

# Text-Free Multi-domain Graph Pre-training: Toward Graph Foundation Models

Xingtong Yu

Singapore Management University  
Singapore  
xingtongyu@smu.edu.sg

Yuan Fang<sup>†</sup>

Singapore Management University  
Singapore  
yfang@smu.edu.sg

Chang Zhou

University of Science and Technology of China  
China  
chouchang21sy@mail.ustc.edu.cn,

Xinming Zhang<sup>†</sup>

University of Science and Technology of China  
China  
inming@ustc.edu.cn

## Abstract

Given the ubiquity of graph data, it is intriguing to ask: Is it possible to train a graph foundation model on a broad range of graph data across diverse domains? A major hurdle toward this goal lies in the fact that graphs from different domains often exhibit profoundly divergent characteristics. Although there have been some initial efforts in integrating multi-domain graphs for pre-training, they primarily rely on textual descriptions to align the graphs, limiting their application to text-attributed graphs. Moreover, different source domains may conflict or interfere with each other, and their relevance to the target domain can vary significantly. To address these issues, we propose MDGPT, a text free Multi-Domain Graph Pre-Training and adaptation framework designed to exploit multi-domain knowledge for graph learning. First, we propose a set of *domain tokens* to align features across source domains for synergistic pre-training. Second, we propose a dual prompts, consisting of a *unifying prompt* and a *mixing prompt*, to further adapt the target domain with unified multi-domain knowledge and a tailored mixture of domain-specific knowledge. Finally, we conduct extensive experiments involving six public datasets to evaluate and analyze MDGPT, which outperforms prior art by up to 37.9%. (Codes are available at <https://anonymous.4open.science/r/MDGPT> for anonymous review.)

## CCS Concepts

• **Information systems** → **Web mining; Data mining; • Computing methodologies** → **Learning latent representations.**

## Keywords

Data mining, graph learning, graph foundation model, multi-domain pre-training, prompt learning, few-shot learning.

<sup>†</sup>Corresponding authors.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
ACM ISBN 978-1-4503-XXXX-X/18/06  
<https://doi.org/XXXXXXX.XXXXXXX>

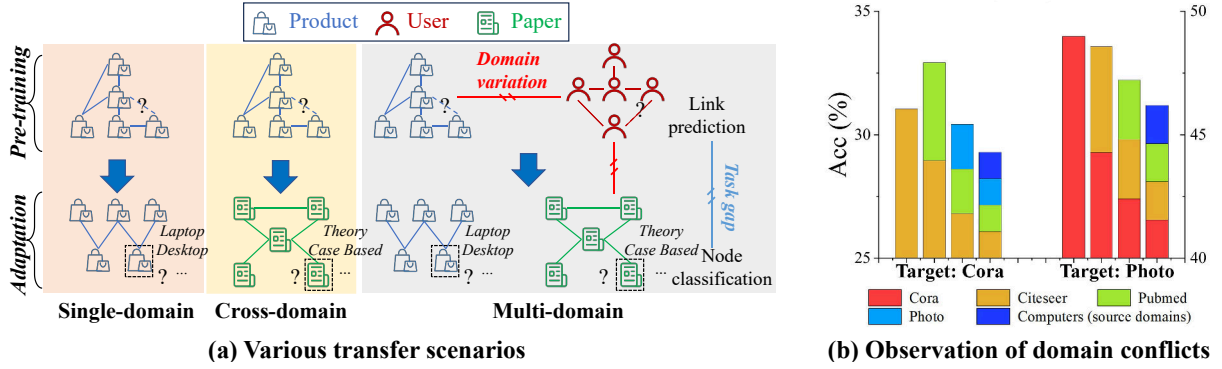
## ACM Reference Format:

Xingtong Yu, Chang Zhou, Yuan Fang<sup>†</sup>, and Xinming Zhang<sup>†</sup>. 2018. Text-Free Multi-domain Graph Pre-training: Toward Graph Foundation Models. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 12 pages. <https://doi.org/XXXXXXX.XXXXXXX>

## 1 Introduction

Graph data are ubiquitous in many domains, owing to their ability to model complex relationships among entities. Thus, applications of graph data are widespread, ranging from e-commerce graphs for recommendation systems [9, 42], to citation graphs for bibliographic study [12, 44], and social networks for user behavioral analysis [4, 24]. These graphs constitute a vast knowledge repository with abundant and comprehensive information across various domains. Toward building a foundation model for graph analysis [17], it is vital to train a universal graph model based on *a broad range of graph data from multiple domains*, which can be further adapted to solve *diverse graph-centric tasks across different downstream fields*.

However, mainstream graph learning methods fall short of achieving this goal. Conventional approaches often train graph neural networks (GNNs) [14, 34, 49] or graph transformers [11, 43, 52] in a supervised fashion, which requires not only re-training for each specific task, but also significant labeled data for each task. Inspired by the advances in pre-trained models for language [3, 13] and vision [1, 7] data, pre-training graph models [10, 35, 46] has been widely explored to overcome the limitations of supervised approaches. They learn task-irrelevant, universal properties on unlabeled graphs, and the pre-trained graph model can be further tailored to different downstream tasks using some task-specific labels through finetuning [35, 46]. To further narrow the *task gap* between pre-training and downstream tasks, prompt learning methods [19, 32] emerge as a popular alternation of finetuning approaches. However, in prevailing solutions to graph pre-training, the pre-training and downstream graphs typically originate from the same dataset consisting of one or more graphs in the same domain, including different sub-graphs of a large graph [19, 35], or a collection of similar graphs [10, 25].



**Figure 1: Motivation of MDGPT. (a) Different scenarios of pre-training and downstream adaptation. (b) Accuracy of one-shot node classification with DGI on two target domains, namely, *Cora* and *Photo*, as more source domains are added to pre-training.**

Therefore, it is imperative to pre-train graph models on a broad spectrum of multi-domain graphs. Yet, graphs from different domains may exhibit profoundly divergent characteristics. For example, the small-world properties of a social network are not applicable to molecular graphs. The distinct nature of graphs from various domains poses a genuine hurdle in integrating broad multi-domain knowledge and achieving universal cross-domain transfer. On one hand, although some works have attempted cross-domain transfer from a single source domain [2, 8, 36, 38, 45], they fail to leverage comprehensive knowledge from a broad range of domains. On the other hand, a few recent studies [16, 33] employ large language models to process or extract node features based on associated textual descriptions in multi-domain graphs, where texts act as a unifying medium across different domains. Consequently, these approaches are limited to text-attributed graphs [40], and do not extend to general graphs that lack explicit textual data. In this work, we propose MDGPT, a foundational graph framework for Text-Free Multi-Domain Graph Pre-Training, to facilitate downstream adaptation to both seen and unseen domains. The solution is non-trivial due to two key challenges.

First, *how do we align multi-domain graphs in the pre-training phase?* Node features in various domains exhibit different distributions, often with unique dimensionality and semantics. For example, features in a citation network denote words in a paper, whereas in a commerce graph, they describe attributes of a product, as shown in Fig. 1(a). Therefore, directly integrating multi-domain graphs may result in conflicts or interferences rather than synergy. As illustrated in Fig. 1(b), adding more source domains often leads to worse performance for conventional graph pre-training methods such as DGI [35]. Although some techniques like singular value decomposition (SVD) [29] can be used to align feature dimensions, the semantic meanings remain misaligned. A recent study [53] proposes a virtual node for each domain, linking all nodes within and then interlinking these virtual nodes. However, this method lacks an explicit mechanism for semantic alignment, leading to suboptimal performance. In MDGPT, we propose a series of *domain tokens* to align node features from different domains. Each domain-specific token is associated with a source domain, serving to bridge the semantic meanings in distinct domains. The token takes the

form of a learnable vector, encouraging a synergistic multi-domain integration.

