HGPrompt: Bridging Homogeneous and Heterogeneous Graphs for Few-shot Prompt Learning

Xingtong Yu[1]*, Yuan Fang[2]†, Zemin Liu[3], Xinming Zhang[1]†

[1]University of Science and Technology of China, China
[2] Singapore Management University, Singapore
[3] National University of Singapore, Singapore
yxt95@mail.ustc.edu.cn, yfang@smu.edu.sg, zeminliu@nus.edu.sg, xinming@ustc.edu.cn

Abstract

Graph neural networks (GNNs) and heterogeneous graph neural networks (HGNNs) are prominent techniques for homogeneous and heterogeneous graph representation learning, yet their performance in an end-to-end supervised framework greatly depends on the availability of task-specific supervision. To reduce the labeling cost, pre-training on self-supervised pretext tasks has become a popular paradigm, but there is often a gap between the pre-trained model and downstream tasks, stemming from the divergence in their objectives. To bridge the gap, prompt learning has risen as a promising direction especially in few-shot settings, without the need to fully fine-tune the pre-trained model. While there has been some early exploration of prompt-based learning on graphs, they primarily deal with homogeneous graphs, ignoring the heterogeneous graphs that are prevalent in downstream applications. In this paper, we propose HGPROMPT, a novel pre-training and prompting framework to unify not only pre-training and downstream tasks but also homogeneous and heterogeneous graphs via a *dual-template* design. Moreover, we propose *dual-prompt* in HGPROMPT to assist a downstream task in locating the most relevant prior to bridge the gaps caused by not only feature variations but also heterogeneity differences across tasks. Finally, we thoroughly evaluate and analyze HGPROMPT through extensive experiments on three public datasets.

1 Introduction

Graph data is ubiquitous due to its ability to model relations between objects, such as bibliographic networks, chemical compound graphs, and social networks. In particular, real-world graphs often involve multiple types of nodes and edges, such as authors, papers and conferences interacting in various ways in a bibliographic network. These graphs are called *heterogeneous* graphs or heterogeneous information networks (HINs) (Sun and Han 2013; Yang et al. 2020), in contrast to conventional *homogeneous* graphs that consist of only one type of node and edge.

On such homogeneous and heterogeneous graphs, graph neural networks (GNNs) (Kipf and Welling 2016; Veličković et al. 2018a) and heterogeneous graph neural networks (HGNNs) (Wang et al. 2019; Lv et al. 2021) have been widely deployed, respectively, for various tasks such as link prediction (LP), node classification (NC) and graph classification (GC). However, when trained in a supervised manner, GNNs and HGNNs depend heavily on extensive task-specific labels, which are challenging or expensive to procure. Moreover, the supervised paradigm would require re-training for different tasks even on the same graph, incurring significant overheads.

Hence, the "pre-train, fine-tune" paradigm (Dong et al. 2019) has emerged as a practical alternative to supervised learning. First, a pre-training step utilizes label-free graphs to learn task-irrelevant general properties on homogeneous (Hu et al. 2020) or heterogeneous graphs (Jiang et al. 2021). Next, a fine-tuning step updates the pre-trained model to adapt it to downstream tasks using some task-specific labels. Apparently, the two steps optimize different objectives, creating a gap between pre-training and downstream tasks, which further leads to subpar performance (Liu et al. 2023a). In addition, fine-tuning a large pre-trained model can be costly, and still require considerable task-specific labels to avoid overfitting.

To address the issues with fine-tuning, prompt-based learning (Brown et al. 2020; Jia et al. 2021) has become popular. Prompt learning seeks to extract the semantic relevance between a downstream task and the pre-trained model, acting as a bridge to align downstream objectives toward the pre-trained model. Under this approach, the parameters of the pre-trained model are frozen, and only a light-weight, task-specific prompt vector is tuned for a downstream task. Given much fewer learnable parameters in a prompt than in the pre-trained model, prompt-tuning can be both efficient and effective especially in few-shot tasks with very scarce task-specific labels.

Fueled by the success of prompt learning in language models, researchers have attempted its application to graphs (Liu et al. 2023b; Sun et al. 2023, 2022; Tan et al. 2023; Yu et al. 2023b). Typically, these approaches first introduce a unification template to bridge pre-training and downstream tasks on graphs. Then, they design a learnable prompt that appends to or modifies the input of the downstream task, to better align the task with the pre-trained graph model. Yet these previous works are only designed for homogeneous

*Work was done while at Singapore Management University.
†Corresponding authors.

graphs, and do not consider the gap brought by tasks on heterogeneous graphs.

In this study, we present a novel few-shot prompt learning framework amenable to heterogeneous graphs, called HGPROMPT. It aims to bridge the gap between homogeneous and heterogeneous graphs. Specifically, HGPROMPT considers two target scenarios as shown in Fig. 1(a): given a heterogeneous graph in a downstream task, we can leverage a pre-trained model that may be trained on either heterogeneous or homogeneous graphs. Note that the inconsistency between pre-training and downstream graphs could stem from several reasons. For instance, the heterogeneity in pre-training graphs may be masked for privacy, or the pre-training graphs are not made available and we only have access to a pre-trained model on homogeneous graphs. In general, we may or may not have control over pre-training, and thus it is more flexible with the two target scenarios. However, the problem is non-trivial due to two key challenges.

First, how do we *unify downstream tasks on heterogeneous graphs with pre-training tasks irrespective of their graph heterogeneity?* In the language domain, the tasks are unified by a common template of masked language modeling (Brown et al. 2020). Similar attempts exist on graph-based tasks. For example, GraphPrompt (Liu et al. 2023b) converts popular graph-based tasks to a common template of predicting subgraph similarity. However, this does not deal with the heterogeneity differences in the input graphs to pre-training and downstream stages. Essentially, we aim to unify not only the template of the tasks, but also the template of graphs, and propose a *dual-template* design. On one hand, we simply follow GraphPrompt to unify link prediction, node classification and graph classification using a common task template based on subgraph similarity. On the other hand, in HGPROMPT we propose a unified graph template to convert a heterogeneous graph into multiple homogeneous subgraphs, as illustrated in Fig. 1(b). To be more specific, we sample one homogeneous subgraph for each type of node/edge, plus an additional one without considering their types. Hence, a heterogeneous graph is unified with homogeneous graphs, yet the heterogeneity is still discernible across graph samples.

