiv preprint arXiv:2310.04560*, 2023.

[70] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics reports*, 659:1–44, 2016.

[71] Kimon Fountoulakis, Amit Levi, Shenghao Yang, Aseem Baranwal, and Aukosh Jagannath. Graph attention retrospective. *arXiv preprint arXiv:2202.13060*, 2022.

[72] Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards foundation models for knowledge graph reasoning. *arXiv preprint arXiv:2310.04562*, 2023.

[73] Mikhail Galkin, Jincheng Zhou, Bruno Ribeiro, Jian Tang, and Zhaocheng Zhu. Zero-shot logical query reasoning on any knowledge graph. *arXiv preprint arXiv:2404.07198*, 2024.

[74] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189. PMLR, JMLR.org, 2015.

[75] Jianfei Gao, Yangze Zhou, and Bruno Ribeiro. Double permutation equivariance for knowledge graph completion. *arXiv preprint arXiv:2302.01313*, 2023.

[76] Shanghua Gao, Zhong-Yu Li, Ming-Hsuan Yang, Ming-Ming Cheng, Junwei Han, and Philip Torr. Large-scale unsupervised semantic segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[77] Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pages 3419–3430. PMLR, 2020.

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

[79] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems*, 36, 2024.

[80] Nate Gruver, Samuel Stanton, Nathan Frey, Tim GJ Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew G Wilson. Protein design with guided discrete diffusion. *Advances in Neural Information Processing Systems*, 36, 2024.

[81] Shurui Gui, Xiner Li, Limei Wang, and Shuiwang Ji. Good: A graph out-of-distribution benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[82] Kilian Konstantin Haefeli, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Diffusion models for graphs benefit from discrete state spaces. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*, 2022.

[83] Jonathan Halcrow, Alexandru Mosoi, Sam Ruth, and Bryan Perozzi. Grale: Designing networks for graph learning. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2523–2532, 2020.

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

[85] Xiaotian Han, Zhimeng Jiang, Ninghao Liu, and Xia Hu. G-mixup: Graph data augmentation for graph classification. In *International Conference on Machine Learning*, pages 8230–8248. PMLR, 2022.

[86] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. *AI Open*, 2:225–250, 2021.

[87] Mingguo He, Zhewei Wei, Hongteng Xu, et al. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. *Advances in Neural Information Processing Systems*, 34:14239–14251, 2021.

[88] Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip Torr, Song Bai, and XIAOJUAN QI. Is synthetic data from generative models ready for image recognition? In *The Eleventh International Conference on Learning Representations*, 2023.

[89] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 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*, pages 639–648, 2020.

[90] Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning, 2023.

[91] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *arXiv preprint arXiv:2402.07630*, 2024.

[92] Justus Isaiah Hibshman, Daniel Gonzalez, Satyaki Sikdar, and Tim Weninger. Joint subgraph-

to-subgraph transitions: Generalizing triadic closure for powerful and interpretable graph modeling. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 815–823, 2021.

[93] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[94] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[95] Tomaž Hočevar and Janez Demšar. A combinatorial approach to graphlet counting. *Bioinformatics*, 30(4):559–565, 2014.

[96] Peter D Hoff, Adrian E Raftery, and Mark S Handcock. Latent space approaches to social network analysis. *Journal of the american Statistical association*, 97(460):1090–1098, 2002.

[97] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[98] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465, 2021.

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

[100] Vincent Tao Hu, David W Zhang, Yuki M Asano, Gertjan J Burghouts, and Cees GM Snoek. Self-guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18413–18422, 2023.

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

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

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

[104] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure

Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

[105] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1857–1867. ACM, 2020.

[106] Hong Huang, Yuxiao Dong, Jie Tang, Hongxia Yang, Nitesh V Chawla, and Xiaoming Fu. Will triadic closure strengthen ties in social networks? *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(3):1–25, 2018.

[107] Hong Huang, Jie Tang, Lu Liu, JarDer Luo, and Xiaoming Fu. Triadic closure pattern analysis and prediction in social networks. *IEEE Transactions on Knowledge and Data Engineering*, 27(12):3374–3389, 2015.

[108] Qian Huang, Hongyu Ren, Peng Chen, Gregor Kržmanc, Daniel Zeng, Percy Liang, and Jure Leskovec. Prodigy: Enabling in-context learning over graphs. *arXiv preprint arXiv:2305.12600*, 2023.

[109] Xingyue Huang, Miguel Romero Orth, Ismail Ilkan Ceylan, and Pablo Barcelo. A theory of link prediction via relational weisfeiler-leman. *arXiv preprint arXiv:2302.02209*, 2023.

[110] Yinan Huang, William Lu, Joshua Robinson, Yu Yang, Muhan Zhang, Stefanie Jegelka, and Pan Li. On the stability of expressive positional encodings for graph neural networks. *arXiv preprint arXiv:2310.02579*, 2023.

[111] Borja Ibarz, Vitaly Kurin, George Papamakarios, Kyriacos Nikiforou, Mehdi Bennani, Róbert Csordás, Andrew Joseph Dudzik, Matko Bošnjak, Alex Vitvitskyi, Yulia Rubanova, et al. A generalist neural algorithmic learner. In *Learning on graphs conference*, pages 2–1. PMLR, 2022.

