HePa: Heterogeneous Graph Prompting For All-level Classification Tasks

Jia Jinghong[1], Lei Song[1], Jiaxing Li[1], Youyong Kong[1, 2*],

[1]Jiangsu Provincial Joint International Research Laboratory of Medical Information Processing,
School of Computer Science and Engineering, Southeast University
[2]Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications
(Southeast University), Ministry of Education, China
{judging0925, 230238577, jiaxing_li, kongyouyong}@seu.edu.cn

Abstract

Heterogeneous graphs, which are common in real-world downstream tasks, have recently sparked a wave of research interest. The performance of end-to-end heterogeneous graph neural networks (HGNNs) greatly relies on supervised training for specific tasks. To reduce the labeling cost, the "pretrain-finetune" paradigm has been widely adopted, but it leads to a knowledge gap between the pre-trained model and downstream tasks. In an effort to address this gap, the "pretrain-prompt" paradigm has emerged as a promising approach. This involves fine-tuning randomly initialized learnable vectors in downstream tasks. However, this approach may result in an insufficient representation of downstream task features. Existing techniques for heterogeneous graph prompting restructure the heterogeneous graph to align with the homogeneous graph prompting scheme. This can potentially introduce the same limitations as homogeneous graph prompt learning. In this paper, we propose HePa, short for **He**terogeneous Graph **P**rompting for **a**ll-level classification tasks. It not only includes a unified prompt *template-graph* adapted for heterogeneous graphs but also introduces a novel *pre-prompt* token optimized during the pre-training phase to convey task information downstream. With these designs, HePa can complete all levels of classification tasks toward few-shot scenarios while activating in-context learning. Finally, we conducted a comprehensive experimental analysis of HePa on three benchmark datasets.

1 Introduction

Graph data can describe relationships between different objects, such as citation networks, social networks, and recommendation networks. In real-world scenarios, we encounter graphs with multiple types of nodes and edges, known as heterogeneous graphs (Sun and Han 2013). These graphs differ from traditional homogeneous graphs by featuring multiple node types and relationships.

Graph neural networks (GNNs) (Kipf and Welling 2016; Veličković et al. 2018; Wang et al. 2020, 2023), and heterogeneous graph neural networks (HGNNs) (Wang et al. 2019; Lv et al. 2021) are widely used for tasks like node classification, link prediction, and graph classification. However, training these networks using supervised methods heavily relies on having high-quality labels for specific tasks. This presents challenges in integrating and transferring knowledge across different tasks.

The *"pretrain-finetune"* paradigm (Dong et al. 2019) was introduced as an alternative to end-to-end supervised learning. In this approach, the model is first pre-trained on unlabeled graphs (Hu et al. 2020; Jiang et al. 2021) to learn general properties and then fine-tuned for specific tasks using labeled data. However, the disparity between pre-training and fine-tuning objectives can lead to a knowledge gap and issues with predictive ability (Liu et al. 2023b). In addition, this paradigm increases the number of parameters to be optimized, resulting in additional overhead.

To bridge the gap between pretraining models and downstream tasks in graph learning, researchers have adopted the *"pretrain, prompt"* paradigm from NLP (Brown et al. 2020; Jia et al. 2021). This approach involves setting a pre-training objective that serves both upstream and downstream tasks (Sun et al. 2022; Liu et al. 2023c; Tan et al. 2023; Yu et al. 2023). After pretraining, model parameters are frozen while the input data are tweaked for different downstream tasks to match the pretraining objective. Additionally, a few tunable prompt vectors are introduced and fine-tuned to customize the model for specific tasks, avoiding the need for broad model adjustments. This technique, known as "**Prompt as token**" (Zi et al. 2024), addresses some challenges but struggles with comprehensive coverage across all levels of downstream classification tasks. Also, the learnable prompts during the downstream phase might not perfectly represent the tasks after fine-tuning due to their random initialization.

Building on previous research, new methods were developed for organizing simple prompt vectors into complex template-graphs (Sun et al. 2023; Huang et al. 2023; Liu et al. 2023a) dubbed "**Prompt as graph**" (Zi et al. 2024). This involves graph prompts containing multiple tokens with internal structures and patterns, enabling better representation of tasks and graph semantic information for classification tasks. The idea introduced by OFA (Liu et al. 2023a) involves using large language models (LLMs) to embed task and category information directly into these template-graphs where LLMs convert task and category prompts into feature vectors, incorporating them as nodes in a graph for in-context learning. However, this technique is primarily designed to handle text-attributed graphs (TAGs) because of

*Corresponding Author.

