GPPT: Graph Pre-training and Prompt Tuning to Generalize Graph Neural Networks

Mingchen Sun[*]
Jilin University
Changchun, China
mcsun20@mails.jlu.edu.cn

Kaixiong Zhou[*]
Rice University
Houston, United States
Kaixiong.Zhou@rice.edu

Xin He
Jilin University
Changchun, China
hexin20@mails.jlu.edu.cn

Ying Wang[†]
Jilin University
Changchun, China
wangying2010@jlu.edu.cn

Xin Wang
Jilin University
Changchun, China
xinwang@jlu.edu.cn

ABSTRACT

Despite the promising representation learning of graph neural networks (GNNs), the supervised training of GNNs notoriously requires large amounts of labeled data from each application. An effective solution is to apply the transfer learning in graph: using easily accessible information to pre-train GNNs, and fine-tuning them to optimize the downstream task with only a few labels. Recently, many efforts have been paid to design the self-supervised pretext tasks, and encode the universal graph knowledge among the various applications. However, they rarely notice the inherent training objective gap between the pretext and downstream tasks. This significant gap often requires costly fine-tuning for adapting the pre-trained model to downstream problem, which prevents the efficient elicitation of pre-trained knowledge and then results in poor results. Even worse, the naive pre-training strategy usually deteriorates the downstream task, and damages the reliability of transfer learning in graph data. To bridge the task gap, we propose a novel transfer learning paradigm to generalize GNNs, namely graph pre-training and prompt tuning (GPPT). Specifically, we first adopt the masked edge prediction, the most simplest and popular pretext task, to pre-train GNNs. Based on the pre-trained model, we propose the graph prompting function to modify the standalone node into a token pair, and reformulate the downstream node classification looking the same as edge prediction. The token pair is consisted of candidate label class and node entity. Therefore, the pre-trained GNNs could be applied without tedious fine-tuning to evaluate the linking probability of token pair, and produce the node classification decision. The extensive experiments on eight benchmark datasets demonstrate the superiority of GPPT, delivering an average improvement of 4.29% in few-shot graph analysis and accelerating the model convergence up to 4.32X. The code is available in: https://github.com/MingChen-Sun/GPPT.

CCS CONCEPTS

• **Information systems → Data mining**; • **Computing methodologies** → *Learning latent representations*; • **Theory of computation** → *Semi-supervised learning*; • **Networks** → *Network algorithms*;

KEYWORDS

Graph neural networks, pre-training, prompt tuning.

ACM Reference Format:
Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. 2022. GPPT: Graph Pre-training and Prompt Tuning to Generalize Graph Neural Networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539249

1 INTRODUCTION

Graph neural networks (GNNs) [5, 7] have become a prominent technique to analyze graph structured data in many real-world systems, including social networks [36], recommender systems [45], and knowledge graph [39]. The general approach of GNNs treats the input graph as an underlying computation graph, and learns the node representations by passing messages across the edges. The generated node representations could be used for different downstream tasks, such as link prediction [42, 43], node classification [2, 48], and biochemical module classification [40, 49].

Recent efforts in transfer learning have advanced GNNs to capture the transferable graph patterns and generalize to different downstream tasks [10, 22, 25]. Specifically, most of them follow the "pre-train, fine-tune" learning strategy: using the easily accessible information as pretext task (e.g., edge prediction) to pre-train GNNs, and fine-tuning over downstream task with the pre-trained model as initialization. By leveraging the substantial corpus of unlabeled structure, the pre-training has become an attractive solution for few-shot graph analysis [11, 44], where the labeled data is scarce. Under the "pre-train, fine-tune" framework, the existing works mainly focus on objective engineering: tailoring the pre-training objectives

[*]Both authors contributed equally to this research.
[†]Corresponding author

KDD '22, August 14–18, 2022, Washington, DC, USA.
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539249

to better inform the downstream applications. The prevalent pre-training approaches include edge prediction [13] and contrastive learning [25].

One of key limitations in "pre-train, fine-tune" is the training objective gap between the constructed pretext and dedicated downstream tasks. This gap would hinder us from directly and efficiently eliciting the pre-trained graph knowledge. We take the pretext edge prediction and downstream node classification as an example. Traditionally, the pre-trained GNNs were transferred by simply replacing the final classification layer, and fine-tuned together with the new parameters. While the pre-trained parameters optimize the binary edge predictions of node pairs, they need to be tuned time consumingly towards the parameter space conducting multi-class node predictions of standalone nodes, especially when the connected nodes have distinct classes. It is observed that such fine-tuning even results in worse downstream performance than training from scratch [13], meaning the negative and failing knowledge elicitation. The previous objective engineering often carefully designs the pretext task close to every new downstream application, which requires both expert knowledge and tedious manual trials.

Instead of being limited in the objective engineering, we propose to explore the novel strategy of "pre-train, prompt, fine-tune" to bridge the task gap in GNNs. The prompt technique is motivated by natural language processing (NLP) [19], where it is applied to reformulate the downstream task looking similar to the pretext one. We use the example of language model pre-trained with masked word prediction, and the downstream classification task with raw input text (e.g., emotion classification of "I missed the bus today"). The prompting function aims at modifying the input text to a prompt by appending the semantic textual template (e.g., "I missed the bus today. I felt so [MASK]"). In this way, the classification task is reformulated to fill the masked emotion word (e.g., "wonderful"), which is in line with the pre-training objective. The pre-trained model could be applied with or without slight fine-tuning to quickly elicit the pre-trained knowledge. This unique advantage of efficient fine-tuning has improved the applicability of generally pre-trained model to a wide series of language tasks [17, 18, 31].

However, it is non-trivial to design the prompting function to generalize GNNs due to following two challenges. First, given the symbolic graph data, it is infeasible to apply the semantic prompting function to bridge the various graph machine learning tasks. One cannot use the textual template (e.g., "I felt so [MASK]" as adopted in NLP) to reformulate the node classification to edge prediction, which are defined to learn the abstract nodes instead of semantic texts. Second, even with the feasible prompting function, it is unaware how to design the informative prompt to better reformulate the downstream application. Using the example of node classification, the prompt should take into account the relative neighborhoods of target node to boost the classification accuracy. In view of these two challenges, we propose the research question: *how to design the graph-aware prompting function to bridge the pretext and downstream tasks in fine-tuning the pre-trained GNNs?*

Presented work. To mitigate the training objective gap, we present graph pre-training and prompt tuning (GPPT) framework.