[112] Paul Jaccard. Distribution comparée de la flore alpine dans quelques régions des alpes occidentales et orientales. *Bulletin de la Murithienne*, (31):81–92, 1902.

[113] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543, 2002.

[114] Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in nlp. In *Association of Computational Linguistics*, pages 264–271. The Association for Computational Linguistics, 2007.

[115] Zhimeng Jiang, Xiaotian Han, Chao Fan, Zirui Liu, Xiao Huang, Na Zou, Ali Mostafavi, and Xia Hu. Topology matters in fair graph learning: a theoretical pilot study.

[116] Zhimeng Jiang, Xiaotian Han, Chao Fan, Zirui Liu, Na Zou, Ali Mostafavi, and Xia Hu. Fmp: Toward fair graph message passing against topology bias. *arXiv preprint arXiv:2202.04187*, 2022.

[117] Lichen Jin, Yizhou Zhang, Guojie Song, and Yilun Jin. Active domain transfer on network embedding. In *Proceedings of The Web Conference 2020*, pages 2683–2689. ACM / IW3C2, 2020.

[118] Wei Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zitao Liu, and Jiliang Tang. Self-supervised learning on graphs: Deep insights and new direction. *arXiv preprint arXiv:2006.10141*, 2020.

[119] Wei Jin, Haitao Mao, Zheng Li, Haoming Jiang, Chen Luo, Hongzhi Wen, Haoyu Han, Hanqing Lu, Zhengyang Wang, Ruirui Li, et al. Amazon-m2: A multilingual multi-locale shopping session dataset for recommendation and text generation. *Advances in Neural Information Processing Systems*, 36:8006–8026, 2023.

[120] Wei Jin, Tong Zhao, Jiayuan Ding, Yozen Liu, Jiliang Tang, and Neil Shah. Empowering graph representation learning with test-time graph transformation. *arXiv preprint arXiv:2210.03561*, 2022.

[121] Wei Jin, Tong Zhao, Jiayuan Ding, Yozen Liu, Jiliang Tang, and Neil Shah. Empowering graph representation learning with test-time graph transformation. In *The Eleventh International Conference on Learning Representations*, 2023.

[122] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *International conference on machine learning*, pages 4839–4848. PMLR, 2020.

[123] Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning*, pages 10362–10383. PMLR, 2022.

[124] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[125] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[126] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[127] Kazi Zainab Khanam, Gautam Srivastava, and Vijay Mago. The homophily principle in

social network analysis. *arXiv preprint arXiv:2008.10383*, 2020.

[128] Dongki Kim, Jinheon Baek, and Sung Ju Hwang. Graph self-supervised learning with accurate discrepancy learning. *Advances in Neural Information Processing Systems*, 35:14085–14098, 2022.

[129] Jinwoo Kim, Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *arXiv*, abs/2207.02505, 2022.

[130] Youngeun Kim, Donghyeon Cho, Priyadarshini Panda, and Sungeun Hong. Progressive domain adaptation from a source pre-trained model. *arXiv preprint arXiv:2007.01524*, 2020.

[131] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[132] Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2016.

[133] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

[134] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, abs/1609.02907, 2017.

[135] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[136] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[137] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[138] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2018.

[139] Kezhi Kong, Guohao Li, Mucong Ding, Zuxuan Wu, Chen Zhu, Bernard Ghanem, Gavin Taylor, and Tom Goldstein. Robust optimization as data augmentation for large-scale graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 60–69, 2022.

[140] Lecheng Kong, Yixin Chen, and Muhan Zhang. Geodesic graph neural network for efficient graph representation learning. *Advances in Neural Information Processing Systems*, 35:5896–5909, 2022.

[141] István A Kovács, Katja Luck, Kerstin Spirohn, Yang Wang, Carl Pollis, Sadie Schlabach, Wenting Bian, Dae-Kyum Kim, Nishka Kishore, Tong Hao, et al. Network-based prediction of protein interactions. *Nature communications*, 10(1):1240, 2019.

[142] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

[143] Nils M Kriege, Fredrik D Johansson, and Christopher Morris. A survey on graph kernels. *Applied Network Science*, 5(1):1–42, 2020.

[144] Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Bhaskar Biswas. Link prediction techniques, applications, and performance: A survey. *Physica A: Statistical Mechanics and its Applications*, 553:124289, 2020.

[145] Anusha Lalitha, Osman Cihan Kilinc, Tara Javidi, and Farinaz Koushanfar. Peer-to-peer federated learning on graphs. *arXiv preprint arXiv:1901.11173*, abs/1901.11173, 2019.

[146] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: an approach to modeling networks. *Journal of Machine Learning Research*, 11(2), 2010.

[147] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9267–9276, 2019.

[148] Juanhui Li, Harry Shomer, Haitao Mao, Shenglai Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. Evaluating graph neural networks for link prediction: Current pitfalls and new benchmarking. *arXiv preprint arXiv:2306.10453*, 2023.

[149] Juanhui Li, Harry Shomer, Haitao Mao, Shenglai Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. Evaluating graph neural networks for link prediction: Current pitfalls and new benchmarking. *arXiv preprint arXiv:2306.10453*, 2023.