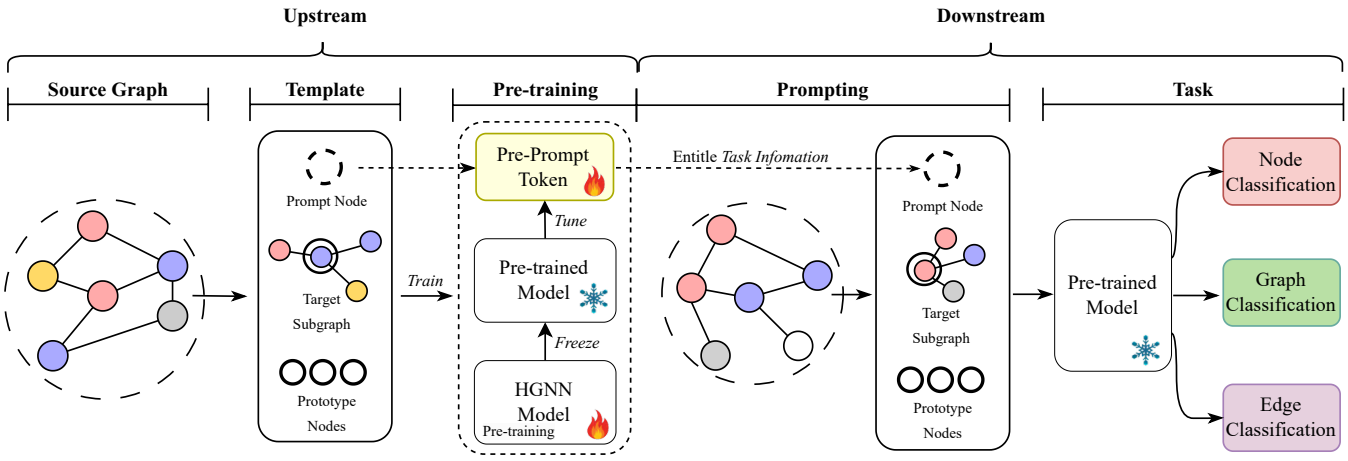

Figure 1: Overall framework of HePa. (1) To unify all-level task data under a single pre-training objective, we construct data into unified prompt template-graphs containing category and empty task information. (2) With the pre-trained model parameters frozen, we optimize the Pre-prompt token separately for every type of task to make it rich in task information. (3) we inject the Pre-prompt task information into the downstream templates for all-level classification tasks.

the language processing ability of LLMs.

In heterogeneous graph learning, researchers are currently trying to apply early graph prompt learning schemes to heterogeneous graphs (Yu et al. 2024; Ma et al. 2024). These methods typically restructure the heterogeneous graph to fit the training objectives of homogeneous graph prompt learning and introduce additional prompt vectors in the downstream phase to learn heterogeneous graphs' multi-relational and topological structure. However, since these methods still use the early "prompt as tokens" method of unified upstream and downstream training objectives for homogeneous graphs, They do not achieve all-level classification in the downstream phase, while still encountering the prompt's random initialization problem mentioned above.

In this paper, we propose a novel heterogeneous graph prompt learning framework **HePa**, which is short for **He**terogeneous graph **P**rompting for **a**ll-level classification tasks. Our framework constructs and trains HGNN models with in-context learning capabilities, targeting multi-task data joint training scenarios, and aiming to use a single training model to solve all levels of downstream tasks. HePa aims to address two major challenges: Heterogeneous graph prompt learning remains confined to the "prompt as tokens" paradigm. **Could we design a template-graph method based on the "Prompt as graph" concept** that integrates task and category information to simultaneously achieve joint training and all-level task classification? General graph prompt learning typically relies on fine-tuning randomly initialized prompts in the downstream phase to capture task-specific information, which may result in incomplete representations. **Could we instead leverage the advantages of template-graphs to extract task-relevant information in the upstream phase**, thus enabling task-specific in-context learning in the downstream phase?

To deal with the challenges outlined above, the contribution of this work is threefold: (1) We design a prompt template-graph that includes heterogeneous graph information adapted to all-level tasks, ensuring that multi-task data can be trained together. (2) We propose a sophisticated virtual prototype specification method to embed the corresponding category information into the prompt template-graph. (3) We introduce Pre-prompt, a learnable token in the upstream phase, to optimize for specific downstream tasks, achieving HePa's in-context learning.

We evaluated the HePa framework on three heterogeneous graph benchmark datasets, covering multi-task and few-shot scenarios. Experiments demonstrate that HePa performs well under these settings. Additionally, we assessed HePa's cross-task knowledge transfer capability, a unique advantage brought by HePa's prompt template. Fig.1 illustrates the overall framework of HePa.

	Het.a	In-context	Joint-train	All-level
GPPT				
GraphPrompt				
HetGPT	✓			
HGPrompt	✓			
PRODIGY		✓		✓
All-in-One			✓	✓
One-for-All		✓	✓	✓
HePa	✓	✓	✓	✓

Table 1: HePa vs. related methods comparison.

2 Related Work

Prompt learning with "Prompt as token". This refers to graph prompts as additional separate tokens that align the downstream and pre-training knowledge gap (Zi et al. 2024).

This technique begins with GPPT(Sun et al. 2022) employing link prediction for pre-training, but its prompts a8e designed solely for downstream node classification. GraphPrompt (Liu et al. 2023c) introduces a unified framework for few-shot learning on graphs by converting various tasks to the prediction similarity of subgraphs. HetGPT (Ma et al. 2024) addresses the challenges of heterogeneous graphs by converting them into multiple homogeneous subgraphs and introducing a dual-prompt mechanism to bridge gaps caused by feature variations and heterogeneity differences. Meanwhile, HGPrompt (Yu et al. 2024) integrates homogeneous and heterogeneous graphs within a unified framework using a dual-template approach, enhancing the applicability of pre-trained models to various graph-based tasks.

Prompt learning with "Prompt as graph". This refers to a graph prompt that has multiple prompt tokens with inner structures and insert patterns. (Zi et al. 2024) The work presented in PRODIGY (Huang et al. 2023) extends this concept by developing a pretraining framework that enables in-context learning over graphs with a novel prompt graph representation. "All-in-One" (Sun et al. 2023) tackles the multi-task problem in graph prompts by unifying the language and graph prompts format to facilitate the transfer of the prompting idea from NLP to graph domains. In contrast, the "One-for-All" (Liu et al. 2023a) framework, i.e., OFA, introduces the Graph Prompting Paradigm to enable in-context learning for graphs, where a prompt graph combines input graphs with additional nodes and edges to encode task-specific information. This paradigm enhances zero-shot learning capabilities, allowing the model to adapt to new tasks without further training.

