Graph Foundation Models: Concepts, Opportunities and Challenges

Jiawei Liu*, Cheng Yang*, Zhiyuan Lu, Junze Chen, Yibo Li,
Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S. Yu, and Chuan Shi

Abstract—Foundation models have emerged as critical components in a variety of artificial intelligence applications, and showcase significant success in natural language processing and several other domains. Meanwhile, the field of graph machine learning is witnessing a paradigm transition from shallow methods to more sophisticated deep learning approaches. The capabilities of foundation models in generalization and adaptation motivate graph machine learning researchers to discuss the potential of developing a new graph learning paradigm. This paradigm envisions models that are pre-trained on extensive graph data and can be adapted for various graph tasks. Despite this burgeoning interest, there is a noticeable lack of clear definitions and systematic analyses pertaining to this new domain. To this end, this article introduces the concept of Graph Foundation Models (GFMs), and offers an exhaustive explanation of their key characteristics and underlying technologies. We proceed to classify the existing work related to GFMs into three distinct categories, based on their dependence on graph neural networks and large language models. In addition to providing a thorough review of the current state of GFMs, this article also outlooks potential avenues for future research in this rapidly evolving domain.

Index Terms—Graph Foundation Models, Large Language Models

1 Introduction

With the rise in computational power and breakthroughs in deep learning techniques, the artificial intelligence (AI) community has introduced the notion of "foundation models": *A foundation model is any model that is trained on broad data and can be adapted to a wide range of downstream tasks [1].* Foundation models enjoy unique attributes like emergence and homogenization, empowering them to serve as the primary building blocks for a myriad of downstream AI applications [1]. Emergence suggests that as a foundation model scales up, it may spontaneously manifest novel capabilities [2]. Meanwhile, homogenization alludes to the model's versatility, enabling its deployment across diverse applications [1]. Thanks to the development of large language models (LLMs), the concept of foundation models first became a reality in natural language processing (NLP). Since then, foundation models have demonstrated impressive versatility, processing not just text but also image data, video data, audio data and multi-modal inputs. This versatility empowers them to excel in tasks ranging from computer vision [3] and audio signal processing [4] to recommender systems [5].

Much like the evolution witnessed in NLP, graph machine learning is also undergoing a paradigm transition. In its early stages, graph tasks predominantly employed shallow methods, such as random walk [6, 7] and matrix factorization [8, 9, 10, 11, 12]. These methods, however, were typically limited to transductive learning [13]. The more recent shift towards deep learning methods has catalyzed the rise of graph neural networks (GNNs). GNNs have revolutionized the landscape by introducing the message-passing mechanism, where nodes iteratively aggregate information from their neighbors. By harnessing GNNs in fully supervised, semi-supervised, or unsupervised settings, researchers have pioneered a variety of customized graph models. These advancements have yielded substantial improvements in tasks like node classification [14], link prediction [15], graph classification [16], and graph clustering [17]. However, certain challenges of GNN models still persist. For example, GNNs are restricted with issues related to expressive power [18] and generalizability [19], especially given the ever-expanding datasets and the widening spectrum of tasks.

The remarkable success of foundation models in varied domains is increasingly garnering the interest of graph machine learning researchers. This naturally evokes the question: Could graph foundation models represent the next frontier in graph machine learning? Such models, if realized, would boast enhanced expressive power, improved transferability, and applicability to more intricate graph data and tasks. As illustrated in Figure 1, a graph foundation model (GFM) is envisioned as a model pre-trained on extensive graph data, primed for adaptation across diverse downstream graph tasks. Drawing parallels with traditional foundation models, a GFM is also anticipated to embody two principal characteristics: emergence and homogenization. Specifically, emergence refers to novel capabilities shown exclusively in large-scale graph models, while homogenization denotes the model's adaptability across different types of graph tasks. Existing deep graph learning methods struggle to encompass these features: their inherent architectures and learning paradigms focus on specific tasks, which restrict the utilization of extensive unlabeled data,

- *Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Ting Bai and Chuan Shi are with School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China. E-mail: {liu_jiawei, yangcheng, luzy, junze, yiboL, baiting, shichuan}@bupt.edu.cn*
- *Mengmei Zhang is with China Telecom Bestpay, Beijing, China. E-mail: zhangmengmei@bestpay.com.cn*
- *Yuan Fang is with School of Computing and Information Systems, Singapore Management University, Singapore. E-mail: yfang@smu.edu.sg*
- *Lichao Sun is with Lehigh University, Bethlehem, Pennsylvania, USA. E-mail: lis221@lehigh.edu*
- *Philip S. Yu is with University of Illinois Chicago, Chicago, USA. E-mail: psyu@uic.edu*
- *Jiawei Liu and Cheng Yang contributed equally to this research.*
- *Corresponding author: Chuan Shi*

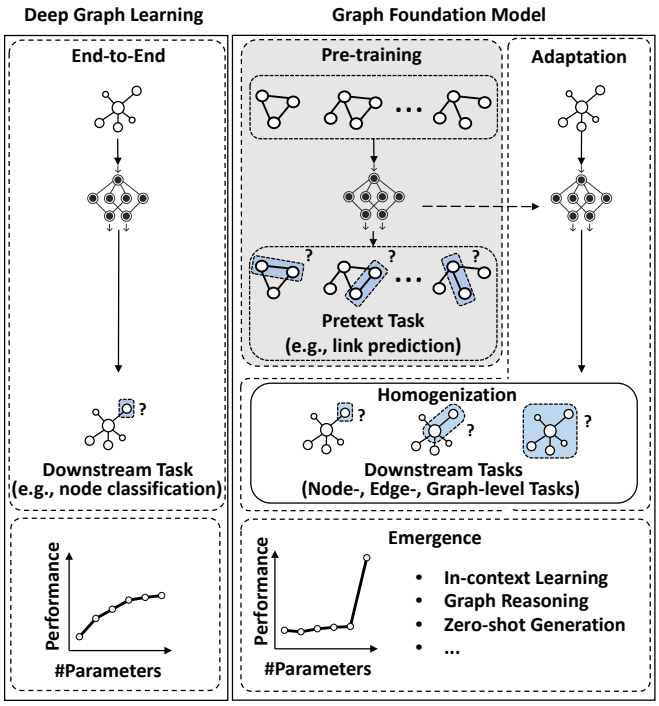


Fig. 1: The distinction between deep graph learning and graph foundation models. Deep graph learning tackles specific tasks on specific datasets through end-to-end training. In contrast, graph foundation models (GFMs) are pre-trained on broad graph data and can be adapted to a wide range of downstream graph tasks, expected to demonstrate emergence and homogenization capabilities.

subsequently limiting their expressive and generalization abilities.

Inspired by the success of LLMs as foundation models in NLP, researchers have explored the possibilities of graph foundation models towards the emergence and homogenization capabilities. These explorations primarily focus on the design of backbone architectures for GFMs, and different learning paradigms including pre-training and adaptation, as they are the key strategies of LLMs to acheive the aforementioned capabilities. First and foremost, the emergent abilities of foundation models typically exist only in backbones with a large number of parameters, whereas the parameter count of GNNs is significantly smaller than that of the language backbones. This implies that the backbone of GFMs may need to be redesigned to achieve more substantial knowledge storage towards emergence. As graph data is typically associated with rich text information, an alternative approach is to use LLMs as GFMs. Nonetheless, it remains uncertain whether LLMs can effectively handle graph data and associated tasks, and it is crucial to determine how to model graph structures in LLMs. Additionally, the homogenization of foundation models necessitates the handling of diverse tasks in a uniform manner. Devising effective pre-training tasks (also called pretext tasks) and downstream task adaptation methods are challenging for graph data, due to the complexity in interconnected nodes and various forms of attributes, as well as the diversity in tasks across node-, edge- and graph-levels. Therefore, there is also a need to design suitable pre-training tasks and adaptation mechanisms.

While there is no definitive solution for designing and implementing GFMs, this paper surveys some related researches and categorizes them into three distinct approaches based on their reliance on GNNs and LLMs: (1) **GNN-based Models**: They aim to enhance existing graph learning paradigms through innovations in the backbone, pre-training, and adaptation aspects; (2) **LLM-based Models**: They explore the feasibility of using an LLM as a GFM by converting graphs into text or tokens; (3) **GNN+LLM-based Models**: They explore various forms of synergy between GNNs and LLMs to empower them with enhanced capabilities.

To the best of our knowledge, this is the first survey towards graph foundation models. Existing surveys of foundation models typically explore modalities such as language and vision [1, 20], rather than graphs. Additionally, there are two surveys [21, 22] dedicated to knowledge graphs and large language models, but knowledge graphs, due to their distinct nature in construction and application, fall outside the scope of this article. We have also noticed a very recent article that mentions the concept of large graph models [23], but it emphasizes opinion statements and lacks a systematic taxonomy. Therefore, the contributions of this article can be summarized as follows:

- This article defines the concept of graph foundation models for the first time, and examines the core issues and characteristics of their capabilities.
- This article introduces a novel taxonomy and discusses the strengths and limitations of each approach towards graph foundation models.
- This article provides promising future directions towards graph foundation models.

The subsequent sections are organized as follows. In Section 2, we introduce the background related to GFMs. Section 3 defines GFMs and highlights their similarities and differences with language foundation models. Sections 4 - 6 delve into the relevant works that consider GNN-based models, LLM-based models and GNN+LLM-based models as GFMs, separately. Section 7 engages in a discussion on the future directions of GFMs. In Section 8, we summarize the key points of this paper.

2 Background

Before introducing GFMs, we review background knowledge on deep graph learning and language foundation models.

2.1 Deep Graph Learning

This section provides a concise overview from three key aspects: graph data, backbone architectures, and learning paradigms.

2.1.1 Graph Data

Graphs capture intricate relationships among entities and possess several key characteristics that make them challenging for machine learning tasks. The primary challenge stems from their **(1) Non-Euclidean Nature**: Graph data is inherently non-Euclidean [24], lacking the rigid geometric structure of traditional data formats such as 1D text, 2D images or tabular data. This means that graph data cannot be adequately described within a simple flat space because its intrinsic structure does not conform to the principles of Euclidean geometry [25]. Unlike Euclidean data, which often comes in predefined shapes (e.g., images of a specific resolution), non-Euclidean data can vary greatly in size and shape. This variability complicates the design of algorithms that often navigate complex topologies, significantly increasing computational cost compared to operations on simpler Euclidean spaces.

Beyond this fundamental structural complexity, two additional challenges are posed by the nature of graph data. **(2) Various Domains**: Graph data appears in domains such as social

networks [26], biology [27], and transportation [28]. It is also used in tasks like 3D human skeleton recognition [29], semantic segmentation [30], and video classification [31]. Domain-specific variability with different node types and edge semantics makes creating a universal model challenging [32]. **(3) Various Types**: graph data includes homogeneous, heterogeneous [33], hyper- [34], and dynamic ones [35]. Such diversity also brings challenges to deep graph learning.

2.1.2 Backbone Architectures

GNNs are the current mainstream backbone architecture for deep graph learning. Most GNNs follow the message-passing framework [36], enabling nodes to exchange information with neighbors. For example, GCN [14] introduces graph convolution layers, GraphSAGE [37] generates node embeddings using inductive learning, and GAT [38] adds an attention mechanism to weigh neighbor importance, enhancing expressive power. These contributions make GNNs versatile tools for deep graph learning.

However, deepening GNNs is challenging. Increasing layers leads to over-smoothing, where node representations become similar [39], and over-squashing, where information is overly compressed [18]. Efforts like DropEdge [40], which randomly removes edges, improve GCN performance and scalability. Graph transformers [41, 42, 43], with their fully connected attention and long-range relationship modeling, help alleviate over-smoothing and over-squashing [44].

2.1.3 Learning Paradigms

The learning paradigms for deep graph learning encompass three primary categories:

Supervised learning. In supervised learning, algorithms use a training dataset with input data and output labels. This is used in tasks like graph classification [45] and graph regression [46]. For instance, in molecular property prediction [47], GNNs predict chemical properties using labeled data, aiding drug development and materials research.

Semi-supervised learning. Semi-supervised learning, as discussed in [48], is a primary focus in deep graph learning. It uses both labeled and unlabeled data to improve model performance, with node classification [14] being a key application. The message-passing mechanism [36] allows GNNs to exchange information among nodes, incorporating both data types for predictions. GNNs can also combine with methods like label propagation for better performance [49].

Unsupervised learning. Unsupervised learning discovers patterns and structures without manual labels. Graph clustering identifies structures based on node relationships, while link prediction estimates missing connections. A subclass, self-supervised learning, generates labels from the data itself [50], allowing GNNs to be trained end-to-end for tasks like graph clustering [17] and link prediction [15].

2.2 Language Foundation Models

AI is currently undergoing a transformative shift marked by the emergence of some specific language models (such as GPT-3) that are trained on extensive and diverse datasets using self-supervised learning. These models, known as foundation models, are able to produce a wide array of outputs, enabling them to tackle a broad spectrum of downstream tasks. In contrast to the deep graph learning pipeline, the foundation model's approach embraces a pre-training and adaptation framework, enabling the model to achieve several significant advancements, including the emergence [2] and homogenization [1]. Foundation models have primarily established themselves in the field of NLP [1], so our discussion will focus on language foundation models in this section.

2.2.1 Language Data

Language data refers to text or spoken content in a human language, encompassing the grammar rules of the natural language and the associated semantics of the words. The quality and quantity of language data play a crucial role in the performance of NLP systems, impacting their accuracy, robustness, and overall effectiveness in various language tasks. In contrast to graph data, language data as Euclidean data is easier to model, and its rich semantic information significantly enhances the knowledge transferability of language models.

2.2.2 Backbone Architectures

An early breakthrough in foundation models is pre-trained language models (PLMs), designed to capture context-aware word representations, which proved remarkably effective as versatile semantic features. Furthermore, researchers have observed that increasing the scale of PLMs, whether by augmenting model size or training data, frequently results in increased model capacity for downstream tasks. These larger PLMs, collectively referred to as LLMs, exhibit emergent abilities [2] compared to their smaller counterparts (e.g., the 1.5B-parameter GPT-2 and 175B-parameter GPT-3). LLMs primarily utilize the Transformer architecture, because highly parallelizable Transformer-based architectures accelerate the pre-training stage and enable the utilization of massive datasets. In Transformer models, tokens serve as the input and represent units at the word level in natural language texts.

2.2.3 Learning Paradigms

As the number of model parameters has rapidly increased, the demand for significantly larger datasets has grown to effectively train these parameters and avoid overfitting. Given the extremely expensive costs associated with building large-scale labeled datasets, the importance of utilizing extensive unlabeled text data has been underscored. Leveraging these unlabeled datasets involves a two-step approach: first, achieving universal representations through self-supervised learning, and subsequently employing these representations for various tasks [51]. Based on different adaptation approaches, learning paradigms can be categorized into two types: pre-train and fine-tune and pre-train, prompt, and predict [52].

Pre-train and Fine-tune. In this paradigm, a model is initially pre-trained as a language model (LM), where it predicts the probability of observed textual data. Following the pre-training phase, we need to tune the model for specific tasks, which is known as fine-tuning. Building upon the success of models like ULMFit [53] and BERT [54], fine-tuning has emerged as the predominant method for adapting pre-trained models. In this framework, the primary emphasis lies in objective engineering, encompassing the design of training objectives for both pre-training and fine-tuning phases. The advantage of fine-tuning is that it can transfer knowledge between source and target tasks (or domains) and benefit the model's performance. For the small size of fine-tuning dataset compared to pre-training dataset, this process can enable adaptation effectively without losing much pre-trained knowledge.

Pre-train, Prompt and Predict. In this paradigm, rather than adjusting PLMs to suit specific downstream tasks, the approach

involves reshaping the downstream tasks to align more closely with those tackled during the original LM training, accomplished by providing textual prompts. From the aspect of prompt engineering, the approaches to create a proper prompts can be classified to manual methods and automated methods. Manual methods involve creating intuitive templates based on human insight, which is the most straightforward approach to crafting prompts. Manual methods face challenges in terms of high cost and precision. To address these issues, some approaches have started to experiment with automated prompt generation. When LLMs have billions of parameters, it is more effective if we can just adapt the downstream tasks without adjusting model parameters. This makes prompt-tuning a promising approach for adapting LLMs.

3 Graph Foundation Models

In this section, we will first formally define the concepts of graph foundation models, including the definition, key characteristics and key technologies. Then, we will discuss the similarities and differences between graph and language foundation models.

3.1 Definition and Key Characteristics

We define a graph foundation model as follows:

Definition A graph foundation model (GFM) is a model that is expected to benefit from the pre-training of broad graph data, and can be adapted to a wide range of downstream graph tasks.

Compared to deep graph learning that employs end-to-end training, GFMs use pre-training to obtain the knowledge from a substantial amount of unlabeled graph data, and then use adaptation techniques to tailor to various downstream tasks. Some studies [55, 56] have already demonstrated that the paradigm of pre-training and adaptation outperform deep graph learning in certain scenarios, e.g., few-shot learning [55], showcasing their superior expressive power and generalization ability. Unlike deep graph learning that aims to achieve better performance on a single task, a GFM is expected to have two key characteristics: emergence and homogenization.