[150] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[151] Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu. Model adaptation: Unsupervised domain adaptation without source data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9641–9650. Computer Vision Foundation / IEEE,

2020.

[152] Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Wei Qian. Finding global homophily in graph neural networks when meeting heterophily. In *International Conference on Machine Learning*, 2022.

[153] Yuwen Li, Miao Xiong, and Bryan Hooi. Graphcleaner: Detecting mislabelled samples in popular graph learning benchmarks. *arXiv preprint arXiv:2306.00015*, 2023.

[154] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 6028–6039. PMLR, PMLR, 2020.

[155] Shuming Liang, Yu Ding, Zhidong Li, Bin Liang, Yang Wang, Fang Chen, et al. Can gnns learn heuristic information for link prediction? 2022.

[156] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. *Advances in Neural Information Processing Systems*, 34:20887–20902, 2021.

[157] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. *Advances in Neural Information Processing Systems*, 34:20887–20902, 2021.

[158] Derek Lim, Joshua David Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. In *The Eleventh International Conference on Learning Representations*, 2022.

[159] Hongyi Ling, Zhimeng Jiang, Meng Liu, Shuiwang Ji, and Na Zou. Graph mixup with soft alignments. In *International Conference on Machine Learning*, pages 21335–21349. PMLR, 2023.

[160] Xiang Ling, Lingfei Wu, Saizhuo Wang, Gaoning Pan, Tengfei Ma, Fangli Xu, Alex X Liu, Chunming Wu, and Shouling Ji. Deep graph matching and searching for semantic code retrieval. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(5):1–21, 2021.

[161] Gang Liu, Eric Inae, Tong Zhao, Jiaxin Xu, Tengfei Luo, and Meng Jiang. Data-centric learning from unlabeled graphs with diffusion model. *arXiv preprint arXiv:2303.10108*, 2023.

[162] Gang Liu, Eric Inae, Tong Zhao, Jiaxin Xu, Tengfei Luo, and Meng Jiang. Data-centric learning from unlabeled graphs with diffusion model. *Advances in neural information processing systems*, 36, 2024.

[163] Gang Liu, Tong Zhao, Jiaxin Xu, Tengfei Luo, and Meng Jiang. Graph rationalization with environment-based augmentations. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1069–1078, 2022.

[164] Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. *arXiv preprint arXiv:2310.00149*, 2023.

[165] Haoyu Liu, Ningyi Liao, and Siqiang Luo. Simga: A simple and effective heterophilous graph neural network with efficient global aggregation. *arXiv preprint arXiv:2305.09958*, 2023.

[166] Hongrui Liu, Binbin Hu, Xiao Wang, Chuan Shi, Zhiqiang Zhang, and Jun Zhou. Confidence may cheat: Self-training on graph neural networks under distribution shift. In *Proceedings of the ACM Web Conference 2022*, pages 1248–1258, 2022.

[167] Jingzhe Liu, Haitao Mao, Zhikai Chen, Tong Zhao, Neil Shah, and Jiliang Tang. Neural scaling laws on graphs. *arXiv preprint arXiv:2402.02054*, 2024.

[168] Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 338–348, 2020.

[169] Ruitong Liu, Yanbin Wang, Haitao Xu, Bin Liu, Jianguo Sun, Zhenhao Guo, and Wenrui Ma. Source code vulnerability detection: Combining code language models and code property graphs. *arXiv preprint arXiv:2404.14719*, 2024.

[170] Songtao Liu, Rex Ying, Hanze Dong, Lanqing Li, Tingyang Xu, Yu Rong, Peilin Zhao, Junzhou Huang, and Dinghao Wu. Local augmentation for graph neural networks. In *International conference on machine learning*, pages 14054–14072. PMLR, 2022.

[171] Yang-Yu Liu, Jean-Jacques Slotine, and Albert-László Barabási. Controllability of complex networks. *nature*, 473(7346):167–173, 2011.

[172] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM Web Conference 2023*, pages 417–428, 2023.

[173] Zhiyuan Liu, Yaorui Shi, An Zhang, Enzhi Zhang, Kenji Kawaguchi, Xiang Wang, and Tat-Seng Chua. Rethinking tokenizer and decoder in masked graph modeling for molecules. *Advances in Neural Information Processing Systems*, 36, 2024.

[174] Shuqi Lu, Zhifeng Gao, Di He, Linfeng Zhang, and Guolin Ke. Highly accurate quantum chemical property prediction with uni-mol+. *arXiv preprint arXiv:2303.16982*, 2023.

[175] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Xiao-Wen Chang, and Doina Precup. When do we need gnn for node classification? *arXiv preprint arXiv:2210.16979*, 2022.

[176] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. In *Advances in Neural Information Processing Systems*.

[177] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Is heterophily a real nightmare for graph neural networks to do node classification? *arXiv preprint arXiv:2109.05641*, 2021.

[178] Sitao Luan, Chenqing Hua, Minkai Xu, Qincheng Lu, Jiaqi Zhu, Xiao-Wen Chang, Jie Fu, Jure Leskovec, and Doina Precup. When do graph neural networks help with node classification: Investigating the homophily principle on node distinguishability. *arXiv preprint arXiv:2304.14274*, 2023.