Second, with unified templates of both tasks and graphs, how do we *design a prompt to narrow the gaps across tasks, caused by not only feature variations but also heterogeneity differences?* In the language domain, a prompt is simply a sequence of word tokens that reformulate the downstream task to narrow the gap with pre-training. Similarly, recent graph prompting methods (Liu et al. 2023b; Sun et al. 2022; Tan et al. 2023) modify the input embeddings to close the task gap, whilst leaving the heterogeneity gap as an open question to solve. In HGPROMPT, we introduce a *dual-prompt* as illustrated in Fig. 1(c). First, a feature prompt modifies the input to the subgraph readout similar to GraphPrompt, based on the observation that different tasks may focus on different features (Liu et al. 2023b). Second, a heterogeneity prompt further modifies the aggregation weights of the multiple homogeneous subgraphs that are converted from the input heterogeneous graph, as different tasks may focus on different facets of heterogeneity.

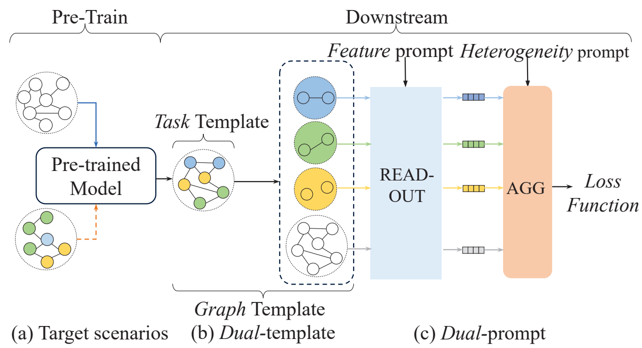


Figure 1: Illustration of HGPROMPT. Black-and-white graphs are homogeneous; colored graphs are heterogeneous, where colors indicate different types of nodes.

To summarize, the contribution of this work is threefold. (1) We propose HGPROMPT, a prompt learning framework on graphs, which bridges the gap between homogeneous and heterogeneous graphs across pre-training and downstream tasks. (2) In HGPROMPT, we design a unified graph template for both heterogeneous and homogeneous graphs, and propose a dual-layer prompt to narrow the gaps caused by the differences in tasks and heterogeneity. (3) We conduct extensive experiments on three benchmark datasets, demonstrating the advantages of HGPROMPT.

2 Related Work

Graph pre-training. Borrowing insights from pre-training methodologies in both vision (Bao et al. 2021) and language domains (Dong et al. 2019), many GNN-based pre-training techniques have been introduced for graph data (Qiu et al. 2020; Bo et al. 2023). Typically, these methods harness the inherent graph structures in a self-supervised fashion, facilitating the transfer of knowledge to downstream tasks through a fine-tuning step. There has also been a surge in the application of pre-training techniques to heterogeneous graphs (Yang et al. 2022; Hwang et al. 2020). Unlike their homogeneous counterparts, heterogeneous graphs are characterized by the presence of heterogeneous nodes and edges. Consequently, these techniques are tailored to leverage the intricate semantic relationships between nodes, originating from the inherent heterogeneity. However, the inconsistency between pre-training and fine-tuning objectives can hinder the performance of downstream tasks (Liu et al. 2023a). While pre-training aims to extract inherent knowledge from the graph in a self-supervised manner, fine-tuning focuses on specific supervision tailored to downstream tasks.

Graph prompt learning. First emerged in the language domain (Brown et al. 2020), prompt learning seeks to reconcile the differences between pre-training and fine-tuning objectives. This is usually achieved by crafting task-specific prompts that guide downstream tasks, while freezing the pre-trained model weights. The success of this technique in language and vision (Jia et al. 2021) has sparked interest in its adoption in graph learning. GPPT (Sun et al. 2022) employs link prediction as its pre-training task, but its prompts are only designed for the downstream node classification task. Both GraphPrompt (Liu et al. 2023b; Yu et al. 2023a) and ProG (Sun et al. 2023) aim to unify pre-training with multiple types of downstream tasks. On one hand, GraphPrompt employs subgraph similarity as its template and devises a learnable prompt for downstream few-shot learning. On the other hand, ProG transforms node and edge-level tasks into graph-level tasks, and assumes a meta-learning setting for prompt learning. Another work VNT (Tan et al. 2023) introduces a learnable virtual node prompt to assimilate task-specific information, while only focusing on downstream node classification in a meta-learning setup. However, none of these methods thoroughly address prompt learning toward bridging the gap between homogeneous and heterogeneous graphs.

3 Preliminaries and Problem Statement

In this section, we present some preliminaries on heterogeneous graph and introduce our problem.

Graph preliminaries. A *heterogeneous graph* (Sun and Han 2012) is represented as $G = (V, E, A, R, \phi, \varphi)$, where V is the set of nodes and E is the set of edges, with a node type mapping function $\phi : V \rightarrow A$ and an edge type mapping function $\varphi : E \rightarrow R$ for a set of node and edge types denoted by A and R, respectively, such that $|A| + |R| > 2$. We also assume an input feature matrix for the nodes, $\mathbf{X} \in \mathbb{R}^{|V| \times d}$. A *homogeneous graph* is defined in the same way, except that $|A| = |R| = 1$, *i.e.*, there is only a single node and edge type.

Problem definition. In our target scenarios, both homogeneous and heterogeneous graphs can be used for pre-training via a link prediction task, given the general availability of links in large-scale graphs without additional annotation costs (Liu et al. 2023b).

In downstream tasks, we deal with heterogeneous graphs, and focus on two popular tasks, namely, node classification and graph classification. For node (or graph) classification on a heterogeneous graph G (or a set of heterogeneous graphs $\mathcal{G}$), let C (or $\mathcal{C}$) be the set of node (or graph) classes, such that a node in G (or a graph in $\mathcal{G}$) is associated with a class in C (or $\mathcal{C}$). We aim to predict the unknown class labels for the instances in the test set in a *few-shot* setting: For each class, we only have k labeled instances for learning.

