

A Survey on Self-Supervised Graph Foundation Models: Knowledge-Based Perspective

Ziwen Zhao ¹, Yixin Su ¹, Yuhua Li ¹, *Member, IEEE*, Yixiong Zou ¹, *Member, IEEE*,
Ruixuan Li ¹, *Member, IEEE*, and Rui Zhang ¹, *Senior Member, IEEE*

(Survey Paper)

Abstract—The field of graph foundation models (GFMs) has seen a dramatic rise in interest in recent years. Their powerful generalization ability is believed to be endowed by self-supervised pre-training and downstream tuning techniques. There is a wide variety of knowledge patterns embedded in the graph data, such as node properties and clusters, which are crucial for learning generalized representations for GFMs. We present a comprehensive survey of self-supervised GFMs from a novel knowledge-based perspective. Our main contribution is a knowledge-based taxonomy that categorizes self-supervised graph models by the specific graph knowledge utilized: microscopic (nodes, links, etc.), mesoscopic (context, clusters, etc.), and macroscopic (global structure, manifolds, etc.). It covers a total of 9 knowledge categories and 300 references for self-supervised pre-training as well as various downstream tuning strategies. Such a knowledge-based taxonomy allows us to more clearly re-examine potential GFM architectures, including large language models (LLMs), as well as provide deeper insights for constructing future GFMs.

Index Terms—Graph foundation models, self-supervised learning, pre-training, graph neural networks, large language models.

I. INTRODUCTION

GRAPHS are prevalent in various real-world applications. They exhibit diverse knowledge patterns due to the inherent topology [1], [2], [3]. Moreover, the availability of features and properties associated with nodes and links, such as textual attributes and centrality measures, further enriches the knowledge present in graphs. Over time, deep graph mining techniques have

Received 16 August 2024; revised 24 February 2025; accepted 27 April 2025. Date of publication 8 May 2025; date of current version 7 July 2025. This work was supported in part by the National Key Research and Development Program of China under Grant 2024YFC3307900, in part by the National Natural Science Foundation of China under Grant 62436003, Grant 62376103, Grant 62206102, and Grant 62302184, in part by the Science and Technology Support Program of Hubei Province under Grant 2022BAA046, in part by Hubei science and technology talent service project under Grant 2024DJC078, in part by Ant Group through CCF-Ant Research Fund, and in part by the HPC Platform of Huazhong University of Science and Technology. Recommended for acceptance by Liqiang Nie. (Ziwen Zhao and Yixin Su are co-first authors.) (Corresponding authors: Yuhua Li; Rui Zhang.)

The authors are with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: zwzhao@hust.edu.cn; yixin.su@outlook.com; idcliyuhua@hust.edu.cn; yixiong@hust.edu.cn; rxli@hust.edu.cn; rayteam@yeah.net).

This article has supplementary downloadable material available at <https://doi.org/10.1109/TKDE.2025.3568147>, provided by the authors.

Digital Object Identifier 10.1109/TKDE.2025.3568147

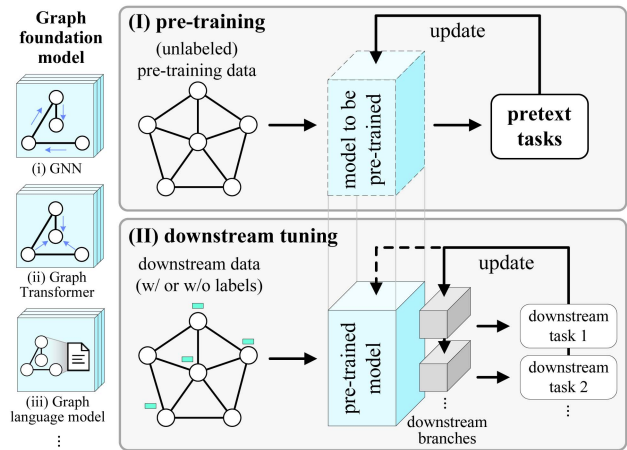


Fig. 1. How self-supervised GFMs are believed to work: pre-training and downstream tuning. Updating the pre-trained model during downstream tuning is optional depending on the tuning strategy.

evolved from graph neural networks (GNNs) [4], [5], [6] to graph Transformers (GTs) [7], [8] and more recent large language model (LLM)-based graph language models [9], [10], [11]. They are motivated by capturing more comprehensive knowledge patterns within the graph data, from local relationships to the global structure.

However, when confronted with various downstream task requirements, researchers often encounter graph data that lacks available labels, such as the field of an article in citation networks. Fortunately, *self-supervised learning* on graphs has emerged as a powerful approach to uncovering underlying patterns in enormous unannotated data [12], [13]. SSL methods design unsupervised tasks – *pretext tasks* – to pre-train a graph model, and adapt the pre-trained model to the specific application scenarios by downstream tuning approaches, as depicted in Fig. 1. Researchers have observed powerful generalization ability within graph models pre-trained with self-supervision [14], as they aim to mine the underlying knowledge patterns of graph data instead of solely relying on manual labels that are limited to specific task spaces. Therefore, self-supervised pre-training and downstream tuning are believed to be the most promising techniques to achieve a *graph foundation model* (GFM) – a highly generalized model that can handle a wide range of application tasks [15].

Previous efforts: The popularity of self-supervised learning and LLMs on graphs in recent years has given rise to a flood of surveys. Early efforts [3], [16], [17] focus on summarizing general self-supervised graph models. [18], [19], [20], [21] systematically summarize the trending direction of graphs meet LLMs, shortly after the sensational debut of ChatGPT. The success of LLMs has also activated heated discussions towards GFMs [14], [22], summarizing key techniques and principles of learning generalized graph models and providing outlooks towards the realization of GFMs. Despite the promising work, we reveal three major shortcomings of the existing surveys:

(1) *Lack of comprehensiveness:* existing surveys in the field of self-supervised graph learning [3], [16], [17] do not cover the latest progress in this fast-developing field. For example, none of these surveys have discussed the new achievements of masked graph autoencoders [23] and learning graph manifolds [24]. A recent survey [25] includes cutting-edge developments in graph contrastive learning, yet it focuses on real-world applications rather than realizing GFMs.

(2) *Unclear categorization:* existing surveys [3], [14], [17] broadly categorize graph pre-training methods as “generative – contrastive (predictive)”. This rough categorization is insufficient to capture the unique characteristics of graphs, which have diverse knowledge patterns embedded in their structure and properties. For instance, predicting *links* requires local relationships between nodes, whereas predicting *clusters* requires the node distribution on the entire graph. However, both generative and contrastive (predictive) frameworks can utilize the knowledge of links [8], [12] and clusters [26], [27], which the aforementioned taxonomy fails to distinguish. On the other hand, recent surveys of GFMs only give a brief summary of existing pre-training and downstream tuning methods: [14] puts the emphasis on the architecture design of graph models, while [18], [22] are closer to outlooks towards future directions of GFMs.

(3) *Limited to specific architectures:* the aforementioned graph self-supervised learning surveys are limited to GNNs/GTs only. On the other hand, LLM-based surveys [19], [20], [21] overemphasize the language model architectures and textual attributes of graphs while overlooking other structural patterns. A recent GFM survey [14] categorizes existing studies into three groups of GNN, LLM, and GNN+LLM, still limited by specific backbone architectures instead of an in-depth perspective towards the ultimate goal – mining generalized graph knowledge. As language models are not designed for mining various types of graph knowledge, it still remains an unanswered question if LLMs are ideal architectures for GFMs. If other promising generalized architectures showed up in the near future (which is happening right now), their architecture-based taxonomy might no longer apply.

Our contributions: Considering the aforementioned issues, it is necessary to provide a comprehensive survey of self-supervised graph models with a clearer categorization and taxonomy, which will offer a better understanding and greater insight into how future GFMs work. We first propose a *knowledge-based taxonomy* that categorizes self-supervised graph pre-training based on the types of knowledge utilized: *microscopic* pre-training (Section III) that focuses on individual nodes and

links; *mesoscopic* pre-training (Section IV) that focuses on local relationships in the graph, such as context and clusters; and *macroscopic* pre-training (Section V) that focuses on the structure and the manifold underlying the entire graph. Such a knowledge-based taxonomy provides a unified perspective to analyze the pre-training and downstream tuning strategies (Section VI) of both GNNs/GTs and recent graph language models (Section VII), providing valuable insights for the future directions of GFMs (Section VIII). Our knowledge-based perspective is also architecture-agnostic, compared to existing surveys which are applicable to only certain types of architectures. Therefore, we provide a more systematic view covering a much wider range of graph models. As illustrated in Fig. 2, we analyze 9 knowledge categories and 300 references ranging from the 2010 s to 2025, which are to our knowledge the most detailed categorization of self-supervised GFMs. All papers included are summarized as tables in the Appendix, available online for better comparison. We also collate more than 500 relevant papers and list them on GitHub.¹ We hope this survey will help researchers exploit more powerful GFMs by exploiting graph-specific knowledge.

II. PRELIMINARY

This section provides basic concepts related to our topic.

Graph: Graph is a data structure consisting of a node (vertex) set and an edge (link) set $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$ indicates if two nodes are connected by a link. For an attributed graph, each node is associated with a row of the feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. For every node $i \in \mathcal{V}$, its (undirected) neighborhood is $\mathcal{N}_i = \{j \in \mathcal{V} | A_{i,j} = 1\}$. A graph dataset can contain one graph only or multiple relatively small graphs.

Graph model and graph foundation model (GFM): A graph model is an encoding function $\mathbf{Z} = f(\mathcal{G}; \Theta)$ that can be parameterized by GNNs [4], [5], [6], GTs [7], [8], graph language models [9], [10], [11], etc. A GFM is an (ideal) graph model pre-trained on various kinds of unsupervised graph data (in any form) to handle various types of graph-related tasks [14].

Pre-training task (pretext): A pretext $\mathcal{L} \in \mathcal{T}$ is a self-supervised task performed during the pre-training phase of a graph model, where \mathcal{T} represents the pretext task set. A pretext should meet two conditions: (1) during the self-supervised pre-training, no manually labeled data is used; (2) its goal is to achieve improved performance on one or multiple downstream tasks $\tilde{\mathcal{L}}$:

$$\sum_{\tilde{\mathcal{L}} \in \tilde{\mathcal{T}}} \min_{\Phi, \Theta^*} \tilde{\mathcal{L}}(\tilde{f} \cdot f^*, \tilde{\mathcal{G}}, \tilde{\mathcal{Y}}), \text{ s.t. } f^* = \sum_{\mathcal{L} \in \mathcal{T}} \arg \min_{\Theta} \mathcal{L}(f, \mathcal{G}) \quad (1)$$

where $\tilde{f}(\tilde{\mathcal{G}}; \Phi)$ denotes some optional downstream branches. “ Θ^* ” is optional depending on whether the pre-trained model parameters are tuned for downstream tasks. “ $\tilde{\mathcal{Y}}$ ” is also optional depending on whether task-specific labels are used, also known as supervised fine-tuning (SFT).

¹[Online]. Available: <https://github.com/Newiz430/Pretext>

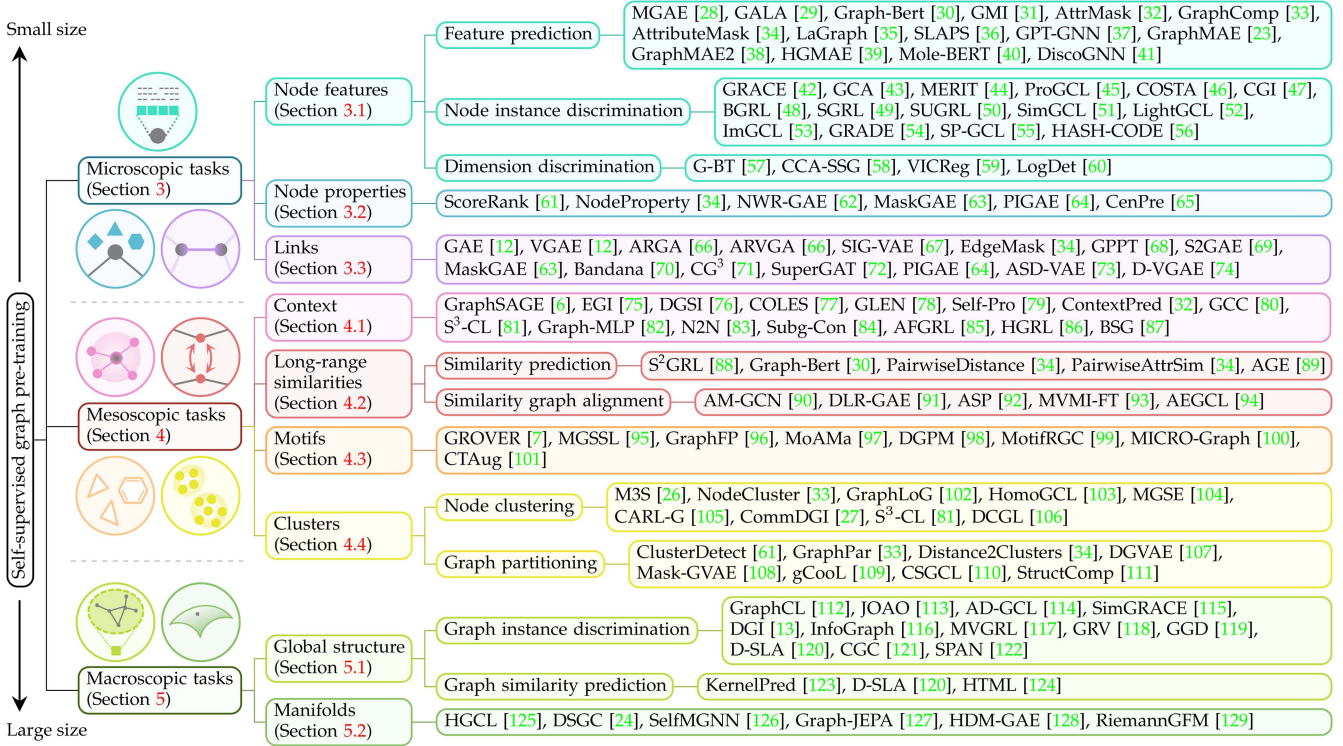


Fig. 2. Our knowledge-based taxonomy of self-supervised graph pre-training with representative literature.

III. MICROSCOPIC PRE-TRAINING TASKS

Microscopic pre-training tasks treat nodes or edges as individual instances. They extract features, properties, and local relationships between these instances.

A. Node Features

Node features are a rich source of semantic information in attributed graphs, encoding domain-specific knowledge such as textual content in citation networks or chemical properties in molecular graphs. The expressiveness and utility of these features heavily depend on their origin and the encoding methods employed.

Feature prediction: Feature prediction serves as a fundamental pretext task in graph autoencoding methods like MGAE [28], GALA [29], and Graph-Bert [30]. These methods reconstruct low-dimensional node representations and match them with the original feature size, minimizing the reconstruction error such as MSE: $\mathcal{L} = \mathbb{E}_{i \in \mathcal{V}} [\|\mathbf{X}_i - \hat{\mathbf{X}}_i\|^2]$, while GMI [31] maximizes the mutual information between the original graph and the output representations by a discriminator network.

Another kind of prediction task focuses on *feature denoising*, where noise is first added to the original features $\tilde{\mathbf{X}} = \mathbf{X} + \epsilon$, and then the model is tasked with recovering the original noise-free features. The success of masked language/image modeling [130], [131] has led to the rise of *masked feature prediction*, also known as masked autoencoding or graph completion [33]. Methods like AttrMask [32], [34], LaGraph [35], and SLAPS [36] sample a noise matrix from a Bernoulli distribution $\mathbf{M} \in \{0, 1\}^{n \times d}$ and obtain masked features $\tilde{\mathbf{X}} = \mathbf{M} \circ \mathbf{X}$.

Then, the original features are reconstructed by an MSE loss. GPT-GNN [37] adopts an autoregressive masking approach, where the masked node attributes and their corresponding edges are generated one-by-one. Recent GraphMAE series [23], [38] introduces a scaled cosine error with a focusing parameter λ to adjust the weight of each sample:

$$\mathcal{L} = \mathbb{E}_{q((1-\mathbf{M}) \circ \mathbf{X} | \mathbf{X})} \left[1 - \left(\frac{\mathbf{X}^\top f(\mathbf{M} \circ \mathbf{X}, \mathbf{A}; \Theta)}{\|\mathbf{X}\| \|f(\mathbf{M} \circ \mathbf{X}, \mathbf{A}; \Theta)\|} \right)^\lambda \right] \quad (2)$$

and this has inspired various new-generation masked autoencoder architectures [39], [40], [132]. DiscoGNN [41] first randomly replaces nodes with different ones. Then, it learns to find and reconstruct the replaced nodes.