Table 1 shows the comparison of HePa and the models mentioned above in terms of various capabilities, where **Het.a** stands for the adaptation of the attributed heterogeneous graph.

3 Preliminaries

Heterogeneous graphs. A heterogeneous graph (Sun and Han 2012) is defined as $G = (V, E, A, R, \phi, \varphi)$, where V is the set of nodes and E is the set of edges. It is associated with a node type mapping function $\phi : V \to A$ and an edge type mapping function $\varphi : E \to R$, in which A and R denote the node type set and edge type set, respectively. Also, to fulfil the heterogeneous graph's requirement, we require $|A| + |R| > 2$. We assume an input feature matrix of which dimension one's size is d for all the nodes as $\mathbf{X} \in \mathbb{R}^{|V| \times d}$.

Graph In-context learning. In-context learning refers to a model's ability to learn a task with only a few provided examples (Dong et al. 2024). For language models, this is primarily achieved through the use of prompts. In the field of graph learning, graph in-context learning refers to the ability to predict task outcomes in the downstream phase by relying solely on a few graph examples contained in the query set (Huang et al. 2023) without the need for additional fine-tuning methods such as prompt fine-tuning.

Target Scenarios. Our work mainly focuses on three levels of few-shot classification tasks: node classification, graph classification, and edge classification tasks. Let us denote a dataset as $D = \{(d_i, y_i)\}_{i=1}^{M}$ where M is the number of data in the dataset, $y_i \in \mathcal{Y}$ is the label of d_i and $\mathcal{Y}$ is the set of all data labels, d_i is a data sample. For a data sample $d_i = (\mathcal{V}_i, \mathcal{E}_i, \mathcal{H}_i, \mathcal{R}_i, \mathcal{T}_i)$, $\mathcal{E}_i$ is the edge set of d_i, $\mathcal{H}_i$ is the feature of d_i, $\mathcal{R}_i$ represents the edge type map of the data sample, and $\mathcal{T}_i$ is the target nodes set of d_i. The model will be pretrained on D_{train} and be evaluated on D_{val} to validate the best model. Then, D_{test} is used to perform downstream tasks. All labels in the test data have never appeared in the training data, meaning $\mathcal{Y}_{\text{train}} \cap \mathcal{Y}_{\text{test}} = \emptyset$. Few-shot scenarios typically handle N-way k-shot tasks, meaning there are k labeled data samples for each of the N classes, $D = d_{i1}^{N \cdot K}$. The model must classify data in the query set based on these support samples.

4 Proposed Framework: HePa

In this section, we introduce HePa, a novel heterogeneous graph prompting framework that enables in-context learning for downstream classification tasks. To address the two challenges outlined in Section 1, HePa consists of the following key components: (1) HePa Template unifies the data for node, graph, and edge level tasks; (2) Our prototype feature specification method injects class information into the prototype nodes of the prompt template; (3) Pre-Prompt activates the contextual learning capability for downstream tasks.

4.1 Overview

We begin with the overall framework of HePa, as seen in Fig. 1, the workflow of HePa is divided into two parts: upstream and downstream.

Upstream Phase. We designate the raw heterogeneous graph data as the Source Graph and organize it into unified prompt template-graphs containing category information using the HePa Template. These template-graphs train the model. After pretraining, we freeze the model parameters and introduce Pre-prompt tokens, optimizing them individually for each task level to incorporate task-specific information.

Downstream Phase. By embedding task information into templates via Pre-prompt tokens for each task type, downstream templates encode both task and category information. Consequently, the downstream template-graphs alone can retrieve task-specific knowledge from the pre-trained model, enabling in-context learning for classification at all levels.

We will cover the construction of the HePa Template in Sect. 4.2, explore the virtual prototype specification method in Sect. 4.3, present the pretraining objectives in Sect. 4.4, and discuss the extraction and injection of task information by Pre-prompt in Sect. 4.5.

4.2 HePa Template

As shown in Fig. 2, the HePa Template mainly consists of four components: (1) a Target subgraph containing the target nodes; (2) a Prompt node, a container for task information; (3) Prototype nodes, containers for category information; (4) k-shot sample graphs for the support set.

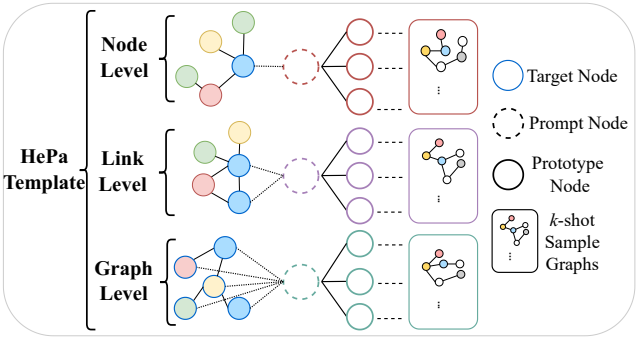

Figure 2: Structure of the HePa template.

Target Subgraph. In classification tasks involving diverse graphs, the focus could be a node, a graph, or an edge. When the focus is a node, it is the one to be classified. If the focus is a graph or an edge, a consistent approach is needed. **Here, we draw inspiration from the approach used in OFA** (Liu et al. 2023a), where the target node for node-level tasks is the one being classified; for edge-level tasks, it is the two nodes at either end of the edge, and for graph-level tasks, it's all nodes within the graph. Once the target nodes are identified, a set of virtual nodes will be linked to them.