Emergence. Emergence means that the graph foundation model will exhibit some new abilities when having a large parameters or trained on more data, which are also referred to as emergent abilities. Inspired by the various emergent abilities [57, 58] possessed by foundation models, we expect GFMs to have similar abilities, including in-context learning, graph reasoning, and zero-shot graph generation, etc. In-context learning allows predictions for various downstream tasks with few examples [59]. Graph reasoning decomposes a complex problem into multiple sub-problems based on the graph structure and addresses them step by step, such as solving graph algorithm problems [60]. Zero-shot graph generation requires the model to generate graphs based on the desired conditions without the need for any examples [61]. Note that although language foundation models have demonstrated various emergent abilities, only a few works [59, 60, 61] have explored emergent abilities of GFMs so far.

Homogenization. Homogenization means that the graph foundation model can be applied to different formats of tasks, such as node classification, link prediction and graph classification. Note that due to the distinct characteristics of tasks on graphs compared to NLP tasks, achieving homogenization is not straightforward. The fundamental question in achieving homogenization is to decide which form to unify different types of graph tasks. Existing works have attempted homogenization through link prediction [55] or graph-level tasks [56], but there is no consensus on which approach is superior.

3.2 Key Technologies

Graph foundation models primarily comprise two key techniques: pre-training and adaptation. This section will provide a brief overview of these two techniques.

Pre-training. Pre-training is a pivotal concept in the development of graph foundation models, akin to its role in language models. It involves pre-training a neural network on a large graph dataset in a self-supervised manner. During pre-training, the model learns to capture the structural information, relationships, and patterns within the graph. There are several pre-training strategies for graph foundation models. Contrastive self-supervised learning [62, 63] leverages the idea of learning representations by contrasting positive samples (e.g., similar node pairs) against negative samples (e.g., dissimilar node pairs). Generative self-supervised learning [64, 65] encourages the model to reconstruct the structure or predict the features of original graph data. If using LLM as a part of the graph foundation model, we can also employ the pre-training methods introduced in Section 2.2.3. These diverse pre-training approaches enable graph foundation models to learn meaningful representations from raw graph data, enhancing their generalization and adaptability across various graph tasks.

Adaptation. The adaptation of graph foundation models involves tailoring these models to specific downstream tasks or domains to enhance their performance. This process includes several techniques, i.e., vanilla fine-tuning, parameter-efficient fine-tuning and prompt-tuning. Vanilla fine-tuning (Vanilla FT) entails training the entire pre-trained model on task-specific data, allowing for the highest level of customization but often requiring substantial data and computational resources. Parameter-efficient fine-tuning (Parameter-efficient FT) [66, 67], on the other hand, adjusts only a subset of the model's parameters, striking a balance between task-specific adaptation and resource efficiency. Prompt-tuning [56, 68] is a versatile approach that relies on external prompts to guide the model's behavior, making it more adaptable and effective. These adaptation techniques enable graph foundation models to excel in a wide range of applications by leveraging their pre-trained knowledge while tailoring their capabilities to specific tasks or domains, making them valuable for diverse downstream applications. Note that although LLMs have developed various types of prompt-tuning methods [52] and some other efficient tuning methods, such as Prefix Tuning [69], there are relatively few prompt tuning methods for graph foundation models.

3.3 Comparison between GFMs and LLMs

Through conceptual comparison, we can observe similarities in the goals and learning paradigms between graph foundation models (GFMs) and language foundation models (commonly referred to as large language models, LLMs). However, the uniqueness of graph data and graph tasks creates fundamental differences between them, which we refer to as their intrinsic differences. Furthermore, due to the relatively limited research on GFMs at present, many issues that have been extensively explored in LLMs remain unresolved, which we refer to as their extrinsic differences. We summarize the similarities and differences between GFMs and LLMs in Table 1, and will delve into them in detail in this section.

TABLE 1: The relationship between language foundation model and graph foundation model.

		Language Foundation Model	Graph Foundation Model
Similarities	Goal	Enhancing the model's expressive power and its generalization across various tasks	
	Paradigm	Pre-training and Adaptation	
Intrinsic differences	Data	Euclidean data (text)	Non-Euclidean data (graphs) or a mixture of Euclidean (e.g., graph attributes) and non-Euclidean data
	Task	Many tasks, similar formats	Limited number of tasks, diverse formats
Extrinsic differences	Backbone Architectures	Mostly based on Transformer	No unified architecture
	Homogenization	Easy to homogenize	Difficult to homogenize
	Domain Generalization	Strong generalization capability	Weak generalization across datasets
	Emergence	Has demonstrated emergent abilities	No/unclear emergent abilities as of the time of writing

3.3.1 Similarities

As shown in Table 1, both language foundation models and graph foundation models share the common goal of enhancing a model's expressive power and improving its ability to generalize across a wide range of tasks. They aim to create versatile, pre-trained models that can be adapted for specific applications. In addition, both follow the pre-training and adaptation paradigm. They begin by pre-training a model on a large, diverse dataset and then adapt it to task-specific data.

3.3.2 Intrinsic Differences

The intrinsic differences between GFM and LLM primarily manifest in two aspects: data and tasks. As for input data, language foundation models are primarily designed for processing Euclidean data, i.e., text. They are trained on vast text corpora, which are inherently sequential and follow a linear order of words or tokens. On the other hand, GFMs are designed to handle non-Euclidean data (represented as graph structures) or a mixture of Euclidean data (like graph attributes) and non-Euclidean data. Compared to text data, graph data can capture complex data relationships and is typically more sparse. Moreover, different graphs may exhibit significant differences in type or structure/geometry, all of which pose challenges in the design of GFMs. Furthermore, language data, even when sourced from texts in different domains, still share a common vocabulary. On the other hand, different graph data may lack this common foundation. For instance, nodes represent atoms in a molecular graph, while nodes represent users in a social network, which are entirely different. Furthermore, graphs from different domains can have different structures. Some graphs have a more hierarchical structure, while others may have a more cyclical structure [70]. Moreover, for a single graph, the different regions can exhibit different structures. For example, In recommender systems, the user-user subgraph and item-item subgraph generally exhibit very distinct structures [71].

In addition, LLMs are typically designed to handle dozens of tasks [72], but these tasks can all be unified under the format of masked language modeling. The reason is that these tasks all involve processing textual data and using the syntax and semantic information within the text. In contrast, GFMs focus a narrower set of tasks but with diverse formats. They excel at graph tasks such as node classification, link prediction and graph classification. The differences in tasks imply that GFMs cannot be learned using methods similar to those in LLMs, significantly increasing the adaptability challenges of GFMs in downstream tasks.

3.3.3 Extrinsic Differences

In addition to the intrinsic differences in data and tasks, there are also some extrinsic differences between GFMs and LLMs, which are due to the lag in technological advancements in GFMs. This section summarizes these differences as follows:

Backbone Architectures. LLMs, such as GPT-3 [73] and LLaMA [74], are mostly based on the Transformer architecture. The advantages of Transformer in terms of expressive power, scalability, parallelizability, and its excellent performance in handling various NLP tasks have made it the mainstream backbone architecture for LLMs. However, for GFMs, using mainstream GNNs as the backbone architecture may not necessarily be suitable. This is mainly because the expressive power and generalization of GNNs have limitations, and their parameter sizes are often too small to exhibit emergent abilities. To enhance the expressiveness of capturing the graph structure, recent works [70, 71, 75] efforts to extend GNN to mixed curvature Riemannian space, or to design graph transformers [63] or models that incorporate LLMs [76]. However, there is still no unified backbone architecture for GFMs.

Homogenization. LLMs are relatively easy to homogenize. This means that various NLP tasks can be formulated as the same task [77], making it possible to use a single model with unified training paradigm for a wide range of tasks. However, due to the poor transferability of graph structural knowledge, homogenization is more challenging for GFMs. Existing works attempt to achieve homogenization by unifying various tasks as link prediction [55] or graph-level tasks [56]. Additionally, constructing a data-task heterogeneous graph [59] may establish connections between tasks, but it is a more complex process.

Domain Generalization. LLMs have demonstrated strong domain generalization capabilities. They can often perform well on tasks and datasets that were not seen during training, showcasing their ability to generalize across various language-related domains. However, due to the diversity and lack of unified vocabulary of graph data, GFMs generally exhibit weaker generalization across datasets, especially when moving to cross-domain graph data [78]. Their performance may degrade significantly when faced with graph structures or characteristics that differ substantially from their training data. Achieving strong domain generalization remains a challenging research problem for GFMs.

Emergence. LLMs have shown emergent abilities, where they can generate coherent and contextually relevant text based on few examples or instructions. Representative emergent abilities include in-context learning [57], chain of thought reasoning [58] and zero-shot generation [79]. However, GFMs have not demon-

strated obvious emergent abilities to the same extent as language foundation models. Only a few recent studies discuss the in-context learning [59], graph reasoning [60] and zero-shot graph generation [61] abilities of GFMs.

3.4 Summary

In this section, we define the concept of graph foundation models and related technologies, and compares graph foundation models with language foundation models. If readers wish to have a more comprehensive understanding of the concept of GFMs, they can refer to our supplementary materials A and B, where we introduce the impact of graph data and graph tasks on GFMs. In the following sections, we will introduce three categories of methods for implementing graph foundation models, along with representative works for each method. Specifically, GNN-based models use GNN as the backbone architecture, while LLM-based models transform the graph into the input format of LLM and use LLM as the backbone architecture. GNN+LLM-based models, on the other hand, utilize both GNN and LLM as the backbone architecture simultaneously. The distinction in backbone architecture also impacts the methods for pre-training and adaptation. Therefore, in the following sections, we will introduce the backbone architectures, pre-training, and adaptation strategies for each category of methods, seperately.

4 GNN-based Models

Thanks to effective model architectures and training paradigms, language models have achieved remarkable performance in natural language processing tasks. The backbone, pre-training and adaptation techniques employed in language models have inspired a series of corresponding efforts in the field of graph-based tasks. In this section, we will delve into GNN-based models, which draw inspiration from the model architectures or training paradigms used in NLP to apply them to graph tasks. Importantly, unlike the LLM-based models and GNN+LLM-based models to be introduced in the following sections, GNN-based models do not explicitly model text data in their pipeline. We have summarized and categorized the works mentioned in this section in supplemental material C.

4.1 Backbone Architectures

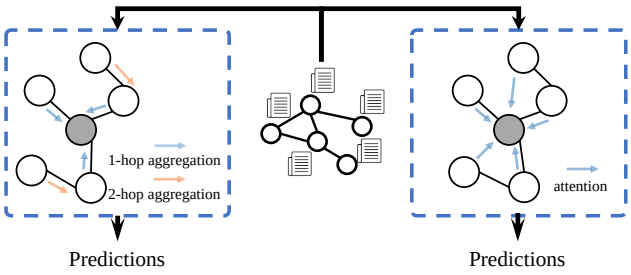
Numerous GNNs have been proposed and widely used in various graph-related downstream tasks. These networks are leveraged for feature extraction, often serving as the foundational components of graph models, commonly referred to as "backbones". In this subsection, we introduce two advanced GNN backbones: message-passing-based and transformer-based methods.

4.1.1 Message Passing-Based Methods

Message Passing Neural Networks (MPNNs) [36] represent a broad category of GNN architectures that operate based on the concept of message passing between nodes in a graph. In the message passing mechanism, each node aggregates information from its neighboring nodes, processes the information, and then sends messages to its neighbors in a series of iterative steps. A typical message passing process can be formulated as :

$$h_v^{k+1} = U^k(h_v^k, M^k_{u\in N(v)}(h_v^k, h_u^k, \mathbf{X}^e_{(u,v)})), \quad (1)$$

where h_v^k and h_v^{k+1} denote the embedding of node v at layer k and layer $k+1$, $\mathbf{X}^e_{(u,v)}$ denotes the edge attribute of edge (u, v),



(a) Message Passing. (b) Graph Transformer.

Fig. 2: A comparison between message passing-based models and graph transformer. A fundamental distinction is that the message passing mechanism is constrained by the graph structure, and the graph transformer treats the graph as a fully-connected network.

$N(v)$ denotes the neighbors of node v, $M^k_{u\in N(v)}(\cdot)$ and $U^k(\cdot)$ denote the message aggregation and update function at layer k.

Many existing GNN-based models utilize message passing-based models as their backbone. Due to the simplicity and effectiveness, several studies [56, 59, 62, 80, 81, 82] adopt GCN [14] as their backbone architecture, where GCN [14] employs a localized first-order approximation of spectral graph convolutions for the dual purpose of capturing graph structure and encoding node features. Several studies [56, 59, 64, 65] adopt GAT [38] as their backbone architecture, where GAT [38] replaces the average aggregation operation in GCN with a weighted aggregation approach, facilitated by an attention mechanism. Additionally, a multi-head attention technique can be further used in GAT to enhance its performance. GPPT [68] and VPGNN[83] uses GraphSAGE [37] as their backbone, which operates by sampling a fixed-size subset of neighboring nodes for each target node and then learns embeddings by aggregating and processing these sampled neighbors' embeddings. Unlike global attention, HGT employs type-specific parameters to define heterogeneous attention over each edge within the graph. To improve the expressive power, a proportion of studies [55, 66, 84, 85, 86, 87] rely on GIN [16] as their primary architecture, where GIN is a message passing-based model with expressive power theoretically equivalent to a 1-WL test [88]. Due to the expressive power of GIN, it is frequently chosen as the backbone for many GNN-based graph models. For an in-depth exploration of message passing-based GNNs, we recommend referring to [50, 89, 90].

4.1.2 Graph Transformer-Based Methods

While GNNs have demonstrated significant success in learning from graph data, they still confront fundamental limitations, including issues related to limited expressive power [16], over-smoothing [39] and over-squashing [18]. In parallel, the transformer architecture [91], which has revolutionized tasks in natural language processing [54, 73] and computer vision [92, 93], achieving the state-of-the-art performance. It has inspired the development of transformer-based models tailored for graph data [41, 42, 43, 94]. Graph transformers have exhibited promising results, particularly in molecular prediction tasks [41], owing to their fully-connected self-attention mechanism. This mechanism enables them to address the shortcomings of traditional message-passing GNNs thanks to its long-range modeling capability and strong expressive power.

The principal distinction between the backbone architectures with message passing mechanism and the graph transformer

lies in their treatment of the underlying graph structure. In the case of the graph transformer, it treats the graph as if it were fully connected, meaning it considers and measures the similarity between every pair of nodes in the graph. Conversely, the message passing mechanism operates under the constraint of the adjacency matrix of the graph. It only propagates information between nodes that are explicitly connected in the graph structure. We illustrate the difference between message passing-based models and graph transformers in Figure 2.

Currently, there are many research efforts focusing on graph transformers. Here we will present part of these studies that employ a pre-training and fine-tuning learning paradigm: Graph-BERT [95] uses intimacy-based and hop-based relative positional embeddings to encode node positions in a subgraph. The intimacy-based relative positional embeddings capture the relative positions of nodes in a subgraph based on their connectivity patterns. The hop-based relative distance embeddings capture the relative positions of nodes in a subgraph based on their hop distances. GROVER [96] uses a variant of MPNN called Directed Message Passing Networks (DyMPNs), which can capture the directed nature of molecular graphs and distinguish different types of edges. The DyMPNs in GROVER are used to compute the node and edge embeddings in the Transformer-style architecture. Graphormer [41] uses spatial encoding to represent node relationships, adding shortest path distance as a bias in softmax attention for better spatial dependency capture. Building upon this foundation, G-Adapter [67] introduces a parameter-efficient fine-tuning approach for graph transformers, utilizing Graphormer as its backbone model. For a more comprehensive exploration, please refer to other literature on graph transformers [97].

4.2 Pre-training

Pre-training in the field of NLP involves exposing a model to a vast amount of unlabeled text data, allowing it to learn general language semantic knowledge in a self-supervised manner. This pre-training step equips the model with a foundational understanding of language, enabling it to transfer this knowledge to downstream tasks. Similarly, the graph domain typically includes many unlabeled nodes and graphs, providing opportunities for pre-training on graphs. Graph pre-training enables the graph models to understand graph structure and semantic information, thus encoding meaningful node or graph embeddings [89, 90]. Recently, some graph pre-training methods have been proposed to learn representations in a self-supervised manner. Based on self-supervised tasks, graph pre-training methods can be categorized into two types: contrastive methods and generative methods.

4.2.1 Contrastive Methods

Specifically, the contrastive graph pre-training methods aim to maximize mutual information between different views [89], which forces the model to capture invariant semantic information across various views. The graph view can vary in scale, encompassing local, contextual or global perspectives. These perspectives correspond to node-level, subgraph-level, or graph-level information within the graph, leading to two distinct categories: (1) Same-scale contrastive learning and (2) Cross-scale contrastive learning. Same-scale contrastive learning compares two graph views at the same level. For example, GCC [84] uses a node's subgraph embedding as its representation, treating subgraphs of the same node as positives and different nodes as negatives. It applies NCE loss to align positives and separate negatives, capturing general patterns. GraphCL [85] and GRACE [62] generate two views by graph augmentation and then employ the InfoNCE loss to contrast node-level embeddings, pushing the graph model to acquire the invariant representations. MA-GCL [82] focuses on manipulating the neural architectures of view encoders instead of perturbing graph inputs or model parameters. GCOPE [98] unifies cross-domain graph pre-training using virtual coordinators and contrastive learning, reducing negative transfer and boosting downstream performance. FUG [99] ensures near-lossless adaptation to diverse graph features with PCA-inspired dimensional encoding and contrastive learning, enabling universal use without preprocessing or model changes. Cross-scale contrastive learning compares two graph views at different levels. For example, DGI [80] utilizes a discriminator to maximize the mutual information between the node embeddings and the graph embedding and minimize the information between node and corrupted graph embedding. Such a contrastive process encourages the encoder to capture information of the whole graph. Although DGI enables the model to capture semantic information about nodes and graphs, it ignores the discrepancies between different nodes.