Node instance discrimination: Instance discrimination, also referred to as “contrastive learning”, has achieved significant success in the visual domain [133], [134] and subsequently becomes a fundamental and general task for graph pre-training. Node instance discrimination aims to perform instance discrimination between node pairs. The workflow involves creating two perturbed versions (views) of an original graph $\mathcal{G}^i, \mathcal{G}^{ii}$. Nodes at the same position across views $(\mathbf{Z}_i^i, \mathbf{Z}_i^{ii})$ form positive pairs, while others $(\mathbf{Z}_i^i, \mathbf{Z}_j^{ii})$ or $(\mathbf{Z}_i^i, \mathbf{Z}_j^i)$ are randomly sampled as negative pairs. The goal is to maximize the similarity between positive pairs and minimize it between negative ones, allowing for the learning of general and perturbation-invariant representations.

The simplest form of node-level instance discrimination is to minimize the MSE between positive pairs $\mathcal{L} = \mathbb{E}_{i \in \mathcal{V}} [\|\mathbf{Z}_i^i - \mathbf{Z}_i^{ii}\|^2]$. However, it can suffer from representation degeneration,

i.e., the output may degenerate to a constant vector regardless of the input. This is often addressed by combining it with other tasks [35], [38], [58], [59]. By contrast, *mutual information* (MI) provides a more effective criterion by capturing non-linear statistical dependence between node instances [135]:

$$I(\mathbf{Z}_i^i; \mathbf{Z}_j^i) = D_{\text{KL}} [p(\mathbf{Z}_i^i, \mathbf{Z}_j^i) \| p(\mathbf{Z}_i^i) p(\mathbf{Z}_j^i)] \quad (3)$$

Calculating MI in a high-dimension space is a challenging task [135]. Therefore, various techniques have been proposed to estimate MI. These techniques mainly include:

(1) *Jenson-Shannon (JS) estimator* [136]: it replaces the KL divergence in (3) with JS divergence and approximates the distributions usually by a discriminator network \mathcal{D} :

$$\mathcal{L} = -\mathbb{E}_{i \in \mathcal{V}} \left[\sigma_+(-\mathcal{D}(\mathbf{Z}_i^i, \mathbf{Z}_i^i)) + \mathbb{E}_{j \in \mathcal{V}_i^-} [\sigma_+(\mathcal{D}(\mathbf{Z}_i^i, \mathbf{Z}_{:j}^i))] \right] \quad (4)$$

(2) *InfoNCE estimator* [137]: it is based on the Noise Contrastive Estimation (NCE) loss. Formally:

$$\mathcal{L} = -\mathbb{E}_{i \in \mathcal{V}} \left[\log \frac{\exp(\langle \mathbf{Z}_i^i, \mathbf{Z}_i^i \rangle)}{\sum_{j \neq i} \exp(\langle \mathbf{Z}_i^i, \mathbf{Z}_j^i \rangle) + \sum_{j=1}^n \exp(\langle \mathbf{Z}_i^i, \mathbf{Z}_j^i \rangle)} \right] \quad (5)$$

where $\langle \cdot, \cdot \rangle$ is the relative similarity of two samples. This estimator is well-known in representative models like GRACE [42], GCA [43], ProGCL [45], and more [44], [46], [47].

(3) *Triplet (margin) estimator* [138]: some contrastive frameworks like SUGRL [50] employ a triplet loss to contrast between the anchor-positive pairs $(\mathbf{Z}, \mathbf{Z}^+)$ the anchor-negative pairs $(\mathbf{Z}, \mathbf{Z}^-)$:

$$\mathcal{L} = \mathbb{E}_{i \in \mathcal{V}} [\langle \mathbf{Z}_i, \mathbf{Z}_i^+ \rangle - \langle \mathbf{Z}_i, \mathbf{Z}_i^- \rangle + \epsilon] \quad (6)$$

where ϵ denotes the distance margin, controlling the lower bound of distance between positive and negative samples.

There are other instance discrimination objectives that have achieved competitive performance in learning node features. For example, the bootstrapping loss [48], [49], [139] generally computes the cosine similarity $\mathcal{L} = -\mathbb{E}_{i \in \mathcal{V}} [\mathbf{Z}_i^i p(\mathbf{Z}_i^i)^\top / \|\mathbf{Z}_i^i\| \|p(\mathbf{Z}_i^i)\|]$ between two asymmetric and momentum-updated networks, aligned by a projector $p(\cdot)$. Other examples include Bayesian Personalized Ranking loss (BPR) [51], [52], [140] and population spectral contrastive loss [55], [56], [141].

Node instance discrimination is one of the most popular and generalizable tasks beyond graph pre-training. Recent literature focusing on downstream tuning has unified multiple tasks such as link prediction, node classification, and graph classification into node instance discrimination [79], [142], [143], as an impactful self-supervised tuning strategy.

Dimension discrimination: Dimension discrimination focuses on distinguishing different dimensions of node representations. This can be seen as a column-wise discrimination approach. Similar to node instance discrimination, it begins with two augmented views. Then, it maximizes the correlation between corresponding dimensions and minimizes it between different ones. This strategy is initially proposed by Barlow Twins [144] and then introduced to the graph domain by G-BT [57], which considers the similarity between dimensions of

different instances. On the other hand, CCA-SSG [58] focuses on the dimensional similarity within a single instance via a covariance regularization term:

$$\mathcal{L} = \|\mathbf{Z}^i - \mathbf{Z}^{ii}\|_F^2 + \lambda \underbrace{\left(\|\mathbf{Z}^{i^\top} \mathbf{Z}^i - \mathbf{I}_d\|_F^2 + \|\mathbf{Z}^{ii^\top} \mathbf{Z}^{ii} - \mathbf{I}_d\|_F^2 \right)}_{\text{dimension discrimination}} \quad (7)$$

Future improvements mainly focus on the effectiveness of the regularization term, such as how to avoid the global or local representation collapse problem [59], [60].

Discussion: For feature prediction, both traditional methods and the latest masked autoencoders tend to preserve shallow features rather than capture deeper semantic information. Despite that node instance discrimination encourages the model to focus on deeper semantic information, it mainly focuses on node feature correlations while neglecting high-order structural knowledge, potentially leading to suboptimal performance in structural tasks like link prediction. Compared to discrimination between instances, dimension discrimination enables augmentation-invariant feature learning by clarifying node dimensions. However, as noted in [58], its benefits may diminish with smaller representation dimensions.

Efficiency analysis: The computational efficiency of node feature-based tasks primarily depends on the size of the feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and the model architecture. The time complexity for the most common InfoNCE-based node instance discrimination is $\mathcal{O}(n^- nd)$, where n^- is the number of negative samples for each node. The huge demand for negative samples results in a prohibitive worst-case cost $\mathcal{O}(n^2 d)$. Similarly, the time complexity for the dimension discrimination term in (7) is $\mathcal{O}(nd^2)$. In contrast, feature prediction and bootstrapping-based methods scale linearly with both n and d ($\mathcal{O}(nd)$), making them more advantageous in terms of scalability. The lightweight linear decoder adopted by new-generation masked autoencoders can further reduce the computational cost [131].

B. Node Properties

While Section III-A discusses pretext tasks that focus on node semantic information, another class of tasks focuses on node properties, which are crucial for understanding their structural roles within the graph.

The *centrality* is a set of node properties that quantify the relative importance between nodes based on local structure. For example, the degree of a node $\text{deg}(\cdot)$ is a common measure of local connectivity, defined as the number of edges incident to that node. There are various kinds of centrality measures including degree, closeness, betweenness, eigenvector, and PageRank [145]. Predicting centralities is one of the most direct methods for capturing the structural importance of nodes. Autoencoding methods such as NWR-GAE [62] and MaskGAE [63] employ an MSE loss to predict the node degree: $\mathcal{L} = \mathbb{E}_{i \in \mathcal{V}} [\|\text{deg}(i) - \hat{\text{deg}}(i)\|^2]$. Besides degree prediction, CenPre [65] leverages the left singular vector of \mathbf{A} as structural representations, similar to the eigenvector centrality, and computes its similarity with the original features. Hu et

al. [61] propose *centrality ranking*, which predicts if a node has a higher or lower centrality score s compared to another node.

There are also other node properties that capture structural features beyond relative importance. For instance, (*local*) *clustering coefficient* [146] measures the gathering tendency of node groups, and predicting the clustering coefficient highlights the local relationship between nodes and guides the model to preserve them [34]. *Node order* is also a special kind of property that holds the permutation-invariant information for more expressive GNNs. PIGAE [64] aligns the order of output node representations with the input node order by adding a learnable permuter into a VGAE [12].

Discussion: Node properties help models comprehend node information by encoding structural roles such as relative importance. However, node properties tend to be more task-specific [34], meaning that they can be difficult to leverage as generalizable knowledge in certain scenarios. Additionally, node properties may not always provide sufficient discriminative power. Graphs with different topologies can have the same degree distribution, making it difficult to distinguish between them solely by the degree.

Efficiency analysis: Calculating different node properties necessitates varying degrees of additional computational costs. For example, the complexity for calculating the degree of all nodes is $\mathcal{O}(|\mathcal{E}|)$, while the complexity for calculating the eigenvector centrality (via eigenvalue decomposition) is $\mathcal{O}(n^3)$. Considering that most of these properties are scalars, the memory required to store them is usually $\mathcal{O}(n)$.

C. Links

From chemical bonds in molecular graphs and social connections in social networks to semantic relationships in knowledge graphs, links play a fundamental role in graphs as they represent basic relationships between nodes.

Link prediction is a fundamental task in graph-based SSL aiming to predict the existence or probability of a link between two nodes. Structure-based autoencoders such as GAE [12] feed the learned node representations into a dot-product decoder $\hat{\mathbf{A}} = \sigma(\mathbf{Z}\mathbf{Z}^\top)$ to predict the existence probability between a pair of nodes:

$$\mathcal{L} = -\mathbb{E}_{(i,j) \in \mathcal{E}} \left[\log \hat{A}_{i,j} \right] + \mathbb{E}_{(i,j) \in \mathcal{E}^-} \left[\log \left(1 - \hat{A}_{i,j} \right) \right] \quad (8)$$

To capture more complicated latent spaces, variational autoencoders such as VGAE [12] and [66], [67], [74] learn a Gaussian model for latent embeddings $q(\mathbf{Z}|\mu, \sigma) = \mathcal{N}(\mu, \sigma^2)$ to approximate the real posterior $p(\mathbf{Z}|\mathbf{X}, \mathbf{A})$. Representation vectors are then sampled from these distributions to maximize the expected log-likelihood $\log p(\mathbf{A})$ bounded by the evidence lower bound (ELBO):

$$\mathcal{J} = \mathbb{E}_{q(\mathbf{Z}|\mu, \sigma)} [\log p(\mathbf{A}|\mathbf{Z})] - D_{\text{KL}} [q(\mathbf{Z}|\mu, \sigma) \| p(\mathbf{Z})] \quad (9)$$

where $p(\mathbf{Z}) = \mathcal{N}(0, \mathbf{I})$ is the preset Gaussian prior. Link prediction usually serves as an auxiliary task for training semi-supervised GNNs [72], discrimination models [71], [93], [94],

and feature-based autoencoders [73], [91]. Instead of a learnable decoder, discrimination losses such as InfoNCE are also applicable to link prediction [8], [147], [148].

Another kind of prediction task, *link denoising*, involves adding random noises to the adjacency matrix. For example, Bandana [70] samples continuous edge noises and predicts the noise values. A more widely adopted approach for link denoising is *masked link prediction*, where a portion p of edges is randomly masked using binary noise $M_{i,j} \sim \text{Bernoulli}(1 - p)$. The objective is similar to that of binary link prediction, with the key difference being that only the masked edges are treated as positive samples during training. EdgeMask [34] and S2GAE [69] learn a decoder $\hat{\mathbf{A}} = \sigma(g(\mathbf{Z}\mathbf{Z}^\top; \Psi))$ to recover the masked edges. MaskGAE [63] captures long-range relationships by randomly masking out paths obtained from random walks.

Edge features in some attributed graphs also provide rich semantics complementing the graph structure, such as the number of co-authored papers or research topics in a co-authorship graph. Methods for node feature learning, such as auto-encoding in PIGAE [64] and ASD-VAE [73], as well as masked feature prediction in AttrMask [32], can be effortlessly applied to *edge feature prediction*.

Discussion: Link prediction has brought significant benefits to structure-based downstream tasks by capturing the structural information of graphs. It explicitly models the relationships between nodes that are not considered in node feature-based methods. Despite its widespread use, link prediction has been criticized for overemphasizing local structure [13], [23]. This highlights the need for exploring more compatible link-based pre-training strategies with feature semantics and higher-order structures.

Efficiency analysis: The memory required for dense and sparse adjacency matrices is $\mathcal{O}(n^2)$ and $\mathcal{O}(|\mathcal{E}|)$, respectively. Link prediction involves calculating the node similarity, such as the dot product and cosine similarity, both of which have a time complexity of $\mathcal{O}(d|\mathcal{E}|)$. As the efficiency of link prediction models is primarily bottlenecked by the edge size, pruning methods and masking link prediction can significantly reduce the memory cost for encoding.

IV. MESOSCOPIC PRE-TRAINING TASKS

In contrast to microscopic tasks that focus on individual nodes and links, mesoscopic pre-training tasks aim to capture properties within a local range. These pretexts learn representations that encode higher-order information and long-range dependencies.

A. Context

Graph context refers to the neighborhood or a broader sub-graph surrounding a node. Most context-based methods leverage the *homophily assumption* [149], where adjacent nodes tend to share similar attributes.

Context discrimination is a pretext that can be traced back to network embedding algorithms, e.g., DeepWalk [150]. They sample node sequences from the graph using random walks and then iteratively update their embeddings using text embedding methods. GraphSAGE [6] redefines “context” from random

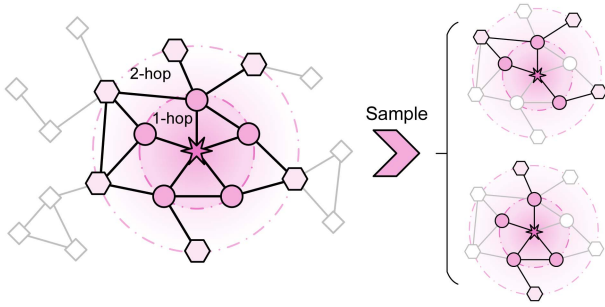


Fig. 3. An illustration of the contextual knowledge. For the central node \star of a 2-hop subgraph (Left), context discrimination often takes its neighboring nodes \circ or \diamond as positive samples and other nodes \diamond as negative samples. Contextual subgraph discrimination samples multiple contextual subgraphs (Right) as positive pairs, while negative ones are sampled from other subgraphs.

walk sequences to the neighborhood subgraphs induced from the graph. It optimizes a negative sampling-based JS estimator loss (4) between the central node i and its contextual nodes $j \in \mathcal{N}_i$:

$$\mathcal{L} = -\mathbb{E}_{\substack{i \in \mathcal{V} \\ j \in \mathcal{N}_i}} \left[\log \sigma(\mathbf{Z}_i^\top \mathbf{Z}_j) + \lambda \sum_{k \in \mathcal{V}^-} \log \sigma(-\mathbf{Z}_i^\top \mathbf{Z}_k) \right] \quad (10)$$

Later efforts [75], [76] improve GraphSAGE with a discriminator network to determine whether one node is the neighbor of another node. COLES [77] captures neighborhood similarity by equipping Laplacian Eigenmaps [151] with negative sampling, which is further generalized by GLEN [78] as a rank optimization problem of representation scatter matrices.

For other MI estimators, Graph-MLP [82] and N2N [83] define their positive sample pairs as every node and its k -hop neighborhood, and employ an InfoNCE loss. Sub-Con [84] selects k -nearest neighbors by personalized PageRank scores as positive samples of a triplet loss. AFGRL [85] selects k -nearest neighbors in the context of both structure and feature as positive samples of a bootstrapping loss. HGRL [86] further leverages the homophily assumption by selecting homophilic neighbors as precise positive samples. BSG [87] uses mean pooling to obtain neighborhood embeddings $\mathbf{Z}_{\mathcal{N}}$ and maximizes the mutual information $I(\mathbf{Z}, \mathbf{Z}_{\mathcal{N}})$ and the conditional entropy $H(\mathbf{Z}|\mathbf{Z}_{\mathcal{N}})$ through MSE and hinge loss, respectively.