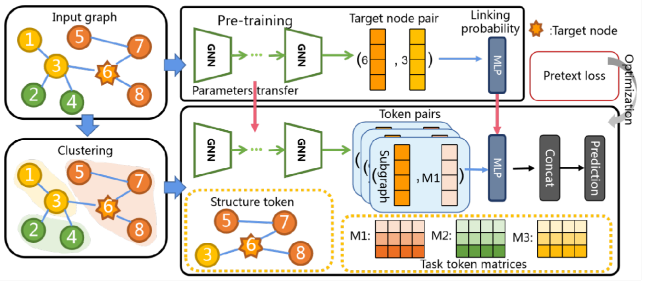

Figure 1: Overview of GPPT. The masked edge prediction is leveraged to pre-train GNNs. To bridge training objective gap, the graph prompting function modifies the standalone node into token pairs, and directly applies the pre-trained model without changing classification layer (i.e., MLP). The token pair contains task token (denoting node label) and structure token (describing node). The node classification is reformulated to measure the linking score of token pair. The task token with the highest score is decided as node label.

Notably, most of the graph pre-training learn the structure knowledge via edge prediction or contrastive learning to evaluate the similarity score of any instance pair, while the downstream application is node classification since the local node labels are scarce. Without loss of generality and following the previous efforts, we adopt the masked edge prediction, the most simplest and universal pretext task, to pre-train GNNs. Based on the pre-trained model, we propose the graph prompting function to reformulate the downstream node classification problem looking the same as edge prediction. As shown in Figure 1, the graph prompting function applies the pairwise token template to modify the standalone node into token pairs: the task token representing the downstream problem and the structure token containing the node information. Specifically, they respectively tackle the above two challenges as below:

First, to unify as edge prediction (challenge #1), the task token represents a node label waiting to be classified with trainable continuous vector, and is appended to the target sample. We measure the linking probability between task token and target sample directly through the pre-trained model. Given the multiple possible labels, the node classification is realized by choosing the task token/node label with the highest linking probability. Meanwhile, the orthogonal prompt initialization and regularization are proposed to separate the trainable vectors of different label tokens.

Second, to inform the node classification (challenge #2), the structure token represents the target sample with its multi-hop neighborhoods. The intuition is that nodes are prone to share the similar patterns with their proximal neighbors. The structure token would make it much easier to accurately classify a node by automatically selecting the relative neighbors.

Finally, we fine-tune the graph prompting function together with the pre-trained GNNs. We design extensive experiments on eight benchmark datasets and two downstream applications. The empirical results show that GPPT consistently outperforms all the other training strategies, including supervised learning, joint training and traditional transfer learning. In the few-shot graph analysis problem, GPPT delivers the average improvement of 4.29%, and saves the fine-tuning time cost up to 4.32X.

2 RELATED WORK

Graph Neural Networks. GNNs apply the recursive message passing across vertices to learn the complex dependencies in graph data [5–7, 21]. Recently, there has been a lot of researches that seek to improve learning the structural information from graph [3, 46, 47], which is known as graph representation learning. The main idea is to optimize the low-dimensional vector mapping so that the low-dimensional embeddings can effectively reflect the structure of the original graph. The learned embeddings can be used as input features for downstream tasks. There are various types of GNNs such as graph recurrent neural networks [23, 38], graph convolutional networks [4], graph autoencoders [26, 37], etc. As an effective technology of graph structured data mining, GNNs are an important foundation of our framework.

Graph Pre-training. Graph pre-training aims to capture several structural patterns across the input graph in a self-supervised manner. The goal is to pre-train a generic GNN that can deal with different tasks [24]. Recently, some representative pre-training models have made great progress in different aspects. Wu et al. [9] develop an effective strategy for pre-training GNNs, and demonstrate its effectiveness for out-of-distribution generalization. GCC [25] leverages contrastive learning to capture the universal network topological properties across multiple networks and transfers the learned prior knowledge to downstream tasks. GPT-GNN [10] introduces a self-supervised attributed graph generation task to pre-train GNN models that can capture the structural and semantic properties of the graph. L2P-GNN[22] utilizes meta-learning to learn the fine-tune strategy during the pre-training process.

Prompt-based Learning. Instead of fine-tuning the pre-trained language models (LMs), researchers gradually focus on a new NLP paradigm in recent years [20], namely "pre-train, prompt, and predict". This paradigm fine-tune LMs via task-specific objective functions on downstream tasks, which makes downstream tasks are reformulated to look more like those solved during the original LM training with the help of a textual prompt. One of the most distinctive and essential aspects of these prompt-based leaning NLP methods is prompt engineering, which means that choosing an appropriate prompt template has a great impact not only on accuracy, but also on the learning process of the model. Recent prompt engineering approaches can be summarized into two categories. Manual Template Engineering is a natural prompt engineering approach to create prompts, which is to manually create intuitive templates based on human introspection. For example, ZSC[20], PET-SGLUE [28] and Petal [27], which create manually crafted prefix prompts to handle a wide variety of downstream tasks. Automated Template Learning searches or learning tokens as prompt templates, such as discrete prompts[35] and continuous prompts[18, 30].

3 PRELIMINARY

In this section, we introduce the preliminary of graph neural networks and the transfer learning scheme of "pre-train, fine-tune".

3.1 Graph Neural Networks

Let tuple $\mathcal{G} = (X, A)$ denote the undirected graph, where $X \in \mathbb{R}^{N\times d}$ is the node features and $A \in \mathbb{R}^{N\times N}$ is the adjacency matrix. Note that N and d denote the number of nodes and feature dimensions, respectively. Each node v_i is described by feature vector $x_i \in \mathbb{R}^d$, which is given by the i-th row in matrix X. The graph link is denoted by $A_{ij} = 1$ if nodes v_i and v_j are connected; otherwise $A_{ij} = 0$.