Second, *how do we adapt multi-domain prior knowledge to downstream tasks in different domains?* The downstream task could be in either a seen or unseen domain. A recent multi-domain pre-training work [53] employs a finetuning or prompt-based approach for downstream adaptation. However, multi-domain prior knowledge contains not only unified global knowledge across domains, but also domain-specific knowledge in each domain. Conventional adaptation approaches for single-domain graph data fall short of effectively transferring such prior knowledge to the target domain. In MDGPT, we propose *dual prompts*, consisting of a *unifying prompt* and a *mixing prompt*, to align the downstream target domain with the source domains, thus facilitating the transfer of multi-domain knowledge to downstream tasks. On one hand, unifying prompt is a learnable vector, serving as a bridge to holistically align the target domain with the unified pre-trained knowledge from all source domains. On the other hand, mixing prompt integrates the individual domain tokens, aligning the target domain with a tailored mixture of knowledge from each source domain, thereby obtaining domain-specific prior knowledge for more precise adaptation.

In summary, the contributions of this work are fourfold. (1) We propose MDGPT, a text-free multi-domain pre-training and prompting framework on graphs, paving a plausible path towards graph foundation models. (2) In pre-training, we design domain tokens to align nodes features, optimizing the pre-training model with multi-domain data. (3) In downstream adaptation, we propose a dual-prompt design with a unifying prompt to align the target domain with the unified pre-trained prior, and a mixing prompt for finer-grained domain-specific alignment. (4) We conduct extensive experiments on five benchmark datasets, demonstrating the superior performance of MDGPT in comparison to the state-of-the-art approaches.

## 2 Related Work

Pre-training methods on graphs [10, 19, 35, 47, 48] capitalize on the intrinsic properties of graph structures via self-supervised learning. Pre-trained knowledge can be transferred to downstream tasks through fine-tuning or parameter-efficient adaptation. However, these methods assume that the pre-training and downstream graphs

come from the same domain (or very similar domains), such as different subgraphs of a large graph [46, 48], or a collection of similar graphs in the same field [10, 25]. As a result, these methods struggle to generalize across multi-domain graphs.

Another line of research on cross-domain learning addresses multiple domains [2, 8, 36, 38]. These studies focus on transferring prior knowledge pre-trained on a single source domain to a different target domain by exploiting domain-invariant representations. However, they are only pre-trained on a single source domain, falling short of utilizing comprehensive multi-domain knowledge. Additionally, they are often tailored to specific tasks or domains [2, 8, 36, 38], or requiring domain expertise [2, 36], which prevents them from generalize to diverse domains and tasks.

Toward multi-domain graph pre-training, OFA [16] and HiGPT [33] employ a language model to derive and process node features from different domains through the medium of natural language, thereby limiting themselves to text-attributed graphs [41, 47]. For text-free graphs, GCOPE [53] introduces a series of domain-specific, interconnecting virtual nodes, which link to other nodes from each domain. While these virtual nodes facilitate some level of connection and alignment across the multiple domains, they do not explicitly align node features from different domains, and thus cannot effectively overcome domain conflicts. Moreover, it only transfers unified pre-trained knowledge to the target domain, overlooking source domain-specific knowledge. Inspired by multi-task pre-training [39, 51], in MDGPT, we propose the notions of domain tokens and dual prompts to overcome the shortcomings of existing multi-domain graph pre-training methods. It is important to note that multi-task pre-training addresses the interference arising from various pre-training tasks within a single domain, which is distinct from our goal to mitigate domain conflicts involving multiple domains.

### 3 Preliminaries

In this section, we present the related preliminaries and our problem definition.

**Graph.** A graph is defined as  $G = (V, E, \mathbf{X})$ , where  $V$  is the set of nodes and  $E$  is the set of edges.  $\mathbf{X} \in \mathbb{R}^{|V| \times d}$  is the feature matrix for the nodes, so that the feature vector of node  $v_i$  in  $V$  is  $\mathbf{x}_i \in \mathbb{R}^d$ . Furthermore, a set of graphs is denoted as  $\mathcal{G}$ .

**Multi-domain pre-training.** Each pre-training graph belongs to a source domain  $D_{S_i}$ , characterized by a node distribution  $P_{S_i}^{(V)}$  and edge distribution  $P_{S_i}^{(E)}$ . Hence, given a set of multi-domain pre-training graphs  $\mathcal{G}_S$  and domains  $\mathcal{D}_S$ , we have  $\{(G_1, D_{S_1}), (G_2, D_{S_2}), \dots, (G_K, D_{S_K})\}$ , where each graph  $G_i \in \mathcal{G}_S$  belongs to domain  $D_{S_i} \in \mathcal{D}_S$ . Note that more than one graph may belong to the same domain, i.e., we may have  $S_i = S_j$  for some  $i, j \leq K$ . In general, different domains follow different node and edge distributions, with varying node feature dimensions and semantics.

**Multi-domain few-shot adaptation.** Given a downstream task, there is a set of graphs  $\mathcal{G}_T$  belonging to a target domain  $D_T$ . The target domain is either *seen* during pre-training (i.e.,  $\exists i D_T = D_{S_i}$ ), or *unseen* (i.e.,  $\forall i D_T \neq D_{S_i}$ ). We address few-shot node and graph classification tasks downstream, where each task only requires a small number of task-specific labels. Specifically, for node

classification within a graph  $G = (V, E, \mathbf{X}) \in \mathcal{G}_T$ , each node  $v \in V$  is associated with a class  $y \in Y$ , where  $Y$  is the set of node classes for the task. For graph classification across a series of graphs  $\mathcal{G}_T$ , each graph  $G \in \mathcal{G}_T$  is associated with a class  $y \in Y$ , where  $Y$  denotes the set of graph classes. In both node and graph classification, each class has only  $m$  labeled examples, where  $m$  is a small number. We call such a task as  $m$ -shot node or graph classification.

## 4 Proposed Approach

In this section, we present our proposed model, MDGPT, starting with an overview, followed by the pre-training and adaptation processes.

### 4.1 Overall framework

We present the overall framework of MDGPT in Fig. 2.

**Pre-training.** As shown in Fig. 2(a), we first align the feature dimensions of graphs from multiple source domains. Next, we propose a series of *domain tokens* to further unify the semantic spaces of features from different domains. Subsequently, we pre-train a graph encoder using a self-supervised task based on a universal task template following previous work [19].

**Downstream adaptation.** As depicted in Fig. 2(b), after an initial dimension alignment for the target domain, we propose *dual prompts*. On one hand, we employ a *unifying prompt*, a learnable vector designed to holistically align the target domain with the unified pre-trained knowledge from all domains. On the other hand, we utilize a *mixing prompt*, a learnable aggregation of domain tokens to align the target domain with a tailored mixture of source domain-specific knowledge for fine-grained adaptation. The dual prompts then modify the downstream features to feed into the pre-trained and frozen graph encoder, adapting the downstream task solely through lightweight prompting.

### 4.2 Multi-domain pre-training

We first introduce the pre-training stage involving multiple distinct source domains. As detailed in Sect. 3, node features across these domains exhibit unique distributions with varying feature dimensionality and semantic meanings, motivating us to align the dimensions and unify the semantics.

**Dimension alignment.** The mismatch in node feature dimensions presents the first obstacle for jointly pre-training multi-domain graphs. A straightforward solution is to use a dimension alignment method to transform the features into consistent dimensions across domains. Specifically, given a graph  $G_i = (V_i, E_i, \mathbf{X}_i)$  from the source domain  $D_{S_i}$ , we transform its feature matrix as follows.

$$\tilde{\mathbf{X}}_i = \text{DA}_{S_i}(\mathbf{X}_i), \quad (1)$$

where  $\text{DA}_{S_i} : \mathbb{R}^{|V| \times d_{S_i}} \rightarrow \mathbb{R}^{|V| \times \tilde{d}}$  is the dimension alignment function specific to domain  $D_{S_i}$ . Here  $d_{S_i}$  is the feature dimension in domain  $D_{S_i}$ , and  $\tilde{d}$  is the aligned dimension that is uniform across all domains. Various choices for the alignment function exist, such as singular value decomposition (SVD) [29], or a learnable multi-layer perceptron (MLP) [15]. In our work, each alignment function  $\text{DA}_{S_i}$  is implemented using SVD. However, dimension alignment only transform the feature spaces into the same dimensions, yet they