4 Proposed Model: HGPROMPT

In this section, we present our proposed model HGPROMPT.

4.1 Overall Framework

We begin with the overall framework of HGPROMPT, as seen in Fig. 2. Given some homogeneous or heterogeneous graphs in Fig. 2(a), we aim to pre-train a GNN model using link prediction in Fig. 2(b), since links are available in all graphs "for free" without extra annotation costs. Here we assume the more general scenario of pre-training on homogeneous graphs[1]. To unify pre-training and downstream stages, we propose a *dual-template* design: first, a *graph* template is applied to convert each heterogeneous graph to multiple homogeneous ones in Fig. 2(c, d); second, a *task* template is applied to convert various tasks to a common subgraph similarity prediction task. In the downstream stage, we further propose a *dual-prompt* formulation consisting of both a *feature* prompt and a *heterogeneity* prompt to bridge the feature variations and heterogeneity differences across tasks.

We will discuss the dual-template design as the basis of our unification (Sect. 4.2). Next, we present the pre-training stage (Sect. 4.3) and introduce the dual-prompt design for the downstream stage (Sect. 4.4).

4.2 Dual-Template Unification

We introduce the basis of unifying pre-training and downstream tasks, namely, the graph and task templates.

Graph template. We propose to unify the heterogeneity of the input graphs through graph template. Our strategy is to convert a heterogeneous graph $G = (V, E, A, R, \phi, \varphi)$ into multiple homogeneous ones based on the node types[2]. Specifically, given a node type $i \in A$, we can extract a homogeneous sugbraph $G^i = (V^i, E^i)$ from G such that

$$V^i = \{v \in V \mid \phi(v) = i\}, \tag{1}$$

$$E^i = \{(a, b) \in E \mid a \in V^i \wedge b \in V^i\}. \tag{2}$$

In this way, the heterogeneous G can be converted into a set of homogeneous graphs $\{G^i : i \in A\}$, each retaining one aspect of the heterogeneity. To also preserve the interactions between different types, we further consider a homogeneous graph $G^0 = (V, E)$, *i.e.*, it retains the full topology of G, but it does not distinguish the types. We call the process of converting a heterogenous graph into $|A| + 1$ homogeneous graph as applying a *graph template*, denoted by $\mathcal{GT}$:

$$\mathcal{GT}(G) = \{G^0\} \cup \{G^i : i \in A\}. \tag{3}$$

To summarize, the graph template unifies downstream heterogeneous graphs into the same format as homogeneous graphs that may be used in pre-training. When there are heterogeneous graphs in pre-training, the same graph template can be applied to them as well.

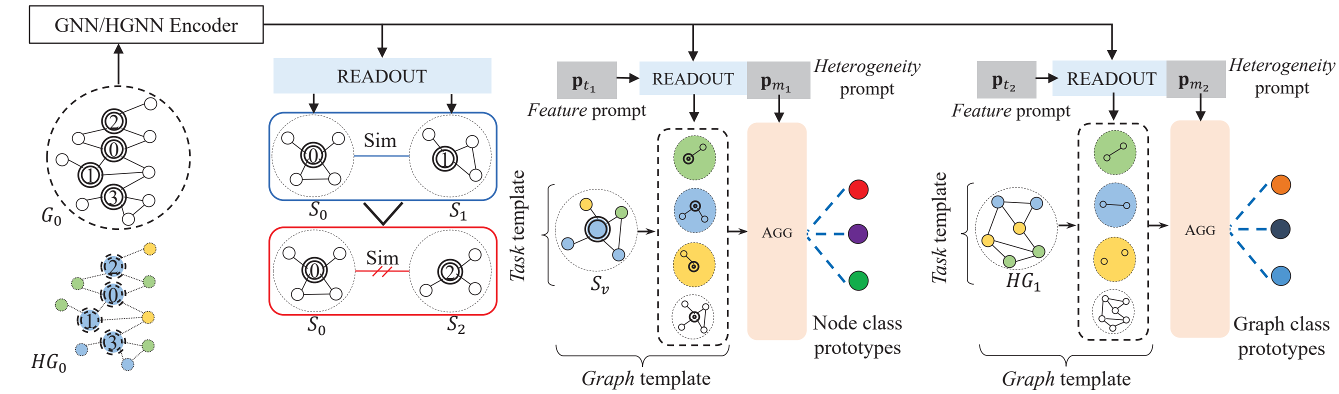
Task template. Next, to unify different tasks, we mainly follow GraphPrompt (Liu et al. 2023b) by converting different task instances into subgraphs, and leverage a common task template of predicting subgraph similarity. However, here we also need to account for the graph template in the formulation of subgraphs and their similarity computation.

First, for node-level tasks including link prediction (LP) and node classification (NC), we can instantiate each node v involved in LP or NC into a context subgraph S_v. A common strategy for S_v is to employ a breadth-first search to extract a δ-hop subgraph centering on v (Liu et al. 2023b). For graph classification (GC), as its instances are already in graph forms, *i.e.*, $S_G = G$ for a graph instance G.

Next, let $\mathbf{s}_v$ denote the embedding vector of a subgraph S_v. If S_v is homogeneous, we can use a direct readout. If S_v

[1]Pre-training on heterogeneous graphs can be achieved by also applying the graph template. See illustration in Appendix A.

[2]Edge types can also be considered just like node types. For brevity, our discussion will illustrate node types only.



(a) Pre-training graphs (b) Pre-training (link prediction) (c) Downstream node classification (d) Downstream graph classification

Figure 2: Overall framework of HGPROMPT. (a) Pre-training graphs can be either homogeneous or heterogeneous. (b) Pre-training task with link prediction on a homogeneous graph[1]. (c) Downstream node classification and (d) graph classification on heterogeneous graphs. Black-and-white graphs are homogeneous; colored graphs are heterogeneous, where colors indicate different types of nodes.

is heterogeneous, the graph template is applied to obtain a set of homogeneous graphs $\mathcal{GT}(S_v)$, which are individually readout and then aggregated into $\mathbf{s}_v$. That is,

$$\mathbf{s}_v = \text{AGG}(\{\text{READOUT}(S_v^i) \mid S_v^i \in \mathcal{GT}(S_v)\}), \quad (4)$$

where the readout layer for a subgraph S is further defined as an aggregation over the features of nodes in S:

$$\text{READOUT}(S) = \text{AGG}_2(\{\mathbf{h}_v : v \in V(S)\}), \quad (5)$$

where $\mathbf{h}_v$ denotes the embedding of node v before the readout layer, and $V(S)$ is the set of nodes in S. Then, the three tasks can be reformulated into predicting the similarity between (sub)graph instances (Liu et al. 2023b).