Another task, *contextual subgraph discrimination*, measures the similarity between two different sampled subgraphs, as shown in Fig. 3. ContextPred [32] samples a “context graph” from the periphery of the k -hop subgraph and matches them as a positive pair. Instead of sampling an additional context graph, GCC [80] directly induces two different subgraphs from the k -hop neighborhood of each node as a positive pair. S³-CL [81] contrasts between intermediate message-passing layers to aggregate neighborhoods of varying scales.

Discussion: Compared to individual links, treating node context as instances facilitates a more complete understanding of the local graph structure. Nonetheless, it is empirically verified that some context learning methods have limited contributions to the performance of message-passing GNNs [34], owing to the inherent capability of message-passing to extract local structural

information. Context learning has the potential to benefit models that put more emphasis on global interactions, e.g., graph Transformers.

Efficiency analysis: Using graph traversal algorithms, one can obtain the k -hop subgraph of any node with a time complexity of $O(n + |\mathcal{E}|)$. A more common approach is to aggregate the embeddings of neighboring nodes by left-multiplying \mathbf{A} , which has a time complexity of $O(k|\mathcal{E}|)$. Since the complexity is independent of n , this approach is particularly suitable for handling sparse networks with a large number of nodes. Existing GFM researchers [8], [152], [153], [154] often reformulate node classification on large networks as predicting subgraph labels around the target node, which is essentially a divide-and-conquer strategy.

B. Long-Range Similarities

Long-range similarities capture relationships between non-neighboring nodes that share semantic relevance. These similarities reveal higher-order dependencies beyond local neighborhoods. For example, in social networks, the small-world property implies that any two individuals are likely connected through a short chain of acquaintances [146].

Similarity prediction: Similarity prediction aims to capture long-range similarities between nodes by directly predicting the similarity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$. Depending on whether two nodes are connected by a path, long-range similarities can be divided into topologically accessible similarities and topologically inaccessible ones. For the former, the *shortest path distance* measures the minimal distance between two connected nodes, and the *Katz index* [155] measures the total number of paths of every length between two connected nodes. S²GRL [88] and PairwiseDistance [34] train the graph model to predict these similarities between all pairs of nodes by a negative log-likelihood loss. For the latter, the *PageRank similarity* [145] quantifies the importance of node pairs in terms of graph structure, while the *Jaccard’s coefficient* [156] measures the overlap between node neighborhoods. There are also feature-based measures that quantify the degree of similarity between two nodes’ features, regardless of their connectivity, such as euclidean distance and cosine similarity [34], [89], [157]. While Graph-Bert [30] directly predicts them by a regressive loss, AGE [89] and PairwiseAttrSim [34] adopt *similarity-based discrimination* that selects a subset of node pairs with the highest (resp. lowest) similarity scores and uses them as positive (resp. negative) samples. These similarities can bridge the disconnected components in the graph data which message-passing cannot.

Similarity graph alignment: Similarity graphs, derived from the original graph based on the pairwise similarities between nodes, usually serve as an alternative structural view of the graph. For instance, the kNN graph reconstructs the edge set by connecting the k -nearest neighbors of each node through various feature-based measures. It shares both commonalities and differences with the original graph structure (and other similarity graphs); therefore, aligning their semantics becomes a principled approach to combine feature semantics and graph structure. AM-GCN [90] and DLR-GAE [91] minimize the discrepancy

between original and similarity graph representations by MSE and cross-entropy. Instance discrimination methods [92], [93], [94] treat the original and similarity graphs as two views to integrate complementary information from both views.

Discussion: Long-range similarities play a crucial role in capturing the dependencies between nodes out of reach for local contexts. It also enables the model to handle sparse graphs or graphs with disconnected components. However, the discrepancy between feature similarity and structural similarity can lead to semantic conflicts. Nodes with similar features may not always have similar structural neighborhoods. Therefore, the choice of similarity measures should depend on the real-world requirements.

Efficiency analysis: Computing all-pairs shortest paths can be computationally expensive for large graphs. The running time of the classic Floyd-Warshall algorithm is $\mathcal{O}(n^3)$ and the memory required is $\mathcal{O}(n^2)$. Using Dijkstra’s algorithm for all nodes results in a complexity of $\mathcal{O}(n|\mathcal{E}|\log n)$. By contrast, calculating feature-based similarities requires $\mathcal{O}(n^2 d)$ time, which is more efficient for dense networks.

C. Motifs

Motifs are small subgraphs that frequently appear and carry significant structural and functional information, such as functional groups in molecular graphs, coregulators in regulatory networks, and cliques of people in social networks.

Motif prediction tasks aim to learn motif-level representations by predicting the motif pseudo-labels of subgraphs. These pseudo-labels are given by unsupervised motif discovery algorithms, e.g., RDKit [158]. GROVER [7] assigns motif pseudo-labels to molecular graphs and trains a GNN for classification, and MoAMA [97] extends this idea by conducting motif-wise feature masking and prediction. DGPM [98] performs a binary node-motif matching task to predict if a node belongs to a motif. Recent literature [95], [96] introduces the concept of “fragment graphs”, whose nodes are aggregated from subgraphs containing specific motifs. The aggregated supernode representations are collected in a motif dictionary. In this way, motif prediction is transformed into a lookup task: the representation vector of each node is associated with an entry in the motif dictionary.

Another line of work employs *motif-based discrimination* that creates contrastive sample pairs for motif-aware representations. MotifRGC [99] designs an adversarial motif generator to generate positive and negative views. Fragment graphs can also serve as contrastive views: MICRO-Graph [100] and GraphFP [96] treat the original graph and its corresponding fragment graph as a positive pair.

Discussion: Most motif-based pretexts are designed specifically for molecular graphs, limiting their applicability to larger-scale networks. The only exception as far as we know is CTAug [101], a contrastive method aiming to preserve cohesive motifs (k-cliques, k-cores, etc.) in social networks. Future research should focus on developing more general motif learning methods to identify and extract common knowledge from different motifs, reducing the size of motif dictionaries while preserving essential structural and functional information.

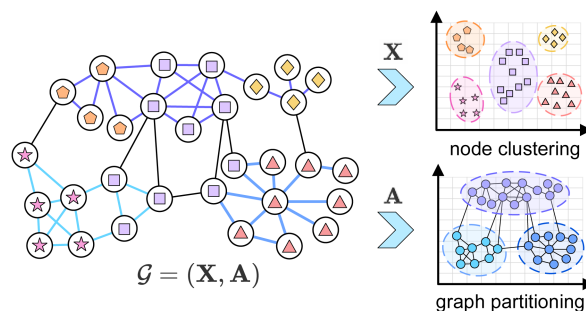


Fig. 4. Node clustering and graph partitioning. The former clusters nodes mainly by feature similarity on \mathbf{X} , marked by node shapes and colors. The latter clusters nodes mainly by connection density on \mathbf{A} , marked by edge colors.

Efficiency analysis: The efficiency of motif learning depends on the algorithm for obtaining motif pseudo-labels. For example, the IFG algorithm [159] called by RDKit has a time complexity $\mathcal{O}(n + |\mathcal{E}|)$ for every molecule. Fragmentation methods traverse all nodes within a graph and map them to supernodes with extra $\mathcal{O}(n)$ time. However, it is worth noting that the diverse range of motifs can lead to large motif dictionaries, incurring extra memory overhead.

D. Clusters

Cluster-based tasks aim to learn representations that capture the inherent clustering structure of the graph, which can be defined based on either feature similarities or link connectivity. Clusters often possess a larger scale compared to motifs, providing a higher-level view of the graph structure.

Node clustering: Node clustering, a classic unsupervised learning task, is introduced as a pretext task by M3S [26] and NodeCluster [33]. They leverage feature-based clustering algorithms (e.g., K -means [160], DeepCluster [161]) to assign a cluster pseudo-label to every node. HomoGCL [103] and MGSE [104] first generate a prototype vector for every cluster by feature aggregation. Then, they optimize an MSE and a cross-entropy-based divergence loss of the cluster assignment probabilities, respectively. CARL-G [105] predicts “cluster validation indices”, a set of measures indicating the compactness and separation of clusters. CommDGI [27], S³-CL [81] and more [103], [106] focus on *cluster-based discrimination* where node embeddings are contrasted with learnable cluster prototypes. GraphLoG [102] models the hierarchical nature of clustering by setting prototypes at different levels and organizing them in a tree structure.

Graph partitioning: Graph partitioning is also known as “non-overlapping community detection” in the scenario of social network mining. Unlike node clustering, graph partitioning is based on structural community patterns and thus is available to unattributed graphs, illustrated in Fig. 4. Early works [33], [61] leverage unsupervised community detection methods, such as spectral clustering and Louvain [162], to generate partition pseudo-labels and learn a community indicator matrix. Distance2Clusters [34] performs a regression task between node representations and community prototypes. Several works have incorporated graph partitioning into more complex frameworks,

e.g., *partition-based discrimination* [109], [110], [111], link prediction VGAE [107] and masked autoencoders [108].

Discussion: Cluster-based pretexts provide graph models with a deeper understanding of the higher-level structural organization. However, most cluster-based pretexts rely on non-overlapping algorithms, assuming that each node belongs to a single cluster. In real-world scenarios, nodes often belong to multiple overlapping communities, which remains a challenge for existing graph models.

Efficiency analysis: The computational cost of clustering/partitioning algorithms can become prohibitive for large networks. Common implementations of K -means have the time complexity of $\mathcal{O}(Knd)$ for every iteration. For graph partitioning, vanilla spectral clustering reaches $\mathcal{O}(n^3)$ time complexity, while the modularity-based algorithms such as Louvain [162] are believed to run in $\mathcal{O}(|\mathcal{E}|)$ time. In practice, however, they still spend a lot of time processing large-scale networks, so clustering methods need to be considered carefully.

V. MACROSCOPIC PRE-TRAINING TASKS

Unlike mesoscopic tasks that focus on the local graph structure, macroscopic pre-training tasks aim to capture global patterns and structures that span the entire graph. These pretexts are designed for a broader understanding of the overall organization and dynamics of the graph.

A. Global Structure

The goal of global structure-based tasks is to capture the overall topology and properties of a graph by learning from its global representations.

Graph instance discrimination: This task learns to distinguish between graph instances by focusing on graph-level representations. They are obtained by aggregating node embeddings by a simple readout function, such as mean pooling and summation. GraphCL [112] matches positive and negative sample pairs from batches of small graphs with an InfoNCE estimator (5), similar to node instance discrimination. Other MI estimators are also suitable for graph instances, such as triplet loss [40] and bootstrapping loss [115]. Subsequent works have explored various aspects of graph instance discrimination, such as adaptive augmentations [113], [114] and negative sample mining [121].

Graph representations can also be used to perform *node-graph discrimination*, also known as cross-scale contrast [3]. Here the global representation can form positive pairs with every node in the graph, and negative pairs are generated by applying one-sided perturbations to either the node or the graph representation. The JS estimator (4) is in widespread use here, pioneered by DGI [13] and InfoGraph [116]. MVGRL [117] performs cross-view contrast by both node-level and graph-level perturbations. GGD [119] and D-SLA [120] propose *group discrimination*, a simplified binary classification approach predicting whether an instance belongs to the original or the perturbed view. This simplification greatly improves the efficiency, as calculating similarities between graph instances is no longer needed. Node-graph discrimination captures the relationships between global

and local representations of a graph, making it applicable to both small and large graphs [76], [122].

Graph similarity prediction: This pretext task leverages various kinds of graph-level similarity functions to learn graph-level representations, first envisioned by [32]. KernelPred [123] predicts various graph kernels, including the graphlet kernel, random walk kernel, WL subtree kernel, etc. These kernels capture different aspects of graph similarity, such as structural similarity, node proximity, and subgraph patterns. D-SLA [120] generates a perturbed graph by adding and removing edges and predicts the graph edit distance kernel – the number of edge modifying steps between the original and perturbed graphs. HTML [124] predicts the isomorphic similarity between graphs based on the Jaccard coefficient.

Discussion: Global structure-based tasks offer a holistic view of the entire graph, capturing its overall topology and properties. This is particularly advantageous when dealing with small graphs or scenarios focusing on global properties, including tasks such as graph classification and graph regression. However, the readout functions used to generate global representations can be coarse-grained, potentially losing important structural information. Moreover, graph perturbations can have a significant impact on the global semantics of small graphs.

Efficiency analysis: Although graph instance discrimination methods define readout functions on the entire graph, their simple formulations imply acceptable computational costs (usually less than $\mathcal{O}(n)$). However, calculating graph kernels on large networks is not an easy task, which makes prediction tasks more constrained by the data size.

B. Manifolds

Manifolds are underlying global topological patterns that euclidean spaces struggle to represent. Recent work explores embedding graphs into non-euclidean manifolds, such as hyperbolic or spherical spaces, to better model hierarchical and tree-like structures of graph data.

Cross-manifold discrimination creates contrastive views in different manifolds, thereby capturing the unique properties of each manifold and their relationships. HGCL [125] uses a pair of hyperbolic GNNs to encode views of the graph, and DSGC [24] uses both euclidean and hyperbolic GNNs to obtain views in both spaces. RiemannGFM [129] contrasts between a hyperbolic and a spherical space. SelfMGNN [126] embeds graphs into a product space that combines euclidean, hyperbolic, and spherical spaces. It enables adaptive learning of the most suitable manifold for each graph based on its structural properties. For prediction tasks, HDM-GAE [128] performs masked feature/link prediction in the hyperbolic space. Graph-JEPA [127] first expresses graph representations as angle vectors in a unit hyperbola and predicts them by a smooth- ℓ_1 loss.

Discussion: Manifold-based tasks offer a promising new direction by capturing complex geometric structures and hierarchical relationships that are difficult to represent in euclidean spaces. There is ample room for exploration, such as investigating more general and flexible approaches for graph manifold

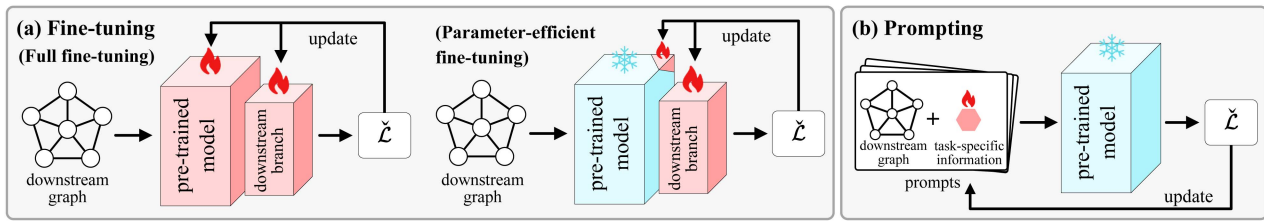


Fig. 5. Different downstream tuning strategies. 🔥: tuned; ❄️: frozen. (a) Graph fine-tuning, where the pre-trained parameters are updated along with downstream branches through the downstream task $\tilde{\mathcal{L}}$. While full fine-tuning updates the entire set of pre-trained parameters, parameter-efficient fine-tuning only updates a small amount of it via specifically designed adapter modules. (b) Graph prompting, where the specifically designed graph prompts are updated, usually carrying downstream task-specific information. Tunable downstream branches are not necessary.

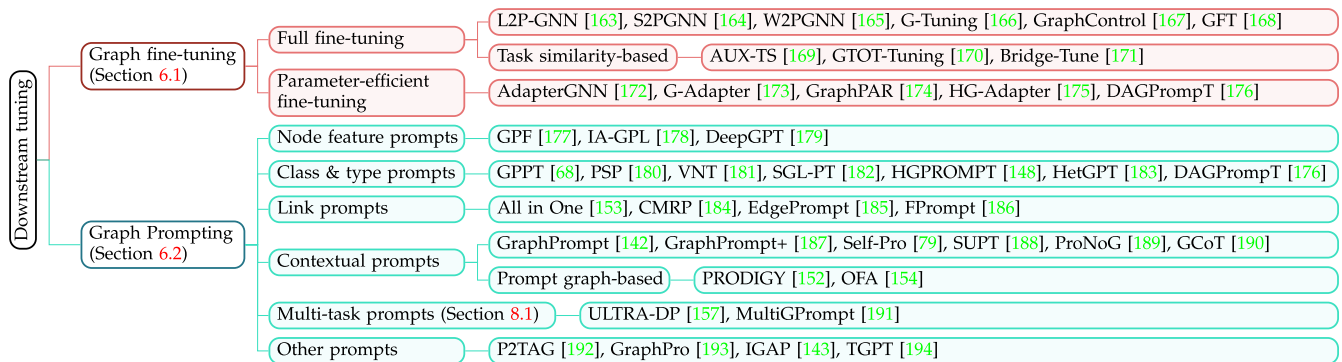


Fig. 6. Our taxonomy of downstream tuning strategies with representative literature.

learning and exploring the integration of different manifolds other than a product space.