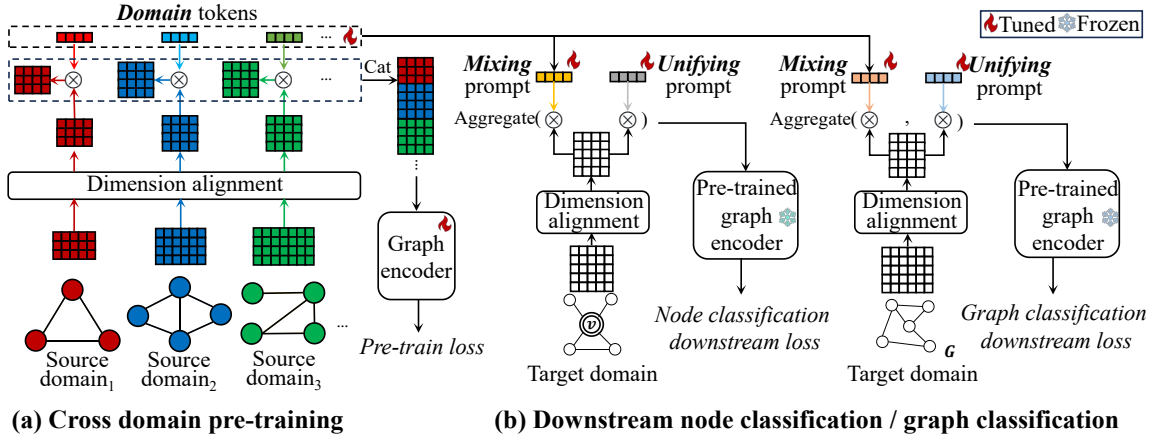


Figure 2: Overall framework of MDGPT.

still occupy distinct semantic spaces across different domains. Direct pre-training on such dimension-aligned feature matrices from multiple domains would result in interference rather than synergy.

**Semantic unification.** To unify the semantics of dimension-aligned features from different domains, we propose the notion of *domain tokens*. We inject a domain token  $\mathbf{t}_{S_i}$  into each source domain  $D_{S_i}$ , leading to a series of domain tokens,  $\mathcal{T} = \{\mathbf{t}_{S_i} : \forall S_i \text{ s.t. } D_{S_i} \in \mathcal{D}_S\}$ . Each domain token adjusts the dimension-aligned features of graphs in the corresponding domain. Specifically, given a graph  $G_i$  from domain  $S_i$ , we further unify the semantics of its dimension-aligned features,  $\hat{\mathbf{X}}_i$ , as follows.

$$\hat{\mathbf{X}}_i = \mathbf{t}_{S_i} \odot \tilde{\mathbf{X}}_i, \quad (2)$$

where  $\hat{\mathbf{X}}_i$  is the unified feature matrix, the domain token  $\mathbf{t}_{S_i}$  is a  $\tilde{d}$ -dimensional learnable vector, and  $\odot$  denotes element-wise multiplication. The unified feature spaces enable a synergistic integration of multi-domain graphs, facilitating the pre-trained model to extract a unified prior along with domain-specific knowledge from a broad range of source domains. Subsequently, given a set of source domain graphs  $\mathcal{G}_S$  and their unified features  $\mathcal{X}_S = \{\hat{\mathbf{X}}_i : \forall i \text{ s.t. } G_i \in \mathcal{G}_S\}$ , we can pre-train a graph encoder using a graph neural network or transformer backbone, as follows.

$$\mathbf{H}_S = \text{GE}(\mathcal{G}_S, \mathcal{X}_S; \Theta), \quad (3)$$

where GE represents graph encoder, and each row of  $\mathbf{H}_S$  corresponds to the output embedding of a node found in the collection of source domain graphs  $\mathcal{G}_S$ . More precisely, let  $\mathbf{h}_{S,v}$  denote the embedding vector of node  $v$ , which is a node in one of the source domain graphs.

**Pre-training loss.** Following previous work [19, 31], we employ link prediction as the pre-training task, leveraging the abundant links in multi-domain graphs as self-supervision. In particular, we adopt the universal task template based on subgraph similarity [19], which enables compatibility among common graph-based tasks, including link prediction, node classification and graph classification. The compatibility of the pre-training task would facilitate adaptation to different downstream tasks.

Specifically, the pre-training data consist of triplets taking the form  $(v, a, B)$ , where  $v, a$  are nodes from the source domain graphs in  $\mathcal{G}_S$ , and  $B = \{b_1, \dots, b_m\}$  represents a set of nodes from  $\mathcal{G}_S$ . Here,  $(v, a)$  is an edge present in  $\mathcal{G}_S$ , while  $B$  serves as the negative examples such that  $(v, b_i)$  is not an edge,  $\forall b_i \in B$ . Such triplets can be readily sampled from  $\mathcal{G}_S$  to generate the pre-training dataset  $\Omega_{\text{pre}}$ . Subsequently, utilizing the universal task template [19], we define the pre-training loss as

$$\mathcal{L}_{\text{pre}}(\Omega_{\text{pre}}; \Theta, \mathcal{T}) = - \sum_{(v,a,B) \in \Omega_{\text{pre}}} \ln \frac{\exp\left(\frac{1}{\tau} \text{sim}(\mathbf{h}_{S,v}, \mathbf{h}_{S,a})\right)}{\sum_{b_i \in B} \exp\left(\frac{1}{\tau} \text{sim}(\mathbf{h}_{S,v}, \mathbf{h}_{S,b_i})\right)}, \quad (4)$$

where  $\tau$  is a temperature hyperparameter,  $\Theta$  denotes the parameters of GraphEncoder,  $\mathcal{T}$  are the learnable domain tokens, and  $\text{sim}$  is a similarity function and is implemented as the cosine similarity following prior work. The pre-training stage yields the pre-trained weights along with the domain tokens, i.e.,  $(\Theta_{\text{pre}}, \mathcal{T}_{\text{pre}}) = \arg \min_{\Theta, \mathcal{T}} \mathcal{L}_{\text{pre}}(\Omega_{\text{pre}}; \Theta, \mathcal{T})$ , which are subsequently utilized in downstream tasks.

### 4.3 Downstream adaptation

In addition to addressing the task gap between pre-training and downstream tasks [19, 31], we focus on transferring pre-trained multi-domain knowledge to downstream tasks, aiming to overcome the domain gap. Likewise, the downstream target domain,  $D_T$ , may have different feature dimensions from the aligned dimensions of the source domains. Therefore, the first step is to apply the same dimension alignment method as used in the pre-training phase. Given a downstream graph  $G = (V, E, \mathbf{X}) \in \mathcal{G}_T$  from target domain  $D_T$ , we transform its feature matrix into  $\tilde{\mathbf{X}} = \text{DA}_T(\mathbf{X})$ .

Afterwards, we propose the notion of *dual prompts*, including unifying prompt and mixing prompt, to further align the target domain with multiple source domains. On one hand, the unifying prompt aims to leverage the unified pre-trained knowledge from all source domains holistically. On the other hand, the mixing prompt aims to leverage a tailored mixture of source domain-specific knowledge for finer-grained adaptation. We elaborate them below.

**Unifying prompt.** Recall that during pre-training, we align node features across multiple source domains, integrating these domains into a unified semantic space for pre-training. To adapt the unified

multi-domain knowledge to the downstream task, we propose a unifying prompt to align the target domain  $D_T$  with the model pre-trained on the source domains. Specifically, we employ a learnable vector  $\mathbf{p}_{\text{uni}} \in \mathbb{R}^{\tilde{d}}$  as the unifying prompt, which can be used to modify the downstream features to achieve adaptation.

**Mixing prompt.** Compared to the unifying prompt for the unified multi-domain knowledge, the mixing prompt prioritizes knowledge specific to certain source domains. For the target domain, a closely related source domain is likely to provide more relevant prior knowledge. Consequently, it is crucial to align the target domain with each source domain to a varying extent, prioritizing the most relevant one. Thus, we define the mixing prompt,  $\mathbf{p}_{\text{mix}} \in \mathbb{R}^{\tilde{d}}$ , as a learnable aggregation of pre-trained domain tokens  $\mathcal{T}_{\text{pre}}$ , where each domain token  $\mathbf{t}_{S_i} \in \mathcal{T}_{\text{pre}}$  can be deemed as a pre-trained “prompt” specific to its corresponding source domain  $D_{S_i}$ . Concretely, let

$$\mathbf{p}_{\text{mix}} = \sum_{i=1}^K \gamma_i \mathbf{t}_{S_i}, \quad (5)$$

where  $\Gamma = \{\gamma_1, \dots, \gamma_K\}$  are the learnable mixing coefficients for the source domains. The mixing prompt then modifies the downstream features to achieve source domain-specific adaptation.