Following the spatial message passing strategy [5], GNNs learn the node representation by recursively aggregating the representations of its neighbors. Formally, at the k-th layer, the representation learning of node v_i is:

$$x_i^{(k)} = \text{AGGREGATE}(\{x_j^{(k-1)}, v_j \in \mathcal{N}(v_i) \cup v_i\}, \theta^{(k)}). \quad (1)$$

$x_i^{(k)} \in \mathbb{R}^d$ is the embedding vector of node v_i learned at the k-th layer; and $x_i^{(0)}$ at the initial layer is given by x_i. $\mathcal{N}(v_i)$ is the set of first-order neighbors adjacent to node v; and $\theta^{(k)} \in \mathbb{R}^{d\times d}$ is the trainable weights. AGGREGATE denotes the aggregation function to pass the neighborhood embeddings to center node v_i, and then combines them (e.g., through sum, mean or max pooling) to generate the new node representation. Suppose the number of graph convolutional layers is K. To facilitate the following expression, we use $h_i = f_\theta(\mathcal{G}, v_i)$ to represent the final node representation learned from K-layer GNNs, where $\theta = \{\theta^{(1)}, \cdots, \theta^{(K)}\}$ denotes the concatenated trainable parameters. It has been widely demonstrated the optimized representation h_i delivers superior performance in many applications, such as edge prediction (e.g., item exploration in recommender system [41]) and node classification (e.g., topic categorization in scientific citation networks [12]).

3.2 Pre-train and Fine-tune GNNs

The supervised learning of node representation usually requires plenty of annotated data from each dedicated downstream task [24]. However, while it is resource expensive to manually tag the plentiful nodes and links in large-scale graphs, it is also inefficient to train GNNs from scratch for each downstream task. The transfer learning framework of "pre-train, fine-tune" has recently shown the potential to provide the attractive solution [10, 22, 25]. The core of pre-training GNNs is to use easily-accessible information to encode the intrinsic graph structure. The pre-trained GNNs could serve as initialization to generalize other downstream applications.

There are several efforts proposed to construct the self-supervised pretext tasks to pre-train GNNs. To encode the intrinsic structure knowledge, masked edge prediction [7] and contrastive learning [32] are the most effective and popular pretext tasks in literature. While the masked edge prediction computes the linking probabilities of node pairs to generate edges, the contrastive learning compares the similarity scores of graph instance pairs. Actually, the contrastive learning could be understood from the view of edge prediction, where the positive graph instance pair should be assigned with the highest similarity (i.e., linking probability). Therefore, without loss of generality and to ease the understanding of graph prompt, we use the masked edge prediction as pretext task to introduce the pre-training. Our work could be easily extended to any other pretext task which optimizes the pairwise score loss.

The edge prediction pretext task works as follows: we first randomly mask partial edges and then train GNNs to reconstruct them. Formally, let $\mathcal{G}^{\text{pre}} = (X, A^{\text{pre}})$ denote the masked graph, where a set of edges is randomly sampled and masked to be $A_{ij} = 0$. Therefore, the node representation learning of node v_i is given by $h_i = f_\theta(\mathcal{G}^{\text{pre}}, v_i)$. The pretext task is to decide whether a node pair is connected, where GNNs is pre-trained to optimize following loss:

$$\min_{\theta,\phi} \sum_{(v_i,v_j)} \mathcal{L}^{\text{pre}}(p_\phi^{\text{pre}}(h_i,h_j); g(v_i,v_j)). \tag{2}$$

Node pair (v_i, v_j) is given by the masked edges, or sampled from the negative unconnected pairs. p_ϕ^{pre} is the projection head with trainable parameters ϕ to evaluate similarity score of node pair (v_i, v_j), such as multiple layer perceptron (MLP) as shown in Figure 1. $\mathcal{L}^{\text{pre}}$ is the pretext loss function, such as cross entropy. $g(v_i, v_j)$ denotes the ground-truth label of node pair, which is indicator A_{ij} in our edge prediction pretext task. The optimized parameters θ^{pre} are expected to encode the graph connectivity structure, and provide a good initialization to be fine-tuned for the downstream tasks.

The downstream application in graph data is often defined by the local-level node classification [13], e.g., classifying node label or attribute, where the ground truth is hard to be obtained. Under the traditional "pre-train, fine-tune" transfer learning framework, GNNs are fine-tuned to optimize the following loss:

$$\begin{aligned} \min_{\theta,\varphi} \quad & \textstyle\sum_{v_i} \mathcal{L}^{\text{down}}(p_\varphi^{\text{down}}(h_i); g(v_i)), \\ \text{s.t.} \quad & \theta^{\text{init}} = \theta^{\text{pre}}. \end{aligned} \tag{3}$$

The constraint means GNNs are initialized by the optimized parameters from pretext task. $p_\varphi^{\text{down}}(h_i)$ denotes the new projection head accompanied with parameters φ, while the pretext projection head is discarded. $\mathcal{L}^{\text{down}}$ is the downstream loss function (e.g., cross entropy), and $g(v_i)$ denotes the ground-truth label of node v_i.

4 GRAPH PROMPTING FRAMEWORK

Due to the training objective gap existing between the pretext and downstream tasks, the pre-trained GNNs may fail to be efficiently leveraged by the new application, and even lead to negative transfer. First, we note that GNNs' parameters θ are optimized to generate close embeddings for connected node pairs, instead of nodes of the same class. If the disconnected pairs share the same class, the pre-trained model requires to be tuned with many epochs to adapted to the new problem. This time-consuming fine-tuning prevents us from efficiently using the pre-trained model. The pre-trained knowledge will also be gradually filtered out in the long tuning process. Second, considering parameters φ of the new projection head, they are hard to be coupled with the pre-trained GNNs at the initial stage. Therefore, the downstream projection head tends to conduct wrong classifications. In the experiment part, we empirically show that the vanilla transfer learning in Eq. (3) even results in poor results, comparing with the plain GNNs without any pre-training.

In this section, we propose to bridge the training objective gap with prompt, which reformulates the downstream task looking the same as pretext task. We are inspired by the successful applications of prompt tuning in NLP, where all the downstream applications are unified according to the pre-trained language model [20]. The details of NLP prompt tuning are in the related work. We define the new transfer learning framework of "pre-train, prompt, fine-tune" for GNNs, and tailor the prompting function to graph data.