[179] Youzhi Luo, Michael McThrow, Wing Yee Au, Tao Komikado, Kanji Uchino, Koji Maruhashi, and Shuiwang Ji. Automated data augmentations for graph classification. *arXiv preprint arXiv:2202.13248*, 2022.

[180] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. Graphdf: A discrete flow model for molecular graph generation. In *International Conference on Machine Learning*, pages 7192–7203. PMLR, 2021.

[181] Jiaqi Ma, Junwei Deng, and Qiaozhu Mei. Subgroup generalization and fairness of graph neural networks. *Advances in Neural Information Processing Systems*, 34:1048–1061, 2021.

[182] Qian Ma, Haitao Mao, Jingzhe Liu, Zhehua Zhang, Chunlin Feng, Yu Song, Yihan Shao, and Yao Ma. Do neural scaling laws exist on graph self-supervised learning? *arXiv preprint arXiv:2408.11243*, 2024.

[183] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? *arXiv preprint arXiv:2106.06134*, 2021.

[184] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? *ArXiv*, abs/2106.06134, 2021.

[185] Yao Ma, Xiaorui Liu, Tong Zhao, Yozen Liu, Jiliang Tang, and Neil Shah. A unified view on graph neural networks as graph signal denoising. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1202–1211, 2021.

[186] Haitao Mao, Xu Chen, Qiang Fu, Lun Du, Shi Han, and Dongmei Zhang. Neuron campaign for initialization guided by information bottleneck theory. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3328–3332,

2021.

[187] Haitao Mao, Zhikai Chen, Wei Jin, Haoyu Han, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. Demystifying structural disparity in graph neural networks: Can one size fit all? *arXiv preprint arXiv:2306.01323*, 2023.

[188] Haitao Mao, Zhikai Chen, Wei Jin, Haoyu Han, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. Demystifying structural disparity in graph neural networks: Can one size fit all? *Advances in Neural Information Processing Systems*, 36, 2024.

[189] Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Michael Galkin, and Jiliang Tang. Graph foundation models. *arXiv preprint arXiv:2402.02216*, 2024.

[190] Haitao Mao, Lun Du, Yujia Zheng, Qiang Fu, Zelin Li, Xu Chen, Shi Han, and Dongmei Zhang. Source free unsupervised graph domain adaptation. *arXiv preprint arXiv:2112.00955*, 2021.

[191] Haitao Mao, Lun Du, Yujia Zheng, Qiang Fu, Zelin Li, Xu Chen, Shi Han, and Dongmei Zhang. Source free graph unsupervised domain adaptation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 520–528, 2024.

[192] Haitao Mao, Juanhui Li, Harry Shomer, Bingheng Li, Wenqi Fan, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. Revisiting link prediction: A data perspective. *arXiv preprint arXiv:2310.00793*, 2023.

[193] Haitao Mao, Juanhui Li, Harry Shomer, Bingheng Li, Wenqi Fan, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. Revisiting link prediction: a data perspective. In *The Twelfth International Conference on Learning Representations*, 2024.

[194] Haitao Mao, Guangliang Liu, Yao Ma, Rongrong Wang, and Jiliang Tang. A data generation perspective to the mechanism of in-context learning. *arXiv preprint arXiv:2402.02212*, 2024.

[195] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *IEEE International Conference on Computer Vision*, pages 2794–2802. IEEE Computer Society, 2017.

[196] Dominic Masters, Josef Dean, Kerstin Klaser, Zhiyi Li, Sam Maddrell-Mander, Adam Sanders, Hatem Helal, Deniz Beker, Ladislav Rampášek, and Dominique Beaini. Gps++: An optimised hybrid mpnn/transformer for molecular property prediction. *arXiv preprint arXiv:2212.02229*, 2022.

[197] Andreas Maurer. A note on the pac bayesian theorem. *arXiv preprint cs/0411099*, 2004.

[198] David McAllester. Simplified pac-bayesian margin bounds. In *Learning Theory and*

Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003. Proceedings, pages 203–215. Springer, 2003.

[199] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

[200] R Thomas McCoy, Shunyu Yao, Dan Friedman, Matthew Hardy, and Thomas L Griffiths. Embers of autoregression: Understanding large language models through the problem they are trained to solve. *arXiv preprint arXiv:2309.13638*, 2023.

[201] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

[202] Filippo Menczer, Santo Fortunato, and Clayton A. Davis. *A First Course in Network Science*. Cambridge University Press, 2020.

[203] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[204] Samarth Mishra, Rameswar Panda, Cheng Perng Phoo, Chun-Fu Richard Chen, Leonid Karlinsky, Kate Saenko, Venkatesh Saligrama, and Rogerio S Feris. Task2sim: Towards effective pre-training and transfer from synthetic data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9194–9204, 2022.

[205] Jose G Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern recognition*, 45(1):521–530, 2012.

[206] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

[207] Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M. Kriege, Martin Grohe, Matthias Fey, and Karsten Borgwardt. Weisfeiler and leman go machine learning: The story so far. *Journal of Machine Learning Research*, 24(333):1–59, 2023.

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

[209] Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampášek. Attending to

graph transformers. *arXiv preprint arXiv:2302.04181*, 2023.