**Prompt tuning.** As discussed, the dual prompts are used to modify the downstream features,  $\tilde{\mathbf{X}}$ , to fully leverage the multi-domain pre-trained knowledge. More concretely, consider a downstream graph  $G \in \mathcal{G}_T$  from the target domain  $D_T$ . We apply the unifying prompt and mixing prompt separately to its dimension-aligned feature matrix,  $\tilde{\mathbf{X}}$ , before feeding them to the pre-trained graph encoder to obtain the final node representations.

$$\mathbf{H} = \text{GE}(G, \mathbf{p}_{\text{uni}} \odot \tilde{\mathbf{X}}; \Theta_{\text{pre}}) + \text{GE}(G, \mathbf{p}_{\text{mix}} \odot \tilde{\mathbf{X}}; \Theta_{\text{pre}}), \quad (6)$$

where  $\Theta_{\text{pre}}$  denotes the pre-trained weights that are kept frozen during downstream adaptation. The final embedding matrix,  $\mathbf{H}$ , results from a combination of both unified and a tailored mixture of pre-trained knowledge and will be used to compute the downstream task loss  $\mathcal{L}_{\text{down}}(\mathbf{H}; \mathbf{p}_{\text{uni}}, \Gamma)$ . To optimize the downstream loss, both the unifying prompt  $\mathbf{p}_{\text{uni}}$  and the mixing coefficients  $\Gamma$  for the mixing prompt are tunable, whereby the pre-trained weights  $\Theta_0$  and domain tokens  $\mathcal{T}_0$  remain frozen. Hence, only a small number of parameters are updated downstream, offering a more parameter-efficient approach than conventional fine-tuning. Hence, our approach MDGPT is especially amenable to few-shot settings, where the downstream task is provided with only a limited number of labeled examples.

In general, for downstream node or graph classification tasks,  $\mathcal{L}_{\text{down}}$  follows the universal task template based on subgraph similarity [19], similar to the pre-training loss  $\mathcal{L}_{\text{pre}}$ . Consider a labeled training set  $\Omega_{\text{down}} = \{(x_1, y_1), (x_2, y_2), \dots\}$ , where  $x_i$  is either a node or a graph, and  $y_i \in Y$  is the class label of  $x_i$  among a set of classes  $Y$ . The loss is then defined as

$$\mathcal{L}_{\text{down}}(\mathbf{H}; \mathbf{p}_{\text{uni}}, \Gamma) = - \sum_{(x_i, y_i) \in \Omega_{\text{down}}} \ln \frac{\exp\left(\frac{1}{r} \text{sim}(\mathbf{h}_{x_i}, \bar{\mathbf{h}}_{y_i})\right)}{\sum_{y \in Y} \exp\left(\frac{1}{r} \text{sim}(\mathbf{h}_{x_i}, \bar{\mathbf{h}}_y)\right)}, \quad (7)$$

where  $\mathbf{h}_{x_i}$  denotes the final embedding of node/graph  $x_i$ , and  $\bar{\mathbf{h}}_y$  is the prototype embedding of class  $y$ , representing the mean of the training instances of class  $y$ . For a node instance  $v$ ,  $\mathbf{h}_v$  is a row of  $\mathbf{H}$ . For a graph instance  $G$ ,  $\mathbf{h}_G$  is a readout of the final embedding matrix  $\mathbf{H}$ .

## 5 Experiments

In this section, we conduct experiments to evaluate MDGPT, and analyze the empirical results.

### 5.1 Experimental Setup

**Datasets.** We conduct experiments on five benchmark datasets. (1) *Cora* [22], (2) *Citeseer* [27] and (3) *Pubmed* [27] are citation graphs from different fields (e.g., computing and biomedical sciences). Each dataset consists of a single graph. Each dataset consists of a single graph, with nodes representing scientific publications and edges denoting citations. In line with previous work [14, 34], we treat the edges as undirected. (4) *Photo* [28] and (5) *Computers* [21] are both e-commerce networks from Amazon in different categories (e.g., photography and computer related products). Each dataset comprises a single graph, where nodes represent products and edges signify frequent co-purchases between products. (6) *Reddit* [6] is a social network, where nodes are post and edges represent interactions such as comments and replies between posts. Note that different citation or e-commerce graphs have distinct features. We present additional details of datasets in Appendix A.

**Setup of multi-domain pre-training.** To evaluate the performance of multi-domain pre-training, we focus on the more challenging scenario of generalizing to unseen domains in downstream tasks. For *Cora*, *Citeseer*, *Pubmed*, *Photos* and *Computers*, we designate each dataset individually as the downstream target domain, while employing the remaining four datasets as the four source domains during multi-domain pre-training. When *Reddit* is used as the target domain, we leverage all five of the previous datasets as source domains. Details of multi-domain transfer to unseen and seen domains are provided in Appendix C and Sect. 5.6, respectively.

**Setup of downstream tasks.** We conduct two types of downstream task, namely, *node classification*, and *graph classification*. We set the tasks in an  $m$ -shot classification, i.e., for each class, we randomly sample  $m$  instances (nodes or graphs) as supervision. Note that each dataset only comprises a single graph, which cannot be directly applied to graph classification. Therefore, following previous work [20, 50], we generate a set of graphs by constructing ego-networks centered on the target nodes (i.e., those with class labels) in each dataset. Subsequently, we conduct graph classification across these ego-networks, each is labeled identically to its corresponding central node. Given that the  $m$ -shot tasks are balanced classification, we evaluate performance using accuracy, consistent with prior studies [18, 37, 48]. Note that we pre-train the graph encoder once for each dataset and then apply the same pre-trained model for all types of downstream tasks.

**Baselines.** We evaluate the performance of MDGPT against state-of-the-art approaches across five primary categories as outlined below. (1) *End-to-end graph neural networks*: GCN [14] and GAT [34] utilize neighborhood aggregation to iteratively gather messages from adjacent nodes, training in an end-to-end manner without pre-training. (2) *Graph pre-training models*: DGI/InfoGraph<sup>1</sup>

<sup>1</sup>Original DGI only operates at the node level, while InfoGraph extends it to the graph level. In our experiments, we apply DGI to node classification, and InfoGraph to graph classification.

**Table 1: Accuracy of one-shot node classification, where each column indicates a target domain.**

Method \ Target domain	Cora	Citeseer	Pubmed	Photo	Computers	Reddit
GCN	28.57 ± 5.07	26.82 ± 5.92	40.03 ± 8.53	46.37 ± 10.58	34.77 ± 11.76	60.12 ± 5.86
GAT	28.40 ± 6.25	23.79 ± 2.96	38.99 ± 4.95	29.42 ± 7.96	31.57 ± 5.87	47.79 ± 6.73
DGI	29.30 ± 5.82	30.03 ± 4.88	41.85 ± 7.78	46.18 ± 7.48	39.37 ± 8.07	60.48 ± 7.38
GRAPHCL	34.94 ± 6.49	30.58 ± 4.58	40.36 ± 7.81	42.26 ± 7.31	<u>45.28</u> ± 6.59	59.80 ± 4.96
Hassani	25.75 ± 6.52	31.67 ± 6.35	42.18 ± 8.46	47.00 ± 8.52	38.14 ± 7.91	52.40 ± 4.88
GPPT	15.37 ± 4.51	23.24 ± 2.94	36.56 ± 5.31	16.19 ± 4.73	19.22 ± 8.71	52.10 ± 7.42
GRAPHPROMPT	35.90 ± 7.10	32.76 ± 7.66	43.34 ± 10.66	<u>49.88</u> ± 8.31	43.03 ± 10.35	<u>65.49</u> ± 6.52
GPF	<u>37.84</u> ± 11.07	<u>37.61</u> ± 8.87	<u>46.36</u> ± 7.48	49.42 ± 7.04	37.00 ± 6.52	64.02 ± 7.98
GCOPE	33.38 ± 6.86	35.56 ± 6.81	42.10 ± 8.07	48.52 ± 7.78	40.22 ± 7.82	61.35 ± 5.30
MDGPT	<b>42.26</b> ± 10.18	<b>42.40</b> ± 9.26	<b>49.82</b> ± 8.38	<b>64.82</b> ± 10.53	<b>49.77</b> ± 11.00	<b>67.80</b> ± 6.84

Results are reported in percent. The best method is bolded and the runner-up is underlined. Table 2 follows the same style.