In the HePa template diagram shown in Fig. 2, the circular blue nodes represent the target nodes, and we denote the set of all target nodes as $\mathcal{T}$. We denote $\mathcal{S}_l(v) = \{V_v^l, E_v^l, R_v^l\}$ as the l-hop subgraph centered on node v. We denote the target subgraph of $\mathcal{T}$ as $\mathcal{G}_\ell(\mathcal{T})$, which is the union of all l-hop subgraphs of $\mathcal{T}$,

$$\mathcal{G}_l(\mathcal{T}) = \bigcup_{v\in\mathcal{T}} \mathcal{S}_l(v) = \left\{ \bigcup_{v\in\mathcal{T}} V_v^l, \bigcup_{v\in\mathcal{T}} E_v^l, \bigcup_{v\in\mathcal{T}} R_v^l \right\} \quad (1)$$

We define $\mathcal{G}_\ell(\mathcal{T}) = \{V_\mathcal{T}^l, E_\mathcal{T}^l, R_\mathcal{T}^l\}$. For the node classification task of the node v, $\mathcal{T} = v$ and $\mathcal{G}_l(\mathcal{T}) = \mathcal{S}_l(v)$. For the edge classification task of the node pair (v_i, v_j), $\mathcal{T} = \{v_i, v_j\}$ and $\mathcal{G}_l(\{v_i, v_j\}) = \mathcal{S}_l(v_i) \cup \mathcal{S}_l(v_j)$. For the graph classification task, $\mathcal{T} = \{V\}$ and $\mathcal{G}_l(\mathcal{V}) = (V, E, R)$.

Virtual Nodes. Our template includes two types of virtual nodes: prompt node and prototype node. After constructing the target subgraph, we connect the target subgraph to the task information container, the prompt node. At the upstream stage, the feature of the prompt node is set to null, the reason for which will be detailed in Sect. 4.5. We denote the prompt node as v_p, which is connected to every target node. We denote these edges as $E_{pmt} = \{(t, r_{pmt}, v_p), (v_p, r_{pmt}, t) \mid t \in \mathcal{T}\}$, where r_{pmt} represents the edge relationship between the prompt node and the target nodes. For both forward and reverse edges, the relationship is the same, i.e., undirected edges.

The virtual nodes responsible for containing the category information are called prototype nodes. As the name implies, the category information contained in these nodes is obtained by calculating class prototype features. The specific calculation method will be discussed in Sect. 4.3. The number of prototype nodes depends on the number of classes to be classified. To combine task information and category information, we connect these prototype nodes to the prompt node. We denote the set of prototype nodes as $V_{ptt} = \{c_i \mid i \in [N]\}$, where c_i is a single prototype node, and N is the number of classes. Thus $E_{ptt} = \{(c_i, r_{ptt}, v_p), (v_p, r_{ptt}, c_i) \mid i \in [N]\}$, where r_{ptt} represents the edge relationship between the prototype nodes and the prompt node.

Support Set. Due to our focus on the few-shot scenario, we also need to connect the k-shot sample data from the support set to our prompt graph template. A single sample in the support set is also a target subgraph. We denote $\mathcal{G}_l^{i,k}\left(\mathcal{T}_k^i\right) = \left\{V_{\mathcal{T}_k^i}^l, E_{\mathcal{T}_k^i}^l, R_{\mathcal{T}_k^i}^l\right\}$ as the i-th class and k-th support sample, where $\mathcal{T}_k^i$ belongs to the i-th class. We connect samples that belong to the same i-th class to the corresponding prototype node c_i,

$$\begin{aligned} E_{\text{sup}} &= \textstyle\bigcup_{i\in[N],k\in[K]} E_{\text{sup}}^{i,k}, \\ &= \textstyle\bigcup_{i\in[N],k\in[K]} \left\{ \left\{ (t, r_{\text{sup}}, c_i), (c_i, r_{\text{sup}}, t) \mid t \in \mathcal{T}_k^i \right\}, E_{\mathcal{T}_k^i}^l \right\}, \end{aligned} \quad (2)$$

Where r_{sup} represents the edge relationship between the target nodes in the support set and the prototype node, so $V_{\text{sup}} = \bigcup_{i\in[N],k\in[K]} \left\{V_{\mathcal{T}_k^i}^l\right\}$, $R_{\text{sup}} = \bigcup_{i\in[N],k\in[K]} \left\{ \left\{R_{\mathcal{T}_k^i}^l\right\}, r_{\text{sup}} \right\}$. We denote $\mathcal{G}_P = (V_P, E_P, R_P)$ to represent the HePa template graph. $\mathcal{G}_P$ consists of the following components,

$$\begin{aligned} V_P &= \{v_p\} \cup V_T^l \cup V_{ptt} \cup V_{\text{sup}}\,, \\ E_P &= E_T^l \cup E_{pmt} \cup E_{ptt} \cup E_{\text{sup}}\,, \\ R_p &= \{r_{pmt}, r_{ptt}\} \cup R_T^l \cup R_{\text{sup}}\,. \end{aligned} \quad (3)$$

4.3 Virtual Prototype

This section outlines the methods for specifying virtual prototype features in our template graph. Virtual prototypes are the features of prototype nodes that accurately represent class information. Proper calculation methods are essential, especially in attributed heterogeneous graphs.