[210] Chaithanya Kumar Mummadi, Robin Hutmacher, Kilian Rambach, Evgeny Levinkov, Thomas Brox, and Jan Hendrik Metzen. Test-time adaptation to distribution shift by confidence maximization and input transformation. *arXiv preprint arXiv:2106.14999*, 2021.

[211] Yohsuke Murase, Hang Hyun Jo, János Török, János Kertész, Kimmo Kaski, et al. Structural transition in social networks. 2019.

[212] Mark EJ Newman. Clustering and preferential attachment in growing networks. *Physical review E*, 64(2):025102, 2001.

[213] Trung-Kien Nguyen and Yuan Fang. Diffusion-based negative sampling on graphs for link prediction. *arXiv preprint arXiv:2403.17259*, 2024.

[214] Maximilian Nickel, Xueyan Jiang, and Volker Tresp. Reducing the rank in relational factorization models by including observable patterns. *Advances in Neural Information Processing Systems*, 27, 2014.

[215] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, pages 4474–4484. PMLR, 2020.

[216] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

[217] Tore Opsahl. Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. *Social networks*, 35(2):159–167, 2013.

[218] Madhur Panwar, Kabir Ahuja, and Navin Goyal. In-context learning through the bayesian prism. In *The Twelfth International Conference on Learning Representations*, 2023.

[219] Hyeonjin Park, Seunghun Lee, Sihyeon Kim, Jinyoung Park, Jisu Jeong, Kyung-Min Kim, Jung-Woo Ha, and Hyunwoo J Kim. Metropolis-hastings data augmentation for graph neural networks. *Advances in Neural Information Processing Systems*, 34:19010–19020, 2021.

[220] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*.

[221] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[222] Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou,

and Jonathan Halcrow. Let your graph do the talking: Encoding structured data for llms. *arXiv preprint arXiv:2402.05862*, 2024.

[223] Oleg Platonov, Denis Kuznedelev, Artem Babenko, and Liudmila Prokhorenkova. Characterizing graph datasets for node classification: Beyond homophily-heterophily dichotomy. *arXiv preprint arXiv:2209.06177*, 2022.

[224] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations*, 2023.

[225] Ben Prystawski and Noah D Goodman. Why think step-by-step? reasoning emerges from the locality of experience. *arXiv preprint arXiv:2304.03843*, 2023.

[226] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1150–1160, 2020.

[227] Joaquin Quinonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. Mit Press, 2008.

[228] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[229] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[230] Md Nakhla Rafi, Dong Jae Kim, An Ran Chen, Tse-Hsun Chen, and Shaowei Wang. Towards better graph neural neural network-based fault localization through enhanced code representation. *arXiv preprint arXiv:2404.04496*, 2024.

[231] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *Association for Computing Machinery Special Interest Group on Knowledge Discovery and Data Mining*, pages 385–394. ACM, 2017.

[232] Pedro Ribeiro, Pedro Paredes, Miguel EP Silva, David Aparicio, and Fernando Silva. A survey on subgraph counting: concepts, algorithms, and applications to network motifs and graphlets. *ACM Computing Surveys (CSUR)*, 54(2):1–36, 2021.

[233] Garry Robins, Pip Pattison, Yuval Kalish, and Dean Lusher. An introduction to exponential random graph (p*) models for social networks. *Social networks*, 29(2):173–191, 2007.

[234] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[235] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. *Advances in neural information processing systems*, 33:12559–12571, 2020.

[236] Ryan Rossi and Nesreen Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[237] Ryan A Rossi, Anup Rao, Sungchul Kim, Eunyee Koh, and Nesreen Ahmed. From closing triangles to closing higher-order motifs. In *Companion Proceedings of the Web Conference 2020*, pages 42–43, 2020.

[238] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[239] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

[240] Benedek Rozemberczki, Oliver Kiss, and Rik Sarkar. Karate club: an api oriented open-source python framework for unsupervised learning on graphs. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 3125–3132, 2020.

[241] Luana Ruiz, Luiz F. O. Chamon, and Alejandro Ribeiro. Transferability properties of graph neural networks. *IEEE Transactions on Signal Processing*, 71:3474–3489, 2023.

[242] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

[243] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 2988–2997. PMLR, PMLR, 2017.

[244] Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*, 2022.

[245] Purnamrita Sarkar, Deepayan Chakrabarti, and Andrew W Moore. Theoretical justification of popular link prediction heuristics. In *IJCAI proceedings-international joint conference on artificial intelligence*, volume 22, page 2722, 2011.

[246] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[247] Xiao Shen, Quanyu Dai, Fu-lai Chung, Wei Lu, and Kup-Sze Choi. Adversarial deep network embedding for cross-network node classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2991–2999, 2020.

[248] Xiao Shen, Quanyu Dai, Sitong Mao, Fu-lai Chung, and Kup-Sze Choi. Network together: Node classification via cross-network deep network embedding. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5):1935–1948, 2020.

[249] Chence Shi, Shitong Luo, Minkai Xu, and Jian Tang. Learning gradient fields for molecular conformation generation. In *International conference on machine learning*, pages 9558–9568. PMLR, 2021.

[250] Hongzhi Shi, Jingtao Ding, Yufan Cao, Li Liu, Yong Li, et al. Learning symbolic models for graph-structured physical mechanism. In *The Eleventh International Conference on Learning Representations*, 2022.