- ***Link prediction (LP)***: Given a graph G and a node triplet (v, a, b) such that (v, a) is an edge in G and (v, b) is not. It is expected that

$$\text{sim}(\mathbf{s}_v, \mathbf{s}_a) > \text{sim}(\mathbf{s}_v, \mathbf{s}_b), \quad (6)$$

  where $\text{sim}(\cdot, \cdot)$ is a similarity function such as cosine similarity in our implementation.
- ***Node Classification (NC)***: Given a graph G with a set of node classes C, and a labeled node set $D = \{(v_1, \ell_1), (v_2, \ell_2), \ldots\}$ such that ℓ_i is the class label of v_i. In the context of k-shot learning, precisely k pairs of $(v_i, \ell_i = c) \in D$ exist for every class $c \in C$. For a class c, we construct a *node class prototype*, which can be understood as a "virtual" subgraph. Its embedding can be computed as $\tilde{\mathbf{s}}_c = \frac{1}{k} \sum_{(v_i, \ell_i) \in D, \ell_i = c} \mathbf{s}_{v_i}$. Subsequently, for an unlabeled node v_j, its class label ℓ_j can be predicted as

$$\ell_j = \arg\max_{c \in C} \text{sim}(\mathbf{s}_{v_j}, \tilde{\mathbf{s}}_c). \quad (7)$$

- ***Graph classification (GC)***: Given a collection of graphs $\mathcal{G}$ and a set of graph classes $\mathcal{C}$, accompanied by a set of labeled graphs $\mathcal{D} = \{(G_1, L_1), (G_2, L_2), \ldots\}$ such that L_i is the class label of G_i. In k-shot learning, there exist precisely k pairs of $(G_i, L_i = c) \in \mathcal{D}$ for every class $c \in \mathcal{C}$. For a class c, we also define a *graph class prototype* which can be represented by $\tilde{\mathbf{s}}_c = \frac{1}{k} \sum_{(G_i, L_i) \in \mathcal{D}, L_i = c} \mathbf{s}_{G_i}$. Then, given an unlabeled graph G_j, its class label L_j can be predicted as

$$L_j = \arg\max_{c \in \mathcal{C}} \text{sim}(\mathbf{s}_{G_j}, \tilde{\mathbf{s}}_c). \quad (8)$$

4.3 Pre-Training Task

We employ the LP task for pre-training, capitalizing on the abundance of links in large-scale graph data as self-supervision (Hamilton, Ying, and Leskovec 2017; Liu et al. 2023b). As shown in Fig. 2(b), consider a triplet (0, 1, 2) where (0, 1) is an edge and (0, 2) is not. It can be used to guide the computation of subgraph similarity toward $\text{sim}(\mathbf{s}_0, \mathbf{s}_1) > \text{sim}(\mathbf{s}_0, \mathbf{s}_2)$, based on the task template. In general, we gather many such triplets from the pre-training graphs to serve as self-supervision $\mathcal{D}_{\text{pre}}$. Then, the pre-training loss is defined as $\mathcal{L}_{\text{pre}}(\Theta) =$

$$-\sum_{(v,a,b) \in \mathcal{D}_{\text{pre}}} \ln \frac{\exp\left(\frac{1}{\tau}\text{sim}(\mathbf{s}_v, \mathbf{s}_a)\right)}{\sum_{u \in \{a,b\}} \exp\left(\frac{1}{\tau}\text{sim}(\mathbf{s}_v, \mathbf{s}_u)\right)}, \quad (9)$$

where τ is a temperature hyperparameter and Θ represents the model weights of GNN/HGNN.

The pre-training stage outputs the trained model weights, *i.e.*, $\Theta_0 = \arg\min_\Theta \mathcal{L}_{\text{pre}}(\Theta)$, which will be deployed in downstream tasks as we shall see next.

4.4 Dual-Prompting for Downstream Tasks

We aim to address NC and GC tasks in downstream, while leveraging an LP-based pre-trained model. Since the pre-training and downstream tasks involve different objectives, we turn to the paradigm of prompt learning to bridge the gap. In our scenario, different tasks may focus on different features, or different facets of heterogeneity in the input graph. Hence, we propose a *dual-prompt* as shown in Fig. 2(c,d), consisting of a *feature* prompt to handle the feature variations and a *heterogeneity* prompt to cope with the heterogeneity differences across different tasks.

Feature prompt. In the language domain, a prompt modifies or reformulates the input to the pre-trained model based on the specific downstream task, to align the task better with the pre-trained model. Similarly, on graphs, a straightforward way is to modify node features in the input or hidden layers of the pre-trained model. Here, we follow GraphPrompt (Liu et al. 2023b) to modify the node features before the readout layer in Eq. (5), which enables different downstream tasks to focus on different sets of features. Specifically, on a given task, let $\mathbf{p}^{\text{feat}}$ be a task-specific learnable vector to serve as the feature prompt for the task. Under the prompt, the readout layer for some subgraph S becomes

$$\text{READOUT}(\{\mathbf{p}^{\text{feat}} \odot \mathbf{h}_v \mid v \in V(S)\}), \quad (10)$$

where $\odot$ denotes the element-wise multiplication. In other words, the feature prompt modifies the feature importance based on the specific needs of the task. Note that the prompt $\mathbf{p}^{\text{feat}}$ has the same dimension as the node embeddings.