4.1 Pre-train, Prompt, Fine-tune

Instead of changing the projection head and classifying the standalone nodes, the graph prompt-based learning framework modifies the input node to token pair, which is applied directly to the pre-trained model. We lay out the general mathematical definition of graph prompting function as below, and then give the concrete prompting function to design the token pair.

Definition 4.1 (Graph prompting function). A graph prompting function f_{prompt} is applied to change the standalone node into prompt: $v_i' = f_{\text{prompt}}(v_i)$. Prompt v_i' has the same input shape to the pretext projection head.

Definition 4.2 (Pairwise prompting function). Considering the edge prediction or contrastive learning pretext task, prompt v_i' is described by the template of token pairs: $v_i' = f_{\text{prompt}}(v_i) = [T_{\text{task}}(y), T_{\text{srt}}(v_i)]$. $T_{\text{task}}(y)$ is the task token to describe the downstream application, and $T_{\text{srt}}(v_i)$ is the structure token to represent target node v_i.

Rethinking the node classification downstream task, the task token $T_{\text{task}}(y)$ could be given by a node label waiting to be classified, while the structure token $T_{\text{srt}}(v_i)$ could be represented by the subgraph surrounding target node v_i to provide more structural information. Given the token pairs $[T_{\text{task}}(y), T_{\text{srt}}(v_i)]$, by embedding them into continuous tensors, one is able to conduct the classification task by fitting the linking probability between the two tokens. Our GPPT shown in Figure 1, the new learning paradigm of "pre-train, prompt, fine-tune" for GNNs, consists of three components:
Prompt addition. In this step, the graph prompting function generates a series of token pairs to be classified. Assuming that there are total C classes $[y_1, \cdots, y_C]$, we construct their corresponding token pairs $[T_{\text{task}}(y_c), T_{\text{srt}}(v_i)]$, for $c = 1, \cdots, C$.
Prompt answer. Given each token pair $[T_{\text{task}}(y_c), T_{\text{srt}}(v_i)]$, we embed them into continuous vectors. We then concatenate them as input to the pre-trained projection head, and obtain the linking probability. We answer and classify target node v_i with label y_c if it obtains the highest probability.
Prompt tuning. Following the pretext training objective, we optimize the following loss to fine-tune GNNs:

$$\min_{\theta,\phi} \sum_{(v_i,y_c)} \mathcal{L}^{\text{pre}}(p_\phi^{\text{pre}}(T_{\text{task}}(y_c), T_{\text{srt}}(v_i)); g(y_c, v_i)). \tag{4}$$

$g(y_c, v_i)$ denotes the ground truth connection between label class y_c and target node v_i. $g(y_c, v_i) = 1$ if node v_i belongs to class y_c; otherwise it is zero. Therefore, we have the same training objective (i.e., edge prediction) with the pretext task, and bridge the optimization objective gap. Below we introduce how to design the task and structure tokens tailored to graph data.

4.2 Graph Prompting Function Design

As defined in Definition 4.2, choosing an appropriate prompting function has great impacts on the learning processes of downstream task and expressive structure. We propose two simple but effective designs for the task and structure token generation. They could be easily replaced depending on the realistic applications on hand.

4.2.1 Task Token Generation. Motivated by the continuous token representation in NLP, the task token $T_{\text{task}}(y_c)$ is embedded into a trainable vector: $e_c = T_{\text{task}}(y_c) \in \mathbb{R}^d$. For the total C classes in the downstream node classification, the task token embeddings are defined by: $E = [e_1, \cdots, e_C]^\top \in \mathbb{R}^{C \times d}$. The task token could be understood as a class-prototype node added to the original graph, where the node classification is performed by querying every class-prototype node. The optimal embedding of task token $T_{\text{task}}(y_c)$ should be at the center of node embeddings of class y_c.

Existing prompt-based learning methods often design the common tokens shared by all training samples [28, 30]. However, it would be hard for the distinct nodes over graph to use and tune the single task token embeddings E. In real-world networks, one of the inherent graph characteristics is cluster structure, and each node belongs to one cluster. Nodes within the same cluster have the dense connections to each other, while they are sparsely linked to the nodes from other clusters. Given the edge prediction pretext task, the pre-trained node embeddings will also be clustered in the embedding space. As explained before, the optimal embedding of task token $T_{\text{task}}(y_c)$ should thus varies with clusters. To better conduct the node classification job at each cluster, we propose the cluster-based task token generation, which consists of three steps:

- First, as shown in Figure 1, we adopt the scalable clustering module (e.g., METIS [14]) to pre-process and split nodes into multiple non-overlapped clusters: $\{\mathcal{G}_1, \cdots, \mathcal{G}_M\}$, where M is the hyper-parameter of cluster number.
- Second, for each cluster m, we train an independent task token embeddings: $E^m = [e_1^m, \cdots, e_C^m]^\top \in \mathbb{R}^{C \times d}$.
- Third, given task token $T_{\text{task}}(y_c)$ of node v_i at cluster m, it is embeded by vector e_c^m.

4.2.2 Structure Token Generation. Instead of exclusively using target node v_i for the downstream classification, we apply structure token $T_{\text{str}}(v_i)$ to leverage the expressive neighborhood information. According to the social influence theory, the proximal nodes tend to possess the similar feature attributes and class patterns. One would be much easier to classify a node by taking into account the positively relative patterns, which also provide the redundant information to make the classification decision robust. Structure token $T_{\text{str}}(v_i)$ denotes the subgraph centered at node v_i. Herein we leverage the first-order adjacent nodes, the most simple subgraph, to formulate $T_{\text{str}}(v_i)$. We then embed structure token $T_{\text{str}}(v_i)$ to continuous vector as:

$$e_{v_i} = a_i * h_i + \sum_{v_j \in \mathcal{N}(v_i)} a_j * h_j. \tag{5}$$

a_i is the weight to aggregate the neighborhood representations, and is learned from attention function. Note that the prompting function in NLP [20] relies on the defined semantic template to inform the downstream task. Comparing with the textual language template, we define the structure token within the graph prompt by the informative neighbors of target nodes.