To fully utilize the neighborhood topology and semantic information of the target nodes, we first introduce the concept of the contextual graph (Liu et al. 2023c). For a target node t, its contextual graph $\mathcal{G}_{\text{tex}}(t)$ is usually obtained by employing a breadth-first search to extract a δ-hop subgraph centering on t, i.e., $\mathcal{G}_{\text{tex}}(t) = \mathcal{S}_\delta(t)$. The contextual graph of t contains rich neighborhood information. Next, we explain our design of the prototype features' specification methods for the prototype nodes under different tasks.

• **Node Classification (NC).** Given a heterogeneous graph G with a set of labeled node classes $\mathcal{Y}$, and a labeled node dataset $D = \{(d_i, y_i)\}_{i=1}^M$ such that y_i is the class label of d_i, as noted in Sect. 2, $d_i = (\mathcal{V}_i, \mathcal{E}_i, \mathcal{H}_i, \mathcal{R}_i, \mathcal{T}_i)$. Since $\mathcal{T}_i$ represents the target nodes set and the node classification task has only one target node, $\mathcal{T}_i = \{t_i\}$. In the few-shot scenario we focus on, the support set of $\mathcal{G}_P$ contains exactly k pairs of $\{(d_i, y_i = c)\} \in D$ for every class $c \in \mathcal{C}$. These samples from the support set are

the source for calculating the prototype. For the contextual graph $\mathcal{G}_{tex}(\mathcal{T}_i)$, its node set can be represented as $V_{tex}(\mathcal{T}_i) = \{u \in \mathcal{V}_i \mid \text{dist}(t_i, u) \leq \delta\}$, the features set as $H_{tex}(\mathcal{T}_i) = \{h_u \mid u \in V_{tex}(\mathcal{T}_i)\}$. Next, we will calculate the features of $\mathcal{G}_{tex}(\mathcal{T}_i)$. First, we compute the weights for the nodes based on their distance to the target node in the contextual graph,

$$W_i = \left\{ w_v = \frac{1}{\text{dist}(t_i, v) + 1} \middle| \; v \in V_{\text{tex}}(\mathcal{T}_i) \right\}, \quad (4)$$

where $\text{dist}(v, u)$ represents the shortest distance between nodes pair (v, u), and W_i is the set of all node weights. For $v \in V_{tex}(\mathcal{T}_i)$, we obtain the feature of $\mathcal{G}_{tex}(\mathcal{T}_i)$ through weighted averaging,

$$x_i = \frac{\sum_{v \in V_{tex}(\mathcal{J}_i)} w_v \cdot h_v}{\sum W_i} \quad (5)$$

where x_i is the feature of $\mathcal{G}_{tex}(\mathcal{T}_i)$.Considering $\mathcal{X}_c^k = \{x_1, x_2, \ldots, x_k\}$ represents the set of features of k contextual graphs corresponding to class $c \in \mathcal{C}$, the NC prototype feature $\tilde{\mathbf{x}}_c$ is given by,

$$\tilde{\mathbf{x}}_c = \frac{\sum x_c^k}{k} \quad (6)$$

• Edge Classification (EC). Specifying class prototypes becomes slightly more complex in edge classification tasks. For the target subgraph, there are two target nodes, which we refer to as the source node and the destination node. We employ a joint computation method to obtain the feature of each edge sample d_i. First, $\mathcal{T}_i = \{t_a, t_b\}$, where t_a represents the source node and t_b represents the destination node. The contextual graphs of these two nodes are $\mathcal{G}_{\text{tex}}(t_a)$ and $\mathcal{G}_{\text{tex}}(t_b)$. We denote $\mathcal{G}_{\text{tex}}(\mathcal{T}_i) = \mathcal{G}_{\text{tex}}(t_a) \cup \mathcal{G}_{\text{tex}}(t_b)$, which has a target node pair (t_a, t_b). We calculate the weights for the nodes based on the minimum distance to either of the central nodes,

$$\begin{aligned} &\text{dist}(v) = \min\left(\text{dist}(t_a, v), \text{dist}(t_b, v)\right) \\ &W_i = \left\{ w_v = \frac{1}{\text{dist}(v) + 1} \middle| \; v \in V_{\text{tex}}(\mathcal{T}_i) \right\} \end{aligned} \quad (7)$$

After that, we calculate the EC prototype using equations (5) and (6).

• Graph Classification (GC). In graph classification tasks, for a data sample d_i, since every node in its target subgraph $\mathcal{G}_l(\mathcal{T}_i)$ is a target node, the contextual graph for this graph is the target subgraph itself. We denote $\mathcal{G}_{\text{tex}}(\mathcal{T}_i) = \mathcal{G}_l(\mathcal{T}_i)$, $H_{\text{tex}}(\mathcal{T}_i) = \{h_u \mid u \in V_{\text{tex}}(\mathcal{T}_i)\}$. The features of $\mathcal{G}_{\text{tex}}(\mathcal{T}_i)$ are obtained directly by averaging the features of all nodes,

$$x_i = \frac{\sum H_{\text{tex}}(\mathcal{T}_i)}{|V_{\mathcal{T}_i}|}. \quad (8)$$

Then, the calculation of the GC prototype is the same as NC and EC, using equations (5) and (6).

We have calculated the prototype features of each class for tasks at the node, edge, and graph levels using different strategies. Then, we assign these prototype features to the corresponding prototype nodes.

Dataset	# Nodes	# Feature Dim	# Edges	# Edge Types	# Node Classes
IMDB	11,616	3,066	34,212	4	3
DBLP	26,128	334	239,566	6	4
ACM	10,924	1,902	547,872	8	3

Table 2: Statistics of datasets.