Efficiency analysis: The time overhead can arise from the switching between different topological spaces. For example, the exponential and logarithmic maps in hyperbolic spaces require multiple calculations of vector norms and inverse trigonometric functions. Although these operations do not alter the upper bound of the algorithm complexity, they introduce additional computations that increase the actual wall-clock time. Furthermore, storing representations in different spaces incurs additional memory costs.

VI. DOWNSTREAM TUNING

Downstream tuning in self-supervised graph models focuses on transferring the knowledge learned from self-supervised pretexts to downstream tasks, as formalized in (1). This section explores two main approaches: graph fine-tuning and graph prompting, illustrated in Figs. 5 and 6. These approaches offer different ways to leverage the pre-trained graph model for specific applications.

A. Graph Fine-Tuning

Fine-tuning adapts pre-trained models to downstream tasks by jointly training them with a generally simple task-specific branch. Traditional GNN fine-tuning methods, known as *full fine-tuning*, update all pre-trained parameters. To improve knowledge transfer, advanced techniques have emerged. For instance, L2P-GNN [163] uses a meta-learning framework that

divides the pre-training data into support and query sets, simulating the adaptation process during pre-training. S2PGNN [164] decomposes fine-tuning into multiple function modules and dynamically identifies the optimal modules for different downstream tasks. W2PGNN [165] and G-Tuning [166] focus on cross-domain transferability by representing fine-tuning as finding a combination of graphon bases. They serve as different dimensions of fundamental transferable patterns across data spaces. GraphControl [167] incorporates a conditional control module to utilize downstream task-specific features effectively. GFT [168] rearranges the input graph data as trees with a virtual root node encoding task-specific information. In this way, downstream task-relevant nodes serve as children of the root node, and their information can be aggregated for various prediction tasks.

A specific line of work quantifies the generalization gap by the *task similarity* between pre-training \mathcal{L} and downstream tasks $\tilde{\mathcal{L}}$. GTOT-Tuning [170] models graph fine-tuning as an optimal transport problem and minimizes the masked Wasserstein distance between tasks. AUX-TS [169] introduces gradient similarity $sim(\mathcal{L}, \tilde{\mathcal{L}}) = \langle \nabla_{\theta} \mathcal{L}, \nabla_{\theta} \tilde{\mathcal{L}} \rangle$ which measures the similarity of loss surfaces between two tasks. If the similarity is positive, it indicates that the optimization directions during the gradient descent are non-conflicting, so two tasks are similar; and vice versa. Bridge-Tune [171] defines representation consistency, the similarity between pairwise node label distributions. A binary label is assigned to each pair of nodes determined by whether their pretext pseudo-labels (or downstream labels) are the same.

Fine-tuning large-scale models can be computationally expensive, and biases from downstream tasks may compromise generalizability. To address these challenges, recent works adopt *parameter-efficient fine-tuning* (PEFT) modules such as adapter [195], [196], which enable models to update only a small subset of pre-trained parameters during fine-tuning. AdapterGNN [172] and G-Adapter [173] introduce adapter modules tailored for GNNs and graph Transformers, respectively. DAGPrompt [176] proposes the Graph Low-Rank Adaptation (GLoRA) module. Different from LoRA [196], both the weights \mathbf{W} and the adjacency matrix \mathbf{A} in GLoRA are decomposed into two low-rank projection matrices $\mathbf{P}, \mathbf{Q} \in \mathbb{R}^{d \times r}$ and $\mathbf{P}_A, \mathbf{Q}_A \in \mathbb{R}^{n \times 1}$.

Discussion: Considering the gap between general graph knowledge and domain-specific downstream knowledge, pre-training and fine-tuning are currently indispensable for building a general graph model. However, graph fine-tuning methods often resort to specific designs in terms of tuning processes and model architectures, limiting their universality across different downstream scenarios. Moreover, fine-tuning the pre-trained parameters may harm the generalization and expressive power of the pre-trained model. Despite that PEFT methods enable precise fine-tuning with minimal resource requirements, they are less common in fine-tuning pure GNNs as they are relatively small in size.

B. Graph Prompting

Prompting is an emerging downstream tuning strategy that has gained popularity with the rise of LLMs. In the graph domain, prompting jointly encodes downstream graph data and corresponding task-specific information as additional learnable components called “prompts”. During downstream training, only the learnable part of the prompts is updated, while the pre-trained model remains frozen. To this end, graph prompts should be first integrated into the downstream data before downstream training by various means (addition [177], element-wise multiplication [142], concatenation [181], weighted aggregation [153], linear transformation [142], etc.), depending on the form of prompts and specific downstream requirements. However, unlike prompts in natural language that follow a deterministic form, graph prompts can take various shapes, increasing the difficulty of prompt design. The following will discuss several graph prompt designs from a knowledge-based perspective.

Node feature prompts: Node feature prompts are learnable vectors $\mathbf{p} \in \mathbb{R}^d$ that share the same size with the node features. Node feature prompts are simple, effective, and generalizable, serving as an important cornerstone of graph prompting. The fundamentality of node features enables their generalization across different data domains and downstream tasks, such as node classification, link prediction, and graph classification. GPF [177] simply adds the node feature prompt $\mathbf{p} \in \mathbb{R}^d$ to every row of the downstream feature matrix, formally $\tilde{\mathbf{X}} \leftarrow [\tilde{\mathbf{X}}_i + \mathbf{p}]_{i \in \mathcal{V}}$ to perform supervised fine-tuning. GPF uses a universal feature prompt for every node, while its variant GPF-plus [177] assigns an independent prompt \mathbf{p}_i to each node generated by a set of learnable prompt bases. IA-GPL [178] generates a feature

prompt for every input node from its representation through a vector quantization-based network. Aside from feature prompts, DeepGPT [179] prepends prefix prompts to every embedding fed into a pre-trained graph Transformer.

Class & type prompts: Class prompts are prototype vectors aggregated from pre-trained node representations that share the same class: $\mathbf{p}_c = \text{agg}(\mathbf{Z}_i | i \in \mathcal{V}, y_i = c)$. They are associated with downstream node classes. Downstream tasks such as node classification can be achieved by matching node representations with these class prompts with cross-entropy [68], [181] or InfoNCE (5) [180], [182]. The selection of objectives often depends on the form of pretexts.

GPPT [68] first divides the original graph into different clusters using node clustering algorithms (Section IV-D), and then defines a set of independent class prompts within each cluster. Unlike GPPT which constructs independent node-class prompt pairs, PSP [180] connects class prototypes to the original graph as virtual class nodes and fine-tunes them by contrastive learning. VNT [181] employs meta-learning on graph data with virtual class nodes, where the class prompts for each target task are aggregated from source tasks through an attention mechanism, aiming to bridge the gap between the source and target domains. SGL-PT [182] leverages masked feature prediction to fine-tune class prompts. The graph classification problem is transformed into the reconstruction of a masked virtual supernode, which serves as the global representation. DAGPrompt [176] concatenates the embeddings as well as class prompts from every GNN layer, and performs InfoNCE-based similarity learning. Type prompting-based methods, such as HGPPROMPT [148] and HetGPT [183], assign a prompt to each node type, similar to class prototypes. Unlike class prompts, node types are considered node properties that are common in heterogeneous graphs and typically do not require manual labeling.

Link prompts: Considering the structural knowledge carried by downstream graph data is largely overlooked by node feature-based prompts, prompts on adjacency matrices begin to thrive. EdgePrompt [185] proposes edge prompts that are learnable attributes on every edge and are integrated into node representations through message propagation. Similar to GPF, edge prompts can be either one shared vector or customized vectors generated by prompt bases. All in One [153] considers pairwise relationships between prompt tokens and constructs a graph prompt $\mathcal{G}_p = (\mathbf{X}_p, \mathbf{A}_p)$, where $\mathbf{X}_p = [\mathbf{p}_k]_k$ and $\mathbf{A}_p = [(\mathbf{p}_k, \mathbf{p}_l)]_{k,l}$. A meta-learning strategy is developed to adapt All in One to miscellaneous downstream scenarios.

Contextual prompts: Context information is often considered in prompting methods in the form of aggregated neighborhood features or embeddings. GPPT [68] and GraphPrompt series [142], [187] design structural prompts that encode one-hop aggregated contextual information for downstream tuning. Self-Pro [79] constructs a 2-hop adjacency matrix $\mathbf{A}_2 = [A_{i,j} = 1 \cap A_{j,k} = 1]_{i,k}$ as the contextual prompt. SUPT [188] builds upon the learnable prompt bases in GPF-plus [177]. However, it uses a message-passing approach to aggregate these bases, preserving the semantic similarity among neighboring nodes in the prompts. ProNoG [189] generates contextual prompts by employing a condition-net, where the input representations are

aggregated from k -hop subgraphs. Building on this, GCoT [190] iteratively aggregates representations from every hidden layer of a pre-trained GNN as “thoughts”, simulating the Chain-of-Thought reasoning in large language models.

Some prompting methods move away from feature vectors and seek other effective forms. For example, PRODIGY [152] and OFA [154] construct a prompt graph², in which each prompt node (data node) represents a sampled k -hop subgraph and each class node represents a class to which the central node belongs. Links between prompt nodes and class nodes indicate the task-specific supervision signals. Learning to predict these links has been demonstrated to be a highly generalizable strategy, applicable both to pre-training and fine-tuning to facilitate both few-shot in-context learning [152] and zero-shot learning [154]. Besides, PRODIGY updates the prompt graph with an auxiliary self-supervised context prediction objective: it predicts if one node belongs to the k -hop subgraph of another target node.

Other prompts: Some cutting-edge prompting methods explore underlying topological properties of graph data. IGAP [143] designs a spectral prompt to transform the low-dimensional pre-training domain to the fine-tuning domain, as the low-frequency domain describes local smooth patterns of graph signals. TGPT [194] captures graphlet information – small motifs that describe local structure patterns of a node – into its node-level and the graph-level prompts. Specifically, node-level topology-aware prompts are generated from a fast graphlet transform matrix, and graph-level ones are aggregated from them.

Discussion: Graph prompting is a novel approach to envisioning graph downstream tuning. It can be viewed as a form of PEFT on graph data instead of model architecture, where prompts are considered tunable adapters across the pre-training and downstream data spaces, achieving both effectiveness and efficiency. Prompting not only bridges data domains but also provides unified fine-tuning templates for various downstream prediction tasks. Such templates include link prediction [79], [152], [154], [176], graph classification [153], subgraph similarity prediction [142], [187], and more. Despite the promising advancements, graph prompting remains a developing area, offering opportunities for further research and improvement. For instance, many graph prompts remain challenging for humans to comprehend, posing challenges for the explainability of graph prompting. Additionally, most proposed “unified prompt templates for downstream tasks” primarily focus on discriminative tasks such as link prediction and graph classification, while neglecting generative and other open-ended tasks that also have widespread demand.

VII. SELF-SUPERVISED GRAPH LANGUAGE MODELS

Previous sections have explored how self-supervised GFMs learn different types of graph knowledge through pre-training

²Unlike the existing survey [197] which categorizes both All in One [153] and PRODIGY [152] as “Prompt as Graphs”, we explicitly distinguish them by different notions: “*graph prompt*”, a graph added on the downstream graph as an entire prompt; and “*prompt graph*”, a new graph comprised of prompt nodes and class nodes.

and downstream tuning. The emergence of large language models (LLMs) has opened up new avenues for constructing graph language models (GLMs) that leverage knowledge patterns, architectures, and training strategies from the natural language domain to process graph data. While traditional GFMs excel at capturing structural patterns, GLMs aim to bridge the gap between graph topology and semantic understanding by combining the strengths of both GNNs and language models.

This section examines self-supervised GLMs from two perspectives shown in Fig. 7: (1) pre-training GLMs with self-supervision, which focuses on incorporating graph knowledge into language modeling pre-training and graph pre-training of GLMs; and (2) tuning GLMs, which focuses on adapting pre-trained language knowledge to graph-specific scenarios through techniques like prompting and fine-tuning. In what follows, we delve into these two directions, particularly focusing on how GLMs integrate different forms of graph knowledge.

A. Pre-Training GLMs With Self-Supervision

GLM pre-training includes two distinct categories: *language modeling pre-training*, and *graph pre-training*. Note that both pre-training schemes can be applied to graph models such as GNNs and graph transformers, as well as to open-source language models (LMs) like T5 [256] and BERT [130].

1) *Language Modeling Pre-Training:* Language modeling pre-training refers to a set of self-supervised pretext tasks in the language domain, known as “language modeling”. They mainly include *autoregressive language modeling* (AR) and *masked language modeling* (MLM). AR, also known as causal language modeling and next-token prediction, predicts the next token in a sequence usually by a maximum likelihood estimation loss. The well-known AR models include GPT series [257], LLaMA [258], and DeepSeek [259]. MLM, exemplified by BERT [130] and RoBERTa [260], predicts masked tokens using bidirectional context, making it ideal for understanding tasks but less suited for generation. Additionally, hybrid methods such as T5 [256] and BART [261] combine AR and MLM to handle both understanding and generation, though at the cost of higher complexity.

Many early GLMs directly utilize open-source or closed-source LLMs pre-trained by the aforementioned tasks. Although pure LLMs have demonstrated preliminary abilities in handling and reasoning on graphs [208], the modality gap between text and graphs makes graph tasks challenging for LLMs without additional support [209], [210]. Therefore, several attempts integrate GNNs as components of LLMs and jointly pre-train them by AR and MLM. UniGraph [132] and P2TAG [192] concatenate an LM and a GNN together and perform MLM on textual node attributes. THLM [198] jointly pre-trains a BERT and a heterogeneous GNN by MLM. On top of this, graph-oriented language pre-training methods are developed. Patton [147] improves traditional MLM to contextualized MLM: it utilizes both representations of the current node and its neighboring nodes to predict the missing tokens of the current node. Path-LLM [199] first generates textual sequences by interconnecting node text in

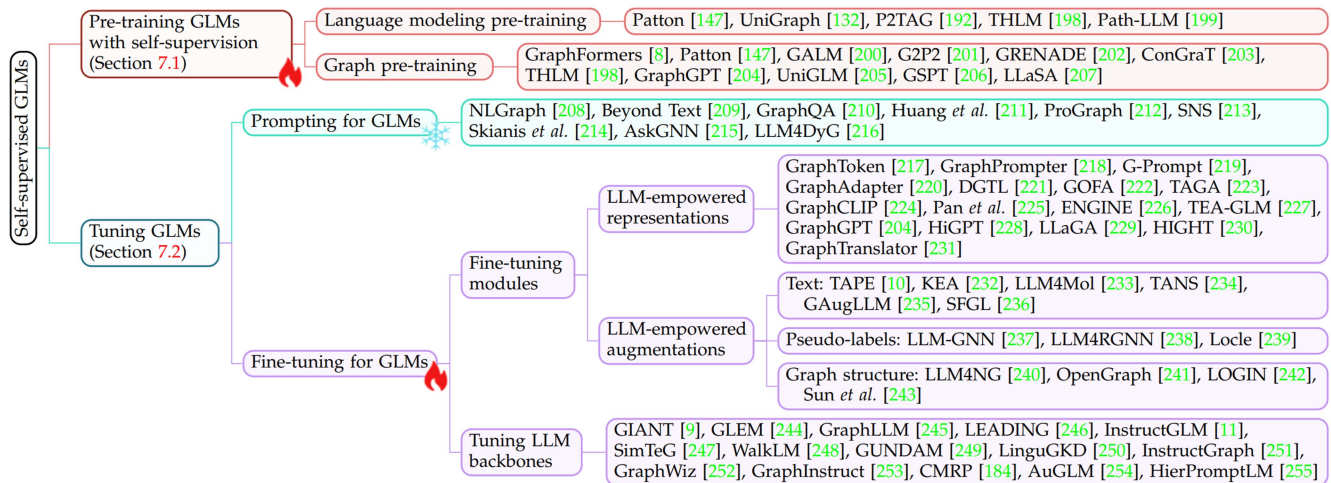


Fig. 7. Our taxonomy of self-supervised GLMs with representative literature. 🔥: network parameter is updated; ❄️: parameter frozen.

the same order as the shortest path, where the earlier node text becomes the prefix tokens for AR pre-training.

2) *Graph Pre-Training*: Graph pre-training features self-supervised pretext tasks in the graph domain, as introduced in Section III–V. They are adapted to pre-train or assist in pre-training language model architectures. Cross-entropy-based and InfoNCE-based link prediction (Section III-C) become the first choice: GALM [200] concatenates an LM and a GNN for joint pre-training, while GraphFormers [8] and Patton [147] incorporate GNN and Transformer layers into a hybrid framework. UniGLM [205] employs node instance discrimination (Section III-A) to pre-train a language model, where positive node samples are selected from multiple hops of the central node by Personalized PageRank.