4.3 Prompt Initialization and Orthogonal Prompt Constraint

Regarding the task token embeddings $E^m = [e_1^m, \cdots, e_C^m]^\top$ at cluster m, the simplest initialization way is to use random initialization, and then train from scratch. However, such vanilla initialization may deteriorate the node classification at the initial training stage, since the optimal task token embeddings should be at the center of node representations. Conceptually, our prompt-based transfer learning paradigm uses the pre-trained GNNs as a good initialization. It follows that pre-trained node representations might serve as good initialization spots. Therefore, we initialize token embedding e_c^m by the mean of node representations, which are given by the training nodes of class y_c at cluster m. This means initialization provides the valid task tokens, and ensures the correct classification at the initial stage.

To conduct correct node classification, the prime condition is the task token embeddings of different classes should be irrelevant to each other. We utilize orthogonal constraint to keep the orthogonality of task token embeddings during model fine-tuning:

$$\mathcal{L}_o = \sum_m \|E^m (E^m)^\top - I\|_F^2. \tag{6}$$

4.4 Overall Learning Process

As mentioned in Section 4.1, the transfer learning framework of "pre-train, prompt, finee-tune" contains three steps, namely prompt addition, prompt answer, and prompt tuning. Based on the designs of graph prompting function and orthogonal prompt constraint, we reiterate the learning process. First, the prompt addition step modifies target node v_i with token pair $[T_{\text{task}}(y), T_{\text{srt}}(v_i)]$, and represents them with embeddings $[e_c^m, e_{v_i}]$. Second, the prompt answer evaluates a series of token pairs for the different classes, and classifies a node by task token accompanied with the highest linking probability. The prompt tuning optimizes the pre-trained GNNs and token embeddings as below:

$$\begin{array}{ll} \min_{\theta,\phi,E^1,\cdots,E^M} & \sum_{(v_i,y_c)} \mathcal{L}^{\text{pre}}(p_\phi^{\text{pre}}(e_c^m, e_{v_i}); g(y_c, v_i)) + \lambda \mathcal{L}_o, \\ \text{s.t.} & \theta^{\text{init}} = \theta^{\text{pre}}, \phi^{\text{init}} = \phi^{\text{pre}}. \end{array} \tag{7}$$

λ is the loss hyper-parameter. ϕ^{pre} is the pre-trained parameters of projection head according to Eq. (2). The pseudo algorithm is listed in Algorithm 1 of Appendix.

Why prompting in GNNs. Besides the benefit of bridging the task gap, the graph prompting function has more other advantages, including tuning efficiency, informative template for downstream applications, and easy deployment. (I) Tuning efficiency. Compared with the traditional transfer learning, the pre-trained GNNs and projection head are both reused without introducing the new projection head. The pre-trained knowledge could be elicited directly to accelerate the convergence in the downstream task. We provide extensive experiments below to demonstrate the efficiency of our GPPT. (II) Informative template. The graph prompting function provides the flexible template to include the informative signals for the downstream task. The task and structure token can be tailored to boost each specific case. (III) Easy deployment. As verified by the following studies, the pre-trained model shows promising results with only a few tuning epochs or even without fine-tuning. One only needs to carefully update the graph prompting function. This property is much preferred in the large-scale graph, where the training of GNNs requires costly computation and memory. We are able to fix GNNs once they are pre-trained, and tune prompt efficiently and scalablely similar to MLP for each new application.

5 EVALUATION

We experiment in the node classification task on benchmark datasets with the goal of answering the following research questions. **Q1:** Compared with supervised training, joint optimization and pre-training baselines, how effective is GPPT to boost the node classification? **Q2:** How efficient is the graph prompting function to adapt the pre-trained model for the downstream application with only a few epochs or even without fine-tuning? **Q3:** How does GPPT perform in the more difficult few-shot setting? **Q4:** How does the graph clustering affect the task token design and the following performance. **Q5:** How does the prompt initialization and regularization affect the prompt tuning? **Q6:** How does GPPT perform in other node classification task?

5.1 Experimental Setup

5.1.1 Dataset. We evaluate the proposed framework GPPT on eight popular benchmark datasets, including citation networks [16] (i.e, Cora, Citeseer, Pubmed), Reddit [7], CoraFull [1], Amazon-Co-Buy [29] (i.e., Computer and Photo), ogbn-arxiv [8].

5.1.2 Approaches. To evaluate the proposed "pre-train, prompt, fine-tune" learning framework of GPPT, we compare with state-of-the-art training schemes, including supervised learning, joint optimization, and pre-training. They are introduced below:

- **Supervised learning:** It uses the training nodes to learn GNNs, which are directly applied for model inference. We consider the following widely-studied GNN baselines, including GraphSAGE(GS) [7], GCN [16], and GAT [33].
- **Joint optimization:** By simply combining the self-supervised learning objective with downstream objective, the joint learning updates GNNs to optimize both tasks. GAE [15] uses graph autoencoder, and jointly optimizes the node classification loss with refactoring loss. We further apply GraphSAGE, the most popular model, as backbone and consider two joint optimization methods: EdgeMask [13] learning the masked edge prediction and DGI [34] leveraging the contrastive learning of deep graph infomax. They have been generally adopted to learn the graph structure. To separate from the following pre-training setting, we use Joint-Edge and Joint-DGI to denote the joint optimization.
- **Pre-training:** Following the transfer learning strategy of "pre-train, fine-tune", the pre-trained models are fine-tuned in the downstream applications. Similar to the joint optimization, we compare with Pre-Edge and Pre-DGI, where EdgeMask DGI are instead applied as pretext tasks, respectively.
- **Prompt-based tuning:** Our GPPT is realized by following the learning paradigm of "pre-train, prompt, fine-tune". Meanwhile, two simple versions are developed: GPPT (w/o C) and GPPT (w/o N). Specifically, GPPT (w/o C) removes the graph clustering and applies a single task token matrix; GPPT (w/o N) replaces the structure token with target node itself to ablate the neighborhoods during classification.