4.4 Pre-training Objective

In our pre-training phase, we input the prompt graph processed by the HePa template into an HGNN model and then use the embeddings of the prototype nodes from the output to perform a binary classification loss. Since our target scenario is an N-way k-shot task, the number of embeddings output by the prototype nodes is N. For a prototype node represented as c_i, its model output embedding is h_{c_i}, where $i \in [N] < |\mathcal{Y}|$, the corresponding label for c_i is y_i. We use an MLP prediction head with a one-dimensional output to perform binary classification for the embedding of each prototype node. The predicting probability $p_{c_i|y_i}$ that y_i is the label of c_i is given as follows,

$$p_{c_i|y_i} = \text{Sigmoid}\left(\text{MLP}\left(h_{c_i}\right)\right), \quad (9)$$

here, the activation function Sigmoid is used to convert the one-dimensional embedding output from the MLP into a probability between 0 and 1. For a template graph with N classes, that is, N prototype nodes, the model predicts the class label of the target $\mathcal{T}$ as follows,

$$L_{\mathcal{T}} = \text{argmax}_{i \in [N]}\left(p_{c_i|y_i}\right). \quad (10)$$

4.5 Pre-prompt: Gathering of Task Information

Using the pretraining objective in Sect. 4.4, we obtained an HGNN model trained from all-level classification task data. However, as mentioned in Sect. 4.3, the feature of the prompt nodes in the samples was initialized to null. So, how does our model autonomously perform node, edge, and graph classification tasks at the downstream stage through in-context learning? Our solution is to introduce the Pre-prompt token, a learnable vector, denoted as $\mathbf{p}^{\text{pre}}$, during the upstream phase, which extracts task information for the three classification tasks mentioned above.

We obtain task information by optimizing the Pre-prompt. To do this, we extract a set of data from all levels of the classification tasks from the training data D_{train} and refer to this set as $D_{\text{pp}} = \{D_{\text{NC}}, D_{\text{EC}}, D_{\text{GC}}\}$, where $D_{\text{pp}} \subseteq D_{\text{train}}$. We optimize the Pre-Prompt using data $D_n \in D_{\text{pp}}$ from each type of task. The loss function for optimizing the Pre-prompt on task type n is denoted by $\mathcal{L}^n(\mathbf{p}^{\text{pre}}) =$

$$-\frac{1}{N}\sum_{i \in [N]}^{D_n \in D_{pp}} y_i \cdot \log\left(p_{c_i|y_i}\right) + (1 - y_i) \cdot \log\left(1 - p_{c_i|y_i}\right), \quad (11)$$

where N is the number of classes in the data for a single task, and $p_{c_i|y_i}$ is the probability that the prototype node c_i is predicted to have the label y_i. Ultimately, we obtain a set of Pre-prompt tokens of different tasks, $\mathcal{P} = \{\mathbf{p}_{NC}^{\text{pre}}, \mathbf{p}_{EC}^{\text{pre}}, \mathbf{p}_{GC}^{\text{pre}}\}$. When performing downstream tasks,

Method	Node Classification						Graph Classification						Edge Classification					
	IMDB		DBLP		ACM		IMDB		DBLP		ACM		IMDB		DBLP		ACM	
GAT	38.64±	5.54	62.98±13.08		32.56±14.12		30.37±	3.83	41.45±17.02		32.98±	2.45	28.98±	2.07	39.15±	3.06	32.02±	4.07
GCN	37.69±	9.11	51.03±13.70		48.33±12.05		29.34±	5.56	38.65±15.53		32.83±	2.40	33.13±	7.52	45.28±	6.22	37.52±	7.19
MAGNN	44.88±	2.56	54.25±	8.48	59.38±	8.39	30.68±10.19		51.67±16.81		36.03±10.40		42.45±	2.58	52.36±	3.08	45.36±	1.78
HAN	42.63±	3.13	61.65±17.89		61.22±	9.01	31.51±	8.38	43.19±14.73		36.17±	9.80	39.51±	3.14	50.24±	4.33	46.88±	2.62
GraphCL	37.49±10.09		67.14±16.54		57.11±	7.45	32.73±13.86		49.84±14.20		33.45±13.67		-		-		-	
HeCo	38.25±11.10		68.95±14.23		61.36±11.78		35.82±14.90		53.40±13.03		34.12±14.89		-		-		-	
GPPT	41.85±	1.58	61.47±	9.23	57.85±	9.97	-		-		-		-		-		-	
GraphPrompt	47.01±	9.31	68.73±	9.28	65.99±10.12		36.21±15.60		54.13±10.62		35.78±15.12		-		-		-	
HGPrompt	51.73±	7.86	75.53±14.60		69.73±10.89		41.62±15.07		56.49±11.68		36.53±15.07		-		-		-	
HePa	**54.53±**	**6.82**	**78.81±**	**6.74**	**74.62±**	**7.31**	**43.74±**	**5.67**	**61.08±**	**8.13**	**40.51±**	**9.76**	**53.39±**	**3.40**	**67.66±**	**3.53**	**57.21±**	**6.05**

Table 3: Evaluation of all-level few-shot classification tasks.

we determine the downstream task based on the number of target nodes in the downstream data and assign the null-feature prompt node as the corresponding task's Pre-prompt token.

5 Experiments

5.1 Experimental Setup

Datasets. We conduct experiments on three benchmark datasets (Fu et al. 2020).