4.2.2 Generative Methods

In addition to contrastive methods, some generative graph pre-training approaches have been proposed. The aim of generative pre-training methods is to enable GNNs to understand the general structural and attribute semantics of graphs. Thus, the GNNs can be adapted to downstream tasks with universal information. Generative learning frameworks for graphs can be classified into two categories based on how they acquire generative targets [90]: graph reconstruction and property prediction.

Graph reconstruction aims to reconstruct specific parts of given graphs, emphasizing fidelity in reproducing the original graph structure or node attributes. For example, VGAE [81] extends the VAE to the graph domain, where it first employs GCN as an encoder to generate node embeddings and then reconstructs the adjacency matrix by the inner product of node embeddings. Furthermore, GPT-GNN [100] proposes the self-supervised edge and attribute generation tasks to push the model to understand the inherent dependencies of attribute and structure. As a result, the model can learn high-order structure and semantic information. GraphMAEs [64, 65] consider that previous generative methods overemphasize structure information, instead, they employ the reconstruction of features and a re-mask decoding strategy in a self-supervised manner. In the property prediction category, models focus on learning and predicting non-trivial properties of provided graphs. For instance, GROVER [96] introduces tasks for nodes and edges, predicting context-aware properties within local subgraphs. The graph-level self-supervision task aims to predict motifs, framing it as a multi-label classification problem with each motif as a label.

Although generative methods are capable of generating novel content, the quality and interpretability of the content are hard to guarantee. In the future, it remains to be explored how to enhance the accuracy and rationality of the generative methods.

4.3 Adaptation

Typically, the objectives of pre-training tasks are different from the downstream ones, which hinders the transferability of pre-training models. To this end, fine-tuning is a common adaptation

approach based on subtle adjustments of model parameters. In addition, the *"pre-train, prompt and predict"* paradigm has attracted considerable attention in recent years. By using prompts, the format of downstream tasks is aligned with that of pre-training tasks, enabling pre-training models to handle downstream tasks in a more effective manner.

4.3.1 Fine-Tuning

For the situation where the model conducts the pre-training and downstream tasks in the same domain, we can utilize a pre-training model to generate node embeddings or graph embeddings, and subsequently fine-tune an external task-specific layer to generalize the pre-training model to downstream tasks. DGI [80] and GRACE [62] utilize the pre-trained encoder to obtain node embeddings, and then fine-tune a logistic regression classifier with labeled data to handle the node classification task. Additionally, there is a more practical scenario where pre-training is performed on the known graphs while tested on unseen graphs. Pre-training models cannot encode unknown graphs appropriately, thus we need to fine-tune the model in this situation. GPT-GNN [100] employs the labeled data to fine-tune a downstream task-specific decoder, which guides the pre-training model to adapt to downstream tasks. Moreover, some parameter-efficient fine-tuning methods have been proposed recently. AdapterGNN [66] employs two parallel adapters before and after the message passing stage to modify the input graph. Such addition-based methods only need to fine-tune the introduced parameters. G-Adapter [67] proposes a parameter-efficient fine-tuning method for graph transformer, which introduces graph structure into the fine-tuning by message passing. G-TUNING [101] is a fine-tuning strategy for GNNs that utilizes graphon reconstruction to preserve generative patterns and address structural divergence between pre-training and downstream datasets

Although the fine-tuning methods have achieved significant success, they typically require sufficient labeled data to tune the model parameters. Moreover, conventional fine-tuning methods necessitate repetitive fine-tuning for each task, incurring significant computational costs. Therefore, more advanced fine-tuning techniques for graph foundation models are still to be explored.

4.3.2 Prompt Tuning

Prompt tuning has recently emerged as a strategy to circumvent the need for full-parameter tuning, facilitating both multi-task adaptation and zero-shot learning [52, 102, 103]. This innovative approach has found significant applications in graph data, where recent studies have concentrated on using prompt tuning to enhance the performance and adaptability of GNN-based models. Following the framework proposed by Guo et al. [86], these methods can be categorized into two distinct groups: pre-prompt methods and post-prompt methods, based on whether the task-specific prompts operate before or after the backbone module.

Pre-prompt methods modify the input graph's topology or node features before message passing to aid downstream tasks or construct a prompt graph to enhance model adaptation. For example, AAGOD [86] proposes to implement the data-centric manipulation by superimposing an amplifier on the adjacency matrix of the original input graph as learnable instance-specific prompts. All In One [56] converts the node-level and graph-level tasks to graph-level tasks. It treats an extra subgraph as a prompt and merges it with the node subgraph. The model subsequently utilizes the combined graph to generate predictions. GPF [87] introduces a uniform feature modification vector to each node in the graph, which can be optimized to adapt pre-training GNN models under any pre-training strategy. Additionally, it features verbalizer-free prompting function, thus aligning the downstream task with the pre-training method's format. PRODIGY [59] is a framework for pre-training an in-context learner over prompt graphs. The goal is to enable a pretrained model to adapt to diverse downstream tasks without optimizing any parameters. IGAP [104] bridges the gap between graph pre-training and inductive fine-tuning by addressing the graph signal and structure gaps using learnable prompts in the spectral space. TPP [105] achieves a replay-free and forget-free GCIL system by storing task-specific knowledge in compact learnable tokens using graph prompts with a frozen pre-trained GNN, avoiding model updates or data replay.

Post-prompt methods use task-specific prompts on representations after message passing to aid downstream task adaptation. For example, GPPT [68] employs a prompting function to generate a token pair for each class, transforming all node classification tasks into link prediction tasks. GraphPrompt [55] unifies pre-training and downstream tasks into a consistent task format based on subgraph similarity, and utilizes labeled data to learn a task-specific prompt vector for each task, which modifies the model's Readout operation and narrows the gap between link prediction and downstream tasks. GraphPrompt+ [106] further enhances GraphPrompt by generalizing pre-training tasks and employing layer-wise prompts to capture hierarchical knowledge across the graph encoder, improving task-specific knowledge extraction for both node and graph classification. ProNoG [107] uses conditional prompting for non-homophilic graphs, leveraging a condition-net to generate node-specific prompts that refine embeddings for fine-grained task adaptation.

Although these methods have improved the performance in few-shot scenarios, further exploration is needed to understand the semantics and interpretability of the graph prompts.

4.4 Discussion

GNN-based models offer several advantages, particularly their ingenious inductive bias and compact parameter size. These models naturally possess essential properties like permutation invariance, enabling them to handle graph-structured data effectively. Additionally, GNN-based models offer the advantage of low-cost training and efficient resource usage, which reduces computational requirements and makes them accessible for deployment even in resource-constrained environments. Moreover, they can generalize well from small amounts of labeled data. By propagating label information through the graph, they can enhance prediction accuracy even when labeled data is sparse.

Furthermore, for more complex graph data such as heterogeneous graphs, hypergraphs, and temporal graphs, preliminary research have investigated GNN-based graph foundation models. For example, CPT-HG [108] and PT-HGNN [109] have designed sophisticated pre-training methods tailored for high-order semantic information in heterogeneous graphs. MultiGPrompt [110] and HGPROMPT [111] both use prompt-based learning to link pre-training and downstream tasks on heterogeneous graphs. PhyGCN [112] and IHP [113] use self-supervised hyperedge prediction and instruction-based prompts respectively to improve node representations in hypergraphs. GraphST [114] and GPT-ST [115] use pre-training to improve spatio-temporal representations for temporal graphs.

Despite their many advantages, GNN-based models have several notable disadvantages. One primary limitation is their lack of explicit text modeling. They often underutilize the rich semantic information embedded in textual attributes associated with nodes or edges, leading to suboptimal exploitation of textual data. Another significant drawback is the limited capacity of GNN-based models to incorporate and utilize general knowledge effectively. Unlike LLMs, which are pre-trained on vast corpora of text and can leverage extensive world knowledge, GNN-based models typically lack such pre-training on diverse and comprehensive datasets. This gap restricts their ability to generalize across different domains and limits their performance in tasks requiring broad contextual understanding or common-sense reasoning.

One promising direction is the integration of LLMs with GNN-based models. LLMs can provide a robust framework for incorporating extensive textual knowledge, enhancing the models' ability to understand and utilize semantic information embedded in text. In the following sections, we will explore graph learning models that incorporate LLMs.

5 LLM-BASED MODELS

Researchers are actively exploring ways to leverage LLMs as core and sole backbone for graph learning [60, 116, 117], for the following advantages that can not be underestimated. Firstly, transformer-based models show a remarkable ability to seamlessly integrate textual information in graph data [116]. Additionally, employing a LLM-liked backbone empowers models to unify diverse graph learning tasks, as these tasks can be described using natural language. Furthermore, recent advancements, such as NLGraph [60], GPT4Graph [117], showcase the LLMs' prowess in preliminary graph reasoning. These advantages mark a highly promising direction for the development of such models. To discover the potential of engaging LLMs into graph learning, these works involve both graph-based properties and textual information as input for the backbone networks. Following some surveys [21, 118], our characterization of the backbone is not confined solely to the narrow definition of LLMs (like GPT-4); it also encompasses certain transformer-based models that leverage textual information. We have summarized and categorized the works mentioned in this section in supplemental material D.

5.1 Backbone Architectures

A central question in employing LLMs for graph data is how to align graph data with natural language so that LLMs can understand them. Given that LLMs initially accept tokens as their inputs and rely on self-attention layers to process input sequences for producing hidden representations, it can be a difficult task to attain a fine-grained modeling of graph structure information [116]. As illustrated in Figure 3, we categorize existing LLM-based methods into two types, graph-to-token and graph-to-text. The key distinction between these two approaches lies in the use of an additional encoder. Graph-to-token methods rely on an additional encoder (e.g., BERT) to generate embedding-level representations for each node, while graph-to-text method directly translates graph representations into natural language input for LLMs without the need for an additional encoder.

5.1.1 Graph-to-token

One approach entails the tokenization of graph information and imitates the standardized input format of transformer-based models. This methodology necessitates not only the serialization of graph data into tokens but also the solutions for encoding the graph's structural information. Since this method uses node representations as unique tokens for the input of backbone models, the backbone need to be either trainable transformers or open source LLMs. For instance, InstructGLM [116] uses LLaMA [74] and T5 [77] to be their backbones for further tuning.

The concept of graph-to-token initially surfaces in GIMLET [119] that treats node representations as tokens and aims to integrate graph data with textual data. Specifically, GIMLET expands the capabilities of LLMs to accommodate both graph and text data by using the transformer architecture, incorporating generalized position embedding and instruction-based pre-training. Furthermore, efforts have been made to integrate graph data with other modalities beyond just text data. For instance, Meta-Transformer [120] introduces a transformer-based architecture designed to incorporate various forms of multimodal data, including graphs, text, and images. However, despite the promising trend indicated by developing unified multimodal intelligence using a transformer backbone, their approaches cannot be considered as graph foundation models because they do not involve any pre-training and adaptation learning paradigm.

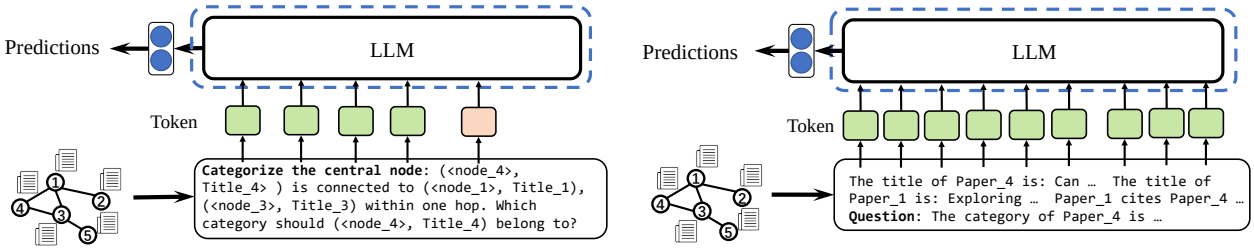
InstructGLM [116] on the other hand, adopts a pre-training and adaptation framework and introduces LLMs to further enhance the model's text processing capabilities, making it a strong contender for the position of a graph foundation model. In this framework, the vocabulary of the LLMs is expanded by incorporating the inherent node feature vectors from the graph as distinct and unique tokens for each node. Leveraging LLMs and the transferability of natural language, InstructGLM makes a valuable contribution to the ongoing movement towards graph foundation model architectures and pipelines that span multiple modalities.

These efforts tokenize graph data to align it with natural language, enabling joint understanding with data from other modalities. Their conclusions showcase promising results for integrating graph data with natural language. However, despite these promising results, how to inform LLMs of underlying graph structures remains an important challenge in this approach.

5.1.2 Graph-to-text

To align graph data with natural language, another approach involves describing graph information using natural language. Several approaches have been developed along this line of thoughts, utilizing prompts to integrate the capabilities of LLMs into classical tasks on graphs. For this method, which exclusively relies on natural language prompts, the associated backbone model can be any LLM, even if it is not open-sourced. For instance, Graph-LLM [118] utilizes multiple language models of different sizes, including BERT [54], DeBERTa [121], Sentence-BERT [122], GPT-4 [123] and LLaMA [74].

Initial attempts mainly use edge list to describe graph structures in natural language and make assessment on various graph tasks. NLGraph [60] also conducts a comprehensive assessment of LLMs across eight graph reasoning tasks as well as popular GNN tasks in natural language. Similarly, utilizing edge lists to describe graph structure, the results once again underscores the limitations of this approach when dealing with complex graph problems. TextForGraph [124] designed two types of prompts, full text and reduced text, to describe information on the graph, effectively compressing the prompt length. When&Why [125] designs several styles of prompts and offers key insights into the use of LLMs for processing structured data. GraphWiz [126] designs different



(a) Graph-to-token. (b) Graph-to-text.

Fig. 3: An illustration of two existing approaches to align graph data with natural language. One approach tokenizes graph data and use node representations (depicted as red tokens) as well as text tokens (depicted as green tokens) to be the input of LLMs. Another approach represents graph data with prompts in natural language and uses text tokens only (depicted as green tokens) as the input of LLMs.

prompts for various tasks on graphs, including cycle detection, subgraph matching, and more.

Moreover, GPT4Graph [117] introduces a novel approach to prompt engineering that combines manually crafted prompts with prompts generated by the language model itself, referred to as handcrafted prompts and automatic prompts. Specifically, for manual prompting, it utilizes description languages such as edge lists, adjacency lists, Graph Modeling Language (GML), and Graph Markup Language (GraphML) to represent graph structures and compare their effectiveness. For automatic prompting, it employs techniques like graph summarization, neighborhood summarization, graph exploration, and graph completion to actively engage LLMs in understanding and manipulating graphs, facilitating graph-based reasoning and learning. The findings indicate that self-prompting is a more effective method for informing LLMs about graph structures. Graph-LLM [118] further supports this conclusion, emphasizing that neighbor summarization is the most effective technique in existing prompt engineering methods.

These studies highlight significant potential for using natural language to describe graph data and using textual tokens as the input of LLMs for graph learning. Nevertheless, a key takeaway from their conclusions is that, at the present moment, the way we use these prompts may not be an effective approach for mining underlying graph structures.

5.2 Pre-Training

The methods discussed in this section solely employ LLMs as the backbone. Hence, the pre-training phase for these methods corresponds to the pre-training phase of LLMs. There are mainly two tasks used in LLM-based models for graph learning, we will provide a concise overview of these two pre-training tasks.

5.2.1 Language Modeling (LM)

Language Modeling (LM) is one of the most common self-supervised task in NLP, and is widely adopted by many state-of-the-art LLMs, such as LLaMA [74] and GPT-3 [127]. LM task can be essentially addressed to the challenge of estimating probability distributions of the next word. While LM represents a broad concept, it frequently pertains specifically to auto-regressive LM or unidirectional LM in practical applications [51]. Many methods involve LM as their pre-training method, namely InstructGLM [116], NLGraph [60], GPT4Graph [117], Graph-LLM [118], TextForGraph [124], When&Why [125], GraphWiz [126] and CGForLLM [128].