**Table 2: Accuracy of one-shot graph classification, where each column indicates a target domain.**

Method \ Target domain	Cora	Citeseer	Pubmed	Photo	Computers	Reddit
GCN	39.74 ± 13.70	25.58 ± 9.13	50.50 ± 10.85	56.97 ± 10.53	39.48 ± 12.19	58.63 ± 5.96
GAT	26.73 ± 9.06	25.33 ± 6.01	38.06 ± 7.14	47.75 ± 11.58	36.59 ± 12.55	66.10 ± 7.71
INFOGRAPH	40.23 ± 10.00	30.28 ± 6.70	49.27 ± 9.75	59.44 ± 10.25	40.58 ± 10.30	64.19 ± 6.88
GRAPHCL	41.46 ± 10.09	32.93 ± 8.04	49.49 ± 9.14	57.58 ± 10.45	42.68 ± 10.55	63.27 ± 5.80
Hassani	40.77 ± 8.04	31.85 ± 6.22	49.79 ± 9.96	60.29 ± 10.77	37.65 ± 10.18	65.64 ± 6.65
GRAPHPROMPT	41.59 ± 9.85	<u>38.92</u> ± 8.97	<u>51.42</u> ± 11.15	<u>60.86</u> ± 9.28	<u>45.93</u> ± 11.70	64.61 ± 6.39
GPF	41.46 ± 10.09	38.87 ± 9.22	48.07 ± 10.52	60.76 ± 9.56	45.65 ± 10.31	65.91 ± 6.28
GCOPE	<u>42.61</u> ± 8.57	35.27 ± 8.48	49.42 ± 9.52	58.50 ± 9.70	45.00 ± 10.30	<u>67.49</u> ± 7.20
MDGPT	<b>48.36</b> ± 11.34	<b>44.28</b> ± 10.16	<b>54.34</b> ± 9.76	<b>64.08</b> ± 9.85	<b>48.29</b> ± 11.35	<b>68.92</b> ± 7.32

[30, 35] and GraphCL [46]. follow the “pre-train, fine-tune” paradigm. Specifically, they first pre-train the GNN to exploit the intrinsic attributes of the graphs and then fine-tune the pre-trained model on downstream tasks according to task-specific labels. (3) *Graph cross-domain model*: Hassani [8] pre-trains a attention-based GNN from both contextual and topological views for cross-domain adaptation. (4) *Graph prompting models*: GPPT<sup>2</sup> [31], GPF[5] and GraphPrompt [19] fall under this category. They employ a self-supervised pre-training tasks and unify downstream tasks into a general template as the pretext task. Subsequently, a single type of prompt is tuned for downstream adaptation. (5) *Multi-domain pre-training models*: GCOPE [53] is pre-trained on multi-domain datasets through self-supervised tasks, followed by fine-tuning or prompting for downstream adaptation.

We provide further descriptions of these baselines in Appendix B, and their implementation and configuration details, along with those of our approach MDGPT, in Appendix C.

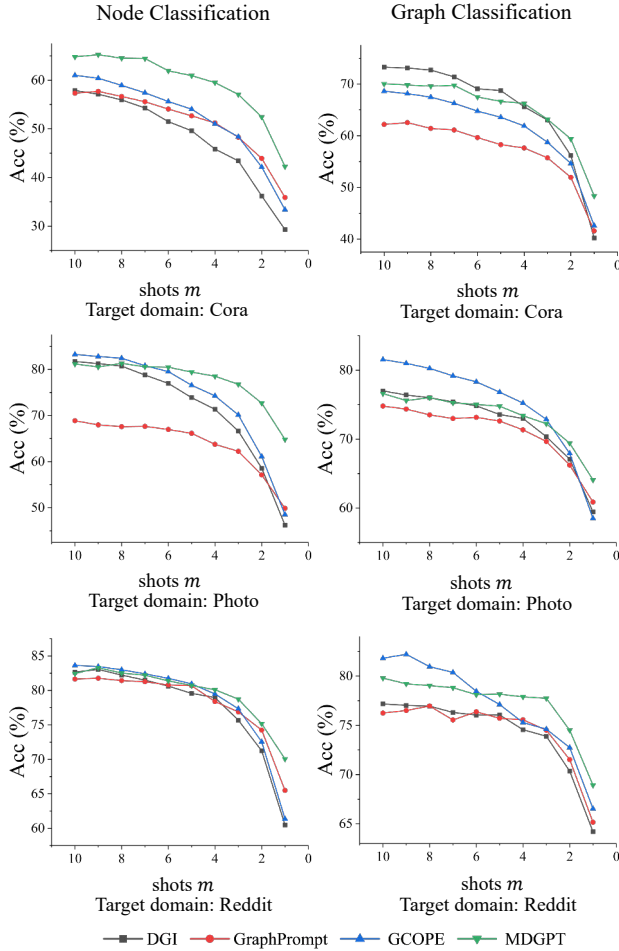
<sup>2</sup>GPPT is specifically designed for downstream node classification task and is not suitable for graph classification. Therefore, in our experiments, we exclusively use GPPT for node classification.

## 5.2 Few-shot performance evaluation

We first evaluate one-shot classification tasks. Subsequently, we vary the number of shots to investigate their impact on performance.

**One-shot performance.** We present the results of one-shot node and graph classification tasks on unseen domains in Tables 1 and 2, respectively. (Additional results on seen domains are presented in Sect. 5.6.) We make the follow observations. (1) MDGPT surpasses all baseline methods across all settings, outperforming the best competitor by 3.5–37.9% on node classification and 4.3–13.9% on graph classification. The results demonstrate its effectiveness in multi-domain pre-training and in downstream adaptation to unseen domains. (2) Another text-free multi-domain pre-training method, GCOPE, significantly lags behind MDGPT. Its suboptimal performance can be attributed to two factors. First, it only performs implicit domain alignment via virtual nodes, without explicitly unifying feature spaces across domains. Second, it overlooks source domain-specific knowledge in downstream adaptation. The results also underscore the importance of our domain tokens in aligning diverse domains and enabling a tailored mixture of source domains. (3) GPPT is only comparable to, if not worse than, other baselines since it is not specifically designed for few-shot learning.





**Figure 3: Impact of number of shots on node classification on three target domains.**

**Few-shot performance.** To examine the performance of MDGPT with more labeled data, we vary the number of shots in both node and graph classification tasks, and report the results in Fig. 3. Our findings are as follows. (1) MDGPT significantly surpasses all baselines in low-shot settings with very limited labeled data (e.g.,  $m \leq 5$ ), highlighting the ideal use case for our approach. (2) As the number of shots increases, the performance of all methods improves as anticipated. However, MDGPT continues to perform competitively, if not better.

### 5.3 Ablation Study

To comprehensively analyze the performance of MDGPT, we conduct two ablation studies as follows.