In contrast to single-modal discrimination, *node-text discrimination* is a more common approach inspired by existing multi-modal pre-training models like CLIP [262]. This approach discriminates between representations derived from a GNN and an LM/LLM to align the two modalities. GraphGPT [204] trains a textual and a graph Transformer by minimizing a cross-entropy-based discrimination loss to align their representation spaces for future instruction tuning. LLaSA [207] unifies different types of structured data into hypergraphs and performs contrastive learning between a hypergraph GNN and a hybrid Transformer.

To achieve more effective modality alignment, researchers agree on the necessity of incorporating higher-order graph-specific knowledge during the self-supervised pre-training. Graph context (Section IV-A) becomes especially crucial for GLMs, as it is easy for LMs to understand compared to other structural knowledge, and provides a clearer perspective of the textual semantics of node entities and their adjacency relationships. G2P2 [201] jointly pre-trains a Transformer and a GCN by context-aware node-text discrimination: it generates a summary embedding by averaging neighborhood text embeddings and performs discrimination among node, text, and summary. GRENADE [202] employs both node-level and neighborhood-level discrimination within and between GNN and BERT. It minimizes the KL divergence of neighbor

similarity distributions. THLM [198] introduces an auxiliary objective (along with MLM) to differentiate contextual and distant nodes. Another common graph knowledge is long-range similarities (Section IV-B). ConGraT [203] jointly pre-trains a sentence Transformer and a GAT by node-text discrimination with a long-range similarity function, namely the number of common neighbors and SimRank. GSPT [206] pre-trains a Transformer by masked feature prediction (Section III-A) on generated random walk sequences. They employ a node feature reconstruction loss rather than predicting the masked token IDs as in MLM.

B. Tuning GLMs

Downstream tuning is the dominant approach for deploying GLMs, given the high computational demands of LLM pre-training. LLMs to be tuned are usually pre-trained by large-scale language modeling pretexts, endowing them with a certain level of world knowledge. They can be utilized for either adapting to graph-specific downstream scenarios or guiding the training of other downstream branches. Some literature categorizes the roles of LLMs as enhancers, predictors, encoders, aligners, etc. [19], [232], [263], viewing pre-trained LLMs as auxiliary modules. Instead, we consider pre-trained LLMs as the core model and treat all subsequent training processes as downstream tuning. This perspective allows us to more clearly demonstrate how graph-specific knowledge assists LLMs in graph downstream tuning.

Note that GLMs discussed in this section do not necessarily employ self-supervised pre-training, and they may be designed for only one specific downstream task. However, here we focus solely on the tuning strategies employed, which we believe have the potential to be applied to self-supervised GLMs or to provide valuable insights.

1) *Prompting for GLMs*: With the advent of LLMs, the earliest GLMs attempt to directly utilize closed-source LLMs such as GPT-3.5 to construct graph learners. Specifically, by designing particular graph-specific prompts, LLMs can get a grasp of

key graph knowledge for open-ended tasks such as question answering and reasoning. Some methods also rely on in-context learning [215], [264], i.e., to provide a few task-specific examples within the prompts to guide the LLM output. Many early attempts, including empirical studies on using LLMs for graph tasks [208], [209], [210], [211], [212], resort to graph-specific prompts rather than fine-tuning strategies.

To achieve effective prompting, the first step is to convert graphs into text using natural language or structured language [11], [264]. However, merely inputting textualized graphs is insufficient for LLMs to fully comprehend graph structural knowledge. Various methods have been proposed to further embed graph knowledge into prompts. For instance, SNS [213] ranks the textual similarity between nodes and selects the top-2 similar neighbors as additional instructions. Skianis et al. [214] textualize the reasoning process of various graph question-answering tasks as pseudo-code functions. AskGNN [215] employs a GNN to select an optimal set of examples for in-context learning.

2) *Fine-Tuning for GLMs*: The rise of open-source LLMs, coupled with the limitations of prompting models in terms of cross-modal and zero-shot generalization capabilities, has facilitated the research of graph-specific fine-tuning for GLMs. *Instruction tuning* is the primary fine-tuning approach for LLMs, wherein appropriate queries for downstream tasks are constructed to elicit the desired LLM predictions, and model parameters are updated by minimizing an error function. However, LLMs typically possess a larger parameter scale than GNNs or GTs, making it challenging for GLMs to perform full fine-tuning. Consequently, parameter-efficient fine-tuning (PEFT) methods have become the predominant approach, where the LLM is frozen and a special fine-tuning module is tuned instead. Apart from intrinsic adapters such as LoRA [196], GLMs often incorporate tailored fine-tuning modules to (1) maximize the retention of pre-trained LLMs' powerful semantic encoding ability, and (2) align the semantic spaces of text and graphs. Types, interfaces, and tuning strategies of fine-tuning modules vary.

Module types: GNNs vs. Non-GNNs: The most commonly used fine-tuning module is GNN, which can capture structural knowledge of graphs, thereby helping to bridge the modality gap. The training of these modules can rely on both downstream task-relevant information and self-supervision signals. GraphToken [217] and GraphPrompter [218] utilize a GNN to generate graph tokens as input for LLMs, and tune³ the GNN by answering graph-related questions. Conversely, G-Prompt [219] uses MLM to tune a GNN following an LM to obtain graph representations through textual prompts. GraphAdapter [220] tunes a GNN by AR as a graph learning branch of a pre-trained LM. Then, it merges the representations with an MLP-based fusion block before passing them to a downstream head. GOFA [222] interleaves GNN layers into a pre-trained LLM encoder as

³By “tuning” we mean that the GNN serves as an auxiliary module of a pre-trained LLM. From the perspective of the GNN, it may be trained from scratch and regarded as pre-training in the original paper. This does not conflict with our statements in this section.

adapters and employs various self-supervised fine-tuning methods, including AR, shortest path distance prediction, and common neighbor prediction. DGTL [221] first generates disentangled graph embeddings by assigning diverse edge weights to each GNN layer. These embeddings are then incorporated into the input tokens of each LLM layer to perform context-aware instruction tuning. TAGA [223] and GraphCLIP [224] employ node-text discrimination fine-tuning to align graph and text. TEA-GLM [227] tunes a GNN using both node instance and dimension discrimination (Section III-A) between GNN embeddings and PCA-processed LLM embeddings. Some methods have developed more complex fine-tuning modules based on GNNs. Pan et al. [225] use a pair of GNNs for two-stage tuning. It first trains a GNN “interpreter” supervised by LLM-extracted keyword information, and then distills the interpreter to another downstream GNN. ENGINE [226] develops a bypass fine-tuning architecture for LLMs, “G-Ladders”, which consists of multiple levels of GNNs and projectors. This structure enhances computational efficiency through node embedding caching.

Unlike GNNs, non-GNN fine-tuning module types such as MLPs lack the ability to handle local graph structures. To achieve modality alignment, these methods often resort to specialized tuning schemes. GraphGPT [204] tunes a linear adapter through “*graph-instruction matching*” before downstream instruction tuning. The LLM is instructed to reorder the list of node text to match textual embeddings obtained from a parallel GNN+Transformer encoder. LLaGA [229] extracts contextual knowledge from a graph using two tokenizers: node embedding concatenation through level-order traversal on a neighborhood tree, and neighborhood embedding aggregation in different hops. HIGHT [230] develops a hierarchical graph tokenizer to combine node, motif, and graph-level information for instruction tuning. GraphTranslator [231] tunes an attention-based adapter, the “translator”, to project graph embeddings into the LLM space. It is guided by LLM-generated text that describes various knowledge patterns such as summaries of nodes, neighborhoods, and other commonalities.

Interfaces: representations vs. augmentations: We have discussed GLMs that leverage LLM-empowered representations for fine-tuning. LLMs can also generate augmented data components for fine-tuning modules. They include:

1) *Text*: TAPE [10] instructs an LLM to explain its decisions in classifying nodes, facilitating an in-depth understanding of node-level information. These explanations are then encoded by a smaller LM to enrich textual features for downstream GNN training. Unlike TAPE which directly uses original adjacency matrices, SFGL [236] generates scale-free graphs from the input data to obtain textual features more aligned with real-world edge distributions. KEA [232] instructs an LLM to generate descriptions of terminologies across different fields. LLM4Mol [233] uses ChatGPT to generate descriptions of chemical molecules, including functional groups and other properties, to fine-tune a small-scale RoBERTa. TANS [234] leverages centralities and clustering coefficients (Section III-B), along with contextual information, to generate descriptive text for nodes using GPT-4o-mini for non-textual graphs. GAUGLLM [235] enhances

textual representations with node context summaries, and uses them to guide feature-level and edge-level augmentations in self-supervised fine-tuning tasks such as node instance discrimination and masked feature prediction.

2) *Pseudo-labels*: LLM-GNN [237] leverages the LLM to generate cluster-aware node pseudo-labels to supervise a downstream GNN, referred to as *label-free node classification*. A set of nodes closer to K -means cluster centers is selected for annotation based on a cluster density metric. Similarly, Locle [239] uses subspace clustering to find the node set. LLM pseudo-labels are then selected based on their information certainty and refined by graph rewiring.

3) *Graph structure*: LLM4NG [240] and OpenGraph [241] both use a pre-trained LLM to generate node samples and associated links to augment the original graph data. While LLM4NG trains an MLP-based edge predictor via cross-entropy-based link prediction, OpenGraph generates new edges through Gibbs sampling and trains a GT by masked link prediction. Sun et al. [243] utilize GPT-3.5-Turbo to assist in removing unreliable edges and adding reliable ones to create a refined graph for GNN input. LOGIN [242] treats the LLM as a consultant for node classification during GNN fine-tuning. If the LLM prediction matches the ground truth, it updates the original feature; if not, it prunes its associated links based on neighbor similarity.

Backbone parameters: frozen vs. tuned: We have discussed GLMs with additional fine-tuning modules, where the LLM backbone remains frozen. Another line of studies designs special textual and graph domain co-training strategies to fine-tune the LLM backbone. GLEM [244] iteratively tunes a GNN and a DeBERTa model using a variational Expectation-Maximization framework. GraphLLM [245] synergistically tunes a GT and a LLaMA 2 model via “prefix-tuning”, i.e., to project graph representations into trainable tokens and prepend them to the keys and values of every Transformer attention layer. LEADING [246] reduces the cost of joint LLM-GNN fine-tuning by decoupling the computation of node embedding from neighborhood embeddings. Instead of joint tuning, SimTeG [247] separates the tuning of LMs and GNNs in a two-stage manner for more distinguishable embedding spaces.

Among all types of graph structural knowledge, link information (Section III-C) is often explicitly extracted to fine-tune LLMs, as it directly indicates relationships between entities and can be easily extracted by both GNNs and LLMs. InstructGLM [11] introduces link prediction into LLM instruction tuning as an auxiliary objective, similar to the training of SuperGAT [72]. CMRP [184] tunes an LLM and a GNN to dynamically select an optimal edge set. The generated edge set is then injected into prompt tokens to iteratively instruct the LLM. GIANT [9] presents *neighborhood prediction* to fine-tune an XR-Transformer, which constructs hierarchical clusters and predicts the rows of the adjacency matrix as a multi-class classification task. AuGLM [254] selects and summarizes neighboring nodes by Personalized PageRank scoring and GNN-generated class prototypes for instruction tuning. Another type of knowledge information involves long-range paths (Section IV-B). WalkLM [248] incorporates long-range information into MLM to fine-tune a DistilRoBERTa model: it first samples attributed

random walk sequences and then textualizes them as a token list. GUNDAM [249] fine-tunes an LLM by reasoning different paths between two nodes, with path labels generated by unsupervised algorithms. LinguGKD [250] develops a distillation architecture for LLM-GNNs: it first leverages node degrees and k-hop neighbors to instruct-tune an LLM. Then, the GNN is fine-tuned by contrasting between every intermediate layer of both models. InstructGraph [251], GraphWiz [252], and GraphInstruct [253] conduct multi-task LLM fine-tuning by combining over a dozen instruction tuning tasks. They include self-supervised predictions on various graph structures (cycles, shortest paths, maximum flows, Hamilton paths, etc.) as well as node-level and link-level downstream signals, aiming to provide the LLM with a comprehensive understanding of the graph domain.

Discussion: While both GLM pre-training and tuning strategies demonstrate the potential of bridging the modality gap between graph and text, it remains underexplored whether GLMs have adequately tapped the potential in LLMs with billion-scale parameters. Some excellent properties of LLMs, e.g., the emergent ability [265], are yet to be discovered on graph model architectures. The fragile side of LLMs such as hallucinations [266] and intervention of spurious factors [208] keeps posing challenges to GLM researchers. To tackle these challenges, future research should focus on developing more powerful graph-specific architectures as well as pre-training and fine-tuning strategies that effectively extract and leverage various types of graph knowledge.

VIII. CHALLENGES AND FUTURE DIRECTIONS

This section discusses potential challenges that graph researchers may encounter and insights for future research directions towards GFMs, as a conclusion to our survey.

A. Combining Different Graph Knowledge Patterns

Despite that a variety of pretext tasks have been proposed for self-supervised graph models, the effectiveness of these pretexts depends on the application scenarios applied to downstream tasks [267]. Therefore, it is crucial to investigate how to effectively combine different graph knowledge to adapt graph models to more application scenarios.

One research direction is to jointly optimize different pretext objectives to obtain different aspects of graph knowledge. This can be formulated as a *multi-task pre-training* problem. Traditional methods simply assign hyperparameters to weigh each pretext task, leading to suboptimal performance. AutoSSL [267] and ParetoGNN [268] are neural parameter search algorithms in order to dynamically find a set of optimal coefficients to combine different pretexts. GraphTCM [269] models a correlation value for every pair of pretexts and optimizes the correlation matrix to find the best parameters. AGSSL [270] and WAS [271] propose a knowledge distillation technique, where the knowledge from different teachers is distilled into a single unified student. Another promising approach is to design specific *multi-task prompting* strategies. ULTRA-DP [157] and MultiGPrompt [191] are multi-task prompting methods that assign a learnable task prompt for

each pretext and pass them to the downstream model. While ULTRA-DP searches for the best task prompt for the downstream task, MultiGPrompt combines all of them by a linear combination or a parameterized network.

B. Knowledge Adaptation Across Graph Types

It is important for future GFMs to explore a wider range of graph knowledge, handling various complex data types beyond simple graphs. Here, we focus on several graph types and their corresponding knowledge patterns. We will show that these patterns and processing methods share commonalities with the graph knowledge discussed in Sections III–V, implying potential data unification in future GFMs.

Heterogeneous & knowledge graphs: Nodes and links in heterogeneous graphs possess unique types that exhibit distinct knowledge patterns. Some existing methods distinguish and handle different node and edge types. For instance, Heterformer [272] and RMR [273] are node instance discrimination methods within specific node and relation types, respectively. HiGPT [228] is a GLM that pre-trains an LLM by matching node types within sampled heterogeneous subgraphs. For downstream tuning, HG-Adapter [175] is a PEFT approach that constructs a heterogeneous structure by calculating correlation scores for every neighboring node type to fine-tune an adapter. HGPROMPT [148] splits a heterogeneous graph into multiple type-specific subgraphs and designs a learnable prompt vector for each node type.

However, these methods do not explicitly mine the higher-order structured knowledge underlying heterogeneous types. For example, *meta-paths* are sequences of nodes and links with specific types, representing composite relations between different entities, such as “co-write: author-paper-author” and “chemical reaction: compound-reaction-compound”. Learning meta-paths is similar to learning links, involving methods like meta-path prediction [39], [255], [274] and instance discrimination with meta-path-based augmentation [275]. Another example is HetGPT [183], a prompting method that aggregates class prompts along meta-paths. Additionally, the *network schema* is a motif-like heterogeneous pattern that contains all relationships of a particular node type. Due to the rich local structural information in the network schema and the ease of instance retrieval and extraction, it is used to generate contrastive instances in PT-HGNN [276] and HeCo [275].

Knowledge graphs are heterogeneous graphs embedded in specific domains and enriched with domain-specific factual knowledge. In knowledge graphs, *relation triples* (head entity, relation, tail entity) serve as the fundamental knowledge units. RotatE [277] employs a negative sampling-based margin loss to discriminate between the head and tail entities in each relation triple. KEPLER [278] further incorporates the margin loss into MLM to fine-tune a RoBERTa model. SelfKG [279] and AutoAlign [280] utilize an InfoNCE and Triplet estimator, respectively, to capture entity information and align them. GNP [281] proposes masked relation prediction, similar to masked link prediction, within constructed positive and negative relation triples.

Dynamic graphs: Real-world graph data often exhibits dynamic characteristics. The structure or size of a graph may change over time, a phenomenon known as *dynamic evolution*. Discrete dynamic evolution manifests as a series of graph snapshots that are subgraphs with evolved features or graph structure, while continuous dynamic evolution is represented by a sequence of events with timestamps.