5.1.3 Implementations. Following the previous efforts and ensuring the fair comparison, we adopt the layer number of 2 and the hidden units of 128 for all GNNs. The Adam optimizer accompanied with weight decay of 5e-4 is applied at both the pre-training and fine-tuning stages. The learning rate is chosen from [0.001,0.005] depending on datasets. For our GPPT, we use the orthogonality prompt constraint coefficient λ of 0.01. The clustering numbers in Cora, Citeseer, Pubmed, CoraFull, Computer, Photo, ogbn-arxiv, Reddit are 7, 6, 3, 5, 10, 8, 10, 10, respectively.

5.2 Performance Analysis

5.2.1 Node Classification Studies. To answer research question **Q1**, we list the comparison between different categories of training methods in Table 1. We make the following major observations.

❶ *Prompt-based learning methods generally obtain the best performance on the benchmarks.* While most of joint optimization methods outperform the supervised learning by leveraging the self-supervised structure signals, the pre-training based methods tend to damage the downstream node classification. As analyzed before, there exists the significant training objective gap between pretext and downstream tasks. Under the paradigm of "pre-train, fine-tune", the pre-trained knowledge may be gradually filtered out during the long fine-tuning process. The pre-training thus fails to server as good initialization for the downstream application, and results in the comparable or even worse classification accuracies. For example, Pre-DGI has 3.55% dropping compared with supervised learning model GraphSAGE on Cora, which means the negative transfer. With the graph prompting function, GPPT directly bridges the task gap to guarantee the effectiveness of pre-trained model in the downstream applications. Comparing with all the other training schemes, the prompt-based learning shows the powerful capability to elicit the pre-trained graph knowledge. The average improvement over the baselines could be up to 3.19%.

❷ *The graph clustering and neighborhood structure are crucial for the informative prompt token designs.* By ablating any of them, it is observed that the generalization performance is generally decreased. The experimental results are in line with our motivations. First, since the node embeddings are pre-trained to memorize the cluster structure, the separating task tokens (which representing the label prototypes) should be adopted for different clusters to better conduct the node classification. The more studies of cluster number are provided in the following experiments. Second, the local neighborhood structure is informative for the node classification. The proximal nodes tend to possess the similar feature attributes and class patterns. One would be much easier to classify a node by taking into account the positively relative patterns, which also provide the redundant information to make the classification decision robust. In the future work, the more task and structure tokens need to be tailored for the specific downstream applications.

5.2.2 Fine-tuning Efficiency Analysis. Compared with other training strategies, the prompt could efficiently adapt the pre-trained model to downstream applications with only a few fine-tuning steps or even without fine-tuning. To answer **Q2**, we compare the training loss dynamics between GraphSAGE, Joint-Edge, Pre-Edge, and GPPT on Cora and Citeseer in Figure 2. The comparisons on other datasets are in Figure 5 of Appendix. Furthermore, given the pre-trained models in pre-training based methods and prompt-learning based approaches, we fix them and apply to the downstream tasks to compare their adaptability. The test accuracy is listed in Table 2. We make the following major observations.

Table 1: Node classification accuracy in percent. The best case is in bold. Imp(%) at the last column means the average improvement of GPPT over baseline.

Training schemes	Methods	Data								Imp(%)
		Cora	**Citeseer**	**Pubmed**	**CoraFull**	**Computer**	**Photo**	**ogbn-arxiv**	**Reddit**	
Supervised	GraphSAGE [7]	81.07±0.30	67.95±0.34	77.24±0.62	65.86±1.73	86.28±0.27	90.95±0.98	65.07±0.67	91.25±0.14	1.60
	GCN [16]	76.32±0.95	65.40±0.13	75.33±0.27	62.79±0.41	84.29±0.61	90.33±0.70	64.26±0.15	87.39±0.28	4.98
	GAT [33]	78.47±1.68	65.61±0.57	75.91±0.14	63.20±0.77	83.95±0.92	88.93±0.50	64.31±1.06	88.00±0.72	4.46
Joint	GAE [15]	78.87±1.22	66.23±0.77	76.37±0.41	65.16±1.30	73.30±0.29	85.29±0.65	65.62±0.44	91.16±1.29	5.63
	Joint-Edge [13]	80.59±1.13	69.11±1.28	78.99±0.52	66.40±0.28	**88.57**±0.69	89.79±0.44	65.51±0.17	91.60±0.16	0.79
	Joint-DGI [34]	78.36±0.38	68.47±0.82	76.66±0.11	64.93±0.57	85.09±0.14	90.38±0.14	65.60±0.28	91.22±0.75	2.40
Pre-train	Pre-Edge [13]	81.00±0.93	67.39±0.98	78.51±0.14	65.62±1.71	86.41±1.46	90.35±0.17	65.59±0.18	91.38±0.16	1.50
	Pre-DGI [34]	76.25±0.90	65.09±1.14	76.42±1.61	63.60±0.73	85.34±0.94	90.94±0.34	65.00±0.18	87.96±0.33	4.23
Prompt	GPPT(w/o C)	80.90±0.75	67.64±0.21	79.28±1.04	65.81±1.33	86.57±0.75	91.58±0.16	65.66±0.88	91.02±0.57	1.16
	GPPT(w/o N)	81.16±0.30	67.74±1.74	79.58±1.43	66.03±1.11	86.95±0.19	**92.26**±0.60	65.96±0.14	**92.51**±0.66	0.57
	GPPT	**81.40**±0.48	**69.21**±0.48	**79.73**±0.38	**66.91**±1.09	87.38±0.85	92.13±1.01	**65.94**±0.24	92.45±0.59	-

Table 2: Test accuracies (in percent) of pre-trained models without fine-tuning. The higher one has better adaptability.

Training schemes	Methods	Data								Imp(%)
		Cora	**Citeseer**	**Pubmed**	**CoraFull**	**Computer**	**Photo**	**ogbn-arxiv**	**Reddit**	
Pre-train	Pre-Edge	46.68±0.39	23.96±0.94	21.96±0.26	21.75±0.24	45.84±0.79	42.69±0.13	38.59±0.22	88.72±0.33	101.26
	Pre-DGI	5.96±0.34	16.95±0.85	40.75±0.18	5.76±0.13	37.26±0.99	22.30±0.25	9.16±0.48	65.39±0.11	396.85
Prompt	GPPT(fix)	**77.19**±0.31	**63.84**±0.45	**78.90**±0.18	**41.36**±0.44	**77.27**±0.25	**87.15**±0.14	**60.22**±0.67	**88.77**±0.42	-

Table 3: Test accuracy in percent of GPPT by ablating prompt initialization (init.) or orthogonal prompt constraint (cons.). The ↓/↑ arrow means the decreasing/improvement compared with the corresponding GPPT in Table 1.