Heterogeneity prompt. When dealing with a heterogeneous (sub)graph S downstream, we need to apply the graph template, which involves an extra layer to aggregate the individual readout outputs from each homogeneous subgraph in $\mathcal{GT}(S)$; see Eq. (4). As various facets of heterogeneity (as suggested by different node/edge types) exist, to distinguish the importance of facets across different tasks, we propose a heterogeneity prompt to modify the input to the aggregation layer. Specifically, let $\mathbf{p}^{\text{het}} = (p_0^{\text{het}}, p_1^{\text{het}}, \ldots, p_{jAj}^{\text{het}})$ denote a task-specific learnable vector as the heterogeneity prompt for a particular task. Note that the prompt vector has $|A| + 1$ dimensions, where each dimension p_i^{het} intends to weigh the i-th subgraph from the graph template $\mathcal{GT}(\cdot)$. Under the prompt, for some subgraph S, the aggregation layer after applying the graph template becomes

$$\text{AGG}(\{(1 + p_i^{\text{het}}) \odot \text{READOUT}(S^i) \mid S^i \in \mathcal{GT}(S)\}), \quad (11)$$

which gives the embedding vector $\mathbf{s}$ for the subgraph S. Hence, the prompt is able to adjust the importance of every facet carried by each subgraph $S^i \in \mathcal{GT}(S)$ based on the task nature.

Prompt tuning. To tune the dual-prompt for a downstream task, we resort to a loss based on subgraph similarity just as pre-training. Consider an NC or GC task with a labeled training set $\mathcal{D}_{\text{down}} = \{(x_1, y_1), (x_2, y_2), \ldots\}$, where each x_i is either a node or a graph, and $y_i \in Y$ is x_i's class label from a set of classes Y. Then, the prompt tuning loss is defined as $\mathcal{L}_{\text{down}}(\mathbf{p}^{\text{feat}}, \mathbf{p}^{\text{het}}) =$

$$-\sum_{(x_i,y_i) 2 D_{\text{down}}} \ln \frac{\exp\left(\frac{1}{\tau}\text{sim}(\mathbf{s}_{x_i}, \tilde{\mathbf{s}}_{y_i})\right)}{\sum_{c 2 Y} \exp\left(\frac{1}{\tau}\text{sim}(\mathbf{s}_{x_i}, \tilde{\mathbf{s}}_c)\right)}. \quad (12)$$

Note that the subgraph and class prototype embeddings $\mathbf{s}_{x_i}$ and $\tilde{\mathbf{s}}_c$ are generated based on the prompt vectors $\mathbf{p}^{\text{feat}}, \mathbf{p}^{\text{het}}$; see Eqs. (10) and (11). During prompt tuning, only the light-weight prompt vectors are tuned, while the pre-trained weights Θ_0 are frozen without any fine-tuning. Such parameter-efficient tuning is amenable to few-shot settings when $\mathcal{D}_{\text{down}}$ only consists of a few training examples.

	# Nodes	# Node Types	# Edges	# Edge Types	Target Type	# Classes
ACM	10,942	4	547,872	8	paper	3
DBLP	26,128	4	239,566	6	author	4
Freebase	180,098	8	1,057,688	36	book	7

Table 1: Summary of datasets.

5 Experiments

In this section, we conduct experiments to evaluate our proposed approach, and analyze the empirical results.

5.1 Experimental Setup

Datasets. We conduct experiments on three benchmark datasets. (1) *ACM* serves as a citation network, which comprises papers from five conferences, classified into three distinct categories: Database, Wireless Communication, and Data Mining. (2) *DBLP* serves as an all-encompassing bibliographic database housing computer science research papers and proceedings. (3) *Freebase* (Bollacker et al. 2008) is a functional and scalable tuple database utilized for organizing comprehensive human knowledge in a structured manner. For all these datasets, we employ the same raw data as Simple-HGN (Lv et al. 2021). We provide a summary of these datasets in Table 1.

Baselines. We assess the performance of HGPROMPT against state-of-the-art approaches from five primary categories as outlined below.

(1) *End-to-end homogeneous graph neural networks*: GCN (Kipf and Welling 2016) and GAT (Veličković et al. 2018a). These homogeneous GNNs leverage the central operation of neighborhood aggregation to iteratively collect messages from neighboring nodes, operating in an end-to-end manner. (2) *End-to-end heterogeneous graph neural networks (HGNNs)*: Simple-HGN (Lv et al. 2021) and HAN (Wang et al. 2019). Compared to homogeneous GNNs, these HGNNs incorporate heterogeneity through heterogeneous neighborhood aggregation w.r.t. edge types or meta-paths. (3) *Homogeneous Graph pre-training models*: DGI/InfoGraph[3] (Veličković et al. 2018b; Sun et al. 2020) and GraphCL (You et al. 2020). These approaches follow the "pre-train, fine-tune" paradigm. Specifically, they first pre-train the GNN models to exploit the intrinsic properties of the graphs, and subsequently fine-tune the pre-trained weights on downstream tasks to adapt to task-specific labels. (4) *Heterogeneous Graph pre-training models*: CPT-HG (Jiang et al. 2021) and HeCo (Wang et al. 2021), which also follow the "pre-train, fine-tune" paradigm. However, they design heterogeneous tasks to learn heterogeneity information during pre-training. (5) *Graph Prompt Models*: GPPT (Sun et al. 2022) and GraphPrompt (Liu et al. 2023b). They employ LP for pre-training and unify downstream tasks with the pre-training task using a consistent template.

[3]Original DGI only works at the node level, while InfoGraph extends it to the graph level. In our experiments, we use DGI for NC, and InfoGraph for GC.