- IMDB (Internet Movie Database) dataset comprises a large collection of movie-related data, including extensive metadata on films, actors, crew members, and user-generated ratings and reviews.
- DBLP (Digital Bibliography & Library Project) dataset is a comprehensive bibliographic database covering major journals and conference proceedings in computer science.
- ACM (Association for Computing Machinery) dataset is an extensive collection of scholarly articles published in ACM conferences and journals. Detailed statistics of these datasets can be found in Table 2.

Baselines. We assess the performance of HePa against methods belonging to four different categories.

- Supervised GNNs: GAT (Veličković et al. 2018), GCN (Kipf and Welling 2016).
- Supervised HGNNs: HAN (Wang et al. 2019), MAGNN (Fu et al. 2020).
- GNN/HGNNs with "Pretrain, fine-tune": GraphCL (You et al. 2020), HeCo (Wang et al. 2021).
- GNN/HGNNs with "Pre-train, prompt": GPPT (Sun et al. 2022), GraphPrompt (Liu et al. 2023c), HGPrompt (Yu et al. 2024).

Implementation Details. To demonstrate the all-level task capability of our framework, we focus on NC, GC, and EC tasks in a few-shot scenario that covers all levels of tasks. We present the results in Table 3. We also employ few-shot link prediction as a downstream task (see Appendix B). In the experimental data, the labels for NC task data come from the dataset labels, the GC task data is obtained by sampling ego-networks and using the dataset labels, and the EC task data labels are selected from the edge types of the dataset. Both upstream and downstream data are from the same dataset, with all task data trained simultaneously. Our model employs R-GCN (Schlichtkrull et al. 2018) as the backbone, more details can be found in Appendix A.

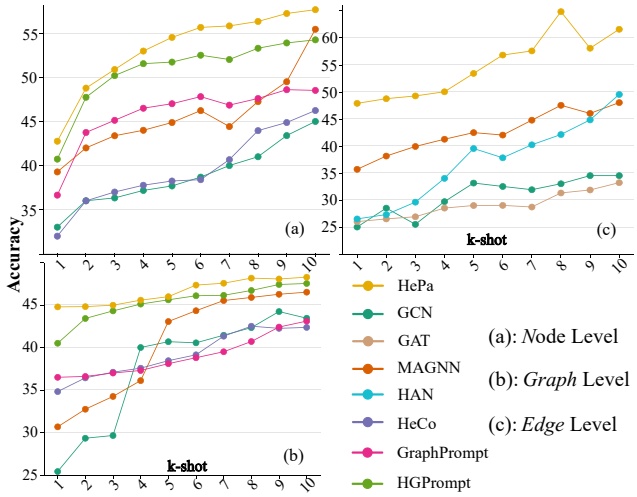

Figure 3: Impact of shots on all-level tasks on *IMDB*.

5.2 Performance Evaluation

We first analyze a fixed shot value and then change the number of shots to analyze the trend in model performance. We adopt Accuracy as the evaluation metric.

Few-shot Node Classification (NC). Following the typical k-shot setup (Wang et al. 2020; Liu et al. 2023a), we randomly select 200 5-shot NC task data for evaluation. The results of the 5-shot NC are presented in Table 3. We make the following observations: Our model, HePa, outperforms baseline models in accuracy across all datasets by effectively

Method	Prompt node	Pre-prompt token	Prototype node	Node-level		Graph-level		Edge-level	
				IMDB	ACM	IMDB	ACM	IMDB	ACM
Prompt.N			✓	47.16	66.34	28.75	30.81	34.31	36.23
Prompt.T	✓		✓	50.72	68.41	35.00	32.73	36.80	42.38
Prototype.N	✓	✓		51.82	72.05	39.73	36.25	47.32	52.19
HePa	✓	✓	✓	**54.53**	**74.62**	**43.77**	**40.51**	**53.39**	**57.21**

Table 4: Ablation study variants and their accuracy results for all-level classification tasks.

capturing and utilizing heterogeneous graph structures and semantic information. It also demonstrates the effectiveness of Pre-prompt and outperforms methods requiring prompt fine-tuning. Additionally, models using heterogeneous information such as HeCo and HGPrompt perform better than their homogeneous graph counterparts but still fall short of HePa, specifically designed with a prompt template for heterogeneous graphs.

Few-shot Graph Classification (GC). Following previous work (Lu et al. 2021), we create a set of graphs by gathering ego-networks centered on the nodes with class labels in each dataset. We randomly select 200 sets of 1-shot GC task data for evaluation. The analysis of model performance for graph classification yields similar conclusions to those observed for NC.

Few-shot Edge Classification (EC). For the edge classification task, we create a set of graphs by gathering ego-networks centered on each end node of the target edge. We selected 200 sets of 5-shot edge classification task data for evaluation. Our model outperforms traditional supervised models across three datasets. The template-graph provided by HePa enabled us to handle task data at three levels simultaneously, allowing us to accomplish previously unattainable edge classification tasks.

Impact of different shots on HePa. To further investigate the impact of shots, we vary the number of shots for all three levels of tasks and benchmark against several competitive baselines in Fig.3 on IMDB (see Appendix C. for other datasets). The key findings from our model demonstrate consistent upward accuracy trends with increasing k-shots, stable performance in graph classification, and robustness across different levels of labeled data.