In the context of a text sequence represented as $s_{1:L} = [s_1, s_2, \cdots, s_L]$, its overall joint probability, denoted as $p(s_{1:L})$, can be expressed as a product of conditional probabilities, as shown in equation:

$$p(s_{1:L}) = \prod_{l=1}^{L} p(s_l | s_{0:l-1}). \tag{2}$$

Here, s_0 represents a distinct token signifying the commencement of the sequence. The conditional probability $p(s_l|s_{0:l-1})$ is essentially a probability distribution over the vocabulary based on the linguistic context $s_{0:l-1}$. To model the context $s_{0:l-1}$, a neural encoder $f_{nenc}(\cdot)$ is employed, and the conditional probability is calculated as follows:

$$p(s_l|s_{0:l-1}) = f_{lm}(f_{nenc}(s_{0:l-1})). \tag{3}$$

In this equation, f_{lm} represents the prediction layer. By training the network using maximum likelihood estimation (MLE) with a large corpus, we can effectively learn these probabilities. Nevertheless, a drawback of unidirectional language models is their encoding of contextual information for each token, which is solely based on the preceding leftward context tokens and the token itself. However, for more robust contextual representations of text, it is preferable to capture contextual information from both the forward and backward directions.

5.2.2 Masked Language Modeling (MLM)

Masked language modeling (MLM) is introduced to address the limitation of the traditional unidirectional language model, frequently denoted as a Cloze task [51]. In MLM, specific tokens within the input sentences are randomly masked, and the model is then tasked with predicting these masked tokens by analyzing the contextual information in the surrounding text. As an effective pre-training task, MLM is adopted in BERT [54] and T5 [77]. Additionally, MLM can be categorized into two subtypes: Sequence-to-Sequence MLM (Seq2Seq MLM) and Enhanced MLM (E-MLM). Seq2Seq MLM involves utilizing the masked sequences as input for a neural encoder, and the resulting output embeddings are used to predict the masked token through a softmax classifier. On the other hand, E-MLM extends the mask prediction task to various types of language modeling tasks or enhances MLM by integrating external knowledge. MLM also faces some drawbacks as this pre-training method may result in a disconnection between the pre-training and fine-tuning stages since the mask token is absent

during fine-tuning. InstructGLM [116] and Graph-LLM [118] use T5/BERT as backbones, and adopt MLM pre-training strategy.

There are also many pre-training tasks like Permuted Language Modeling (PLM) [129], Denoising Autoencoder (DAE) [130], Text-Text Contrastive Learning (TTCL), [131] and others. These pre-training tasks are currently not adopted by existing LLM-based models in graph learning, and thus not within the scope of our discussion in this section. However, we believe that in the future, more research will be developed on graph tasks involving these pre-training tasks, offering additional possibilities for the establishment and refinement of graph foundation models.

5.3 Adaptation

The adaptation phase plays a pivotal role in enhancing the performance of LLM-based models in graph learning. LLMs are primarily trained on textual corpora, which results in a significant gap between the pre-training phase and their deployment on graph tasks. Both the graph-to-token and graph-to-text methods are accompanied by specific adaptation techniques designed to enhance the LLM's ability to understand graph data effectively. As these methods share a fundamentally similar training procedure that utilizes prompts, we classify these adaptation strategies in the aspect of prompt engineering: manual and automatic.

5.3.1 Manual Prompting

Methods mentioned here use manually created prefix style prompts. For instance, LLMtoGraph [132] and NLGraph [60] employ node and edge lists incorporating other graph properties described in natural language to form a comprehensive prompt. GPT4Graph [117] goes a step further by utilizing additional description languages to represent graph data, such as edge list, adjacency list, GML and GraphML, providing a more extensible framework for manual graph prompts. Furthermore, InstructGLM [116] employs instruction prompting to involve the design of a set of graph descriptions centered around a central node, coupled with task-specific descriptions. Graph-LLM [118], TextForGraph [124], When&Why [125], GraphWiz [126] and CGForLLM [128] also use natural language instructions and subsequently conduct a series of comprehensive experiments.

5.3.2 Automatic Prompting

Creating prompts manually is a time-consuming task, and these prompts can sometimes be sub-optimal [133]. To address these drawbacks, automatically generated prompts have been introduced for further adaptation. GPT4Graph [117] firstly tries to employ three different types of prompts generated by LLM itself, namely graph summarization, graph exploration and graph completion, in graph tasks. Specifically, graph summarization generates a summary of the given graph by extracting key features or a summary of the neighborhood information of target nodes. Graph exploration means generating a sequence of queries or actions to retrieve information from the given graph. Graph completion generates partial graphs and prompt itself to complete the missing parts. By leveraging these self-prompting strategies, LLMs can actively engage in the understanding and manipulation of graphs, facilitating graph-based reasoning and learning. Graph-LLM [118] uses automatic prompts as well in the form of neighbor summary, and their experimental results once again emphasize the efficiency of automatic prompting.

Additionally, there are various adaptation approaches based on fine-tuning, including Vanilla Fine-Tuning [54], Intermediate Fine-Tuning (IFT) [134], Multi-task Fine-Tuning (MTFT) [135], and Parameter Efficient Fine-Tuning [136]. These methods offer efficient ways to adapt pre-trained models to downstream tasks, although they have not been applied to graph tasks at this time. However, we anticipate that future research will explore the integration of these adaptation approaches into graph tasks, further advancing the development of the graph foundation model.

5.4 Discussion

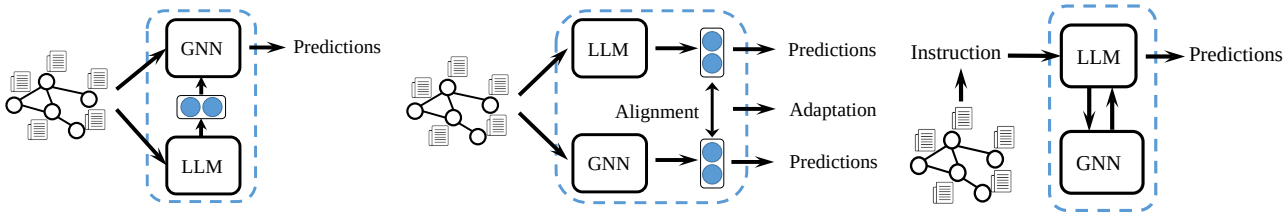
Efforts of aligning graph data with natural language and using sole LLMs as graph learners has paved the way for exciting developments. The integration of graph data, text, and other modalities into transformer-based models presents a promising way, with the potential to connect techniques from the GNN field with advancements in the LLM domain. Moreover, leveraging LLMs allows for the unification of various graph tasks, as these tasks can all be described in natural language. This makes LLM-based backbones a more competitive selection for building GFMs.

Nonetheless, it is essential to acknowledge that the current ways of utilizing LLMs as backbones for graph learning also presents inherent limitations. These limitations encompass the inability of LLMs to effectively process the lengthy textual information required to describe graph structures, their incapacity to engage in multi-hop logical reasoning through graph links, the challenge they face in capturing the topological structures prevalent in highly connected graphs, and their struggle in handling the dynamic nature of graphs that evolve over time. Furthermore, graph-to-text methods are constrained by the LLM's input length, limiting the size of the graph data they can handle. In contrast, graph-to-token methods incur higher computational costs but can process large-scale graph data encountered in real-world scenarios, as each node can usually be represented by just a single token [116]. These shortcomings underscore the need for further research in using LLM-based models for graph learning.

Future research directions for LLM-based approaches include enhancing the ability of LLMs to more effectively and efficiently understand critical information in graphs, including node features and topological structures. Considering that LLMs cannot directly comprehend graphs, and flattened natural language description of graphs are likely to result in information loss, efficient and structured modeling techniques for graphs need to be developed. These methods are expected to help bridge the gap between natural language prompts and the comprehensive information present in graph data. Moreover, existing research, such as LLM4DYG [137], has explored the application of LLMs to complex graph data, specifically temporal graphs. However, more diverse types of graph data, such as hypergraphs and heterogeneous graphs, need to be explored.

6 GNN+LLM-based Models

GNN-based models lack the ability to process text and thus cannot directly make predictions based on textual data. Additionally, they cannot make predictions based on natural language instructions provided by users. Consequently, exploring the performance of models with a substantial parameter count in graph-related tasks is imperative. On the other hand, LLM-based models for graph learning have their inherent limitations. These limitations include the incapability of LLMs to process precise mathematical calculations and the inability to handle multi-hop logical reasoning, etc. These shortcomings underline the necessity for further research



(a) GNN-centric methods.

(b) Symmetric methods, where the aligned embeddings can be further utilized for downstream tasks.

(c) LLM-centric methods, which take an instruction as input and output an answer.

Fig. 4: An illustration of GNN+LLM-based models.

and innovation in this domain. To overcome these limitations and harness the strengths of both language understanding from LLMs and structural analysis from GNNs, integrating LLMs and GNNs can potentially lead to a more comprehensive and powerful model. We summarize and categorize the works mentioned in this section in supplemental material E.

6.1 Backbone Architectures

To simultaneously utilize information from both the graph and text and accomplish a variety of tasks, we need to design a framework that effectively integrates LLM and GNN. Depending on the prediction model, GNN+LLM-based methods can be classified as: 1) GNN-centric Methods, 2) Symmetric Methods, and 3) LLM-centric Methods, as illustrated in Figure 4.

6.1.1 GNN-centric Methods

Several works aim to utilize LLM to extract node features from raw data and make predictions using GNN. These approaches are denoted as GNN-centric models. For example, GraD [138] performs a parameter-efficient fine-tuning of an LLM on the textual dataset of a TAG (text-attributed graph). The textual dataset T is annotated with task-specific labels $\mathbf{Y}$, where $G = (V, E, T)$ and T is the set of texts with each element aligned with a node in V. Then the downstream task loss for fine-tuning is:

$$\begin{aligned} \text{Loss}_{\text{CLS}} &= \mathcal{L}_\theta(\phi(\text{LLM}(T)), \mathbf{Y}), \\ \text{Loss}_{\text{LINK}} &= \mathcal{L}_\theta\left(\phi\left(\text{LLM}\left(T_{\text{src}}\right), \text{LLM}\left(T_{\text{dst}}\right)\right), \mathbf{Y}\right), \end{aligned} \tag{4}$$

where $\phi(\cdot)$ is the classifier for the classification task or similarity function for the link prediction task, T_{src} and T_{dst} are the texts of the source node and the target node, respectively, Loss_{CLS} and $\text{Loss}_{\text{LINK}}$ are the loss of classification and link prediction task, respectively. Thus we can get the node representations $\mathbf{X}$ with fine-tuned LLM, achieved by removing the head layer. Then we can train GNN with the loss:

$$\begin{aligned} \text{Loss}_{\text{CLS}} &= \mathcal{L}_\theta(\phi(\text{GNN}(\text{LLM}(T))), \mathbf{Y}), \\ \text{Loss}_{\text{LINK}} &= \mathcal{L}_\theta\left(\phi\left(\text{GNN}\left(\text{LLM}(T_{\text{src}})\right), \text{GNN}\left(\text{LLM}(T_{\text{dst}})\right)\right), \mathbf{Y}\right), \end{aligned} \tag{5}$$

where $\phi(\cdot)$ is the classifier for the classification task or similarity function for the link prediction task, $\mathbf{Y}$ is the task-specific labels, T is the set of texts, T_{src} and T_{dst} are the texts of the source node and the target node, respectively.

For LLMs that do not provide direct access to their embeddings such as ChatGPT, TAPE [76] engages these LLMs through text interactions. Specifically, TAPE first utilizes an LLM to generate a ranked prediction list and explanation based on the original text, and then an LM is utilized and fine-tuned to transform the original text and additional features of predictions and explanation generated by LLM into node features. Subsequently, downstream GNNs can utilize the features for prediction tasks. TAPE extracts graph-agnostic features and cannot capture correlations between graph topology and raw features. To this end, GIANT [139] utilizes a graph-structure aware self-supervised learning method to finetune the LM. Consequently, the text representations encompass graph-related information. WTGIA [140] focuses on text-level Graph Injection Attacks (GIAs), enhancing the interpretability and real-world applicability of graph injection attacks. In GALM [141], the focus is on exploring pre-training approaches for models that combine text and graph data, particularly on extensive heterogeneous graphs enriched with rich textual data. OFA [142] introduces text-attributed graphs that use natural language to describe nodes and edges, unified by language models into a common embedding space. Heterformer [143] integrates contextualized text encoding and heterogeneous structure encoding within a single model. It incorporates heterogeneous structure information into each Transformer layer as it encodes node texts. Edgeformers [144], which are based on graph-enhanced Transformers, perform edge and node representation learning by contextually modeling texts associated with edges. LLMRec [145] improves recommender systems by using three straightforward yet powerful LLM-based graph augmentation techniques, addressing the issues of sparse implicit feedback and low-quality side information commonly found in recommendation systems. WalkLM [146] conducts attributed random walks on the graph and uses an automated program to generate approximately meaningful textual sequences from these walks. It then fine-tunes a language model (LM) with these textual sequences and extracts embedding vectors from the LM, capturing both attribute semantics and graph structures. TOUCHUP-G [147] enhances the node features derived from a pre-trained model for downstream graph tasks and introduces. Multiplex graph neural networks initialize node attributes as feature vectors for node representation learning, but they fall short in capturing the full semantics of the nodes' associated texts. METERN [148] addresses this by using a single text encoder to model the shared knowledge across relations and employing a small number of parameters per relation to generate relation-specific representations. Another research [149] investigates the use of LLMs to improve graph topological structures, a relatively unexplored area. A label-free pipeline, LLM-GNN [150], uses

LLMs for annotation and supplies training signals to GNNs for subsequent prediction.

6.1.2 Symmetric Methods

Also, there are some works that align the embeddings of GNN and LLM to make better predictions or utilize the embeddings for other downstream tasks, denoted as symmetric methods. Most GNN-centric based methods involve two sequential steps: text encoding and graph aggregation. It is important to note that during the generation of text embeddings, there is no exchange of information between nodes. To consider the interrelated nature of connected nodes, several works try to utilize GNN and LLM together to get structure-aware text features. GraphFormer [151] fuses text embedding and graph aggregation as an iterative workflow. During each iteration, the interconnected nodes will engage in information exchange within the layerwise GNN component, formulated as $\hat{z}^l = \mathrm{GNN}(z^l)$, where $\mathbf{z}^l$ is the output of l-th layer of GNN.

As a result, each node will incorporate information from its neighboring nodes. The Transformer component then operates on these enhanced node features, enabling the generation of progressively more informative node representations as $z^{l+1} = \mathrm{TRM}(\mathrm{CONCAT}(\hat{z}^l, h^l))$, where TRM is the transformer, and h^l is the output of l-th layer of transformer. However, this method suffers from scalability issues because the memory complexity is proportional to the graph size as neighborhood texts are also encoded. GLEM [152] employs a variational EM framework to alternatively update the LLMs and GNNs, thus essentially capturing the node label distribution conditioned on the local text attributes. In contrast, GNN uses the text and label information of neighboring nodes to predict labels, thus characterizing global conditional label distribution. By doing so, GLEM efficiently incorporates local textual data and global structural information into its components and can ease the scalability issue.

Other studies employ distinct encoders for graph nodes and texts, training them to align their representations within a shared latent space. G2P2 [153] jointly pre-trains a graph-text model utilizing three graph interaction-based contrastive strategies, and then explores prompting for the downstream tasks. [154] utilizes GNN to model the structural information of nodes, which is then integrated with the corresponding text fragment encoded by a language model. The model subsequently predicts the masked token. ENGINE [155] integrates large language models and graph neural networks using an adjustable side structure. This approach significantly reduces training complexity while maintaining the capacity of the combined model. To address this, PATTON [156] incorporates two pre-training strategies: network-contextualized masked language modeling and masked node prediction, aiming to capture the inherent relationship between textual attributes and network structure. OpenGraph [157] enhances the graph learning paradigm by developing a flexible graph foundation model. This model can understand complex topological patterns in diverse graph data, enabling it to excel in zero-shot graph learning tasks across a range of downstream datasets. RLMRec [158] improves the recommendation performance of current recommender systems by utilizing large language models (LLMs) and aligning their semantic space with collaborative relation modeling to achieve better representation learning. Some other works [159, 160, 161] also utilize GNN and LLM to learn representations for molecules. These models employ a contrastive learning strategy to effectively pre-train on a dataset containing pairs of molecular graphs and corresponding textual descriptions. By simultaneously learning the chemical structures of molecules and their associated text through this approach, these models can then be applied to various downstream tasks. Furthermore, MolCA [162] allows a language model (LM) to comprehend both text-based and graph-based molecular information through the use of a cross-modal projector. GIT-Mol [163] encompasses all three modalities in molecular science—graph, image, and text—supporting tasks such as molecule generation, molecule captioning, molecular image recognition, and molecular property prediction.

6.1.3 LLM-centric Methods

While LLMs have shown impressive performance in various natural language tasks, they struggle with precise mathematical calculations, multi-step logic reasoning, spatial and topological perception, and handling temporal progression. Hence some works utilize GNNs to enhance the performance of LLM, denoted as LLM-centric methods. For example, GraphTranslator [164] utilizes a Graph Model to efficiently manage predefined tasks and takes advantage of the extended interface of Large Language Models to support a variety of open-ended tasks for the Graph Model. GraphGPT [165] integrates large language models with graph structural knowledge through graph instruction tuning, enabling LLMs to understand complex graph structures and improving adaptability across various datasets and tasks. THLM [166] introduces a novel pre-training framework for language models that explicitly incorporates the topological and heterogeneous information found in text-attributed heterogeneous graphs. GraphPrompter [167] aligns graph information with LLMs via soft prompts. InstructGraph [168] enhances LLMs with graph reasoning and generation capabilities through instruction tuning and preference alignment. RELM [169] utilizes the chemical knowledge embedded in LMs to support GNNs, thereby improving the accuracy of real-world chemical reaction predictions. TEA-GLM [170] pretrains a GNN using contrastive learning to capture structural and semantic graph information, then employs a linear projector to map GNN representations into unified task-specific instructions for LLMs, enabling effective cross-dataset and cross-task generalization without fine-tuning the LLM. G-Retriever [171] introduces G-Retriever, a retrieval-augmented generation (RAG) framework that enables question answering on real-world textual graphs through a conversational interface, mitigating hallucinations and scaling efficiently to large graphs.