[251] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

[252] Nima Shoghi, Adeesh Kolluru, John R. Kitchin, Zachary W. Ulissi, C. Lawrence Zitnick, and Brandon M. Wood. From molecules to materials: Pre-training large generalizable models for atomic property prediction, 2023.

[253] Harry Shomer, Yao Ma, Haitao Mao, Juanhui Li, Bo Wu, and Jiliang Tang. Lpformer: an adaptive graph transformer for link prediction. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2686–2698, 2024.

[254] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[255] Yu Song, Haitao Mao, Jiachen Xiao, Jingzhe Liu, Zhikai Chen, Wei Jin, Carl Yang, Jiliang Tang, and Hui Liu. A pure transformer pretraining framework on text-attributed graphs. *arXiv preprint arXiv:2406.13873*, 2024.

[256] Yu Song and Donglin Wang. Learning on graphs with out-of-distribution nodes. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1635–1645, 2022.

[257] Balasubramaniam Srinivasan and Bruno Ribeiro. On the equivalence between positional node embeddings and structural graph representations. *arXiv preprint arXiv:1910.00452*,

2019.

[258] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. Gppt: Graph pre-training and prompt tuning to generalize graph neural networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 1717–1727, New York, NY, USA, 2022. Association for Computing Machinery.

[259] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. All in one: Multi-task prompting for graph neural networks. 2023.

[260] Shiyin Tan, Dongyuan Li, Renhe Jiang, Ying Zhang, and Manabu Okumura. Community-invariant graph contrastive learning. *arXiv preprint arXiv:2405.01350*, 2024.

[261] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models. *arXiv preprint arXiv:2310.13023*, 2023.

[262] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Association for Computing Machinery Special Interest Group on Knowledge Discovery and Data Mining*, pages 990–998. ACM, 2008.

[263] Wenzhuo Tang, Haitao Mao, Danial Dervovic, Ivan Brugere, Saumitra Mishra, Yuying Xie, and Jiliang Tang. Cross-domain graph data scaling: A showcase with diffusion models. *arXiv preprint arXiv:2406.01899*, 2024.

[264] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[265] Brandon Trabucco, Kyle Doherty, Max A Gurinas, and Ruslan Salakhutdinov. Effective data augmentation with diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024.

[266] Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

[267] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *Computer Science*, abs/1412.3474, 2014.

[268] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(2605):2579–2605, 2008.

[269] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[270] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *stat*, 1050:20, 2017.

[271] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *ICLR (Poster)*, 2(3):4, 2019.

[272] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*, 2023.

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

[274] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10:3152676, 2017.

[275] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

[276] Haonan Wang, Jieyu Zhang, Qi Zhu, and Wei Huang. Augmentation-free graph contrastive learning. *arXiv preprint arXiv:2204.04874*, 2022.

[277] Haonan Wang, Jieyu Zhang, Qi Zhu, and Wei Huang. Can single-pass contrastive learning work for both homophilic and heterophilic graph? *arXiv preprint arXiv:2211.10890*, 2022.

[278] Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. In *International Conference on Learning Representations*, 2021.

[279] Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. *arXiv preprint arXiv:2203.00199*, 2022.

[280] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. Can language models solve graph problems in natural language? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[281] Hui Wang and Zichun Le. Seven-layer model in complex networks link prediction: A survey. *Sensors*, 20(22):6560, 2020.

[282] Junfu Wang, Yuanfang Guo, Liang Yang, and Yunhong Wang. Understanding heterophily for graph neural networks. *arXiv preprint arXiv:2401.09125*, 2024.

[283] Xiyuan Wang, Haotong Yang, and Muhan Zhang. Neural common neighbor with completion for link prediction. *arXiv preprint arXiv:2302.00890*, 2023.

[284] Xiyuan Wang, Haotong Yang, and Muhan Zhang. Neural common neighbor with completion for link prediction. In *The Twelfth International Conference on Learning Representations*, 2024.

[285] Rongzhe Wei, Haoteng Yin, Junteng Jia, Austin R Benson, and Pan Li. Understanding non-linearity in graph neural networks from the bayesian-inference perspective. In *Advances in Neural Information Processing Systems*.

[286] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23:69–101, 1996.

[287] Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, 37:1–12, 2020.

[288] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[289] Man Wu, Shirui Pan, Chuan Zhou, Xiaojun Chang, and Xingquan Zhu. Unsupervised domain adaptive graph convolutional networks. In *The Web Conference 2020*, pages 1457–1467. ACM / IW3C2, 2020.

[290] Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. Handling distribution shifts on graphs: An invariance perspective. *arXiv preprint arXiv:2202.02466*, 2022.

[291] Wei Wu, Bin Li, Chuan Luo, and Wolfgang Nejdl. Hashing-accelerated graph neural networks for link prediction. In *Proceedings of the Web Conference 2021*, pages 2910–2920, 2021.

[292] Xinyi Wu, Zhengdao Chen, William Wang, and Ali Jadbabaie. A non-asymptotic analysis of oversmoothing in graph neural networks. *arXiv preprint arXiv:2212.10701*, 2022.