**Data ablation.** We evaluate the impact of utilizing more source domains, by iteratively adding *Citeseer*, *Pubmed*, *Photo*, and *Computers* to pre-training, in this order, while the target domain is fixed to *Cora*. As the number of source domains grows from one to four, we perform node classification on *Cora* using several strong baselines

**Table 3: Analysis on homophilic and heterophilic data.**

Target domain	Source domain	Accuracy (%)	
		GCOPE	MDGPT
Pubmed	Cora	40.92	45.59
	Cora + Citeseer	40.58	46.72
	Cora + Citeseer + Wisconsin	43.14	48.51
Wisconsin	Texas	27.37	27.75
	Texas + Cornell	27.55	27.73
	Texas + Cornell + Cora	27.14	29.78
Chameleon	Wisconsin	23.14	24.81
	Wisconsin + Squirrel	24.13	24.95
	Wisconsin + Squirrel + Cora	22.61	25.36

and MDGPT, as shown in Fig. 4. We observe that for all baselines, adding more datasets tends to cause domain conflicts. Specifically, DGI performs worse when *Photo* and *Computers* are added, GraphPrompt performs worse when *Computers* is added, and GCOPE performs worse when *PubMed* is added. In contrast, MDGPT consistently performs better when more source domains are introduced, demonstrating the effectiveness of our multi-domain pre-training.

**Model ablation.** We compare MDGPT to its ablated variants obtained by excluding the use of domain tokens, mixing prompts, or unifying prompts. These variants and their corresponding results on downstream tasks are presented in Table 4. The findings confirm the importance of each design element. First, the use of domain tokens and mixing prompt is crucial. Notably, Variant 3, which includes domain tokens, significantly outperforms Variants 1 and 2, demonstrating the effectiveness of domain tokens in aligning multiple source domains. Furthermore, Variant 4 surpasses Variant 3, highlighting the additional benefit of using mixing prompts for downstream adaptation. Second, omitting the unifying prompt results in reduced performance, as evidenced by the higher accuracy of Variant 2 compared to Variant 1. Lastly, the dual prompts with both mixing and unifying prompts are beneficial, enabling MDGPT to achieve superior performance.

### 5.4 Hyperparameter Analysis

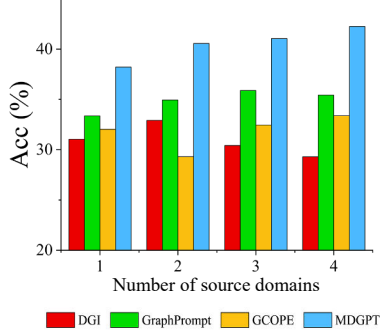
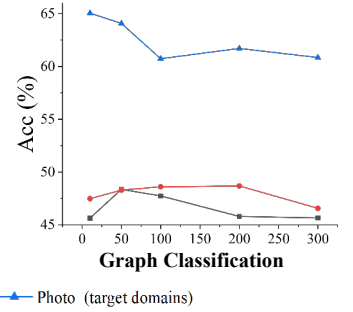
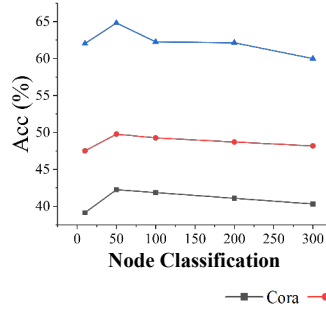
We evaluate the impact of the feature dimension,  $\tilde{d}$ , after performing domain alignment in Eq. (1). We vary  $\tilde{d}$  and report the corresponding performance in Fig. 5. We observe that, for both node and graph classification, as  $\tilde{d}$  grows from a low value, the performance improves initially since higher dimensions increases model capacity. However, after reaching a peak (around  $\tilde{d} = 50$ ), accuracy starts to gradually decline as  $\tilde{d}$  grows further, due to the amplified semantic gap in these feature dimensions across various domains. Based on the above observation,  $\tilde{d} = 50$  is a robust setting in most datasets and tasks.

### 5.5 Homophily Sensitivity

To further evaluate the robustness of MDGPT across homophilic and heterophilic graphs, we conduct cross-domain experiments on homophilic datasets (*Cora*, *Citeseer*, *Pubmed*) and heterophilic datasets (*Texas* [23], *Wisconsin* [23], *Cornell* [23], *Squirrel* [26],

**Table 4: Model ablation study on the effects of key components.**

Methods	Domain token	Mixing prompt	Unified prompt	Target domain for node classification					Target domain for graph classification				
				Cora	Citeseer	Pubmed	Photo	Computers	Cora	Citeseer	Pubmed	Photo	Computers
VARIANT 1	×	×	×	32.11	32.45	37.47	49.69	38.98	43.12	33.36	52.89	56.04	40.79
VARIANT 2	×	×	✓	35.90	38.16	43.34	49.88	43.03	41.59	38.92	51.42	60.86	45.93
VARIANT 3	✓	×	×	36.50	40.05	47.02	50.64	43.37	43.39	39.18	53.07	57.57	43.38
VARIANT 4	✓	✓	×	37.24	41.47	48.22	56.59	41.30	43.94	42.45	49.95	56.83	44.11
MDGPT	✓	✓	✓	<b>42.26</b>	<b>42.40</b>	<b>49.82</b>	<b>64.82</b>	<b>49.77</b>	<b>48.36</b>	<b>44.28</b>	<b>54.34</b>	<b>64.08</b>	<b>48.29</b>

**Figure 4: Data ablation study with a growing number of source domains.****Figure 5: Sensitivity study of  $\tilde{d}$ , the aligned feature dimension across domains, on three target domains.****Table 5: Accuracy of one-shot node classification on seen target domains.**

Method \ Target domain	Cora	Citeseer
DGI	33.57 ± 6.83	32.90 ± 6.10
GRAPHCL	38.73 ± 7.43	32.46 ± 4.75
GRAPHPROMPT	45.98 ± 9.43	36.45 ± 8.38
GCOPE	34.95 ± 7.45	35.18 ± 7.76
MDGPT	<b>47.44 ± 10.03</b>	<b>44.96 ± 10.12</b>

**Table 6: Comparison of the number of tunable parameters during the downstream adaptation stage.**

Method \ Target domain	Cora	Citeseer	Computers	Reddit
GCN	14,592	14,336	15,360	23,296
GRAPHCL	1,792	1,536	2,560	10,496
GRAPHPROMPT	256	256	256	256
GCOPE	1,792	1,536	2,560	10,496
MDGPT	56	56	56	57

*Chameleon* [26]). The results are shown in Table 3. We observe that regardless of whether the source or target domain graphs are homophilic or heterophilic, MDGPT effectively leverages multi-domain knowledge and bridges the gap between various domains and consistently outperforms GCOPE. This further demonstrates the effectiveness of MDGPT.

## 5.6 Node Classification on Seen Domains

We further conduct one-shot node classification tasks on seen target domains. Specifically, we pre-train on the source domains including *Cora*, *Citeseer*, *Pubmed*, *Photo* and *Computers*, and evaluate node classification on a seen target domain, *Cora* or *Citeseer*, separately. We illustrate the results of several competitive baselines and MDGPT in Table 5, and observe the same trend as those observed in in Sect. 5.2, showing the robustness of MDGPT in dealing with both seen and unseen domains.

## 5.7 Parameters Efficiency

We evaluate the parameter efficiency of MDGPT compared to representative baselines. We present the number of tunable parameters during downstream adaptation in Table 6. For GCN, as it follows an end-to-end supervised paradigm, all parameters in the model have to be updated, resulting in the most parameters. For GraphCL and GCOPE, we update only the downstream classifier while freezing the pre-trained model, significantly reducing the number of tunable parameters. For GraphPrompt, since it only tunes a light-weight prompt containing fewer parameters than a classifier, the number of parameters is further reduced. Finally, MDGPT is the most parameter-efficient adaptation, as only the dual prompts are updated. Specifically, the unifying prompt has the same dimension as the unified features, which is generally smaller than the original feature dimension, and the mixing prompt’s dimension is same as the number of source domains, which is a quite small compared to the feature dimension. Despite having the fewest parameters, MDGPT significantly outperforms all baselines, further demonstrating the effectiveness of our design.



## 6 Conclusions and Future Work

In this paper, we explored multi-domain pre-training on graphs, with the objective of learning a comprehensive range of knowledge from diverse domains, and adapting to unseen domains in downstream applications. Our proposed approach MDGPT leverages a series of domain tokens to unify the semantic spaces of multiple source domains, and extract domain-specific knowledge. Moreover, we introduced dual prompts to align the target domain with pre-trained knowledge, including both unified multi-domain knowledge and a tailored mixture of source domain-specific knowledge. Finally, we conducted extensive experiments on six public datasets, demonstrating that MDGPT significantly outperforms various state-of-the-art baselines.