6.2 Pre-training

To train the model and enable it to handle both graph and text information, we need to train the model on a large amount of data. LLM and GNN can be pre-trained on textual data and graph data respectively, and the GNN+LLM-based methods can be pre-trained on both data. In this subsection, we category the pre-training strategies as GNN or LLM-based, and alignment-based.

6.2.1 GNN or LLM-based

Other frameworks leverage pre-trained LLMs to obtain text embeddings. The majority of existing models [138, 139, 151, 152, 159, 160, 161, 162] employ Masked Language Modeling (MLM) during pre-training. Some models, like TAPE and Graph-ToolFormer, opt for Language Modeling (LM) in the pre-training phase. Additionally, SimTeG integrates Text-Text Contrastive Learning (TTCL), a technique that leverages certain observed text

pairs exhibiting more semantic similarity than randomly selected pairs during the pre-training phase as:

$$\text{Loss}_{\text{TTCL}} = \mathbf{E}_{x,y^+,y^-}\left[-\log\frac{\exp\big(k\big(x,y^+\big)\big)}{\exp(k(x,y^+))+\exp(k(x,y^-))}\right], \quad (6)$$

where $\mathbf{E}$ is the expectation, k is the score function, y^+ is the positive sample and y^- is the negative sample. Additionally, GALM [141] utilizes graph reconstruction for pre-training on extensive graph datasets, and thus can incorporate the graph information into the pre-trained LLMs.

6.2.2 Alignment-based

Symmetric methods of LLM and GNN like Text2Mol [159], MoleculeSTM [160], and CLAMP [161] are pre-trained on large datasets with Graph-Text Contrastive Learning (GTCL), which aligns the embeddings of the graph encoder and the text encoder. The embeddings involve rich information about graph structure and text, thus demonstrating appealing performance on downstream datasets. For a molecule example, CLAMP minimizes the contrastive loss as below:

$$\begin{aligned}\text{Loss}_{\text{NCE}} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(k(\text{LLM}(\mu_i), \text{GNN}(\xi_i))) \\ + (1-y_i)\log(1-k(\text{LLM}(\mu_i), \text{GNN}(\xi_i))),\end{aligned} \quad (7)$$

where μ_i is the text representation, ξ_i is the graph representation, and k is a score function to predict the activity of a molecule. The contrastive loss promotes the active molecules on a bioassay to have similar embeddings to the specific bioassay, while ensuring that inactive molecules have dissimilar embeddings to it.

6.3 Adaptation

The adaptation phase plays a pivotal role in optimizing GNN+LLM-based models for efficient graph learning. Apart from some works [159, 160, 161] which test the model's performance on zero-shot tasks such as zero-shot structure-text retrieval and zero-shot text-based molecule editing, models in most cases need adaptation. In this subsection, we categorize these adaptation strategies into two main types: fine-tuning and prompt-tuning.

6.3.1 Fine-tuning

To adapt to the downstream tasks, some works [139, 141, 143, 144, 146, 151, 151, 152] utilize vanilla fine-tuning methods for node classification tasks. However, vanilla fine-tuning methods involve adjusting a broad range of model parameters, which can be computationally intensive and resource-demanding. So other works utilize parameter-efficient fine-tuning methods for downstream tasks, resulting in a more efficient and resource-friendly approach. Specifically, several studies [159, 160, 161] align the embedding space of GNN and LLM utilizing paired molecule graph-text data, while other research [138, 145, 148, 150, 156, 166] is tuned on TAGs with classification task. Additionally, some works [162, 164, 167] adapt to downstream tasks by generating text captions or descriptions.

6.3.2 Prompt-Tuning

The prompt-tuning approach is employed in certain studies [153, 154, 163, 165, 168]. For example, G2P2 [153] leverages prompt-tuning to automatically optimize prompts with limited labeled data for efficient adaptation to downstream tasks. Other studies [76, 142, 145, 169] exclusively focus on utilizing Tuning-Free Prompting to generate text. These approaches leverage the inherent capabilities of language models without any additional fine-tuning or parameter adjustments, thereby relying solely on the pre-trained knowledge embedded within the models to produce text outputs. For example, in TAPE [76], the initial text features are incorporated into a specialized prompt to interrogate a language model, generating a ranked list of predictions along with explanations. Subsequently, the expanded text features are utilized for finetuning on an LLM.

6.4 Discussion

To summarize, LLMs excel in capturing complex linguistic patterns and semantics from textual data, allowing the GNN+LLM-based models to generate embeddings that involve rich text, structure information, and even external knowledge of LLMs, thus leading to better model performance. Also, when integrated with GNN, LLM's reasoning capabilities over graphs may be enhanced. At the same time, these models can also be regarded as multimodal models to accomplish cross-modal tasks, such as text-graph retrieval tasks. The embeddings can be then utilized for a bunch of downstream tasks.

Also, it is challenging to align LLMs and GNNs into a common representational space. To tackle this problem, it is essential to establish a robust standard for measuring the alignment between LLM and GNN representations. This standard should evaluate the degree to which the embeddings from both models capture similar semantic and structural information. Additionally, it is crucial to design effective methods for achieving this alignment. By doing so, we can ensure that the combined model leverages the strengths of both LLMs and GNNs, ultimately enhancing performance on various downstream tasks. Moreover, existing work has begun to extend the GNN+LLM approach to heterogeneous graphs and hypergraphs. For heterogeneous graphs, HiGPT [172] introduces an in-context heterogeneous graph tokenizer and a heterogeneity-aware instruction-tuning framework to address distribution shifts, enhancing generalization and performance across various heterogeneous graph learning scenarios, and GHGRL [173] employs LLM to automatically summarize and classify different data formats and types of heterogeneous graph data. For hypergraph, HyperBERT [174] augments a pre-trained BERT model with specialized hypergraph-aware layers for the task of node classification.

7 Challenges and Future Directions

Although the previous sections have discussed the concepts and a lot of related works towards graph foundation models, there are still many avenues for future exploration in this research area. This section will delve into these issues.

7.1 Challenges about Data and Evaluation

7.1.1 Data Quantity and Quality

The improvements in data quantity and data quality are the key factors contributing to the effectiveness of foundation models [1]. At present, there is still a limited amount of open-source large-scale graph data [175, 176], and each dataset is primarily concentrated in a single domain. This poses a challenge to learn graph foundation models for diverse data domains. Hence, it is necessary to collect and organize a unified, massive dataset that covers

graph data and related corpora across different domains. Note that some works have constructed cross-domain datasets [177, 178], which aid in developing cross-domain graph foundation models. Additionally, if the graph data are noisy, incomplete, or not properly curated, it will negatively affect the performance of graph foundation models. To enhance the data quality of GFMs, efforts have been made to propose augmentation strategies from various perspectives, including graph structure learning, feature competion and label mixing, etc. However, since existing data augmentation techniques are typically tailored for individual GNN-based models, there is a need for further exploration on how to effectively augment graph data for LLM-based or GNN+LLM-based models.

7.1.2 Evaluation

With the help of natural language instructions and powerful generation capabilities, LLMs can support a variety of open-ended tasks [74]. This presents new opportunities for graph foundation models based on LLM. However, due to the lack of labels in open-ended tasks, evaluating the performance of GFMs in such tasks is a challenge. When using LLM as a language foundation model, the evaluation of its performance on open-ended tasks has evolved from human evaluation [127] to meta-evaluation [179]. The question of whether existing LLM evaluation methods [127, 179] can be applied to GFMs remains to be explored. Beyond evaluating the performance of GFMs, it is also worth evaluating their robustness, trustworthiness, or holistic performance, similar to the current practices for language foundation models [180, 181, 182].

7.2 Challenges about Models

7.2.1 Model Architectures

As mentioned above, the designs of backbone architectures and learning paradigms are crucial for the implementation of GFMs. Although this article has outlined some potential solutions, it does not rule out the possibility of better ones. For example, regarding the backbone architecture, recent works have proposed model architectures that go beyond the Transformer, offering improved performance [183] or interpretability [184]. However, it is still unknown whether these backbone architectures can be used for dealing with graph data. Additionally, when utilizing GNN+LLM-based models, it is worth exploring how to more effectively align the outputs of both models. Furthermore, there is limited research regarding the emergent abilities or neural scaling law [185, 186] of GNN-based [59] or LLM-based [60] graph foundation models. It is yet unclear whether GNN+LLM-based models may have greater potential for emergence. Furthermore, considering the diverse types of graphs (such as heterogeneous graphs [172], temporal graphs [137], and hypergraphs [174]), designing a GFM capable of handling multiple types of graphs is a valuable direction for future research. A potential solution is to use a mixture of experts (MoE) model [187], where each expert handles one type of graph. Finally, given that current multimodal foundation models [188] primarily handle text, images, audio, and other modalities, it is an interesting research direction to explore whether GNNs can be employed to further expand the diversity of modalities covered by multimodal foundation models or enhance the capabilities of foundation models for multimodal learning [189].

7.2.2 Model Training

In order to achieve homogeneity and make effective use of pre-training data, it is crucial to design appropriate pre-training tasks in pre-training. Unlike many language foundation models, which often use LM [127] or MLM [54] as pre-training tasks, there are now various forms of pre-training tasks tailored to different GFM model architectures. Whether each type of pre-training task has its own applicable scope and whether there will be a unified pre-training task are worth further exploration. Additionally, enabling graph foundation models to support cross-domain data is a vital concern. Some works use data from different domains as model input for pre-training [98, 190], or enable adaptation to data from different domains through methods such as LLM-based embedding [142], condition generation [191] and zero-shot transfer [172]. Finally, apart from fine-tuning and prompting that are introduced in this article, there are other potential training techniques that can be applied to improve efficiency or update knowledge, such as knowledge distillation [192], reinforcement learning from human feedback (RLHF) [127] and model editing [193]. Whether the above-mentioned techniques can be applied to graph foundation models will be a focal point of future research.

7.3 Challenges about Applications

7.3.1 Killer Applications

In comparison to the outstanding performance of language foundation models in tasks like text translation [194] and text generation [195], whether GFMs can similarly catalyze groundbreaking applications in graph tasks is not yet clear. For scenarios that are well-suited for the application of GNNs, such as e-commerce [196] and finance [197], potential research directions include leveraging graph-based models integrated with LLMs to better support open-ended tasks [164], or enhancing the reasoning capabilities of LLMs through graph learning techniques [198]. Furthermore, GFMs have the potential to make breakthroughs in some emerging fields. For example, drug development is a time-consuming and costly process [199], and language foundation models have already been successfully used for related tasks like target identification and side effect prediction [1]. Given the 3D geometric structure of proteins [200], GFMs hold the promise of enhancing the drug discovery pipeline by leveraging their ability to model graph structure information [201], potentially speeding up the process further. Additionally, urban computing may also represent a crucial application scenario for GFMs. It is worth noting that traditional traffic prediction techniques have been primarily focused on addressing individual tasks such as travel demand prediction [202] and traffic flow prediction [203], lacking a comprehensive understanding of the entire transportation system. Given that the transportation system can be viewed as a spatio-temporal graph, graph foundation models hold the potential to capture the participation behavior of actors in the transportation system [204], thereby offering a unified approach to addressing various issues in urban computing.

7.3.2 Trustworthiness

Despite the strong performance of LLM-based foundation models, their black-box nature [205] introduces a host of safety concerns, such as hallucination and privacy leaks. The hallucination refers to the output appearing plausible but deviating from user input, context, or facts [206]. Existing research suggests that this phenomenon is associated with multiple factors, such as the model's overconfidence in its own behavior [207] and the misunderstanding of false correlations [208]. Similarly, recent work has pointed out that pre-trained GNNs also pose certain trustworthy risks about

fairness [209] and robustness against attacks [210, 211]. Given the unique nature of graph data, we may require certain techniques to prevent or mitigate security risks on GFMs, such as confidence calibration [212] or counterfactual reasoning [213]. Additionally, given that existing research has indicated privacy risks in both GNN [214, 215] and LLM [216], enhancing the privacy of GFMs is also a critical concern. Some potential solutions include federated learning [217], RLHF [127] and red teaming [218], but whether these methods can be applied to GFMs is still unknown. Finally, graph data in real-world applications frequently encounter challenges such as noise [219], class imbalance [220], data incompleteness [221], and multi-modal features [222]. Developing methods to utilize these graph data for constructing GFMs, or adapting existing GFMs to accommodate these characteristics, will be a critical area of focus for future research.

8 Conclusions

The development of foundation models and graph machine learning has spurred the emergence of a new research direction, with the aim to train on broad graph data and apply it to a wide range of downstream graph tasks. In this article, we propose the concept of graph foundation models (GFMs) for the first time, and provide an introduction to relevant concepts and representative methods. We summarize existing works towards GFMs into three main categories based on their reliance on graph neural networks (GNNs) and large language models (LLMs): GNN-based models, LLM-based models, and GNN+LLM-based models. For each category of methods, we introduce their backbone architectures, pre-training, and adaptation strategies separately. After providing a comprehensive overview of the current landscape of graph foundation models, this article also points out the future directions for this evolving field.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No.U20B2045, 62192784, 62236003), Young Elite Scientists Sponsorship Program (No.2023QNRC001) by CAST, NSF under grants III-2106758 and POSE-2346158.

References

[1] R. Bommasani, D. A. Hudson, E. Adeli *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.
[2] J. Wei, Y. Tay, R. Bommasani *et al.*, "Emergent abilities of large language models," *TMLR*, 2022.
[3] W. Wang, Z. Chen, X. Chen *et al.*, "Visionllm: Large language model is also an open-ended decoder for vision-centric tasks," in *Proc. of NeurIPS*, vol. 36, 2023.
[4] H. Zhang, X. Li, and L. Bing, "Video-llama: An instruction-tuned audio-visual language model for video understanding," in *Proc. of EMNLP demo*, 2023, pp. 543–553.
[5] Z. Zhao, W. Fan, J. Li *et al.*, "Recommender systems in the era of large language models (llms)," *IEEE TKDE*, 2024.
[6] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. of KDD*, 2014.
[7] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. of KDD*, 2016.
[8] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE TPAMI*, 2006.
[9] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proc. of ICML*. PMLR, 2016, pp. 2071–2080.
[10] M. Nickel, V. Tresp, H.-P. Kriegel *et al.*, "A three-way model for collective learning on multi-relational data." in *Proc. of ICML*, vol. 11, no. 10.5555, 2011, pp. 3 104 482–3 104 584.
[11] C. Yang, Z. Liu, D. Zhao *et al.*, "Network representation learning with rich text information." in *Proc. of IJCAI*, 2015, pp. 2111–2117.
[12] J. Wang, Z. Zhang, Z. Shi, J. Cai, S. Ji, and F. Wu, "Duality-induced regularizer for semantic matching knowledge graph embeddings," *IEEE TPAMI*, vol. 45, no. 2, pp. 1652–1667, 2022.
[13] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.
[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. of ICLR*, 2017.
[15] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proc. of ICONIP*, 2018.
[16] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. of ICLR*, 2019.
[17] C. Wang, S. Pan, R. Hu *et al.*, "Attributed graph clustering: a deep attentional embedding approach," in *Proc. of IJCAI*, 2019.
[18] U. Alon and E. Yahav, "On the bottleneck of graph neural networks and its practical implications," in *Proc. of ICLR*, 2021.
[19] L. Yang, J. Zheng, H. Wang *et al.*, "Individual and structural graph information bottlenecks for out-of-distribution generalization," *IEEE TKDE*, 2023.
[20] C. Zhou, Q. Li, C. Li *et al.*, "A comprehensive survey on pretrained foundation models: A history from bert to chatgpt," *IJMLC*, pp. 1–65, 2024.
[21] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," *IEEE TKDE*, 2024.
[22] J. Z. Pan, S. Razniewski, J.-C. Kalo *et al.*, "Large language models and knowledge graphs: Opportunities and challenges," *TGDK*, pp. 1–38, 2023.
[23] Z. Zhang, H. Li, Z. Zhang, Y. Qin, X. Wang, and W. Zhu, "Graph meets llms: Towards large graph models," in *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*, 2023.
[24] Z. Wu, S. Pan, F. Chen *et al.*, "A comprehensive survey on graph neural networks," *IEEE TNNLS*, vol. 32, no. 1, pp. 4–24, 2020.
[25] S. Zafeiriou, M. Bronstein, T. Cohen *et al.*, "Guest editorial: Non-euclidean machine learning," *IEEE TPAMI*, pp. 723–726, 2022.
[26] L. Freeman, "The development of social network analysis," *A Study in the Sociology of Science*, 2004.
[27] G. Muzio, L. O'Bray, and K. Borgwardt, "Biological network analysis with deep learning," *Briefings in bioinformatics*, 2021.
[28] W. Jiang and J. Luo, "Graph neural network for traffic forecasting: A survey," *Expert Systems with Applications*, 2022.
[29] M. Li, S. Chen, X. Chen *et al.*, "Symbiotic graph neural networks for 3d skeleton-based human action recognition and motion prediction," *IEEE TPAMI*, pp. 3316–3333, 2021.
[30] B. Zhang, J. Xiao, J. Jiao, Y. Wei, and Y. Zhao, "Affinity attention graph neural network for weakly supervised semantic segmentation," *IEEE TPAMI*, 2021.
[31] J. Gao, T. Zhang, and C. Xu, "Learning to model relationships for zero-shot video classification," *IEEE TPAMI*, vol. 43, no. 10, pp. 3476–3491, 2020.
[32] S. Fan, X. Wang, C. Shi *et al.*, "Generalizing graph neural networks on out-of-distribution graphs," *IEEE TPAMI*, 2023.
[33] C. Shi, Y. Li, J. Zhang *et al.*, "A survey of heterogeneous information network analysis," *IEEE TKDE*, 2016.
[34] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *Proc. of AAAI*, 2019.
[35] R. Lu, Z. Cheng, B. Chen, and X. Yuan, "Motion-aware dynamic graph neural network for video compressive sensing," *IEEE TPAMI*, 2024.
[36] J. Gilmer, S. S. Schoenholz, P. F. Riley *et al.*, "Neural message passing for quantum chemistry," in *Proc. of ICML*, 2017.
[37] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Proc. of NeurIPS*, 2017.
[38] P. Velickovic, G. Cucurull, A. Casanova *et al.*, "Graph attention networks," in *Proc. of ICLR*, 2018.
[39] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc.*