[293] Jun Xia, Chengshuai Zhao, Bozhen Hu, Zhangyang Gao, Cheng Tan, Yue Liu, Siyuan Li, and Stan Z. Li. Mole-BERT: Rethinking pre-training graph neural networks for molecules. In *The Eleventh International Conference on Learning Representations*, 2023.

[294] Han Xie, Jing Ma, Li Xiong, and Carl Yang. Federated graph classification over non-iid graphs. *Advances in Neural Information Processing Systems*, 34, 2021.

[295] Jiarong Xu, Renhong Huang, Xin Jiang, Yuxuan Cao, Carl Yang, Chunping Wang, and Yang Yang. Better with less: A data-active perspective on pre-training graph neural networks. *Advances in Neural Information Processing Systems*, 36:56946–56978, 2023.

[296] Keylu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? *ICLR 2020*, 2020.

[297] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*.

[298] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[299] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[300] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pages 5453–5462. PMLR, 2018.

[301] Minghao Xu, Hang Wang, Bingbing Ni, Hongyu Guo, and Jian Tang. Self-supervised graph-level representation learning with local and global structure. In *International Conference on Machine Learning*, pages 11548–11558. PMLR, 2021.

[302] Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. In *International Conference on Learning Representations*, 2022.

[303] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 1287–1292. IEEE, 2022.

[304] Ruichao Yang, Xiting Wang, Yiqiao Jin, Chaozhuo Li, Jianxun Lian, and Xing Xie. Reinforcement subgraph reasoning for fake news detection. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2253–2262, 2022.

[305] Shiqi Yang, Joost van de Weijer, Luis Herranz, Shangling Jui, et al. Exploiting the intrinsic neighborhood structure for source-free domain adaptation. *Advances in Neural Information Processing Systems*, 34:29393–29405, 2021.

[306] Shiqi Yang, Yaxing Wang, Joost van de Weijer, Luis Herranz, and Shangling Jui. Unsupervised domain adaptation without source data by casting a bait. *Computer Vision and Pattern Recognition*, abs/2010.12427, 2020.

[307] Shuwen Yang, Guojie Song, Yilun Jin, and Lun Du. Domain adaptive classification on heterogeneous information networks. In *International Joint Conference on Artificial Intelligence*, pages 1410–1416. ijcai.org, 2020.

[308] Hongbin Ye, Ningyu Zhang, Hui Chen, and Huajun Chen. Generative knowledge graph construction: A review. *arXiv preprint arXiv:2210.12714*, 2022.

[309] Haoteng Yin, Muhan Zhang, Jianguo Wang, and Pan Li. Surel+: Moving from walks to sets for scalable subgraph-based graph representation learning. *arXiv preprint arXiv:2303.03379*, 2023.

[310] Haoteng Yin, Muhan Zhang, Yanbang Wang, Jianguo Wang, and Pan Li. Algorithm and system co-design for efficient subgraph-based graph representation learning. *arXiv preprint arXiv:2202.13538*, 2022.

[311] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

[312] Jiaxuan You, Jonathan M Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10737–10745, 2021.

[313] Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In *International conference on machine learning*, pages 7134–7143. PMLR, 2019.

[314] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. In *International Conference on Machine Learning*, pages 12121–12132. PMLR, 2021.

[315] Zebin You, Yong Zhong, Fan Bao, Jiacheng Sun, Chongxuan Li, and Jun Zhu. Diffusion models and semi-supervised learners benefit mutually with few labels. *Advances in Neural Information Processing Systems*, 36, 2024.

[316] Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G Hauptmann, et al. Language model beats diffusion–tokenizer is key to visual generation. *arXiv preprint arXiv:2310.05737*, 2023.

[317] Seongjun Yun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J Kim. Neo-gnns: Neighborhood overlap-aware graph neural networks for link prediction. *Advances in Neural Information Processing Systems*, 34:13683–13694, 2021.

[318] You Yuning, Chen Tianlong, Wang Zhangyang, and Shen Yang. Graph domain adaptation via theory-grounded spectral regularization. In *The Eleventh International Conference on Learning Representations*, 2023.

[319] Guangyao Zhai, Evin Pinar Örnek, Shun-Cheng Wu, Yan Di, Federico Tombari, Nassir Navab, and Benjamin Busam. Commonscenes: Generating commonsense 3d indoor scenes with scene graphs. *arXiv preprint arXiv:2305.16283*, 2023.

[320] Guangyao Zhai, Evin Pınar Örnek, Shun-Cheng Wu, Yan Di, Federico Tombari, Nassir

Navab, and Benjamin Busam. Commonscenes: Generating commonsense 3d indoor scenes with scene graphs. *Advances in Neural Information Processing Systems*, 36, 2024.

[321] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12104–12113, 2022.

[322] Bohang Zhang, Jingchu Gai, Yiheng Du, Qiwei Ye, Di He, and Liwei Wang. Beyond weisfeiler-lehman: A quantitative framework for GNN expressiveness. In *The Twelfth International Conference on Learning Representations*, 2024.