Methods	Node classification						Graph classification					
	ACM		DBLP		Freebase		ACM		DBLP		Freebase	
	MicroF	MacroF	MicroF	MacroF	MicroF	MacroF	MicroF	MacroF	MicroF	MacroF	MicroF	MacroF
GCN	50.2±12.1	44.2±16.6	51.3±15.4	48.6±17.2	17.1±11.8	15.3± 9.5	33.5± 1.0	17.7± 2.7	38.1±15.0	35.0±16.2	17.3± 2.9	16.5± 2.5
GAT	38.5± 7.8	28.0±12.4	65.0±11.6	62.8±13.2	17.9± 8.5	16.5± 6.8	33.5± 1.0	17.2± 1.5	46.4±14.5	40.3±16.8	16.3± 2.6	16.0± 2.1
SIMPLE-HGN	45.5±10.6	40.5±14.2	59.7±15.2	57.3±16.2	17.8± 9.1	16.0± 7.8	33.3±10.8	16.8± 6.8	44.5±15.8	39.5±17.1	17.8± 2.8	16.7± 2.4
HAN	62.2± 9.1	57.1±11.7	62.4±18.9	60.6±19.7	18.7±12.4	16.8± 8.1	35.3± 8.9	22.2± 5.6	47.0±16.5	42.0±17.9	18.4± 3.1	17.3± 2.6
DGI/INFOGRAPH	59.4±18.8	56.2±24.5	68.2±15.1	65.3±16.1	18.3± 9.2	16.8± 6.7	34.7±16.3	20.6±16.6	51.0±15.6	45.1±16.0	18.2± 2.8	17.3± 2.4
GRAPHCL	58.5±16.6	55.4±21.3	66.5±15.1	64.6±15.2	18.4± 9.2	16.9± 6.2	33.1±13.4	21.7±12.0	50.6±15.7	44.7±16.2	18.3± 2.8	17.4± 2.3
CPT-HG	62.5±15.4	58.4±20.1	70.6±13.4	67.0±12.5	19.0± 8.5	17.4± 5.5	35.7±19.5	27.0±18.8	53.6±14.9	46.9±15.5	18.9± 2.7	17.9± 2.1
HECO	63.3±15.1	59.1±17.4	70.7±11.6	67.6±12.3	19.4± 8.1	18.1± 6.1	35.4±15.9	27.3±14.4	53.7±13.2	47.5±15.2	19.3± 2.6	18.5± 2.3
GPPT	54.8±10.8	51.6±12.2	61.4± 9.2	63.2±10.6	17.5± 7.7	16.5± 5.1	-	-	-	-	-	-
GRAPHPROMPT	65.6±13.6	61.1±14.8	73.7± 9.9	69.2±10.6	19.8± 6.0	18.1± 4.4	36.1±15.7	27.7±15.4	56.6±11.6	49.0±13.6	19.7± 2.4	18.5± 1.8
HGPROMPT	<u>68.2</u>±11.3	<u>63.5</u>±13.0	<u>77.6</u>± 9.7	<u>73.2</u>±10.9	<u>21.2</u>± 7.0	<u>19.8</u>± 4.8	<u>37.5</u>±15.0	<u>28.9</u>±14.8	<u>58.4</u>±10.6	<u>47.7</u>± 8.8	<u>20.9</u>± 2.8	<u>19.8</u>± 2.1
HGPROMPT+	**74.0**±12.4	**74.4**±14.2	**82.3**±10.6	**77.4**±12.0	**22.6**± 7.9	**21.0**± 5.7	**39.7**±15.0	**32.9**±18.0	**60.5**±11.4	**52.0**±13.9	**21.7**± 3.3	**20.9**± 3.2

Table 2: Evaluation (%) of node and graph classification. The best method is bolded and the runner-up is underlined.

It is worth noting that GPPT exclusively aligns the downstream NC with their pre-training task. (See Appendix B for more details).

We highlight that other few-shot methods on graphs, such as Meta-GNN (Zhou et al. 2019), AMM-GNN (Wang et al. 2020), RALE (Liu et al. 2021), VNT (Tan et al. 2023) and ProG (Sun et al. 2023), rely on a meta-learning paradigm (Finn, Abbeel, and Levine 2017). Thus, they are not applicable to our setting since they need a large amount of labeled data in their base classes for meta-training. Conversely, our approach exclusively employs label-free graphs for pre-training.

Settings and parameters. To assess if our framework generalizes well to different downstream tasks, we focus on few-shot NC and GC following previous work (Liu et al. 2023b). To further test the robustness of our approach, we also adopt few-shot LP as a downstream task, although it does not have a typical few-shot setting. Hence, we will present the results on NC and GC here, and defer the results of LP to Appendix D. Furthermore, we name our model HGPROMPT and HGPROMPT+: HGPROMPT deals with the target scenario where pre-training graphs are homogeneous, while HGPROMPT+ copes with the scenario where pre-training graphs are heterogeneous (by also applying the graph template in pre-training); see Fig. 1(a).

The downstream tasks adhere to a k-shot learning setting. Further elaboration on the task construction process will be provided when presenting the results of each kind of task in Sect. 5.2. We adopt MicroF and MacroF (Pedregosa et al. 2011; Lv et al. 2021) as the evaluation metrics.

For hyperparameter settings and other implementation details about the baselines and HGPROMPT, see Appendix C.

5.2 Performance Evaluation

For both few-shot NC and GC tasks, we first evaluate a fixed-shot setting, and subsequently vary the number of shots to observe the performance trends.

Few-shot node classification. Following the typical k-shot setup (Wang et al. 2020; Liu et al. 2021, 2023b), we randomly generate 100 one-shot tasks for model training and validation. The results of one-shot NC are presented in Table 2. We make the following observations. (1) HGPROMPT surpasses all baselines on the three datasets, indicating the efficacy of HGPROMPT in bridging homogeneous and heterogeneous graphs across pre-training and downstream tasks. (2) HGPROMPT+ outperforms HGPROMPT, demonstrating the benefit of further leveraging heterogeneity in pre-training (assuming made available) via graph templates. Similarly, CPT-HG and HeCo also make use of heterogeneity in pre-training, and perform better than their homogeneous counterparts DGI/InfoGraph and GraphCL. However, CPG-HG and HeCo still lag behind HGPROMPT and HGPROMPT+ significantly despite an already unified graph format (i.e., heterogeneous graphs in both pre-training and downstream tasks), further implying the importance of the task template and our dual-prompt design in few-shot settings. (3) HAN outperforms other supervised methods because it leverages meta-paths, which are pre-defined using domain knowledge. Also note that GAT and Simple-HGN often perform poorly due to their large model size, which are not intended for few-shot settings.