of AAAI, 2018.
[40] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *Proc. of ICLR*, 2020.
[41] C. Ying, T. Cai, S. Luo *et al.*, "Do transformers really perform badly for graph representation?" *Proc. of NeurIPS*, 2021.
[42] D. Chen, L. O'Bray, and K. Borgwardt, "Structure-aware transformer for graph representation learning," in *Proc. of ICML*, 2022.
[43] D. Kreuzer, D. Beaini, W. Hamilton, V. Létourneau, and P. Tossou, "Rethinking graph transformers with spectral attention," in *Proc. of NeurIPS*, 2021.
[44] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a general, powerful, scalable graph transformer," *Proc. of NeurIPS*, 2022.
[45] J. B. Lee, R. Rossi, and X. Kong, "Graph classification using structural attention," in *Proc. of KDD*, 2018.
[46] X. Jiang, P. Ji, and S. Li, "Censnet: convolution with edge-node switching in graph neural networks," in *Proc. of IJCAI*, 2019.
[47] O. Wieder, S. Kohlbacher, M. Kuenemann *et al.*, "A compact review of molecular property prediction with graph neural networks," *Drug Discovery Today: Technologies*, 2020.
[48] Z. Song, X. Yang, Z. Xu, and I. King, "Graph-based semi-supervised learning: A comprehensive review," *IEEE TNNLS*, 2022.
[49] C. Yang, J. Liu, and C. Shi, "Extract the knowledge of graph neural networks and go beyond it: An effective knowledge distillation framework," in *Proc. of WWW*, 2021.
[50] Y. Liu, M. Jin, S. Pan *et al.*, "Graph self-supervised learning: A survey," *IEEE TKDE*, 2022.
[51] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *Science China Technological Sciences*, 2020.
[52] P. Liu, W. Yuan, J. Fu *et al.*, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM CSUR*, 2023.
[53] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proc. of ACL*, 2018.
[54] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proc. of NAACL*, 2019.
[55] Z. Liu, X. Yu, Y. Fang, and X. Zhang, "Graphprompt: Unifying pre-training and downstream tasks for graph neural networks," in *Proc. of WWW*, 2023.
[56] X. Sun, H. Cheng, J. Li *et al.*, "All in one: Multi-task prompting for graph neural networks," in *Proc. of KDD*, 2023.
[57] Q. Dong, L. Li, D. Dai *et al.*, "A survey on in-context learning," in *Proc. of EMNLP*, 2024, pp. 1107–1128.
[58] J. Wei, X. Wang, D. Schuurmans *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," in *Proc. of NeurIPS*, 2022.
[59] Q. Huang, H. Ren, P. Chen *et al.*, "PRODIGY: Enabling in-context learning over graphs," in *Proc. of NeurIPS*, 2023.
[60] H. Wang, S. Feng, T. He *et al.*, "Can language models solve graph problems in natural language?" in *Proc. of NeurIPS*, vol. 36, 2023.
[61] B. Su, D. Du, Z. Yang *et al.*, "A molecular multimodal foundation model associating molecule graphs with natural language," *arXiv preprint arXiv:2209.05481*, 2022.
[62] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Deep Graph Contrastive Representation Learning," in *ICML Workshop on Graph Representation Learning and Beyond*, 2020.
[63] M. Sun, J. Xing, H. Wang, B. Chen, and J. Zhou, "Mocl: data-driven molecular fingerprint via knowledge-aware contrastive learning from molecular graph," in *Proc. of KDD*, 2021.
[64] Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, and J. Tang, "Graphmae: Self-supervised masked graph autoencoders," in *Proc. of KDD*, 2022.
[65] Z. Hou, Y. He, Y. Cen, X. Liu, Y. Dong, E. Kharlamov, and J. Tang, "Graphmae2: A decoding-enhanced masked self-supervised graph learner," in *Proc. of WWW*, 2023.
[66] S. Li, X. Han, and J. Bai, "Adaptergnn: Parameter-efficient fine-tuning improves generalization in gnns," in *Proc. of AAAI*, vol. 38, no. 12, 2024, pp. 13 600–13 608.
[67] A. Gui, J. Ye, and H. Xiao, "G-adapter: Towards structure-aware parameter-efficient transfer learning for graph transformer networks," in *Proc. of AAAI*, 2024.
[68] M. Sun, K. Zhou, X. He, Y. Wang, and X. Wang, "Gppt: Graph pre-training and prompt tuning to generalize graph neural networks," in *Proc. of KDD*, 2022.
[69] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proc. of ACL*, 2021.
[70] L. Sun, Z. Zhang, J. Ye *et al.*, "A self-supervised mixed-curvature graph neural network," in *Proc. of AAAI*, 2022.
[71] J. Ye, Z. Zhang, L. Sun, Y. Yan, F. Wang, and F. Ren, "Sincere: sequential interaction networks representation learning on co-evolving riemannian manifolds," in *Proc. of WWW*, 2023.
[72] Q. Lhoest, A. V. del Moral, Y. Jernite *et al.*, "Datasets: A community library for natural language processing," in *Proc. of EMNLP*, 2021.
[73] T. Brown, B. Mann, N. Ryder *et al.*, "Language models are few-shot learners," *Proc. of NeurIPS*, 2020.
[74] H. Touvron, T. Lavril, G. Izacard *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.
[75] L. Sun, Z. Huang, Z. Wang, F. Wang, H. Peng, and S. Y. Philip, "Motif-aware riemannian graph neural network with generative-contrastive learning," in *Proc. of AAAI*, 2024.
[76] X. He, X. Bresson, T. Laurent *et al.*, "Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning," in *Proc. of ICLR*, 2024.
[77] C. Raffel, N. Shazeer, A. Roberts *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *JMLR*, 2020.
[78] D. Zhou, L. Zheng, D. Fu *et al.*, "Mentorgnn: Deriving curriculum for pre-training gnns," in *Proc. of CIKM*, 2022.
[79] J. Li, D. Li, S. Savarese, and S. C. H. Hoi, "BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models," in *Proc. of ICML*, 2023.
[80] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *Proc. of ICLR*, 2019.
[81] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *NIPS Workshop on Bayesian Deep Learning*, 2016.
[82] X. Gong, C. Yang, and C. Shi, "Ma-gcl: Model augmentation tricks for graph contrastive learning," in *Proc. of AAAI*, 2023.
[83] Z. Wen, Y. Fang, Y. Liu, Y. Guo, and S. Hao, "Voucher abuse detection with prompt-based fine-tuning on graph neural networks," in *Proc. of CIKM*, 2023.
[84] J. Qiu, Q. Chen, Y. Dong *et al.*, "Gcc: Graph contrastive coding for graph neural network pre-training," in *Proc. of KDD*, 2020.
[85] Y. You, T. Chen, Y. Sui *et al.*, "Graph contrastive learning with augmentations," *Proc. of NeurIPS*, 2020.
[86] Y. Guo, C. Yang, Y. Chen, J. Liu, C. Shi, and J. Du, "A data-centric framework to endow graph neural networks with out-of-distribution detection ability," in *Proc. of KDD*, 2023.
[87] T. Fang, Y. M. Zhang, Y. Yang *et al.*, "Universal prompt tuning for graph neural networks," in *Proc. of NeurIPS*, 2023.
[88] B. Weisfeiler and A. Leman, "The reduction of a graph to canonical form and the algebra which appears therein," *nti, Series*, 1968.
[89] L. Wu, H. Lin, C. Tan, Z. Gao, and S. Z. Li, "Self-supervised learning on graphs: Contrastive, generative, or predictive," *IEEE TKDE*, 2021.
[90] Y. Xie, Z. Xu, J. Zhang, Z. Wang, and S. Ji, "Self-supervised learning of graph neural networks: A unified review," *IEEE TPAMI*, 2022.
[91] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Proc. of NeurIPS*, 2017.
[92] A. Dosovitskiy, L. Beyer, A. Kolesnikov *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. of ICLR*, 2021.
[93] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. of ICCV*, 2021.
[94] J. Kim, D. Nguyen, S. Min, S. Cho, M. Lee, H. Lee, and S. Hong, "Pure transformers are powerful graph learners," in *Proc. of NeurIPS*, 2022.
[95] J. Zhang, H. Zhang, C. Xia, and L. Sun, "Graph-bert: Only attention is needed for learning graph representations," *arXiv preprint arXiv:2001.05140*, 2020.
[96] Y. Rong, Y. Bian, T. Xu *et al.*, "Self-supervised graph transformer on large-scale molecular data," *Proc. of NeurIPS*, 2020.
[97] L. Müller, M. Galkin, C. Morris, and L. Rampásek, "Attending

to graph transformers," *TMLR*, 2024.
[98] H. Zhao, A. Chen, X. Sun, H. Cheng, and J. Li, "All in one and one for all: A simple yet effective method towards cross-domain graph pretraining," in *Proc. of KDD*, 2024, pp. 4443–4454.
[99] J. Zhao, D. Jin, M. Ge *et al.*, "Fug: Feature-universal graph contrastive pre-training for graphs with diverse node features," in *Proc. of NeurIPS*, 2024.
[100] Z. Hu, Y. Dong, K. Wang *et al.*, "Gpt-gnn: Generative pre-training of graph neural networks," in *Proc. of KDD*, 2020.
[101] Y. Sun, Q. Zhu, Y. Yang, C. Wang, T. Fan, J. Zhu, and L. Chen, "Fine-tuning graph neural networks by preserving graph generative patterns," in *Proc. of AAAI*, 2024, pp. 9053–9061.
[102] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proc. of EMNLP*, 2021.
[103] X. Liu, K. Ji, Y. Fu, W. Tam, Z. Du, Z. Yang, and J. Tang, "P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks," in *Proc. of ACL*, 2022, pp. 61–68.
[104] Y. Yan, P. Zhang, Z. Fang, and Q. Long, "Inductive graph alignment prompt: Bridging the gap between graph pre-training and inductive fine-tuning from spectral perspective," in *Proc. of WWW*, 2024, pp. 4328–4339.
[105] C. Niu, G. Pang, L. Chen, and B. Liu, "Replay-and-forget-free graph class-incremental learning: A task profiling and prompting approach," in *Proc. of NeurIPS*, 2024.
[106] X. Yu, Z. Liu, Y. Fang, Z. Liu, S. Chen, and X. Zhang, "Generalized graph prompt: Toward a unification of pre-training and downstream tasks on graphs," *IEEE TKDE*, 2024.
[107] X. Yu, J. Zhang, Y. Fang, and R. Jiang, "Non-homophilic graph pre-training and prompt learning," in *Proc. of KDD*, vol. 34, 2025, pp. 22 667–22 681.
[108] X. Jiang, Y. Lu, Y. Fang, and C. Shi, "Contrastive pre-training of gnns on heterogeneous graphs," in *Proc. of CIKM*, 2021.
[109] X. Jiang, T. Jia, Y. Fang *et al.*, "Pre-training on large-scale heterogeneous graph," in *Proc. of KDD*, 2021.
[110] X. Yu, C. Zhou, Y. Fang, and X. Zhang, "Multigprompt for multi-task pre-training and prompting on graphs," in *Proc. of WWW*, 2024, pp. 515–526.
[111] X. Yu, Y. Fang, Z. Liu, and X. Zhang, "Hgprompt: Bridging homogeneous and heterogeneous graphs for few-shot prompt learning," in *Proc. of AAAI*, 2024.
[112] Y. Deng, R. Zhang, P. Xu, J. Ma, and Q. Gu, "Pre-trained hypergraph convolutional neural networks with self-supervised learning," *TMLR*, 2024.
[113] M. Yang, Z. Liu, L. Yang *et al.*, "Instruction-based hypergraph pretraining," in *Proc. of SIGIR*, 2024.
[114] Q. Zhang, C. Huang, L. Xia *et al.*, "Spatial-temporal graph learning with adversarial contrastive adaptation," in *Proc. of ICML*, 2023.
[115] Z. Li, L. Xia, Y. Xu, and C. Huang, "Gpt-st: Generative pre-training of spatio-temporal graph neural networks," in *Proc. of NeurIPS*, 2023.
[116] R. Ye, C. Zhang, R. Wang, S. Xu, and Y. Zhang, "Language is all a graph needs," in *Proc. of EACL*, 2024.
[117] J. Guo, L. Du, and H. Liu, "Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking," *arXiv preprint arXiv:2305.15066*, 2023.
[118] Z. Chen, H. Mao, H. Li *et al.*, "Exploring the potential of large language models (llms) in learning on graphs," *ACM SIGKDD Explorations Newsletter*, vol. 25, no. 2, pp. 42–61, 2024.
[119] H. Zhao, S. Liu, M. Chang *et al.*, "Gimlet: A unified graph-text model for instruction-based molecule zero-shot learning," in *Proc. of NeurIPS*, vol. 36, 2023.
[120] Y. Zhang, K. Gong, K. Zhang, H. Li, Y. Qiao, W. Ouyang, and X. Yue, "Meta-transformer: A unified framework for multimodal learning," *arXiv preprint arXiv:2307.10802*, 2023.
[121] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," in *Proc. of ICLR*, 2021.
[122] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proc. of EMNLP*, 2019.
[123] OpenAI, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.
[124] F. Wenkel, G. Wolf, and B. Knyazev, "Pretrained language models to solve graph tasks in natural language," in *ICML 2023 Workshop on Structured Probabilistic Inference {\&} Generative Modeling*, 2023.
[125] J. Huang, X. Zhang, Q. Mei, and J. Ma, "Can llms effectively leverage graph structural information: when and why," *NeurIPS2023 workshop*, 2023.
[126] N. Chen, Y. Li, J. Tang, and J. Li, "Graphwiz: An instruction-following language model for graph problems," in *Proc. of KDD*, 2024.
[127] L. Ouyang, J. Wu, X. Jiang *et al.*, "Training language models to follow instructions with human feedback," in *Proc. of NeurIPS*, 2022.
[128] A. Antonucci, G. Piqué, and M. Zaffalon, "Zero-shot causal graph extrapolation from text via llms," *AAAI 2024 XAI4Sci workshop*, 2024.
[129] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Proc. of NeurIPS*, 2019.
[130] M. Lewis, Y. Liu, N. Goyal *et al.*, "BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proc. of ACL*, 2020.
[131] N. Saunshi, O. Plevrakis, S. Arora, M. Khodak, and H. Khandeparkar, "A theoretical analysis of contrastive unsupervised representation learning," in *Proc. of ICML*, 2019.
[132] C. Liu and B. Wu, "Evaluating large language models on graphs: Performance insights and comparative analysis," *arXiv preprint arXiv:2308.11224*, 2023.
[133] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, "How can we know what language models know?" *TACL*, 2020.
[134] C. Poth, J. Pfeiffer, A. Rücklé, and I. Gurevych, "What to pre-train on? efficient intermediate task selection," in *Proc. of EMNLP*, 2021.
[135] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," in *Proc. of ACL*, 2019.
[136] N. Houlsby, A. Giurgiu, S. Jastrzebski *et al.*, "Parameter-efficient transfer learning for nlp," in *Proc. of ICML*, 2019.
[137] Z. Zhang, X. Wang, Z. Zhang, H. Li, Y. Qin, S. Wu, and W. Zhu, "Llm4dyg: Can large language models solve problems on dynamic graphs?" in *Proc. of KDD*, 2024.
[138] C. Mavromatis, V. N. Ioannidis, S. Wang *et al.*, "Train your own GNN teacher: Graph-aware distillation on textual graphs," in *Proc. of KDD*, 2023.
[139] E. Chien, W. Chang, C. Hsieh, H. Yu, J. Zhang, O. Milenkovic, and I. S. Dhillon, "Node feature extraction by self-supervised multi-scale neighborhood prediction," in *Proc. of ICLR*, 2022.
[140] R. Lei, Y. Hu, Y. Ren, and Z. Wei, "Intruding with words: Towards understanding graph injection attacks at the text level," in *Proc. of NeurIPS*, 2024.
[141] H. Xie, D. Zheng, J. Ma *et al.*, "Graph-aware language model pre-training on a large graph corpus can help multiple graph applications," in *Proc. of KDD*, 2023.
[142] H. Liu, J. Feng, L. Kong, N. Liang, D. Tao, Y. Chen, and M. Zhang, "One for all: Towards training one graph model for all classification tasks," in *Proc. of ICLR*, 2024.
[143] B. Jin, Y. Zhang, Q. Zhu, and J. Han, "Heterformer: Transformer-based deep node representation learning on heterogeneous text-rich networks," in *Proc. of KDD*, 2023.
[144] B. Jin, Y. Zhang, Y. Meng, and J. Han, "Edgeformers: Graph-empowered transformers for representation learning on textual-edge networks," in *Proc. of ICLR*, 2023.
[145] W. Wei, X. Ren, J. Tang *et al.*, "Llmrec: Large language models with graph augmentation for recommendation," in *Proc. of WSDM*, 2024, pp. 806–815.
[146] Y. Tan, Z. Zhou, H. Lv, W. Liu, and C. Yang, "Walklm: A uniform language model fine-tuning framework for attributed graph embedding," in *Proc. of NeurIPS*, 2023.
[147] J. Zhu, X. Song, V. Ioannidis, D. Koutra, and C. Faloutsos, "Touchup-g: Improving feature representation through graph-centric finetuning," in *Proc. of SIGIR*, 2024, pp. 2662–2666.
[148] B. Jin, W. Zhang, Y. Zhang, Y. Meng, H. Zhao, and J. Han, "Learning multiplex embeddings on text-rich networks with one text encoder," in *NeurIPS Workshop*, 2023.
[149] S. Sun, Y. Ren, C. Ma, and X. Zhang, "Large language models as topological structure enhancers for text-attributed graphs," *arXiv preprint arXiv:2311.14324*, 2023.
[150] Z. Chen, H. Mao, H. Wen, H. Han, W. Jin, H. Zhang, H. Liu, and J. Tang, "Label-free node classification on graphs with