[323] Duo Zhang, Xinzijian Liu, Xiangyu Zhang, Chengqian Zhang, Chun Cai, Hangrui Bi, Yiming Du, Xuejian Qin, Jiameng Huang, Bowen Li, Yifan Shan, Jinzhe Zeng, Yuzhi Zhang, Siyuan Liu, Yifan Li, Junhan Chang, Xinyan Wang, Shuo Zhou, Jianchuan Liu, Xiaoshan Luo, Zhenyu Wang, Wanrun Jiang, Jing Wu, Yudi Yang, Jiyuan Yang, Manyi Yang, Fu-Qiang Gong, Linshuang Zhang, Mengchao Shi, Fu-Zhi Dai, Darrin M. York, Shi Liu, Tong Zhu, Zhicheng Zhong, Jian Lv, Jun Cheng, Weile Jia, Mohan Chen, Guolin Ke, Weinan E, Linfeng Zhang, and Han Wang. Dpa-2: Towards a universal large atomic model for molecular and material simulation. *arXiv preprint arXiv:2312.15492*, 2023.

[324] Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and Philip S Yu. From canonical correlation analysis to self-supervised graph neural networks. *Advances in Neural Information Processing Systems*, 34:76–89, 2021.

[325] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[326] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning. *Advances in Neural Information Processing Systems*, 34:9061–9073, 2021.

[327] Shengyu Zhang, Kun Kuang, Jiezhong Qiu, Jin Yu, Zhou Zhao, Hongxia Yang, Zhongfei Zhang, and Fei Wu. Stable prediction on graphs with agnostic distribution shift. *arXiv preprint arXiv:2110.03865*, 2021.

[328] Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. Graph-less neural networks: Teaching old mlps new tricks via distillation. *ArXiv*, abs/2110.08727, 2021.

[329] Yanci Zhang, Yutong Lu, Haitao Mao, Jiawei Huang, Cien Zhang, Xinyi Li, and Rui Dai. Company competition graph. *arXiv preprint arXiv:2304.00323*, 2023.

[330] Yizhou Zhang, Guojie Song, Lun Du, Shuwen Yang, and Yilun Jin. Dane: Domain adaptive network embedding. *International Joint Conference on Artificial Intelligence*, abs/1906.00684:4362–4368, 2019.

[331] Yufeng Zhang, Fengzhuo Zhang, Zhuoran Yang, and Zhaoran Wang. What and how does in-context learning learn? bayesian model averaging, parameterization, and generalization. *arXiv preprint arXiv:2305.19420*, 2023.

[332] He Zhao, Lan Du, and Wray Buntine. Leveraging node attributes for incomplete relational data. In *International conference on machine learning*, pages 4072–4081. PMLR, 2017.

[333] Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. Graphtext: Graph reasoning in text space. *arXiv preprint arXiv:2310.01089*, 2023.

[334] Qifang Zhao, Weidong Ren, Tianyu Li, Xiaoxiao Xu, and Hong Liu. Graphgpt: Graph learning with generative pre-trained transformers. *arXiv preprint arXiv:2401.00529*, 2023.

[335] Tong Zhao, Gang Liu, Daheng Wang, Wenhao Yu, and Meng Jiang. Learning from counterfactual links for link prediction. In *International Conference on Machine Learning*, pages 26911–26926. PMLR, 2022.

[336] Shuxin Zheng, Jiyan He, Chang Liu, Yu Shi, Ziheng Lu, Weitao Feng, Fusong Ju, Jiaxi Wang, Jianwei Zhu, Yaosen Min, He Zhang, Shidi Tang, Hongxia Hao, Peiran Jin, Chi Chen, Frank Noé, Haiguang Liu, and Tie-Yan Liu. Towards predicting equilibrium distributions for molecular systems with deep learning. *arXiv preprint arXiv:2306.05445*, 2023.

[337] Wenqing Zheng, Edward W Huang, Nikhil Rao, Zhangyang Wang, and Karthik Subbian. You only transfer what you share: Intersection-induced graph transfer learning for link prediction. *arXiv preprint arXiv:2302.14189*, 2023.

[338] Yiwu Zhong, Jing Shi, Jianwei Yang, Chenliang Xu, and Yin Li. Learning to generate scene graph from natural language supervision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1823–1834, 2021.

[339] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71:623–630, 2009.

[340] Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. Graph neural networks with heterophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11168–11176, 2021.

[341] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.

[342] Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. Interpreting and unifying graph neural networks with an optimization framework. In *Proceedings of the Web Conference 2021*, pages 1215–1226, 2021.

[343] Qi Zhu, Natalia Ponomareva, Jiawei Han, and Bryan Perozzi. Shift-robust gnns: Overcoming the limitations of localized graph training data. *Advances in Neural Information Processing Systems*, 34:27965–27977, 2021.

[344] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020.

[345] Zhaocheng Zhu, Xinyu Yuan, Louis-Pascal Xhonneux, Ming Zhang, Maxime Gazeau, and Jian Tang. Learning to efficiently propagate for reasoning on knowledge graphs. *arXiv preprint arXiv:2206.04798*, 2022.

[346] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems*, 34:29476–29490, 2021.

[347] Lixin Zou, Haitao Mao, Xiaokai Chu, Jiliang Tang, Wenwen Ye, Shuaiqiang Wang, and Dawei Yin. A large scale search dataset for unbiased learning to rank. *Advances in Neural Information Processing Systems*, 35:1127–1139, 2022.