Methods	**Cora**	**Citeseer**	**Pubmed**	**CoraFull**	**Computer**	**Photo**	**ogbn-arxiv**	**Reddit**
GPPT(w/o cons.)	80.81±0.14(↓)	67.73±0.90(↓)	78.72±0.26(↓)	65.11±0.67(↓)	86.82±0.59(↓)	91.71±0.98(↓)	64.82±0.76(↓)	92.10±0.32(↓)
GPPT(w/o init.)	79.30±0.10(↓)	67.01±0.08(↓)	77.82±0.08(↓)	66.15±0.07(↓)	87.07±0.12(↓)	91.70±0.04(↓)	66.21±0.22(↑)	92.53±0.15(↑)

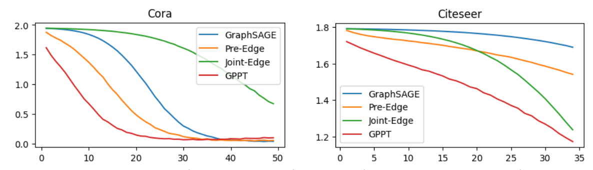

Figure 2: Training losses with epochs on Cora and Citeseer.

❸ *GPPT consistently shows the fastest convergence towards the optimal downstream points.* In general, GPPT only takes 23 fine-tuning epochs to optimize the downstream tasks, given the average convergence steps of 100 in other training strategies. GPPT speeds up the fine-tuning efficiency by up to 4.32X. This is because the graph prompting function in GPPT directly bridges the task gap, which enables the pre-trained GNNs and projection head could be applied to downstream task without any modification. Therefore, the pre-trained knowledge is directly elicited in the downstream application, which brings the high fine-tuning efficiency.

❹ *The prompt-based learning paradigm even enables the pre-trained GNNs to be transferred without any tuning, comparing with the traditional pre-training methods.* In Table 2, Pre-Edge and Pre-DGI are fine-tuned with one step to adapt the new node classification layer to the downstream task. In contrast, all the pre-trained parameters are fixed in GPPT since the node classification has already been reformulated to the pretext edge prediction. We only learn the graph prompting function, a small adaption module easily to be updated. It is observed that GPPT delivers the significant improvements over Pre-Edge and Pre-DGI. The pre-training based methods without careful fine-tuning have very poor performances, e.g., the only 5.96% accuracy of Pre-DGI on Cora. These observations positively validate the applicability and reliability of GPPT to real-world systems: Without conducting the message passing and training GNNs again, the pre-trained model (or node embeddings) could be applied directly to any downstream task by only tuning the prompt similar to MLP. Such efficiency is much preferred for the large-scale graph, where the training in the whole graph is time consuming and memory expensive.

5.2.3 Few-Shot Setting Analysis. The self-supervised pretext task provides unlabeled information to better tackle the few-shot learning problem. To answer research question **Q3**, we consider the more difficult few-shot node classification setting, by randomly masking a portion of training labels on the concerned datasets. The test accuracy obtained with 30% and 50% masking ratio is listed in Tables 5 and 6 in the Appendix. We make the following observations.

❺ *Supervised learning and joint optimizing methods have poor results in the few-shot setting.* The general supervised learning methods utilize empirical risk minimization. As the amount of labeled data decreases, the prior knowledge that the model can acquire will also decrease, which will lead to worse performance.

❻ *The prompt-based learning methods effectively elicit the pre-trained knowledge to improve few-shot learning.* Compared with the supervised learning and joint optimization, the pre-training approaches purely optimize the pre-text objective to capture the graph structure knowledge, without suffering from the masked training labels. The pre-trained model serves as good initialization to the following few-shot problem. Furthermore, compared with pre-training approaches, the prompt-based methods train the extra prompting function to better inform the downstream node classification, could thus reduce the need of costly annotated data. GPPT delivers the average improvements ranging from 0.57% to 7.90%.

5.2.4 The Crucial Role of Graph Clustering in Task Token. The appropriate choice of cluster number, i.e., hyperparameter M, is crucial to the task token designs and fine-tuning results. To answer **Q4**, we evaluate GPPT with hyperparameter range $[1, 10]$ on Cora, and list the results in the right of Figure 3. Furthermore, we visualize the node embeddings and prompt variables on Cora in the left of Figure 3. The experiments on other citation network datasets are in Figure 4 of Appendix. We make the following observations.

❼ *Without clustering, the graph prompting function has relatively poor result.* The cluster number of $M = 1$ means the graph is treated integrally without clustering, where all the nodes share the same set of task tokens. Due to the inherent clustering characteristic in graph data, the pretext task of edge prediction will assign the close embeddings for nodes within the same cluster. Meanwhile, the embedding distance between clusters is generally large. It is hard to tune the single task token of specific class, which should be close to node embeddings of the corresponding class but far away from the diverse clusters. Thus, GPPT without clustering consistently has worse node classification accuracy.

❽ *The proper hyperparameter M varies with datasets.* The adoption of hyperparameter M depends on the inherent cluster structure in the given graph data. The usage of M should minimize the cross-cluster connections, while maximizing the intra-cluster edges.

❾ *The task token embeddings are close to their corresponding nodes.* Using t-SNE, we visualize the high-dimensional embeddings obtained from one specific cluster on Cora. The colored circles denote nodes and the colored stars represent task tokens. The different colors are applied to indicate label classes. It is observed that task token and nodes of the same classes are tuned to be proximal to each other. This property demonstrates the reliability of using task token to perform the node classification, since there exists high linking probability between a node and its corresponding task token.