large language models (llms)," in *Proc. of ICLR*, 2024.
[151] J. Yang, Z. Liu, S. Xiao *et al.*, "Graphformers: Gnn-nested transformers for representation learning on textual graph," *Proc. of NeurIPS*, 2021.
[152] J. Zhao, M. Qu, C. Li, H. Yan, Q. Liu, R. Li, X. Xie, and J. Tang, "Learning on large-scale text-attributed graphs via variational inference," in *Proc. of ICLR*, 2023.
[153] Z. Wen and Y. Fang, "Augmenting low-resource text classification with graph-grounded pre-training and prompting," in *Proc. of SIGIR*, 2023, pp. 506–516.
[154] X. Huang, K. Han, Y. Yang *et al.*, "Can gnn be good adapter for llms?" in *Proc. of WWW*, 2024, pp. 893–904.
[155] Y. Zhu, Y. Wang, H. Shi, and S. Tang, "Efficient tuning and inference for large language models on textual graphs," in *Proc. of IJCAI*, 2024.
[156] B. Jin, W. Zhang, Y. Zhang, Y. Meng, X. Zhang, Q. Zhu, and J. Han, "Patton: Language model pretraining on text-rich networks," in *Proc. of ACL*, 2023, pp. 7005–7020.
[157] L. Xia, B. Kao, and C. Huang, "Opengraph: Towards open graph foundation models," in *Findings of EMNLP*, 2024, pp. 2365–2379.
[158] X. Ren, W. Wei, L. Xia *et al.*, "Representation learning with large language models for recommendation," in *Proc. of WWW*. ACM, 2024, pp. 3464–3475.
[159] C. Edwards, C. Zhai, and H. Ji, "Text2mol: Cross-modal molecule retrieval with natural language queries," in *Proc. of EMNLP*, 2021.
[160] S. Liu, W. Nie, C. Wang *et al.*, "Multi-modal molecule structure–text model for text-based retrieval and editing," *NMI*, vol. 5, no. 12, pp. 1447–1457, 2023.
[161] P. Seidl, A. Vall, S. Hochreiter, and G. Klambauer, "Enhancing activity prediction models in drug discovery with the ability to understand human language," in *Proc. of ICML*, 2023.
[162] Z. Liu, S. Li, Y. Luo *et al.*, "Molca: Molecular graph-language modeling with cross-modal projector and uni-modal adapter," in *Proc. of EMNLP*, 2023, pp. 15 623–15 638.
[163] P. Liu, Y. Ren, J. Tao, and Z. Ren, "Git-mol: A multi-modal large language model for molecular science with graph, image, and text," *Comput. Biol. Medicine*, vol. 171, p. 108073, 2024.
[164] M. Zhang, M. Sun, P. Wang *et al.*, "Graphtranslator: Aligning graph model to large language model for open-ended tasks," in *Proc. of WWW*, 2024, pp. 1003–1014.
[165] J. Tang, Y. Yang, W. Wei, L. Shi, L. Su, S. Cheng, D. Yin, and C. Huang, "Graphgpt: Graph instruction tuning for large language models," in *Proc. of SIGIR*, 2024.
[166] T. Zou, L. Yu, Y. Huang, L. Sun, and B. Du, "Pretraining language models with text-attributed heterogeneous graphs," in *Findings of EMNLP*, 2023, pp. 10 316–10 333.
[167] Z. Liu, X. He, Y. Tian, and N. V. Chawla, "Can we soft prompt llms for graph learning tasks?" in *Proc. of WWW*, 2024.
[168] J. Wang, J. Wu, Y. Wu *et al.*, "Instructgraph: Boosting large language models via graph-centric instruction tuning and preference alignment," in *Findings of ACL*, 2024.
[169] Y. Shi, A. Zhang, E. Zhang, Z. Liu, and X. Wang, "Relm: Leveraging language models for enhanced chemical reaction prediction," in *Findings of EMNLP*, 2023, pp. 5506–5520.
[170] D. Wang, Y. Zuo, F. Li, and J. Wu, "Llms as zero-shot graph learners: Alignment of GNN representations with LLM token embeddings," in *Proc. of NeurIPS*, 2024.
[171] X. He, Y. Tian, Y. Sun *et al.*, "G-retriever: Retrieval-augmented generation for textual graph understanding and question answering," in *Proc. of NeurIPS*, 2024.
[172] J. Tang, Y. Yang, W. Wei, L. Shi, L. Xia, D. Yin, and C. Huang, "Higpt: Heterogeneous graph language model," in *Proc. of KDD*, 2024, pp. 2842–2853.
[173] H. Gao, C. Zhang, F. Wu, J. Zhao, C. Zheng, and H. Liu, "Bootstrapping heterogeneous graph representation learning via large language models: A generalized approach," in *Proc. of AAAI*, 2025.
[174] A. Bazaga, P. Liò, and G. Micklem, "Hyperbert: Mixing hypergraph-aware layers with language models for node classification on text-attributed hypergraphs," in *Findings of EMNLP*, 2024.
[175] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *Proc. of NeurIPS*, 2020.
[176] A. Khatua, V. S. Mailthody, B. Taleka, T. Ma, X. Song, and W.-m. Hwu, "Igb: Addressing the gaps in labeling, features, heterogeneity, and size of public graph datasets for deep learning research," in *Proc. of KDD*, 2023, pp. 4284–4295.
[177] Z. Li, Z. Gou, X. Zhang, Z. Liu, S. Li, Y. Hu, C. Ling, Z. Zhang, and L. Zhao, "Teg-db: A comprehensive dataset and benchmark of textual-edge graphs," in *Proc. of NeurIPS*, 2024.
[178] X. Li, W. Chen, Q. Chu, H. Li, Z. Sun, R. Li, C. Qian, Y. Wei, Z. Liu, C. Shi *et al.*, "Can large language models analyze graphs like professionals? a benchmark, datasets and models," in *Proc. of NeurIPS*, 2024.
[179] Z. Zeng, J. Yu, T. Gao, Y. Meng, T. Goyal, and D. Chen, "Evaluating large language models at evaluating instruction following," in *Proc. of ICLR*, 2024.
[180] B. Wang, C. Xu, S. Wang *et al.*, "Adversarial glue: A multi-task benchmark for robustness evaluation of language models," in *Proc. of NeurIPS*, 2021.
[181] B. Wang, W. Chen, H. Pei *et al.*, "Decodingtrust: A comprehensive assessment of trustworthiness in gpt models," in *Proc. of NeurIPS*, 2023.
[182] R. Bommasani, P. Liang, and T. Lee, "Holistic evaluation of language models," *Annals of the New York Academy of Sciences*, 2023.
[183] T. Dao and A. Gu, "Transformers are ssms: Generalized models and efficient algorithms through structured state space duality," in *Proc. of ICML*, 2024.
[184] Y. Yu, S. Buchanan, D. Pai, T. Chu, Z. Wu, S. Tong, B. Haeffele, and Y. Ma, "White-box transformers via sparse rate reduction," in *Proc. of NeurIPS*, vol. 36, 2024.
[185] Z. Wang, Y. Li, B. Ding, Y. Li, and Z. Wei, "Exploring neural scaling law and data pruning methods for node classification on large-scale graphs," in *Proc. of WWW*, 2024, pp. 780–791.
[186] H. Mao, Z. Chen, W. Tang *et al.*, "Position: Graph foundation models are already here," in *Proc. of ICML*, 2024.
[187] H. Wang, Z. Jiang, Y. You *et al.*, "Graph mixture of experts: learning on large-scale graphs with explicit diversity modeling," in *Proc. of NeurIPS*, 2023, pp. 50 825–50 837.
[188] N. Fei, Z. Lu, Y. Gao *et al.*, "Towards artificial general intelligence via a multimodal foundation model," *Nature Communications*, 2022.
[189] Y. Yuan, "On the power of foundation models," in *Proc. of ICML*. PMLR, 2023, pp. 40 519–40 530.
[190] A. Davies, R. Green, N. Ajmeri, and T. S. Filho, "Its all graph to me: Single-model graph representation learning on multiple domains," in *NeurIPS Workshop*, 2023.
[191] Y. Zhu, Y. Wang, H. Shi, Z. Zhang, D. Jiao, and S. Tang, "Graphcontrol: Adding conditional control to universal graph pre-trained models for graph domain transfer learning," in *Proc. of WWW*, 2024, pp. 539–550.
[192] Y. Jiang, C. Chan, M. Chen, and W. Wang, "Lion: Adversarial distillation of proprietary large language models," in *Proc. of EMNLP*, 2023, pp. 3134–3154.
[193] Y. Yao, P. Wang, B. Tian, S. Cheng, Z. Li, S. Deng, H. Chen, and N. Zhang, "Editing large language models: Problems, methods, and opportunities," in *Proc. of EMNLP*, 2023.
[194] A. Hendy, M. Abdelrehim, A. Sharaf *et al.*, "How good are gpt models at machine translation? a comprehensive evaluation," *arXiv preprint arXiv:2302.09210*, 2023.
[195] H. Zhang, H. Song, S. Li, M. Zhou, and D. Song, "A survey of controllable text generation using transformer-based pre-trained language models," *ACM CSUR*, 2023.
[196] D. Zhang, X. Huang, Z. Liu *et al.*, "AGL: A scalable system for industrial-purpose graph machine learning," *Proc. VLDB Endow.*, 2020.
[197] J. Wang, S. Zhang, Y. Xiao, and R. Song, "A review on graph neural network methods in financial applications," *Journal of Data Science*, vol. 20, no. 2, pp. 111–134, 2022.
[198] J. Yu, R. He, and Z. Ying, "Thought propagation: An analogical approach to complex reasoning with large language models," in *Proc. of ICLR*, 2024.
[199] O. J. Wouters, M. McKee, and J. Luyten, "Estimated research and development investment needed to bring a new medicine to market, 2009-2018," *Jama*, 2020.
[200] S. Liu, H. Wang, W. Liu *et al.*, "Pre-training molecular graph representation with 3d geometry," in *Proc. of ICLR*, 2022.
[201] J. Xia, Y. Zhu, Y. Du, and S. Z. Li, "A systematic survey of chemical pre-trained models," in *Proc. of IJCAI*, 2023.
[202] D. Zhuang, S. Wang, H. Koutsopoulos *et al.*, "Uncertainty

quantification of sparse travel demand prediction with spatial-temporal graph neural networks," in *Proc. of KDD*, 2022.
[203] X. Wang, Y. Ma, Y. Wang, W. Jin, X. Wang, J. Tang, C. Jia, and J. Yu, "Traffic flow prediction via spatial temporal graph neural network," in *Proc. of WWW*, 2020.
[204] X. Wang, D. Wang, L. Chen, F.-Y. Wang, and Y. Lin, "Building transportation foundation model via generative graph transformer," in *Proc. of ITSC*. IEEE, 2023, pp. 6042–6047.
[205] T. Sun, Y. Shao, H. Qian *et al.*, "Black-box tuning for language-model-as-a-service," in *Proc. of ICML*, 2022.
[206] Y. Zhang, Y. Li, L. Cui *et al.*, "Siren's song in the ai ocean: A survey on hallucination in large language models," *arXiv preprint arXiv:2309.01219*, 2023.
[207] R. Ren, Y. Wang, Y. Qu *et al.*, "Investigating the factual knowledge boundary of large language models with retrieval augmentation," *arXiv preprint arXiv:2307.11019*, 2023.
[208] S. Li, X. Li, L. Shang *et al.*, "How pre-trained language models capture factual knowledge? a causal-inspired analysis," in *Findings of ACL*, 2022, pp. 1720–1732.
[209] Z. Zhang, M. Zhang, Y. Yu, C. Yang, J. Liu, and C. Shi, "Endowing pre-trained graph models with provable fairness," in *Proc. of WWW*, 2024, pp. 1045–1056.
[210] Z. Zhang, M. Chen, M. Backes, Y. Shen, and Y. Zhang, "Inference attacks against graph neural networks," in *USENIX Security*, 2022, pp. 4543–4560.
[211] Z. Zhang, X. Wang, H. Zhou, Y. Yu, M. Zhang, C. Yang, and C. Shi, "Can large language models improve the adversarial robustness of graph neural networks?" in *Proc. of KDD*, 2025.
[212] X. Wang, H. Liu, C. Shi, and C. Yang, "Be confident! towards trustworthy graph neural networks via confidence calibration," *Proc. of NeurIPS*, 2021.
[213] J. Tan, S. Geng, Z. Fu *et al.*, "Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning," in *Proc. of WWW*, 2022.
[214] C. Wu, F. Wu, L. Lyu, T. Qi, Y. Huang, and X. Xie, "A federated graph neural network framework for privacy-preserving personalization," *Nature Communications*, 2022.
[215] B. Yan, Y. Cao, H. Wang, W. Yang, J. Du, and C. Shi, "Federated heterogeneous graph neural network for privacy-preserving recommendation," in *Proc. of WWW*, 2024.
[216] R. Staab, M. Vero, M. Balunovic, and M. Vechev, "Beyond memorization: Violating privacy via inference with large language models," in *Proc. of ICLR*, 2024.
[217] H. Chen, T. Zhu, T. Zhang, W. Zhou, and P. S. Yu, "Privacy and fairness in federated learning: on the perspective of trade-off," *ACM CSUR*, 2023.
[218] Z. Shi, Y. Wang, F. Yin, X. Chen, K.-W. Chang, and C.-J. Hsieh, "Red teaming language model detectors with language models," *TACL*, vol. 12, pp. 174–189, 2024.
[219] B. Fatemi, L. El Asri, and S. M. Kazemi, "Slaps: Self-supervision improves structure learning for graph neural networks," in *Proc. of NeurIPS*, vol. 34, 2021, pp. 22 667–22 681.
[220] L. Zeng, L. Li, Z. Gao, P. Zhao, and J. Li, "Imgcl: Revisiting graph contrastive learning on imbalanced node classification," in *Proc. of AAAI*, vol. 37, no. 9, 2023, pp. 11 138–11 146.
[221] W. Tu, B. Xiao, X. Liu, S. Zhou, Z. Cai, and J. Cheng, "Revisiting initializing then refining: an incomplete and missing graph imputation network," *IEEE TNNLS*, 2024.
[222] Z. Lian, L. Chen, L. Sun, B. Liu, and J. Tao, "Gcnet: Graph completion network for incomplete multimodal learning in conversation," *IEEE TPAMI*, vol. 45, no. 7, pp. 8419–8432, 2023.
[223] X. Wang, H. Ji, C. Shi *et al.*, "Heterogeneous graph attention network," in *Proc. of WWW*, 2019.
[224] J. M. Thomas, A. Moallemy-Oureh, S. Beddar-Wiesing, and C. Holzhüter, "Graph neural networks designed for different graph types: A survey," *TMLR*, 2023.
[225] H. Yang, "Aligraph: A comprehensive graph neural network platform," in *Proc. of KDD*, 2019.
[226] V. P. Dwivedi, L. Rampášek, M. Galkin *et al.*, "Long range graph benchmark," in *Proc. of NeurIPS*, 2022.
[227] R. Zhu, X. Jiang, J. Lu, and S. Li, "Cross-domain graph convolutions for adversarial unsupervised domain adaptation," *IEEE TNNLS*, 2021.
[228] J. You, Z. Ying, and J. Leskovec, "Design space for graph neural networks," *Proc. of NeurIPS*, 2020.
[229] P. Li, Y. Wang, H. Wang, and J. Leskovec, "Distance encoding: Design provably more powerful neural networks for graph representation learning," *Proc. of NeurIPS*, 2020.
[230] X. Zou, X. Zhao, P. Liò, and Y. Zhao, "Will more expressive graph neural networks do better on generative tasks?" in *Proc. of LoG*. PMLR, 2024, pp. 21–1.