Few-shot graph classification. Following previous work (Lu et al. 2021), we create a set of graphs by gathering ego-networks centered on the target nodes (i.e., those with class labels) in each dataset. Each ego-network is assigned the same label as its corresponding ego-node. Again, we ran-

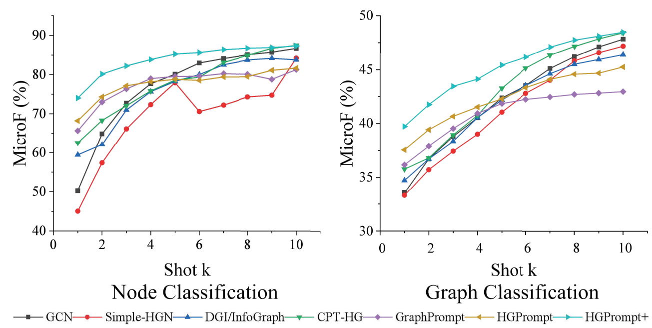


Figure 3: Impact of shots on NC and GC tasks on *ACM*.

Methods	Graph template (pre-training)	Heterogeneity prompt	Graph template (downstream)	Feature prompt	Node classification ACM	Node classification DBLP	Node classification Freebase	Graph classification ACM	Graph classification DBLP	Graph classification Freebase
VARIANT 1	×	×	×	×	55.92	56.49	17.43	33.92	40.47	16.78
VARIANT 2	×	×	×	✓	61.10	69.26	18.18	35.18	56.68	19.77
VARIANT 3	×	×	✓	✓	62.34	70.77	18.63	35.70	56.23	19.03
HGPROMPT	×	✓	✓	✓	**63.57**	**73.22**	**19.81**	**37.57**	**58.49**	**20.91**
VARIANT 1+	✓	×	×	×	61.79	60.71	18.09	35.48	48.56	17.81
VARIANT 2+	✓	×	×	✓	68.47	72.05	19.88	36.85	58.24	20.85
VARIANT 3+	✓	×	✓	✓	71.37	74.36	20.73	37.60	59.08	20.32
HGPROMPT+	✓	✓	✓	✓	**74.47**	**77.44**	**21.08**	**39.72**	**60.53**	**21.70**

Table 3: Variants used in ablation study, and corresponding results in MicroF (%) on node and graph classification.

domly generate 100 one-shot GC tasks for model training and validation. We report the results of one-shot GC in Table 2. We observe similar patterns to those on the NC results, showing robustness of our approach on both node- and graph-level tasks.

Performance with different shots. To further investigate the impact of shots, we vary the number of shots for both NC and GC tasks, and benchmark against several competitive baselines in Fig. 3 on *ACM* (see Appendix D for other datasets). We make the following observations. (1) HGPROMPT outperforms the baselines when very limited labeled data are given (*e.g.*, fewer than 5 shots), and remains competitive when more shots are available. Note that CPT-HG utilizes heterogeneous information in pre-training, and hence has an advantage to HGPROMPT, and HGPROMPT+ serves as a better comparison. (2) HGPROMPT+ outperforms all baselines even when increasing to 10 shots, by also making use of the heterogeneity via graph templates in pre-training. (3) Generally, the performance advantages of HGPROMPT and HGPROMPT+ diminish as more shots are used, which is expected as they are especially designed for label-scarce settings.

5.3 Model Analyses

Ablation study. To study the impact of each component within our model, we undertake an ablation study. We modify HGPROMPT against design variations associated with the key components of dual-template and dual-prompt, as shown in Table 3. Note that for all variants, the task template is always applied as that is the basis of our framework. Furthermore, certain constraints exist, *e.g.*, a heterogeneity prompt is only meaningful if the graph template is applied downstream.

The results in Table 3 show that various components are useful, as follows. (1) The absence of a heterogeneity prompt diminishes performance. This is evident when contrasting HGPROMPT with Variant 3 and HGPROMPT+ with Variant 3+, respectively, demonstrating the need to bridge the heterogeneity differences across tasks. (2) Omitting graph template for downstream tasks similarly impacts outcomes adversely, given the generally better performance of Variant 3 against Variant 2, and that of Variant 3+ against Variant 2+. (3) Applying graph templates during pre-training yields clear benefits. Notably, all variants annotated with "+" consistently outperform their counterparts without "+", demonstrating that graph templates can also leverage heterogeneous information in pre-training effectively. (4) Variants 1/1+ typically achieve the least optimal performance, underscoring the necessity of graph templates and dual-prompt.

Backbone	Method	Node Classification MicroF	Node Classification MacroF	Graph Classification MicroF	Graph Classification MacroF
GCN	SUPERVISED	50.26	44.21	33.56	17.72
	GRAPHPROMPT	65.67	61.10	36.18	27.70
	HGPROMPT	68.23	63.57	37.57	28.90
	HGPROMPT+	**74.04**	**74.47**	**39.72**	**32.96**
GAT	SUPERVISED	38.50	28.01	33.52	17.25
	GRAPHPROMPT	38.87	31.97	34.52	23.44
	HGPROMPT	38.98	33.85	35.75	24.56
	HGPROMPT+	**45.86**	**42.50**	**37.10**	**25.33**
SIMPLE-HGN	SUPERVISED	45.57	40.58	33.31	16.83
	GRAPHPROMPT	49.62	45.25	34.47	21.64
	HGPROMPT	51.79	47.17	35.38	22.83
	HGPROMPT+	**54.81**	**51.06**	**36.95**	**24.15**

Table 4: Evaluation (%) of different backbones for NC and GC tasks on *ACM*. SUPERVISED means end-to-end training.

Flexibility on backbones. To further assess the flexibility and robustness of HGPROMPT, we examine its performance on different GNN and HGNN backbones, including GCN, GAT and SIMPLE-HGN. We report the results in Table 4 on *ACM* (see Appendix D for other datasets). We observe that irrespective of the backbone employed, HGPROMPT+ consistently emerges as the top performer, followed by HGPROMPT, implying the robustness of our proposed framework.

Parameter efficiency. We further investigate the number of parameters that require updating during downstream tasks. See Appendix D for more details.

6 Conclusions

In this work, we delved into few-shot prompting on heterogeneous graphs. We introduced HGPROMPT, aiming to bridge the gap between homogeneous and heterogeneous graphs. We proposed dual-template to unify downstream tasks with pre-training irrespective of graph heterogeneity. We proposed dual-prompt to narrow the gap caused by feature and heterogeneity variations. Comprehensive evaluations on three benchmark datasets further illustrate the advantages of HGPROMPT.