5.3 Model Analysis

Ablation study. To study the impact of each component within HePa, we undertake an ablation study as shown in Table 4. Variant *Prompt.N* only uses prototype nodes to store category information, with the target subgraph directly connected to this node. Variant *Prompt.T* does not use Pre-Prompt for extracting and injecting task information. Variant *Prototype.N* does not use our prototype calculation strategy but is directly represented by the average features of the labeled nodes. In Table 4, we have the following observations: (1) The *Prompt.N* variant shows a significant drop in performance across all tasks compared to the full model, demonstrating the necessity of the complete prompt mechanism. (2) Without the Pre-prompt token, the model lacks the nuanced understanding of task-specific contexts, underperforming relative to the full. (3) The *Prototype.N* variant's low performance suggests that explicit computation of prototype features is essential for accurately representing class characteristics.

Method	Edge-Level	Node-Level	Graph-Level
GCN	27.43	31.02	28.63
GAT	28.05	33.76	29.34
MAGNN	37.65	37.93	30.85
HAN	41.21	37.60	31.51
HePa	**51.85**	**41.18**	**38.96**

Table 5: Evaluation of cross-task capability on IMDB. The model is trained on graph-level data.

Cross-Task capability. To further investigate our model's cross-task classification capability, we train the model solely on graph classification data and then perform classification tasks at all three levels in the downstream phase. We present the results on IMDB in Table 5 (for other datasets, please see Appendix D). We observe the followings: (1) Although the training data does not include edge classification data, the results for the edge classification task are still excellent and very close to the results of joint training. (2) The performance of the node classification task is still better than other comparison models, but there is a significant gap compared to joint training. (3) The accuracy of the model trained solely on graph-level data for graph classification tasks is slightly lower than the results of joint training.

These phenomena indicate that HePa has good cross-task capability, and the jointly trained HePa model can enhance downstream task performance by integrating information from different task data through message passing.

6 Conclusions

In this study, we present HePa, a heterogeneous graph prompting framework for all levels of classification tasks to address the challenges of few-shot graph prompting. First, we introduce a unified prompt template-graph for both upstream training and downstream tasks. Then, we propose the method to specify the task and class information. In addition, by incorporating task-specific Pre-prompt tokens, HePa enables In-context learning for all-level tasks. Experimental results on benchmark datasets demonstrate HePa's superior performance and robustness compared to existing models.

Acknowledgments

This work is supported by grant 62471133 National Natural Science Foundation of China. This work is also supported by grant 2242024K40020 Central University Basic Research Fund of China, and the Big Data Computing Center of Southeast University.

References

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.

Dong, Q.; Li, L.; Dai, D.; Zheng, C.; Ma, J.; Li, R.; Xia, H.; Xu, J.; Wu, Z.; Chang, B.; Sun, X.; Li, L.; and Sui, Z. 2024. A Survey on In-context Learning. arXiv:2301.00234.

Fu, X.; Zhang, J.; Meng, Z.; and King, I. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *Proceedings of The Web Conference 2020*, WWW '20, 2331–2341. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-7023-3.

Hu, Z.; Dong, Y.; Wang, K.; Chang, K.-W.; and Sun, Y. 2020. GPT-GNN: Generative Pre-Training of Graph Neural Networks. In *The ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1857–1867.

Huang, Q.; Ren, H.; Chen, P.; Kržmanc, G.; Zeng, D.; Liang, P. S.; and Leskovec, J. 2023. PRODIGY: Enabling In-context Learning Over Graphs. *Advances in Neural Information Processing Systems*, 36: 16302–16317.

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.; Jia, T.; Fang, Y.; Shi, C.; Lin, Z.; and Wang, H. 2021. Pre-Training on Large-Scale Heterogeneous Graph. In *The ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 756–766.

Kipf, T. N.; and Welling, M. 2016. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

Liu, H.; Feng, J.; Kong, L.; Liang, N.; Tao, D.; Chen, Y.; and Zhang, M. 2023a. One For All: Towards Training One Graph Model For All Classification Tasks. In *The Twelfth International Conference on Learning Representations*.

Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; and Neubig, G. 2023b. Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Computing Surveys*, 55(9): 1–35.

Liu, Z.; Yu, X.; Fang, Y.; and Zhang, X. 2023c. 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.

Ma, Y.; Yan, N.; Li, J.; Mortazavi, M.; and Chawla, N. V. 2024. HetGPT: Harnessing the Power of Prompt Tuning in Pre-Trained Heterogeneous Graph Neural Networks. In *Proceedings of the ACM on Web Conference 2024*, WWW '24, 1015–1023. New York, NY, USA: Association for Computing Machinery. ISBN 9798400701719.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; van den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling Relational Data with Graph Convolutional Networks. In Gangemi, A.; Navigli, R.; Vidal, M.-E.; Hitzler, P.; Troncy, R.; Hollink, L.; Tordai, A.; and Alam, M., eds., *The Semantic Web*, 593–607. Cham: Springer International Publishing. ISBN 978-3-319-93417-4.

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. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wang, K.; Liang, Y.; Li, X.; Li, G.; Ghanem, B.; Zimmermann, R.; zhou, Z.; Yi, H.; Zhang, Y.; and Wang, Y. 2023. Brave the Wind and the Waves: Discovering Robust and Generalizable Graph Lottery Tickets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

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.

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.; Fang, Y.; Liu, Z.; and Zhang, X. 2024. HGPrompt: Bridging Homogeneous and Heterogeneous Graphs for Few-Shot Prompt Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(15): 16578–16586.

Yu, X.; Zhou, C.; Fang, Y.; and Zhang, X. 2023. MultiGPrompt for Multi-Task Pre-Training and Prompting on Graphs. arXiv:2312.03731.

Zi, C.; Zhao, H.; Sun, X.; Lin, Y.; Cheng, H.; and Li, J. 2024. ProG: A Graph Prompt Learning Benchmark. arXiv:2406.05346.