Jiawei Liu received the B.S. degree in computer science and technology from Beijing University of Posts and Telecommunications, Beijing, China, in 2020. He is currently pursuing the Ph.D. degree in computer science and technology from Beijing University of Posts and Telecommunications, Beijing, China. His research interests include graph data mining and machine learning.

Cheng Yang is an Associate Professor of Computer Science at Beijing University of Posts and Telecommunications (BUPT). He received his Bachelor and Ph.D. degrees from Tsinghua University in 2014 and 2019, respectively. Cheng's research interests include data mining, natural language processing and social computing. He has published 60+ papers in top journals and conferences, such as NeurIPS, ICLR, KDD and ACL.

Zhiyuan Lu received the BS degree in communication engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2022. He is currently pursuing the PhD degree in computer science and technology from Beijing University of Posts and Telecommunications, Beijing, China. His current research interests are in graph neural networks, data mining and machine learning.

Junze Chen is a master's student at the School of Computer Science, Beijing University of Posts and Telecommunications (BUPT). He obtained his Bachelor's degree in Computer Science and Technology from BUPT in 2022. His primary research interests are in data mining, graph neural networks, and natural language processing.

Yibo Li received the B.S. degree from the Beijing University of Posts and Telecommunications, China, in 2022. She is currently working toward the master's degree with the Beijing University of Posts and Communications, China. Her current research interests are in graph neural networks and large language models.

Mengmei Zhang received her Ph.D. degree in computer science and technology from Beijing University of Posts and Telecommunications in 2023. She is currently a senior researcher at China Telecom Bestpay. Her research interests include graph mining, large language models, and risk control.

Ting Bai received her Ph.D. degree from Renmin University of China in 2019. She currently is an associate professor in the School of Computer Science, Beijing University of Posts and Telecommunications. Her major research interests are in recommender systems and Human behavior analysis. She has published several papers on SIGIR, WWW, KDD, CIKM, WSDM, TKDE, and so on.

Yuan Fang received the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign in 2014. He is currently an assistant professor with the School of Computing and Information Systems, Singapore Management University. His current research focuses on graph data mining, machine learning and their applications.

Lichao Sun is an Assistant Professor in Computer Science and Engineering at Lehigh University. He obtained his Ph.D. degree in Computer Science at University of Illinois Chicago in 2020. His research interests are Trustworthy AI and Medical AI in various applications. He have published more than 90 papers in top-tier journals and conferences, such as Nature Medicine, NeurIPS, ICML, ICLR, AAAI and IJCAI.

Philip S. Yu is a Distinguished Professor at the University of Illinois Chicago, holding the Wexler Chair in information and Technology. He is a Fellow of the ACM and IEEE. He has published over 2,000 papers and holds or has applied for over 300 US patents. His main research interests include big data, data mining, privacy preserving publishing and mining, data streams, database systems, Internet applications and technologies.

Chuan Shi received the Ph.D. degree from the ICT of Chinese Academic of Sciences in 2007. He joined the Beijing University of Posts and Telecommunications in 2007 and is a professor and deputy director of Beijing Key Lab of Intelligent Telecommunications Software and Multimedia at present. His research interests are in data mining and machine learning. He has published more than 100 papers in refereed journals and conferences, such as TPAMI, TKDE, KDD, WWW and NeurIPS.

Supplemental Materials:

A. Impact of Graph Data on GFMs

The success of foundation models depends on high-quality training data, and foundation models exhibit significantly different performance on various types of test data. In this section, we discuss the impact of graph data on graph foundation models from three aspects: graph type, graph scale and graph diversity.

Graph Type. Based on the number of categories of nodes and edges in a graph, we can categorize graphs into homogeneous graphs and heterogeneous graphs. In homogeneous graphs, all nodes and edges belong to the same category. For example, in a social graph where nodes represent individuals (users) and edges represent friendship relationships, it is a homogeneous graph because all nodes are individuals and all edges represent friendship relationships. Heterogeneous graphs, on the other hand, have more than one type of nodes or edges, representing different types of entities and relationships [33]. For instance, an e-commerce graph may include nodes for users, products, and purchase relationships, forming a heterogeneous graph. For GFMs, handling heterogeneous graphs poses greater challenges and typically requires the design of specific backbone architectures and optimization objectives. Nonetheless, utilizing the meta-path based approach [223], a heterogeneous graph can be mapped into multiple homogeneous graphs, one for each meta-path. For example, one can apply the GFM trained on homogeneous graphs to each of these meta-path induced homogeneous graphs, separately, to get node embedding. Then, these embeddings on homogeneous graphs under different meta-paths can be fused together. However, beyond homogeneous graphs and heterogeneous graphs, there are some more complex types of graphs in the real world, such as dynamic graphs and hypergraphs [224], which poses additional challenges for GFM.

Graph Scale. Based on the number of nodes and edges in a graph, we can categorize graphs into relatively small graphs and large graphs. Small graphs are of smaller scale, typically containing dozens to hundreds of nodes and edges. For example, chemical molecular graphs represent the structure of small molecules and typically consist of dozens to hundreds of atoms. Large graphs, on the other hand, refer to graphs with a significant number of nodes and edges, often encompassing millions or even billions of nodes and edges. For instance, e-commerce graph in Alibaba includes billons of nodes and hundreds of billion edges [196]. For graph foundation models, large graphs impose higher demands on the capacities of graph foundation models. Firstly, large graphs, due to their numerous nodes and typically sparser edges, introduce more noise and pose greater challenges in terms of storage and computation [225]. Additionally, large graphs often exhibit long-range dependency relationships [226], requiring more neural network layers and a higher number of parameters, which exacerbates the over-smoothing [39] and over-squashing [18] problem of GNN-based models.

Graph Diversity. Based on whether a graph dataset originates from the same domain, we can categorize graphs into same-domain graphs and cross-domain graphs. Same-domain graphs refer to graph data from similar or related domains, typically containing nodes and edges of similar types. For example, the social graphs of Facebook and WeChat come from similar domains. Cross-domain graphs [227], on the other hand, involve graph data from different domains or data sources, often comprising nodes and edges of different types, aimed at addressing multi-domain problems or cross-domain tasks. For example, academic networks and molecular graphs come from different domains. For graph foundation models, supporting cross-domain graphs presents greater challenges because graphs from different domains lack a unified underlying semantics. This can result in weak transfer performance or even negative transfer when applying the model to a new dataset [78]. Therefore, addressing the heterogeneity of different domains and enabling the same GFM to be applied to graphs from different domains is a significant challenge for GFMs.

B. Impact of Graph Tasks on GFMs

Language foundation models can be widely applied to various NLP tasks, while for graph foundation models, the formats of graph tasks are also quite diverse and can be categorized into three classes: node-level tasks, edge-level tasks, and graph-level tasks.

Node-level Tasks. Node-level tasks refer to the classification, regression, or prediction performed on each node. Common node-level tasks include node classification, node regression, and clustering coefficient prediction. For example, in social networks, graph nodes can represent users, and node classification can be used to identify users from different social circles.

Edge-level Tasks. Edge-level tasks involve the classification, regression, or prediction performed on each individual edge. Common edge-level tasks include edge classification, link prediction, shortest path prediction, connectivity prediction, and maximum flow prediction. For example, in e-commerce, link prediction can be used to predict products that users may be interested in.

Graph-level Tasks. Graph-level tasks focus on the entire graph. Common graph-level tasks include graph classification, graph regression, graph generation, graph clustering, graph condensation and average clustering coefficient prediction. For example, in bioinformatics, graph property prediction can be used to predict the biological activity or toxicity of molecular compounds, thereby accelerating the drug discovery process.

In summary, the format of tasks in graphs are highly diverse and can be categorized into three types: node-level, edge-level, and graph-level, each of which has wide-ranging applications. This undoubtedly increases the challenge of homogenization for GFMs. For example, in graph classification and node classification tasks on synthetic datasets, modeling structural information is often more crucial [228]. On the other hand, when dealing with node classification tasks on graphs with rich node features, modeling feature information becomes more important [228]. Furthermore, tasks that are more similar to each other will also have a lower transfer difficulty, implying that these tasks are more likely to be addressed using the same GFM. While increasing expressive power holds promise for improving the performance of many node-level, edge-level, and graph-level tasks [229], there is also some work suggesting that overly strong expressive power may not be necessary for graph generation tasks [230].

C. Details of approaches involved as GNN-based models

We categorize the GNN-based methods in Table 1.

D. Details of approaches involved as LLM-based models

We categorize the LLM-based methods in Table 2.

E. Details of approaches involved as GNN+LLM-based models

We categorize the GNN+LLM-based methods in Table 3.

TABLE 1: Details of approaches involved as GNN-based models.

Model	Backbone Architecture	Pre-training	Adaptation
All In One [56]	GCN, GAT, Graph Transformer	Same-Scale CL	Prompt Tuning
PRODIGY [59]	GCN, GAT	Graph Reconstruction, Supervised	Prompt Tuning
DGI [80]	GCN	Cross-Scale CL	Parameter-Efficient FT
GRACE [62]	GCN	Same-Scale CL	Vanilla FT
FUG [99]	GCN	Same-Scale CL	Vanilla FT
VGAE [81]	GCN	Graph Reconstruction, Property Prediction	Vanilla FT
MA-GCL [82]	GCN	Same-Scale CL	Vanilla FT
MultiGPrompt [110]	GCN	Cross-Scale CL, Graph Reconstruction	Prompt Tuning
IGAP [104]	GCN, GAT, GraphSAGE	Same-Scale CL, Cross-Scale CL, Graph Reconstruction	Prompt Tuning
HGPROMPT [111]	GCN, GAT, SimpleHGN	Graph Reconstruction	Prompt Tuning
GraphMAE [64]	GAT	Graph Reconstruction	Parameter-Efficient FT
GraphMAE2 [65]	GAT	Graph Reconstruction	Parameter-Efficient FT
GPPT [68]	GraphSAGE	Graph Reconstruction, Cross-Scale CL	Prompt Tuning
VPGNN [83]	GraphSAGE	Cross-Scale CL	Prompt Tuning
GPT-GNN [100]	HGT	Graph Reconstruction	Vanilla FT
PT-HGNN [109]	HGT	Same-Scale CL	Vanilla FT
CPT-HG [108]	HGT	Same-Scale CL	Vanilla FT
GraphPrompt [55]	GIN	Graph Reconstruction	Prompt Tuning
IHP [113]	PHC	Graph Reconstruction	Prompt Tuning
GraphPrompt+ [106]	GIN	Graph Reconstruction, Cross-Scale CL, Same-Scale CL	Prompt Tuning
ProNoG+ [107]	FAGCN	Same-Scale CL	Prompt Tuning
GCC [84]	GIN	Same-Scale CL	Vanilla FT
GraphCL [85]	GIN	Same-Scale CL	Parameter-Efficient FT
AdapterGNN [66]	GIN	Cross-Scale CL, Graph Reconstruction, Same-Scale CL	Parameter-Efficient FT
PhyGCN [112]	HyperGCN	Graph Reconstruction	Parameter-Efficient FT
GPT-ST [115]	GPT-ST	Graph Reconstruction	Parameter-Efficient FT
GraphST [114]	GraphST	Same-Scale CL	Parameter-Efficient FT
AAGOD [86]	GIN	Same-Scale CL, Supervised	Prompt Tuning
GPF [87]	GIN	Cross-Scale CL, Graph Reconstruction	Prompt Tuning
GCOPE [98]	FAGCN	Same-Scale CL	Prompt Tuning
FOTOM [190]	GIN	Same-Scale CL	Parameter-Efficient FT
TPP [105]	SGC	Same-Scale CL	Prompt Tuning
GraphControl [191]	GIN	Same-Scale CL	Parameter-Efficient FT
G-TUNING [101]	GIN	Same-Scale CL, Graph Reconstruction	Customized FT
Graph-BERT [95]	Graph Transformer	Graph Reconstruction, Supervised	Vanilla FT
GROVER [96]	Graph Transformer	Property Prediction	Vanilla FT
G-Adapter [67]	Graph Transformer	Supervised, Graph Reconstruction, Property Prediction	Parameter-Efficient FT

TABLE 2: Details of approaches involved as LLM-based models.

Model	Backbone Architecture			Pre-training	Adaptation
GIMLET [119]	Graph-to-token	+	Transformer	-	-
InstructGLM[116]	Graph-to-token	+	Flan-T5/LLaMA	MLM,LM	Manual Prompt Tuning
NLGraph[60]	Graph-to-text	+	GPTs	LM	Manual Prompt Tuning
TextForGraph [124]	Graph-to-text	+	GPTs	LM	Manual Prompt Tuning
When&Why [125]	Graph-to-text	+	GPTs	LM	Maunal Prompt Tuning
GraphWiz [126]	Graph-to-text	+	LLaMA, Mistral	LM	Maunal Prompt Tuning
CGForLLM [128]	Graph-to-text	+	GPT4	LM	Maunal Prompt Tuning
LLM4DYG [137]	Graph-to-text	+	LLaMA, Vicuna, GPT-3.5	LM	Manual Prompt Tuning
GPT4Graph[117]	Graph-to-text	+	GPT-3	LM	Manual Prompt Tuning + Automatic Prompt Tuning
Graph-LLM[118]	Graph-to-text	+	BERT, DeBERTa, Sentence-BERT, GPTs, LLaMA	MLM,LM	Manual Prompt Tuning + Automatic Prompt Tuning

TABLE 3: Details of approaches involved as GNN+LLM-based models.

Model	Backbone Architecture	Pre-training	Adaptation
TAPE [76]	GNN-centric	LM	Tuning-free Prompting + Parameter-Efficient FT
GIANT [139]	GNN-centric	MLM	Vanilla FT
GraD [138]	GNN-centric	MLM	Parameter-Efficient FT
GALM [141]	GNN-centric	Graph Reconstruction	Vanilla FT
OFA [142]	GNN-centric	MLM	Tuning-free Prompting
Heterformer [143]	GNN-centric	LM	Vanilla FT
edgeformers [144]	GNN-centric	LM	Vanilla FT
LLMRec [145]	GNN-centric	LM	Tuning-free Prompting + Parameter-Efficient FT
WalkLM [146]	GNN-centric	MLM	Vanilla FT
METERN [148]	GNN-centric	MLM	Parameter-Efficient FT
LLM-GNN [150]	GNN-centric	LM	Parameter-Efficient FT
WTGIA [140]	GNN-centric	LM	Parameter-Efficient FT
GHGRL [173]	GNN-centric	LM	Vanilla FT
GLEM [152]	Symmetric	MLM	Vanilla FT
GraphFormer [151]	Symmetric	MLM	Vanilla FT
G2P2 [153]	Symmetric	GTCL	Prompt Tuning
Text2Mol [159]	Symmetric	MLM + GTCL	Parameter-Efficient FT
MoleculeSTM [160]	Symmetric	MLM + GTCL	Parameter-Efficient FT
MolCA [162]	Symmetric	LM	Parameter-Efficient FT
CLAMP [161]	Symmetric	MLM + GTCL	Parameter-Efficient FT
GIT-Mol[163]	Symmetric	LM	Prompt Tuning
PATTON [156]	Symmetric	MLM	Parameter-Efficient FT
ENGINE [155]	Symmetric	LM	Parameter-Efficient FT
OpenGraph [157]	Symmetric	LM	Vanilla FT
RLMRec [158]	Symmetric	LM	Parameter-Efficient FT
GraphTranslator [164]	LLM-centric	LM	Parameter-Efficient FT
THLM [166]	LLM-centric	MLM	Parameter-Efficient FT
GraphGPT [165]	LLM-centric	MLM	Prompt Tuning
InstructGraph[168]	LLM-centric	LM	Prompt Tuning
RELM [169]	LLM-centric	LM	Tuning-Free Prompting
GraphPrompter [167]	LLM-centric	LM	Parameter-Efficient FT
HiGPT [172]	LLM-centric	LM	Parameter-Efficient FT
G-Retriever[171]	LLM-centric	LM	Prompt-Tuning
TEA-GLM[170]	LLM-centric	LM	Parameter-Efficient FT
HyperBERT[174]	LLM-centric	LM	Parameter-Efficient FT