Dynamic node features are coupled with temporal knowledge. GPT-ST [282] and STGP [283] perform masked feature prediction across the time dimension of the feature matrix. DDGCL [284] introduces a temporal similarity function to the InfoNCE estimator, weighted by a time gap penalty. STGP [283] and DyGPrompt [285] design dual prompts that capture both downstream node features and time information. LLM4DyG [216] prompts an LLM to sequentially consider node and time information, significantly improving the LLM’s reasoning abilities on dynamic graphs. For dynamic evolution on graph structures, GraphPro [193] constructs a graph prompt that concatenates various graph snapshots with the original structure, and incorporates temporal weights into downstream message passing.

Hypergraphs: Hypergraphs involve *hyperedges* that connect more than two nodes. They are better suitable for capturing higher-order structural relationships beyond simple pairwise interactions. Hyperedges can be considered as a special type of context, as they represent local commonalities among nodes. ViLLain [286] learns a balanced and distinctive pseudo-label distribution by propagating the prototype vectors along the hyperedges. For instance discrimination models, HyperGCL [287] focuses on generating hypergraph augmentations with a variational autoencoder, while TriCL [288] focuses on performing contrast within and between nodes and hyperedges based on the connection membership. HypeBoy [289] predicts if a node belongs to a hyperedge formed by another set of nodes by minimizing the similarity between their projected embeddings.

C. Tackling Potential Biases in Future GFMs

Potential biases in graph representations can significantly affect the generalization ability of graph models [53], [290], presenting challenges in building GFMs. These biases may be inherent in the graph structural data itself or introduced during pre-training or downstream tuning, mainly including the following aspects.

Imbalanced data distributions: Potential biases manifest when the feature or structural properties of graphs exhibit uneven distributions. They mainly include imbalanced class distributions, node degree distributions, and sensitive representation distributions. To mitigate potential biases, current methods involve constructing balanced training data distributions through structure-aware augmentations. GRADE [54] capitalizes on feature similarity to balance nodes with imbalanced degrees, including removing dissimilar connections of high-degree nodes and aligning the neighbor distribution of similar low-degree nodes. For long-tailed class distributions, ImGCL [53] first divides a graph into several K-means clusters, each one referring to a latent class. Node representations are then sampled within each

cluster based on PageRank scores. FPrompt [186] is a fair graph prompting approach where the structural prompt is obtained by masking links within different sensitive groups.

Another way is to design specific training strategies to distinguish or emphasize imbalanced instances. CM-GCL [291] prunes model parameters during contrastive pre-training to capture minority samples and employs a focal loss to emphasize these samples during fine-tuning. Graphair [292] introduces a learnable graph augmentation network and a discriminative network to identify sensitive attributes in the augmented graphs. It utilizes adversarial training to update both networks. GraphPAR [174] is a fair PEFT (Section VI-A) approach that generates multiple sensitive attribute vectors for every node and minimizes the distance between them during fine-tuning.

Vulnerability to attacks: Graph models that are not robust against perturbations are vulnerable to malicious attacks, such as injecting noise into features or textual attributes, and inserting nodes and links into the graph structure to disrupt the underlying knowledge patterns [293], [294]. Structural attacks often become more effective than feature-based ones [290].

To mitigate the impact of structural attacks, GRV [118] introduces a robustness quantification metric based on mutual information and uses it to guide the training of a DGI model [13]. RES [290] demonstrates the effectiveness of random edge dropping in defense against structural attacks and applies it to node-level and graph-level instance discrimination. The recent emergence of GLMs has prompted further research in terms of their robustness. Although LLMs provide a certain degree of robustness compared to GNNs, their performance can still decline considerably under structural attacks [238], [295]. To improve structural robustness, LLM4RGNN [238] tunes a local LLM and a small LM, which are used to identify and remove malicious links, as well as to re-add missing important links in the perturbed graph data.

D. Building Powerful and Explainable GFMs

There is a pressing need for better explainability of GFMs. It not only ensures the model reliability in practical applications but also provides valuable insights into understanding graph knowledge. A promising and underexplored avenue is Reasoning on Graphs (RoG). Drawing from the huge success of LLM reasoning, RoG researchers tend to employ natural language instructions, such as Chain-of-Thoughts and Tree-of-Thoughts, to guide GLMs in reasoning over graph structures [245], [296], [297]. Reasoning not only enhances the model explainability by reducing hallucinations but also enriches the knowledge provided, thereby boosting the generalization ability of GFMs in various open-ended tasks. Additionally, recent GLMs leverage the power of external knowledge bases through Retrieval-Augmented Generation (RAG) [298], [299], [300]. These models retrieve knowledge relevant to the queries from knowledge graph databases to enhance reasoning explainability. By advancing in these directions, we can pave the way for versatile GFMs and even graph agents that are capable of handling diverse, complex, and domain-specific graph tasks with unprecedented effectiveness and adaptability.

REFERENCES

- [1] L. Wu et al., "Graph neural networks: Foundation, frontiers and applications," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2022, pp. 4840–4841.
- [2] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 249–270, Jan. 2022.
- [3] Y. Liu et al., "Graph self-supervised learning: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 6, pp. 5879–5900, Jun. 2023.
- [4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.
- [5] P. Veličković et al., "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018.
- [6] W. Hamilton et al., "Inductive representation learning on large graphs," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.
- [7] Y. Rong et al., "Self-supervised graph transformer on large-scale molecular data," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 12559–12571.
- [8] J. Yang et al., "GraphFormers: GNN-nested transformers for representation learning on textual graph," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 28798–28810.
- [9] E. Chien et al., "Node feature extraction by self-supervised multi-scale neighborhood prediction," in *Proc. Int. Conf. Learn. Representations*, 2022.
- [10] X. He et al., "Harnessing explanations: LLM-to-LM interpreter for enhanced text-attributed graph representation learning," in *Proc. Int. Conf. Learn. Representations*, 2024.
- [11] R. Ye et al., "Language is all a graph needs," in *Proc. Eur. Conf. Field Comput. Linguistics Findings*, 2024, pp. 1955–1973.
- [12] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *Proc. Int. Conf. Neural Inf. Process. Syst. Workshop*, 2016.
- [13] P. Veličković et al., "Deep graph infomax," in *Proc. Int. Conf. Learn. Representations*, 2019.
- [14] J. Liu et al., "Towards graph foundation models: A survey and beyond," 2023, *arXiv:2310.11829*.
- [15] R. Bommasani et al., "On the opportunities and risks of foundation models," 2021, *arXiv:2108.07258*.
- [16] J. Xia et al., "A survey of pretraining on graphs: Taxonomy, methods, and applications," 2022, *arXiv:2202.07893*.
- [17] Y. Xie, Z. Xu, J. Zhang, Z. Wang, and S. Ji, "Self-supervised learning of graph neural networks: A unified review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 2412–2429, Feb. 2023.
- [18] Z. Zhang et al., "Graph meets LLMs: Towards large graph models," in *Proc. Int. Conf. Neural Inf. Process. Syst. Workshop*, 2023.
- [19] B. Jin, G. Liu, C. Han, M. Jiang, H. Ji, and J. Han, "Large language models on graphs: A comprehensive survey," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 12, pp. 8622–8642, Dec. 2024.
- [20] W. Fan et al., "Graph machine learning in the era of large language models (LLMs)," 2024, *arXiv:2404.14928*.
- [21] X. Ren et al., "A survey of large language models for graphs," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2024, pp. 6616–6626.
- [22] H. Mao et al., "Position: Graph foundation models are already here," in *Proc. Int. Conf. Mach. Learn.*, 2024, pp. 34670–34692.
- [23] Z. Hou et al., "GraphMAE: Self-supervised masked graph autoencoders," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2022, pp. 594–604.
- [24] H. Yang et al., "Dual space graph contrastive learning," in *Proc. Int. Conf. World Wide Web*, 2022, pp. 1238–1247.
- [25] W. Ju et al., "Towards graph contrastive learning: A survey and beyond," 2024, *arXiv:2405.11868*.
- [26] K. Sun et al., "Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2020, pp. 5892–5899.
- [27] T. Zhang et al., "CommDGI: Community detection oriented deep graph infomax," in *Proc. Conf. Inf. Knowl. Manage.*, 2020, pp. 1843–1852.
- [28] C. Wang et al., "MGAE: Marginalized graph autoencoder for graph clustering," in *Proc. Conf. Inf. Knowl. Manage.*, 2017, pp. 889–898.
- [29] J. Park, M. Lee, H. J. Chang, K. Lee, and J. Y. Choi, "Symmetric graph convolutional autoencoder for unsupervised graph representation learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6519–6528.
- [30] J. Zhang et al., "Graph-BERT: Only attention is needed for learning graph representations," 2020, *arXiv:2001.05140*.
- [31] Z. Peng et al., "Graph representation learning via graphical mutual information maximization," in *Proc. Int. Conf. World Wide Web*, 2020, pp. 259–270.

- [32] W. Hu et al., "Strategies for pre-training graph neural networks," in *Proc. Int. Conf. Learn. Representations*, 2019.
- [33] Y. You et al., "When does self-supervision help graph convolutional networks?," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 10871–10880.
- [34] W. Jin et al., "Self-supervised learning on graphs: Deep insights and new direction," 2020, *arXiv: 2006.10141*.
- [35] Y. Xie et al., "Self-supervised representation learning via latent graph prediction," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 24460–24477.
- [36] B. Fatemi et al., "SLAPS: Self-supervision improves structure learning for graph neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 22667–22681.
- [37] Z. Hu et al., "GPT-GNN: Generative pre-training of graph neural networks," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 1857–1867.
- [38] Z. Hou et al., "GraphMAE2: A decoding-enhanced masked self-supervised graph learner," in *Proc. Int. Conf. World Wide Web*, 2023, pp. 737–746.
- [39] Y. Tian et al., "Heterogeneous graph masked autoencoders," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2023, pp. 9997–10005.
- [40] J. Xia et al., "Mole-BERT: Rethinking pre-training graph neural networks for molecules," in *Proc. Int. Conf. Learn. Representations*, 2023.
- [41] J. Xia et al., "DiscoGNN: A sample-efficient framework for self-supervised graph representation learning," in *Proc. Int. Conf. Data Eng.*, 2024, pp. 2876–2888.
- [42] Y. Zhu et al., "Deep graph contrastive representation learning," in *Proc. Int. Conf. Mach. Learn. Workshop (GRL+)*, 2020.
- [43] Y. Zhu et al., "Graph contrastive learning with adaptive augmentation," in *Proc. Int. Conf. World Wide Web*, 2021, pp. 2069–2080.
- [44] M. Jin et al., "Multi-scale contrastive siamese networks for self-supervised graph representation learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2021, pp. 1477–1483.
- [45] J. Xia et al., "ProGCL: Rethinking hard negative mining in graph contrastive learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 24332–24346.
- [46] Y. Zhang et al., "COSTA: Covariance-preserving feature augmentation for graph contrastive learning," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2022, pp. 2524–2534.
- [47] C. Wei et al., "Contrastive graph structure learning via information bottleneck for recommendation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 20407–20420.
- [48] S. Thakoor et al., "Large-scale representation learning on graphs via bootstrapping," in *Proc. Int. Conf. Learn. Representations*, 2022.
- [49] D. He et al., "Exploitation of a latent mechanism in graph contrastive learning: Representation scattering," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2024, pp. 115351–115376.
- [50] Y. Mo et al., "Simple unsupervised graph representation learning," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2022, pp. 7797–7805.
- [51] J. Yu et al., "Are graph augmentations necessary? Simple graph contrastive learning for recommendation," in *Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2022, pp. 1294–1303.
- [52] X. Cai et al., "LightGCL: Simple yet effective graph contrastive learning for recommendation," in *Proc. Int. Conf. Learn. Representations*, 2023.
- [53] L. Zeng et al., "ImGCL: Revisiting graph contrastive learning on imbalanced node classification," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2023, pp. 11138–11146.
- [54] R. Wang et al., "Uncovering the structural fairness in graph contrastive learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 32465–32473.
- [55] H. Wang et al., "Single-pass contrastive learning can work for both homophilic and heterophilic graph," *Trans. Mach. Learn. Res.*, 2023.
- [56] P. Zhang et al., "High-frequency-aware hierarchical contrastive selective coding for representation learning on text attributed graphs," in *Proc. Int. Conf. World Wide Web*, 2024, pp. 4316–4327.
- [57] P. Bielak et al., "Graph Barlow Twins: A self-supervised representation learning framework for graphs," *Knowl.-Based Syst.*, vol. 256, 2022, Art. no. 109631.
- [58] H. Zhang et al., "From canonical correlation analysis to self-supervised graph neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 76–89.
- [59] A. Bardes et al., "VICReg: Variance-invariance-covariance regularization for self-supervised learning," in *Proc. Int. Conf. Learn. Representations*, 2022.
- [60] Y. Zhang et al., "Geometric view of soft decorrelation in self-supervised learning," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2024, pp. 4338–4349.
- [61] Z. Hu et al., "Unsupervised pre-training of graph convolutional networks," in *Proc. Int. Conf. Learn. Representations Workshop Representation Learn. Graphs Manifolds*, 2019.
- [62] M. Tang et al., "Graph auto-encoder via neighborhood Wasserstein reconstruction," in *Proc. Int. Conf. Learn. Representations*, 2022.
- [63] J. Li et al., "What's behind the mask: Understanding masked graph modeling for graph autoencoders," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2023, pp. 1268–1279.
- [64] R. Winter et al., "Permutation-invariant variational autoencoder for graph-level representation learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 9559–9573.
- [65] B. Liang et al., "Centrality-guided pre-training for graph," in *Proc. Int. Conf. Learn. Representations*, 2025.
- [66] S. Pan et al., "Adversarially regularized graph autoencoder for graph embedding," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 2609–2615.
- [67] A. Hasanzadeh et al., "Semi-implicit graph variational autoencoders," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 10712–10723.
- [68] M. Sun et al., "GPPT: Graph pre-training and prompt tuning to generalize graph neural networks," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2022, pp. 1717–1727.
- [69] Q. Tan et al., "S2GAE: Self-supervised graph autoencoders are generalizable learners with graph masking," in *Proc. Int. Conf. Web Search Data Mining*, 2023, pp. 787–795.
- [70] Z. Zhao et al., "Masked graph autoencoder with non-discrete bandwidths," in *Proc. Int. Conf. World Wide Web*, 2024, pp. 377–388.
- [71] S. Wan et al., "Contrastive and generative graph convolutional networks for graph-based semi-supervised learning," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2021, pp. 10049–10057.
- [72] D. Kim and A. Oh, "How to find your friendly neighborhood: Graph attention design with self-supervision," in *Proc. Int. Conf. Learn. Representations*, 2021.
- [73] X. Jiang et al., "Incomplete graph learning via attribute-structure decoupled variational auto-encoder," in *Proc. Int. Conf. Web Search Data Mining*, 2024, pp. 304–312.
- [74] Y.-S. Cho, "Decoupled variational graph autoencoder for link prediction," in *Proc. Int. Conf. World Wide Web*, 2024, pp. 839–849.
- [75] Q. Zhu et al., "Transfer learning of graph neural networks with ego-graph information maximization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 1766–1779.
- [76] W. Zhao et al., "Deep graph structural infomax," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2023, pp. 4920–4928.
- [77] H. Zhu et al., "Contrastive Laplacian Eigenmaps," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 5682–5695.
- [78] H. Zhu and P. Koniusz, "Generalized Laplacian Eigenmaps," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 30783–30797.
- [79] C. Gong et al., "Self-Pro: A self-prompt and tuning framework for graph neural networks," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2024, pp. 197–215.
- [80] J. Qiu et al., "GCC: Graph contrastive coding for graph neural network pre-training," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 1150–1160.
- [81] K. Ding et al., "Eliciting structural and semantic global knowledge in unsupervised graph contrastive learning," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2023, pp. 7378–7386.
- [82] Y. Hu et al., "Graph-MLP: Node classification without message passing in graph," 2021, *arXiv:2106.04051*.
- [83] W. Dong et al., "Node representation learning in graph via node-to-neighbourhood mutual information maximization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 16620–16629.
- [84] Y. Jiao, Y. Xiong, J. Zhang, Y. Zhang, T. Zhang, and Y. Zhu, "Sub-graph contrast for scalable self-supervised graph representation learning," in *Proc. IEEE Int. Conf. Data Mining*, 2020, pp. 222–231.
- [85] N. Lee et al., "Augmentation-free self-supervised learning on graphs," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2022, pp. 7372–7380.
- [86] J. Chen et al., "Towards self-supervised learning on graphs with heterophily," in *Proc. Conf. Inf. Knowl. Manage.*, 2022, pp. 201–211.
- [87] H. Jung and H. Park, "Balancing graph embedding smoothness in self-supervised learning via information-theoretic decomposition," in *Proc. Int. Conf. World Wide Web*, 2025, pp. 2621–2632.
- [88] Z. Peng et al., "A new self-supervised task on graphs: Geodesic distance prediction," *Inf. Sci.*, vol. 607, pp. 1195–1210, 2022.
- [89] G. Cui et al., "Adaptive graph encoder for attributed graph embedding," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 976–985.