In this work, we focused on text-free graphs, which are more challenging than text-attributed graphs due to the lack of textual descriptions. However, text-attributed graphs are also important, providing auxiliary knowledge and a gateway to connect with large language models. Therefore, a promising future direction is to integrate both text-free and text-attributed graphs across multiple domains for pre-training, in order to learn a more comprehensive range of knowledge.

## References

- [1] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. 2021. Pre-trained image processing transformer. In *CVPR*.
- [2] Kaize Ding, Kai Shu, Xuan Shan, Jundong Li, and Huan Liu. 2021. Cross-domain graph anomaly detection. *IEEE TNNLS* (2021).
- [3] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *NeurIPS* (2019).
- [4] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *WWW*.
- [5] Taoran Fang, Yunchao Zhang, Yang Yang, Chunping Wang, and Lei Chen. 2024. Universal Prompt Tuning for Graph Neural Networks. In *NeurIPS*.
- [6] Will Hamilton, Zhitaoying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.
- [7] Bao Hangbo, Dong Li, Piao Songhao, and Wei Furu. 2022. BEiT: BERT Pre-Training of Image Transformers. In *ICLR*.
- [8] Kaveh Hassani. 2022. Cross-domain few-shot graph classification. In *AAAI*.
- [9] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*.
- [10] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. Cpt-gnn: Generative pre-training of graph neural networks. *SIGKDD* (2020).
- [11] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *WWW*.
- [12] Anshul Kanakia, Zhihong Shen, Darrin Eide, and Kuansan Wang. 2019. A scalable hybrid research paper recommender system for microsoft academic. In *WWW*.
- [13] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL-HLT* (2019).
- [14] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
- [15] Rudolf Kruse, Sanaz Mostaghim, Christian Borgelt, Christian Braune, and Matthias Steinbrecher. 2022. Multi-layer perceptrons. *Computational intelligence: a methodological introduction* (2022).
- [16] Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhao Zhang. 2024. One for All: Towards Training One Graph Model for All Classification Tasks. In *ICLR*.
- [17] Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S Yu, et al. 2023. Towards graph foundation models: A survey and beyond. *arXiv preprint arXiv:2310.11829* (2023).
- [18] Zemin Liu, Yuan Fang, Chenghao Liu, and Steven CH Hoi. 2021. Relative and absolute location embedding for few-shot node classification on graph. *AAAI* (2021).
- [19] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. 2023. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *WWW*.
- [20] Yuanfu Lu, Xunqiang Jiang, Yuan Fang, and Chuan Shi. 2021. Learning to pre-train graph neural networks. In *AAAI*.
- [21] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*.
- [22] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* (2000).
- [23] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287* (2020).
- [24] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*.
- [25] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *SIGKDD*.
- [26] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-scale attributed node embedding. *Journal of Complex Networks* (2021), cnab014.
- [27] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* (2008).
- [28] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).
- [29] Gilbert W Stewart. 1993. On the early history of the singular value decomposition. *SIAM review* (1993).
- [30] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. 2019. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *ICLR*.
- [31] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. 2022. Gpnt: Graph pre-training and prompt tuning to generalize graph neural networks. In *SIGKDD*.
- [32] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. 2023. All in One: Multi-Task Prompting for Graph Neural Networks. *SIGKDD* (2023).
- [33] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Long Xia, Dawei Yin, and Chao Huang. 2024. HiGPT: Heterogeneous Graph Language Model. *arXiv preprint arXiv:2402.16024* (2024).
- [34] Petar Velicković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.
- [35] Petar Velicković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep Graph Infomax. In *ICLR*.
- [36] Chen Wang, Yueqing Liang, Zhiwei Liu, Tao Zhang, and S Yu Philip. 2021. Pre-training graph neural network for cross domain recommendation. In *CogML*.
- [37] Ning Wang, Minnan Luo, Kaize Ding, Lingling Zhang, Jundong Li, and Qinghua Zheng. 2020. Graph Few-shot Learning with Attribute Matching. In *CIKM*.
- [38] Qizhou Wang, Guansong Pang, Mahsa Salehi, Wray Buntine, and Christopher Leckie. 2023. Cross-domain graph anomaly detection via anomaly-aware contrastive alignment. In *AAAI*.
- [39] Zeyuan Wang, Qiang Zhang, HU Shuang-Wei, Haoran Yu, Xurui Jin, Zhichen Gong, and Huajun Chen. 2022. Multi-level Protein Structure Pre-training via Prompt Learning. In *ICLR*.
- [40] Zhihao Wen and Yuan Fang. 2023. Augmenting Low-Resource Text Classification with Graph-Grounded Pre-training and Prompting. In *SIGIR*.
- [41] Zhihao Wen and Yuan Fang. 2023. Prompt tuning on graph-augmented low-resource text classification. *arXiv preprint arXiv:2307.10230* (2023).
- [42] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph neural networks in recommender systems: a survey. *Comput. Surveys* (2022).
- [43] Yuxia Wu, Yuan Fang, and Lizi Liao. 2024. On the Feasibility of Simple Transformer for Dynamic Graph Modeling. In *WWW*.
- [44] Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *WWW*.
- [45] Baoyao Yang and Pong C Yuen. 2019. Cross-domain visual representations via unsupervised graph alignment. In *AAAI*.
- [46] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *NeurIPS* (2020).
- [47] Xingtong Yu, Yuan Fang, Zemin Liu, Yuxia Wu, Zhihao Wen, Jianyuan Bo, Xinming Zhang, and Steven CH Hoi. 2024. Few-Shot Learning on Graphs: from Meta-learning to Pre-training and Prompting. *arXiv preprint arXiv:2402.01440* (2024).
- [48] Xingtong Yu, Zhenghao Liu, Yuan Fang, Zemin Liu, Sihong Chen, and Xinming Zhang. 2023. Generalized Graph Prompt: Toward a Unification of Pre-Training and Downstream Tasks on Graphs. *arXiv preprint arXiv:2311.15317* (2023).
- [49] Xingtong Yu, Zemin Liu, Yuan Fang, and Xinming Zhang. 2023. Learning to count isomorphisms with graph neural networks. *AAAI* (2023).
- [50] Xingtong Yu, Zemin Liu, Yuan Fang, and Xinming Zhang. 2024. HGPROMPT: Bridging Homogeneous and Heterogeneous Graphs for Few-shot Prompt Learning. In *AAAI*.
- [51] Xingtong Yu, Chang Zhou, Yuan Fang, and Xinming Zhang. 2024. MultiGPrompt for Multi-Task Pre-Training and Prompting on Graphs. In *WWW*.
- [52] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. *NeurIPS* (2019).
- [53] Haihong Zhao, Aochuan Chen, Xiangguo Sun, Hong Cheng, and Jia Li. 2024. All in One and One for All: A Simple yet Effective Method towards Cross-domain Graph Pretraining. In *SIGKDD*.

## Appendices

### A Further Descriptions of Datasets

In this section, we provide a comprehensive description of datasets.

(1) *Cora*<sup>3</sup> [22] includes 2,708 computing publications, each assigned to one of seven categories. The citation network has 5,429 links. Each publication in the dataset is depicted by a binary word vector, which shows the presence or absence of words from a dictionary containing 1,433 distinct words.

(2) *Citeseer*<sup>4</sup> [27] comprises 3,312 computing publications, each classified into one of six categories, which are different from those in *Cora*. The citation network includes 4,732 links. Each publication is described by a binary word vector that indicates the presence or absence of words from a dictionary containing 3,703 unique words.

(3) *PubMed*<sup>5</sup> [27] is composed of 19,717 biomedical publications from diabetes, each classified into one of three categories. The citation network comprises 44,338 links. Each publication in the dataset is depicted by a TF/IDF weighted word vector from a dictionary, indicating the presence of 500 unique words from the dictionary.

(4) *Amazon Photo*<sup>6</sup> [28] comprises 7,487 photography related products, categorized into one of eight categories. The co-purchasing network consists of 119,043 edges, which represent frequently bought-together product relationships. Each product is described by a feature vector derived from its metadata and reviews, and is labeled according to its category.