Acknowledgments

This research / project is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (Proposal ID: T2EP20122-0041). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore. This work is also supported in part by the National Key Research and Development Program of China under Grant 2020YFB2103803. The authors wish to thank Dr. Deyu Bo from Beijing University of Posts and Telecommunications for his valuable comments on this work.

References

Bao, H.; Dong, L.; Piao, S.; and Wei, F. 2021. BEiT: BERT pre-training of image transformers. In *International Conference on Learning Representations*.

Bo, D.; Fang, Y.; Liu, Y.; and Shi, C. 2023. Graph contrastive learning with stable and scalable spectral encoding. *Advances in Neural Information Processing Systems*, 36.

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *the ACM SIGMOD International Conference on Management of Data*, 1247–1250.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33: 1877–1901.

Dong, L.; Yang, N.; Wang, W.; Wei, F.; Liu, X.; Wang, Y.; Gao, J.; Zhou, M.; and Hon, H.-W. 2019. Unified language model pre-training for natural language understanding and generation. *Advances in Neural Information Processing Systems*, 32: 13063–13075.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 1126–1135.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30: 1025–1035.

Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V.; and Leskovec, J. 2020. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*.

Hwang, D.; Park, J.; Kwon, S.; Kim, K.; Ha, J.-W.; and Kim, H. J. 2020. Self-supervised auxiliary learning with meta-paths for heterogeneous graphs. *Advances in Neural Information Processing Systems*, 33: 10294–10305.

Jia, C.; Yang, Y.; Xia, Y.; Chen, Y.-T.; Parekh, Z.; Pham, H.; Le, Q.; Sung, Y.-H.; Li, Z.; and Duerig, T. 2021. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, 4904–4916.

Jiang, X.; Lu, Y.; Fang, Y.; and Shi, C. 2021. Contrastive pre-training of GNNs on heterogeneous graphs. In *the ACM International Conference on Information and Knowledge Management*, 803–812.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; and Neubig, G. 2023a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9): 1–35.

Liu, Z.; Fang, Y.; Liu, C.; and Hoi, S. C. 2021. Relative and absolute location embedding for few-shot node classification on graph. In *the AAAI Conference on Artificial Intelligence*, 4267–4275.

Liu, Z.; Yu, X.; Fang, Y.; and Zhang, X. 2023b. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *the ACM Web Conference*, 417–428.

Lu, Y.; Jiang, X.; Fang, Y.; and Shi, C. 2021. Learning to pre-train graph neural networks. In *the AAAI Conference on Artificial Intelligence*, 4276–4284.

Lv, Q.; Ding, M.; Liu, Q.; Chen, Y.; Feng, W.; He, S.; Zhou, C.; Jiang, J.; Dong, Y.; and Tang, J. 2021. Are we really making much progress? Revisiting, benchmarking and refining heterogeneous graph neural networks. In *the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1150–1160.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830.

Qiu, J.; Chen, Q.; Dong, Y.; Zhang, J.; Yang, H.; Ding, M.; Wang, K.; and Tang, J. 2020. GCC: Graph contrastive coding for graph neural network pre-training. In *the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1150–1160.

Sun, F.-Y.; Hoffman, J.; Verma, V.; and Tang, J. 2020. InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*.

Sun, M.; Zhou, K.; He, X.; Wang, Y.; and Wang, X. 2022. GPPT: Graph pre-training and prompt tuning to generalize graph neural networks. In *the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1717–1727.

Sun, X.; Cheng, H.; Li, J.; Liu, B.; and Guan, J. 2023. All in one: Multi-task prompting for graph neural networks. In *the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2120–2131.

Sun, Y.; and Han, J. 2012. *Mining heterogeneous information networks: Principles and methodologies*. Morgan & Claypool Publishers.

Sun, Y.; and Han, J. 2013. Mining heterogeneous information networks: a structural analysis approach. *ACM SIGKDD Explorations Newsletter*, 14(2): 20–28.

Tan, Z.; Guo, R.; Ding, K.; and Liu, H. 2023. Virtual Node Tuning for Few-shot Node Classification. In *the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2177–2188.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018a. Graph attention networks. In *International Conference on Learning Representations*.

Veličković, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2018b. Deep graph infomax. In *International Conference on Learning Representations*.

Wang, N.; Luo, M.; Ding, K.; Zhang, L.; Li, J.; and Zheng, Q. 2020. Graph few-shot learning with attribute matching. In *the ACM International Conference on Information and Knowledge Management*, 1545–1554.

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019. Heterogeneous graph attention network. In *The Web Conference*, 2022–2032.

Wang, X.; Liu, N.; Han, H.; and Shi, C. 2021. Self-supervised heterogeneous graph neural network with co-contrastive learning. In *the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1726–1736.

Yang, C.; Xiao, Y.; Zhang, Y.; Sun, Y.; and Han, J. 2020. Heterogeneous network representation learning: A unified framework with survey and benchmark. *IEEE Transactions on Knowledge and Data Engineering*, 34(10): 4854–4873.

Yang, Y.; Guan, Z.; Wang, Z.; Zhao, W.; Xu, C.; Lu, W.; and Huang, J. 2022. Self-supervised heterogeneous graph pre-training based on structural clustering. *Advances in Neural Information Processing Systems*, 35: 16962–16974.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33: 5812–5823.

Yu, X.; Liu, Z.; Fang, Y.; Liu, Z.; Chen, S.; and Zhang, X. 2023a. Generalized graph prompt: Toward a unification of pre-training and downstream tasks on graphs. arXiv:2311.15317.

Yu, X.; Zhou, C.; Fang, Y.; and Zhang, X. 2023b. MultiGPrompt for multi-task pre-training and prompting on graphs. arXiv:2312.03731.

Zhou, F.; Cao, C.; Zhang, K.; Trajcevski, G.; Zhong, T.; and Geng, J. 2019. Meta-GNN: On few-shot node classification in graph meta-learning. In *the ACM International Conference on Information and Knowledge Management*, 2357–2360.