- [90] X. Wang et al., "AM-GCN: Adaptive multi-channel graph convolutional networks," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 1243–1253.
- [91] Z. Chen et al., "Dual low-rank graph autoencoder for semantic and topological networks," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2023, pp. 4191–4198.
- [92] J. Chen and G. Kou, "Attribute and structure preserving graph contrastive learning," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2023, pp. 7024–7032.
- [93] X. Fan, M. Gong, Y. Wu, and H. Li, "Maximizing mutual information across feature and topology views for representing graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 10, pp. 10735–10747, Oct. 2023.
- [94] W.-Z. Li, C.-D. Wang, J.-H. Lai, and P. S. Yu, "Towards effective and robust graph contrastive learning with graph autoencoding," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 2, pp. 868–881, Feb. 2024.
- [95] Z. Zhang et al., "Motif-based graph self-supervised learning for molecular property prediction," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 15870–15882.
- [96] K.-D. Luong and A. Singh, "Fragment-based pretraining and finetuning on molecular graphs," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2023, pp. 17584–17601.
- [97] E. Inae et al., "Motif-aware attribute masking for molecular graph pre-training," in *Proc. 3rd Learn. Graphs Conf.*, 2024.
- [98] P. Yan et al., "Empowering dual-level graph self-supervised pretraining with motif discovery," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2024, pp. 9223–9231.
- [99] L. Sun et al., "Motif-aware Riemannian graph neural network with generative-contrastive learning," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2024, pp. 9044–9052.
- [100] S. Zhang et al., "Motif-driven contrastive learning of graph representations," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 8, pp. 4063–4075, Aug. 2024.
- [101] Y. Wu et al., "Graph contrastive learning with cohesive subgraph awareness," in *Proc. Int. Conf. World Wide Web*, 2024, pp. 629–640.
- [102] M. Xu et al., "Self-supervised graph-level representation learning with local and global structure," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 11548–11558.
- [103] W. Li et al., "HomoGCL: Rethinking homophily in graph contrastive learning," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2023, pp. 11548–11558.
- [104] Q. Wen et al., "From coarse to fine: Enable comprehensive graph self-supervised learning with multi-granular semantic ensemble," in *Proc. Int. Conf. Mach. Learn.*, 2024, pp. 52801–52818.
- [105] W. Shiao et al., "CARL-G: Clustering-accelerated representation learning on graphs," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2023, pp. 2036–2048.
- [106] M. Chen et al., "Deep contrastive graph learning with clustering-oriented guidance," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2024, pp. 11364–11372.
- [107] J. Li et al., "Dirichlet graph variational autoencoder," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 5274–5283.
- [108] J. Li et al., "Mask-GVAE: Blind denoising graphs via partition," in *Proc. Int. Conf. World Wide Web*, 2021, pp. 3688–3698.
- [109] B. Li et al., "Graph communal contrastive learning," in *Proc. Int. Conf. World Wide Web*, 2022, pp. 1203–1213.
- [110] H. Chen et al., "CSGCL: Community-strength-enhanced graph contrastive learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2023, pp. 2059–2067.
- [111] S. Zhang et al., "StructComp: Substituting propagation with structural compression in training graph contrastive learning," in *Proc. Int. Conf. Learn. Representations*, 2024.
- [112] Y. You et al., "Graph contrastive learning with augmentations," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 5812–5823.
- [113] Y. You et al., "Graph contrastive learning automated," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12121–12132.
- [114] S. Suresh et al., "Adversarial graph augmentation to improve graph contrastive learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 15920–15933.
- [115] J. Xia et al., "SimGRACE: A simple framework for graph contrastive learning without data augmentation," in *Proc. Int. Conf. World Wide Web*, 2022, pp. 1070–1079.
- [116] F. Sun et al., "InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," in *Proc. Int. Conf. Learn. Representations*, 2020.
- [117] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 4116–4126.
- [118] J. Xu et al., "Unsupervised adversarially robust representation learning on graphs," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2022, pp. 4290–4298.
- [119] Y. Zheng et al., "Rethinking and scaling up graph contrastive learning: An extremely efficient approach with group discrimination," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 10809–10820.
- [120] D. Kim et al., "Graph self-supervised learning with accurate discrepancy learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 14085–14098.
- [121] H. Yang et al., "Generating counterfactual hard negative samples for graph contrastive learning," in *Proc. Int. Conf. World Wide Web*, 2023, pp. 621–629.
- [122] L. Lin et al., "Spectral augmentation for self-supervised learning on graphs," in *Proc. Int. Conf. Learn. Representations*, 2023.
- [123] N. Navarin et al., "Pre-training graph neural networks with kernels," 2018, *arXiv: 1811.06930*.
- [124] J. Li et al., "Hierarchical topology isomorphism expertise embedded graph contrastive learning," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2024, pp. 13518–13527.
- [125] J. Liu et al., "Enhancing hyperbolic graph embeddings via contrastive learning," in *Proc. Int. Conf. Neural Inf. Process. Syst. Workshop*, 2021, pp. 4575–4589.
- [126] L. Sun et al., "A self-supervised mixed-curvature graph neural network," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2022, pp. 4146–4155.
- [127] G. Skenderi et al., "Graph-level representation learning with joint-embedding predictive architectures," *Trans. Mach. Learn. Res.*, 2025.
- [128] R. Gong et al., "Graph representation learning in hyperbolic space via dual-masked," in *Proc. Int. Conf. Comput. Linguistics*, 2025, pp. 637–646.
- [129] L. Sun et al., "RiemannGFM: Learning a graph foundation model from structural geometry," in *Proc. Int. Conf. World Wide Web*, 2025, pp. 1154–1165.
- [130] J. Devlin et al., "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2019, pp. 4171–4186.
- [131] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 16000–16009.
- [132] Y. He et al., "UniGraph: Learning a unified cross-domain foundation model for text-attributed graphs," 2024, *arXiv:2402.13630*.
- [133] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 9726–9735.
- [134] T. Chen et al., "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1597–1607.
- [135] M. I. Belghazi et al., "Mutual information neural estimation," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 531–540.
- [136] S. Nowozin et al., "f-GAN: Training generative neural samplers using variational divergence minimization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 271–279.
- [137] A. V. D. Oord et al., "Representation learning with contrastive predictive coding," 2018, *arXiv: 1807.03748*.
- [138] F. Schroff et al., "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 815–823.
- [139] J.-B. Grill et al., "Bootstrap your own latent - A new approach to self-supervised learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 21271–21284.
- [140] S. Rendle et al., "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. Conf. Uncertainty Artif. Intell.*, 2009, pp. 452–461.
- [141] J. Z. HaoChen et al., "Provable guarantees for self-supervised deep learning with spectral contrastive loss," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2023, pp. 5000–5011.
- [142] Z. Liu et al., "GraphPrompt: Unifying pre-training and downstream tasks for graph neural networks," in *Proc. Int. Conf. World Wide Web*, 2023, pp. 417–428.
- [143] Y. Yan et al., "Inductive graph alignment prompt: Bridging the gap between graph pre-training and inductive fine-tuning from spectral perspective," in *Proc. Int. Conf. World Wide Web*, 2024, pp. 4328–4339.

- [144] J. Zbontar et al., “Barlow Twins: Self-supervised learning via redundancy reduction,” in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12310–12320.
- [145] S. Brin et al., “The anatomy of a large-scale hypertextual web search engine,” in *Proc. Int. Conf. World Wide Web*, 1998, pp. 107–117.
- [146] D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’ networks,” *Nature*, vol. 393, pp. 440–442, 1998.
- [147] B. Jin et al., “Patton: Language model pretraining on text-rich networks,” in *Proc. Assoc. Comput. Linguistics*, 2023, pp. 7005–7020.
- [148] X. Yu et al., “HGPROMPT: Bridging homogeneous and heterogeneous graphs for few-shot prompt learning,” in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2024, pp. 16578–16586.
- [149] Y. Ma et al., “Is homophily a necessity for graph neural networks?,” in *Proc. Int. Conf. Learn. Representations*, 2022.
- [150] B. Perozzi et al., “DeepWalk: Online learning of social representations,” in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 701–710.
- [151] M. Belkin and P. Niyogi, “Laplacian eigenmaps and spectral techniques for embedding and clustering,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2001, pp. 585–591.
- [152] Q. Huang et al., “PRODIGY: Enabling in-context learning over graphs,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2023, pp. 16302–16317.
- [153] X. Sun et al., “All in One: Multi-task prompting for graph neural networks,” in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2023, pp. 2120–2131.
- [154] H. Liu et al., “One for All: Towards training one graph model for all classification tasks,” in *Proc. Int. Conf. Learn. Representations*, 2024.
- [155] L. Katz, “A new status index derived from sociometric analysis,” *Psychometrika*, vol. 18, pp. 39–43, 1953.
- [156] P. Jaccard, “The distribution of the flora in the alpine zone,” *New Phytologist*, vol. 11, pp. 37–50, 1912.
- [157] M. Chen et al., “ULTRA-DP: Unifying graph pre-training with multi-task graph dual prompt,” 2023, *arXiv:2310.14845*.
- [158] N. Brown, *In Silico Medicinal Chemistry: Computational Methods to Support Drug Design*. London, U.K.: Royal Society of Chemistry, 2015.
- [159] P. Ertl, “An algorithm to identify functional groups in organic molecules,” *J. Cheminformatics*, vol. 9, pp. 1–7, 2017.
- [160] A. Coates and A. Y. Ng, “Learning feature representations with K-Means,” in *Neural Networks: Tricks of the Trade*, Berlin, Germany: Springer, 2012, pp. 561–580.
- [161] M. Caron et al., “Deep clustering for unsupervised learning of visual features,” in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 132–149.
- [162] V. D. Blondel et al., “Fast unfolding of communities in large networks,” *J. Statist. Mechanics, Theory Experiment*, vol. 2008, 2008, Art. no. P10008.
- [163] Y. Lu et al., “Learning to pre-train graph neural networks,” in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2021, pp. 4276–4284.
- [164] Z. Wang, S. Di, L. Chen, and X. Zhou, “Search to fine-tune pre-trained graph neural networks for graph-level tasks,” in *Proc. IEEE 40th Int. Conf. Data Eng.*, 2024, pp. 2805–2819.
- [165] Y. Cao et al., “When to pre-train graph neural networks? From data generation perspective!,” in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2023, pp. 142–153.
- [166] Y. Sun et al., “Fine-tuning graph neural networks by preserving graph generative patterns,” in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2024, pp. 9053–9061.
- [167] Y. Zhu et al., “GraphControl: Adding conditional control to universal graph pre-trained models for graph domain transfer learning,” in *Proc. Int. Conf. World Wide Web*, 2024, pp. 539–550.
- [168] Z. Wang et al., “GFT: Graph foundation model with transferable tree vocabulary,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2024, pp. 107403–107443.
- [169] X. Han et al., “Adaptive transfer learning on graph neural networks,” in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2021, pp. 565–574.
- [170] J. Zhang et al., “Fine-tuning graph neural networks via graph topology induced optimal transport,” in *Proc. Int. Joint Conf. Artif. Intell.*, 2022, pp. 3705–3711.
- [171] R. Huang et al., “Measuring task similarity and its implication in fine-tuning graph neural networks,” in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2024, pp. 12617–12625.
- [172] S. Li et al., “AdapterGNN: Parameter-efficient fine-tuning improves generalization in GNNs,” in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2024, pp. 13600–13608.
- [173] A. Gui et al., “G-Adapter: Towards structure-aware parameter-efficient transfer learning for graph transformer networks,” in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2024, pp. 12226–12234.
- [174] Z. Zhang et al., “Endowing pre-trained graph models with provable fairness,” in *Proc. Int. Conf. World Wide Web*, 2024, pp. 1045–1056.
- [175] Y. Mo et al., “HG-Adapter: Improving pre-trained heterogeneous graph neural networks with dual adapters,” in *Proc. Int. Conf. Learn. Representations*, 2025.
- [176] Q. Chen et al., “DAGPrompt: Pushing the limits of graph prompting with a distribution-aware graph prompt tuning approach,” in *Proc. Int. Conf. World Wide Web*, 2025, pp. 4346–4358.
- [177] T. Fang et al., “Universal prompt tuning for graph neural networks,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2023, pp. 52464–52489.
- [178] J. Li et al., “Instance-aware graph prompt learning,” *Trans. Mach. Learn. Res.*, 2025.
- [179] R. Shirkavand and H. Huang, “Deep prompt tuning for graph transformers,” 2023, *arXiv:2309.10131*.
- [180] Q. Ge et al., “PSP: Pre-training and structure prompt tuning for graph neural networks,” in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2024, pp. 423–439.
- [181] Z. Tan et al., “Virtual node tuning for few-shot node classification,” in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2023, pp. 2177–2188.
- [182] Y. Zhu et al., “SGL-PT: A strong graph learner with graph prompt tuning,” 2023, *arXiv:2302.12449*.
- [183] Y. Ma et al., “HetGPT: Harnessing the power of prompt tuning in pre-trained heterogeneous graph neural networks,” in *Proc. Int. Conf. World Wide Web*, 2024, pp. 1015–1023.
- [184] W. Jiang et al., “Killing two birds with one stone: Cross-modal reinforced prompting for graph and language tasks,” in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2024, pp. 1301–1312.
- [185] X. Fu et al., “Edge prompt tuning for graph neural networks,” in *Proc. Int. Conf. Learn. Representations*, 2025.
- [186] Z. Li et al., “Fairness-aware prompt tuning for graph neural networks,” in *Proc. Int. Conf. World Wide Web*, 2025, pp. 3586–3597.
- [187] 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 Trans. Knowl. Data Eng.*, vol. 36, no. 11, pp. 6237–6250, Nov. 2024.
- [188] J. Lee et al., “Subgraph-level universal prompt tuning,” 2024, *arXiv:2402.10380*.
- [189] X. Yu et al., “Non-homophilic graph pre-training and prompt learning,” in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2025, pp. 1844–1854.
- [190] X. Yu et al., “GCoT: Chain-of-thought prompt learning for graphs,” 2025, *arXiv:2502.08092*.
- [191] X. Yu et al., “MultiGPrompt for multi-task pre-training and prompting on graphs,” in *Proc. Int. Conf. World Wide Web*, 2024, pp. 515–526.
- [192] H. Zhao et al., “Pre-training and prompting for few-shot node classification on text-attributed graphs,” in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2024, pp. 4467–4478.
- [193] Y. Yang et al., “GraphPro: Graph pre-training and prompt learning for recommendation,” in *Proc. Int. Conf. World Wide Web*, 2024, pp. 3690–3699.
- [194] J. Wang et al., “A novel prompt tuning for graph transformers: Tailoring prompts to graph topologies,” in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2024, pp. 3116–3127.
- [195] N. Houlsby et al., “Parameter-efficient transfer learning for NLP,” in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2790–2799.
- [196] E. Hu et al., “LoRA: Low-rank adaptation of large language models,” in *Proc. Int. Conf. Learn. Representations*, 2022.
- [197] X. Sun et al., “Graph prompt learning: A comprehensive survey and beyond,” 2023, *arXiv:2311.16534*.
- [198] T. Zou et al., “Pretraining language models with text-attributed heterogeneous graphs,” in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2023, pp. 10316–10333.
- [199] W. Shang et al., “Path-LLM: A shortest-path-based LLM learning for unified graph representation,” 2024, *arXiv:2408.05456*.
- [200] H. Xie et al., “Graph-aware language model pre-training on a large graph corpus can help multiple graph applications,” in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2023, pp. 5270–5281.
- [201] Z. Wen and Y. Fang, “Augmenting low-resource text classification with graph-grounded pre-training and prompting,” in *Proc. Int. ACM SIGIR Conf. Res. Develop. Informat. Retrieval*, 2023, pp. 506–516.
- [202] Y. Li et al., “GRENADE: Graph-centric language model for self-supervised representation learning on text-attributed graphs,” in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2023, pp. 2745–2757.
- [203] W. Brannon et al., “ConGraT: Self-supervised contrastive pretraining for joint graph and text embeddings,” in *Proc. Assoc. Comput. Linguistics Workshop*, 2024, pp. 19–39.