(5) *Amazon Computers*<sup>7</sup> [28] comprises 13,752 computer-related products, categorized into ten groups. The co-purchasing network contains 245,861 edges, illustrating frequently bought-together product relationships. Each product is represented by a feature vector generated from its metadata and reviews, and is labeled according to its specific category.

(6) *Reddit*<sup>8</sup> [6] consists of 232,965 posts from the social media platform Reddit, categorized into 41 different subreddits. The interaction network includes 11,606,919 edges, representing interactions such as comments and replies between posts. Each post in the dataset is described by a feature vector derived from its textual content and metadata, consisting of 602 unique words.

### B Further Descriptions of Baselines

In this section, we summarize the datasets in Table 7 and present more details for the baselines used in our experiments.

#### (1) End-to-end Graph Neural Networks

- **GCN** [14]: GCN utilizes a mean-pooling strategy for neighborhood aggregation to integrate information from neighboring nodes.
- **GAT** [34]: GAT also leverages neighborhood aggregation for end-to-end node representation learning, uniquely assigns varying attention weights to different neighbors, thereby adjusting their impact on the aggregation process.

#### (2) Graph Pre-training Models

<sup>3</sup><https://relational.fit.cvut.cz/dataset/CORA>

<sup>4</sup><https://nrvs.com/download/data/labelled/citeseer.zip>

<sup>5</sup><https://github.com/kimiyoung/planetoid/raw/master/data>

<sup>6</sup>[https://github.com/shchur/gnn-benchmark/blob/master/data/npz/amazon\\_electronics\\_photo.npz](https://github.com/shchur/gnn-benchmark/blob/master/data/npz/amazon_electronics_photo.npz)

<sup>7</sup>[https://github.com/shchur/gnn-benchmark/blob/master/data/npz/amazon\\_electronics\\_computers.npz](https://github.com/shchur/gnn-benchmark/blob/master/data/npz/amazon_electronics_computers.npz)

<sup>8</sup><https://data.dgl.ai/dataset/reddit.zip>

Table 7: Summary of datasets.

	Nodes	Edges	Feature dimension	Node Classes
Cora	2,708	10,556	1,433	7
Citeseer	3,327	9,104	3,703	6
Pubmed	19,717	88,648	500	3
Photo	7,650	238,162	745	8
Computers	13,752	491,722	767	10
Reddit	232,965	114,615,892	602	41

- **DGI** [34]: DGI operates as a self-supervised pre-training methodology tailored for homogeneous graphs. It is predicated on the maximization of mutual information (MI), aiming to enhance the estimated MI between locally augmented instances and their global counterparts.
- **InfoGraph** [30]: Expanding upon DGI, InfoGraph is centered on graph-level tasks, endeavoring to maximize the alignment between node and graph embeddings.
- **GraphCL** [46]: GraphCL leverages a variety of graph augmentations for self-supervised learning, tapping into the intrinsic structural patterns of graphs. The overarching goal is to amplify the concordance between different augmentations throughout graph pre-training.

#### (3) Graph Cross-domain Model

- **Hassani** [8]: Hassani proposes an attention-based graph encoder that utilizes contextual and topological views of graph to learn task-specific knowledge for rapid adaptation and task-agnostic knowledge for knowledge transfer.

#### (4) Graph Prompt Models

- **GPPT** [31]: GPPT utilize a GNN model, pre-trained through a link prediction task. The downstream prompt module is specifically tailored for node classification task, which is unified as the pre-training link prediction task.
- **GPF** [5]: GPF operates as a universal prompt-based tuning methodology tailored for pre-trained GNN models. It is predicated on the adaptation of the input graph’s feature space, aiming to achieve an effect equivalent to any form of prompting function.
- **GraphPrompt** [19]: GraphPrompt employs subgraph similarity calculation as a unified template to narrow the gap between pre-training and downstream tasks, inclusive of node and graph classification. A learnable prompt is subsequently tuned during the downstream adaptation to incorporate task-specific knowledge.

#### (5) Multi-domain graph pre-training

- **GCOPE** [53]: GCOPE is a multi-domain pre-training approach that cooperates graph datasets across multiple domains through a series of domain-specific interconnect virtual nodes that link nodes within the same domain. Its primary goal is to improve performance in downstream applications by leveraging the multiple domain knowledge in multiple source domains.

**Table 8: Settings of multi-domain transfer to unseen domains.**

Target domain	Cora	Citesser	Pubmed	Photo	Computers
Cora	×	✓	✓	✓	✓
Citeseer	✓	×	✓	✓	✓
Pubmed	✓	✓	×	✓	✓
Photo	✓	✓	✓	×	✓
Computers	✓	✓	✓	✓	×
Reddit	✓	✓	✓	✓	✓

## C Implementation Details

**General settings Optimizer.** For all experiments, we use the Adam optimizer.

**Environment.** The environment in which we run experiments is:

- Linux version: 5.15.0-78-generic
- Operating system: Ubuntu 18.04.5 LTS
- CPU information: Intel(R) Xeon(R) Platinum 8352V
- GPU information: GeForce RTX 4090 (24 GB)

**Further experiments setup.** We conduct multi-domain pre-training with different source domains for various target domains, the correspondence is listed in Table 8.

To show the effectiveness of multi-domain pre-training, for *Cora*, *Citeseer*, *Pubmed*, *Photos* and *Computers*, we cooperate four datasets for multi-domain pre-training, while the rest for downstream adaptation. For further evaluate the performance of cross-domain adaptation, we leverage *Cora*, *Citeseer*, *Pubmed*, *Photos* and *Computers* for multi-domain pre-training, while *Reddit* for cross domain adaptation. Note that all pre-training methods are pre-trained on the above multi-domain datasets.

For downstream node and graph classification tasks, we conduct  $m = 1$  shot tasks. Additionally, we also vary the number of shots for  $1 \leq m \leq 10$ , to evaluate the robustness of MDGPT. We repeat the sampling 100 times to construct 100  $m$ -shot tasks for both node and graph classification. For each task, we run with five different random seeds, leading to a total of 500 outcomes for node classification and graph classification tasks, and we report the average and standard deviation across these 500 results.

**Details of baselines.** We utilize the officially provided code for all open-source baselines and reproduce the non-open-source GCOPE. Each model is tuned based on the settings recommended in their respective literature to achieve optimal performance.

For the baseline GCN [14], we employ a 3-layer architecture, and set the hidden dimensions to 256. For GAT [34], we employ a 2-layer architecture and set the hidden dimension to 64. Additionally, we apply 8 attention heads in the first GAT layer.

For DGI [34], we utilize a 1-layer GCN as the base model and set the hidden dimensions to 256. Additionally, we employ prelu as the activation function. For InfoGraph [30], a 3-layer GCN is used as the base model, with its hidden dimensions set to 256. For GraphCL [46], a 1-layer GCN is also employed as its base model, with the hidden dimensions set to 256. Specifically, we select edge dropping as the augmentations, with a default augmentation ratio of 0.2.

For Hassani [8], We employ a 3-layer GCN is used as the base model for all datasets, with the hidden dimensions set to 256.

For GPPT [31], we utilize a 2-layer GraphSAGE as its base model, setting the hidden dimensions to 256. For base GraphSAGE, we also employ a mean aggregator. For GraphPrompt [19], a 3-layer GCN is used as the base model for all datasets, with the hidden dimensions set to 256. For GPF [5], employs a 5-layer GCN as the base model for all datasets, following the recommended settings. The hidden dimensions are set to 256.

For GCOPE [53], we employ a 2-layer GCN as the base model and set the hidden dimensions to 100. Downstream adaptation is achieved through fine-tuning, as it is reported to yield the best performance in their literature.

For all baselines except for GCOPE, we set the unified feature dimensions to 50, the same as our MDGPT. For GCOPE, we adhere to the recommended settings and set the unified feature dimensions to 100.

**Details of MDGPT.** For our proposed MDGPT, we utilize a 3-layer GCN as the base model for all datasets, with the hidden dimensions set to 256. We set the unified feature dimensions to 50.