- [204] J. Tang et al., "GraphGPT: Graph instruction tuning for large language models," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Informat. Retrieval*, 2024, pp. 491–500.
- [205] Y. Fang et al., "UniGLM: Training one unified language model for text-attributed graphs," in *Proc. Int. Conf. Web Search Data Mining*, 2025, pp. 973–981.
- [206] Y. Song et al., "A pure transformer pretraining framework on text-attributed graphs," in *Proc. Learn. Graphs Conf.*, 2024.
- [207] Y. Xu et al., "LLaSA: Large language and structured data assistant," 2024, *arXiv:2411.14460*.
- [208] H. Wang et al., "Can language models solve graph problems in natural language?," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2023, pp. 30840–30861.
- [209] Y. Hu et al., "Beyond text: A deep dive into large language models' ability on understanding graph data," in *Proc. Int. Conf. Neural Inf. Process. Syst. Workshop New Front. Graph Learn.*, 2023.
- [210] B. Fatemi et al., "Talk like a graph: Encoding graphs for large language models," in *Proc. Int. Conf. Learn. Representations*, 2024.
- [211] J. Huang et al., "Can LLMs effectively leverage graph structural information through prompts, and why?," *Trans. Mach. Learn. Res.*, 2024.
- [212] X. Li et al., "Can large language models analyze graphs like professionals? A benchmark, datasets and models," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2024, pp. 141045–141070.
- [213] R. Li et al., "Similarity-based neighbor selection for graph LLMs," 2024, *arXiv:2402.03720*.
- [214] K. Skianis et al., "Graph reasoning with large language models via pseudo-code prompting," 2024, *arXiv:2409.17906*.
- [215] Z. Hu et al., "Let's ask GNN: Empowering large language model for graph in-context learning," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2024, pp. 1396–1409.
- [216] Z. Zhang et al., "LLM4DyG: Can large language models solve spatial-temporal problems on dynamic graphs?," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2024, pp. 4350–4361.
- [217] B. Perozzi et al., "Let your graph do the talking: Encoding structured data for LLMs," 2024, *arXiv:2402.05862*.
- [218] Z. Liu et al., "Can we soft prompt LLMs for graph learning tasks?," in *Proc. Int. Conf. World Wide Web*, 2024, pp. 481–484.
- [219] X. Huang et al., "Prompt-based node feature extractor for few-shot learning on text-attributed graphs," 2023, *arXiv:2309.02848*.
- [220] X. Huang et al., "Can GNN be good adapter for LLMs?," in *Proc. Int. Conf. World Wide Web*, 2024, pp. 893–904.
- [221] Y. Qin et al., "Disentangled representation learning with large language models for text-attributed graphs," 2023, *arXiv:2310.18152*.
- [222] L. Kong et al., "GOFA: A generative one-for-all model for joint graph language modeling," in *Proc. Int. Conf. Learn. Representations*, 2025.
- [223] Z. Zhang et al., "TAGA: Text-attributed graph self-supervised learning by synergizing graph and text mutual transformations," 2024, *arXiv:2405.16800*.
- [224] Y. Zhu et al., "GraphCLIP: Enhancing transferability in graph foundation models for text-attributed graphs," in *Proc. Int. Conf. World Wide Web*, 2025, pp. 2183–2197.
- [225] B. Pan et al., "Distilling large language models for text-attributed graph learning," in *Proc. Conf. Inf. Knowl. Manage.*, 2024, pp. 1836–1845.
- [226] Y. Zhu et al., "Efficient tuning and inference for large language models on textual graphs," in *Proc. Int. Joint Conf. Artif. Intell.*, 2024, pp. 5734–5742.
- [227] D. Wang et al., "LLMs as zero-shot graph learners: Alignment of GNN representations with LLM token embeddings," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2024, pp. 5950–5973.
- [228] J. Tang et al., "HiGPT: Heterogeneous graph language model," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2024, pp. 2842–2853.
- [229] R. Chen et al., "LLaGA: Large language and graph assistant," in *Proc. Int. Conf. Mach. Learn.*, 2024, pp. 7809–7823.
- [230] Y. Chen et al., "Improving molecule-language alignment with hierarchical graph tokenization," 2024, *arXiv:2406.14021*.
- [231] M. Zhang et al., "GraphTranslator: Aligning graph model to large language model for open-ended tasks," in *Proc. Int. Conf. World Wide Web*, 2024, pp. 1003–1014.
- [232] Z. Chen et al., "Exploring the potential of large language models (LLMs) in learning on graphs," *KDD Explorations Newsl.*, vol. 25, pp. 42–61, 2024.
- [233] C. Qian et al., "Can large language models empower molecular property prediction?," 2023, *arXiv:2307.07443*.
- [234] Z. Wang et al., "Can LLMs convert graphs to text-attributed graphs?," 2024, *arXiv:2412.10136*.
- [235] Y. Fang et al., "GAugLLM: Improving graph contrastive learning for text-attributed graphs with large language models," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2024, pp. 747–758.
- [236] J. Lu et al., "Scale-free graph-language models," in *Proc. Int. Conf. Learn. Representations*, 2025.
- [237] Z. Chen et al., "Label-free node classification on graphs with large language models (LLMs)," in *Proc. Int. Conf. Learn. Representations*, 2024.
- [238] Z. Zhang et al., "Can large language models improve the adversarial robustness of graph neural networks?," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2025, pp. 2008–2019.
- [239] T. Zhang et al., "Leveraging large language models for effective label-free node classification in text-attributed graphs," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Informat. Retrieval*, 2025.
- [240] J. Yu et al., "Leveraging large language models for node generation in few-shot learning on text-attributed graphs," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2025, pp. 13087–13095.
- [241] L. Xia et al., "OpenGraph: Towards open graph foundation models," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2024, pp. 2365–2379.
- [242] Y. Qiao et al., "LOGIN: A large language model consulted graph neural network training framework," in *Proc. Int. Conf. Web Search Data Mining*, 2025, pp. 232–241.
- [243] S. Sun et al., "Large language models as topological structure enhancers for text-attributed graphs," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2025.
- [244] J. Zhao et al., "Learning on large-scale text-attributed graphs via variational inference," in *Proc. Int. Conf. Learn. Representations*, 2023.
- [245] Z. Chai et al., "GraphLLM: Boosting graph reasoning ability of large language model," 2023, *arXiv:2310.05845*.
- [246] R. Xue et al., "Efficient large language models fine-tuning on graphs," 2023, *arXiv:2312.04737*.
- [247] K. Duan et al., "SimTeG: A frustratingly simple approach improves textual graph learning," 2023, *arXiv:2308.02565*.
- [248] Y. Tan et al., "WalkLM: A uniform language model fine-tuning framework for attributed graph embedding," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2023, pp. 13308–13325.
- [249] S. Ouyang et al., "GUNDAM: Aligning large language models with graph understanding," 2024, *arXiv:2409.20053*.
- [250] S. Hu et al., "Large language model meets graph neural network in knowledge distillation," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2025, pp. 17295–17304.
- [251] J. Wang et al., "InstructGraph: Boosting large language models via graph-centric instruction tuning and preference alignment," in *Proc. Assoc. Comput. Linguistics Findings*, 2024, pp. 13492–13510.
- [252] N. Chen et al., "GraphWiz: An instruction-following language model for graph problems," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2024, pp. 353–364.
- [253] Z. Luo et al., "GraphInstruct: Empowering large language models with graph understanding and reasoning capability," 2024, *arXiv:2403.04483*.
- [254] Z. Xu et al., "How to make LLMs strong node classifiers?," 2024, *arXiv:2410.02296*.
- [255] Q. Zhu et al., "HierPromptLM: A pure PLM-based framework for representation learning on heterogeneous text-rich networks," 2025, *arXiv:2501.12857*.
- [256] C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, pp. 1–67, 2020.
- [257] A. Radford et al., "Improving language understanding by generative pre-training," *OpenAI*, 2018.
- [258] H. Touvron et al., "LLaMA: Open and efficient foundation language models," 2023, *arXiv:2302.13971*.
- [259] DeepSeek-AI, "Deepseek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning," 2025, *arXiv:2501.12948*.
- [260] Y. Liu et al., "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.
- [261] M. Lewis et al., "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proc. Assoc. Comput. Linguistics*, 2020, pp. 7871–7880.
- [262] A. Radford et al., "Learning transferable visual models from natural language supervision," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8748–8763.

- [263] Y. Li et al., “A survey of graph meets large language model: Progress and future directions,” in *Proc. Int. Joint Conf. Artif. Intell.*, 2024, pp. 8123–8131.
- [264] J. Zhao et al., “GraphText: Graph reasoning in text space,” 2023, *arXiv:2310.01089*.
- [265] J. Wei et al., “Emergent abilities of large language models,” *Trans. Mach. Learn. Res.*, 2022.
- [266] Z. Ji et al., “Survey of hallucination in natural language generation,” *ACM Comput. Surveys*, vol. 55, pp. 1–38, 2023.
- [267] W. Jin et al., “Automated self-supervised learning for graphs,” in *Proc. Int. Conf. Learn. Representations*, 2022.
- [268] M. Ju et al., “Multi-task self-supervised graph neural networks enable stronger task generalization,” in *Proc. Int. Conf. Learn. Representations*, 2023.
- [269] T. Fang et al., “Exploring correlations of self-supervised tasks for graphs,” in *Proc. Int. Conf. Mach. Learn.*, 2024, pp. 12957–12972.
- [270] L. Wu et al., “Automated graph self-supervised learning via multi-teacher knowledge distillation,” 2022, *arXiv:2210.02099*.
- [271] T. Fan et al., “Decoupling weighing and selecting for integrating multiple graph pre-training tasks,” in *Proc. Int. Conf. Learn. Representations*, 2024.
- [272] B. Jin et al., “Heterformer: Transformer-based deep node representation learning on heterogeneous text-rich networks,” in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2023, pp. 1020–1031.
- [273] H. Duan et al., “Reserving-masking-reconstruction model for self-supervised heterogeneous graph representation,” in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2024, pp. 689–700.
- [274] D. Hwang et al., “Self-supervised auxiliary learning with meta-paths for heterogeneous graphs,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 10294–10305.
- [275] X. Wang et al., “Self-supervised heterogeneous graph neural network with co-contrastive learning,” in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2021, pp. 1726–1736.
- [276] X. Jiang et al., “Pre-training on large-scale heterogeneous graph,” in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2021, pp. 756–766.
- [277] Z. Sun et al., “RotatE: Knowledge graph embedding by relational rotation in complex space,” in *Proc. Int. Conf. Learn. Representations*, 2019.
- [278] X. Wang et al., “KEPLER: A unified model for knowledge embedding and pre-trained language representation,” in *Proc. Trans. Assoc. Comput. Linguistics*, vol. 9, pp. 176–194, 2021.
- [279] X. Liu et al., “SelfKG: Self-supervised entity alignment in knowledge graphs,” in *Proc. Int. Conf. World Wide Web*, 2022, pp. 860–870.
- [280] R. Zhang et al., “AutoAlign: Fully automatic and effective knowledge graph alignment enabled by large language models,” *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 6, pp. 2357–2371, Jun. 2024.
- [281] Y. Tian et al., “Graph neural prompting with large language models,” in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2024, pp. 19080–19088.
- [282] Z. Li et al., “GPT-ST: Generative pre-training of spatio-temporal graph neural networks,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2023, pp. 70229–70246.
- [283] J. Hu et al., “Prompt-based spatio-temporal graph transfer learning,” in *Proc. Conf. Inf. Knowl. Manage.*, 2024, pp. 890–899.
- [284] S. Tian et al., “Self-supervised representation learning on dynamic graphs,” in *Proc. Conf. Inf. Knowl. Manage.*, 2021, pp. 1814–1823.
- [285] X. Yu et al., “Node-time conditional prompt learning in dynamic graphs,” in *Proc. Int. Conf. Learn. Representations*, 2025.
- [286] G. Lee et al., “VilLain: Self-supervised learning on homogeneous hypergraphs without features via virtual label propagation,” in *Proc. Int. Conf. World Wide Web*, 2024, pp. 594–605.
- [287] T. Wei et al., “Augmentations in hypergraph contrastive learning: Fabricated and generative,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 1909–1922.
- [288] D. Lee and K. Shin, “I’m me, we’re us, and I’m us: Tri-directional contrastive learning on hypergraphs,” in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2023, pp. 8456–8464.
- [289] S. Kim et al., “HypeBoy: Generative self-supervised representation learning on hypergraphs,” in *Proc. Int. Conf. Learn. Representations*, 2024.
- [290] M. Lin et al., “Certifiably robust graph contrastive learning,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2023, pp. 17008–17037.
- [291] Y. Qian et al., “Co-modality graph contrastive learning for imbalanced node classification,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 15862–15874.
- [292] H. Ling et al., “Learning fair graph representations via automated data augmentations,” in *Proc. Int. Conf. Learn. Representations*, 2023.
- [293] H. Zhang et al., “Graph contrastive backdoor attacks,” in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 40888–40910.
- [294] X. Lyu et al., “Cross-context backdoor attacks against graph prompt learning,” in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2024, pp. 2094–2105.
- [295] K. Guo et al., “Learning on graphs with large language models (LLMs): A deep dive into model robustness,” 2024, *arXiv:2407.12068*.
- [296] B. Jin et al., “Graph chain-of-thought: Augmenting large language models by reasoning on graphs,” in *Proc. Assoc. Comput. Linguistics Findings*, 2024, pp. 163–184.
- [297] L. Luo et al., “Reasoning on graphs: Faithful and interpretable large language model reasoning,” in *Proc. Int. Conf. Learn. Representations*, 2024.
- [298] B. Peng et al., “Graph retrieval-augmented generation: A survey,” 2024, *arXiv:2408.08921*.
- [299] X. He et al., “G-Retriever: Retrieval-augmented generation for textual graph understanding and question answering,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2024, pp. 132876–132907.
- [300] X. Jiang et al., “RAGraph: A general retrieval-augmented graph learning framework,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2024, pp. 29948–29985.



Ziwen Zhao received the BS degree in bioinformatics and computer science from the Huazhong University of Science and Technology, Wuhan, China. He is currently working toward the master’s degree with the School of Computer Science and Technology, Huazhong University of Science and Technology. His primary research interests include graph mining, self-supervised learning, and graph foundation models.



officially included in the PaddlePaddle model repository.

Yixin Su received the PhD degree in computer science from the University of Melbourne. His research focuses on recommender systems and personalized large language models. He is currently an associate professor with the Huazhong University of Science and Technology. Prior to academia, he worked as a machine learning scientist with PayPal and Baidu, where he led the design of generative auction mechanisms in billion-scale ad systems. He has published in top-tier venues including SIGKDD, SIGIR, and *Nature Machine Intelligence*, with one of his models



senior member of the China Computer Federation (CCF). Her research interests include data mining, social networks, machine learning, and Big Data.

Yuhua Li (Member, IEEE) received the PhD degree in computer application technology from the Huazhong University of Science and Technology, Wuhan, China, in 2006. She is currently a professor with the School of Computer Science and Technology, Huazhong University of Science and Technology. She was a visiting scholar with the University of California, Santa Barbara. She has published more than 60 journal and conference papers (NeurIPS, *IEEE Transactions on Knowledge and Data Engineering*, SIGIR, WWW, ICDM, IJCAI). She is also a



vision.

Yixiong Zou (Member, IEEE) received the BS degree from Nankai University, and the PhD degree from the School of Electrical Engineering and Computer Science, Peking University, Beijing, China. He was a visiting scholar with Carnegie Mellon University. He has published more than 40 journal and conference papers. He is currently a lecturer with the School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests include multimodal large language models, few-shot learning, open-world learning and computer



Rui Zhang (Senior Member, IEEE) is a distinguished professor with the Huazhong University of Science and Technology and a visiting professor with Tsinghua University. His research interests include data management and mining, and AI. His inventions have been adopted by major IT companies in their products such as Microsoft, Amazon, AT&T. He has won several awards including Google Faculty Research Award, Chris Wallace Award for Outstanding Research, and Australian Future Fellowship.



He has published more than 500 journal and conference papers (NeurIPS, KDD, ICDM, IJCAI). He is also a member of ACM.

Ruixuan Li (Member, IEEE) received the BS, MS, and PhD degrees in computer science from the Huazhong University of Science and Technology in 1997, 2000, and 2004, respectively. From 2009 to 2010, he was a visiting researcher with the Department of Electrical and Computer Engineering, University of Toronto. He is currently a professor with the School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests include cloud and edge computing, Big Data management, and distributed system security.