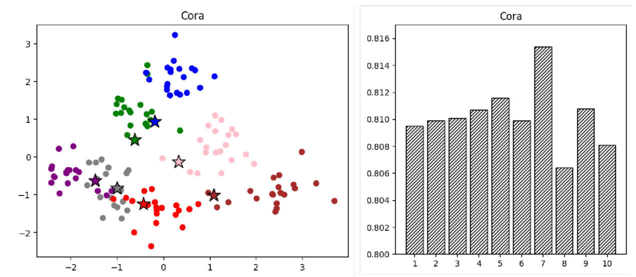

Figure 3: Left: Node and task token visualization on Cora. Right: Cluster number study on Cora.

5.2.5 Prompt Tuning Analysis. To study the impacts of prompt initialization and orthogonal regularization on the prompt tuning and to answer **Q5**, we respectively ablate these two components, and show the test accuracies in Table 3.

❿ *The proper prompt initialization and orthogonal regularization can effectively improve the performance of GPPT.* Compared with GPPT in Table 1, the ablation of any of them generally results in performance dropping. The dropping scale range from 0.3% to 3.2%. Since the optimal task token embeddings should be at the center of node representations, the vanilla initialization may deteriorate model at the initial training stage, and misleads the following training process. The orthogonal constrain in Eq. (6) is important to separate task tokens to accurately conduct the node classification.

5.2.6 Node Degree Classification. Besides the node label classification defined in the benchmark datasets, there are many other problems concerned in graph data, such as node degree prediction and pairwise node distance estimation [13]. To answer **Q6**, we adopt the node degree prediction as new downstream application, and test the generality of pre-trained models. Specifically, we randomly mask part of edges, and train GNNs to predict the ground-truth degrees of nodes. We list the comparison results on the three citation networks in Table 7 in the appendix. It is observed GPPT shows the powerful generality to the new application. The improvement obtained by GPPT is up to 46%. This empirical property greatly improve the applicability of pre-trained: By training on one pretext task, GPPT can easily adapt it to new application by only changing the prompting function.

6 CONCLUSION AND FUTURE WORK

We innovatively propose GPPT, the first transfer learning paradigm of "pre-train, prompt, fine-tune" for GNNs. The graph prompting function is developed to reformulate the downstream task looking similar to the pretext one, which aims to reduce their training objective gap. Moreover, we design the task and structure token generation methods, which are used to define node prompt for the node classification applications. The mean prompt initialization and orthogonal regularization are proposed to improve the prompt tuning. Extensive experiments demonstrate that GPPT consistently outperforms the traditional training paradigms on benchmark graphs, accompanied with improved tuning efficiency and better adaptability to the downstream tasks. In the future work, we will explore the graph prompting function in the more challenging knowledge graph, and try meta learning to improve prompt tuning.

7 ACKNOWLEDGMENTS

We express gratitude to the anonymous reviewers for their hard work and kind comments. This work is supported by the National Natural Science Foundation of China (No. 61872161, No. 61976102), the Foundation of the Major Project of Science and Technology Innovation 2030 - New Generation of Artificial Intelligence (No. 2021ZD0112500), the Interdisciplinary and integrated innovation of JLU (No. JLUXKJC2020207).

REFERENCES

[1] Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In 6th International Conference on Learning Representations, ICLR. OpenReview.net.
[2] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In International Conference on Machine Learning. PMLR, 1725–1735.
[3] Tianlong Chen, Kaixiong Zhou, Keyu Duan, Wenqing Zheng, Peihao Wang, Xia Hu, and Zhangyang Wang. 2022. Bag of Tricks for Training Deeper Graph Neural Networks: A Comprehensive Benchmark Study. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022).

[4] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-Scale Learnable Graph Convolutional Networks. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018. ACM, 1416–1424.
[5] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In International conference on machine learning. PMLR, 1263–1272.
[6] Kai Guo, Kaixiong Zhou, Xia Hu, Yu Li, Yi Chang, and Xin Wang. 2021. Orthogonal Graph Neural Networks. arXiv preprint arXiv:2109.11338 (2021).
[7] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems. 1024–1034.
[8] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. Advances in neural information processing systems 33, 22118–22133.
[9] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In 8th International Conference on Learning Representations, ICLR. OpenReview.net.
[10] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. GPT-GNN: Generative Pre-Training of Graph Neural Networks. In KDD ’20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. ACM, 1857–1867.
[11] Kexin Huang and Marinka Zitnik. 2020. Graph meta learning via local subgraphs. Advances in Neural Information Processing Systems 33 (2020), 5862–5874.
[12] Meng Jiang. 2021. Cross-Network Learning with Partially Aligned Graph Convolutional Networks. In KDD ’21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. ACM, 746–755.
[13] Wei Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zitao Liu, and Jiliang Tang. 2020. Self-supervised learning on graphs: Deep insights and new direction. arXiv preprint arXiv:2006.10141 (2020).
[14] George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on scientific Computing 20, 1 (1998), 359–392.
[15] Thomas N Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. NIPS Workshop on Bayesian Deep Learning (2016).
[16] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In 5th International Conference on Learning Representations, ICLR. OpenReview.net.
[17] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. arXiv preprint arXiv:2104.08691 (2021).
[18] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP. Association for Computational Linguistics, 4582–4597.
[19] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. arXiv preprint arXiv:2107.13586 (2021).
[20] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. CoRR abs/2107.13586 (2021).
[21] Zirui Liu, Kaixiong Zhou, Fan Yang, Li Li, Rui Chen, and Xia Hu. 2021. EXACT: Scalable graph neural networks training via extreme activation compression. In International Conference on Learning Representations.
[22] Yuanfu Lu, Xunqiang Jiang, Yuan Fang, and Chuan Shi. 2021. Learning to Pre-train Graph Neural Networks. In Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021. AAAI Press, 4276–4284.
[23] Yao Ma, Ziyi Guo, Zhaochun Ren, Jiliang Tang, and Dawei Yin. 2020. Streaming Graph Neural Networks. In Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020. ACM, 719–728.
[24] George Panagopoulos, Giannis Nikolentzos, and Michalis Vazirgiannis. 2021. Transfer Graph Neural Networks for Pandemic Forecasting. In Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021. AAAI Press, 4838–4845.
[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 KDD ’20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. ACM, 1150–1160.
[26] Arindam Sarkar, Nikhil Mehta, and Piyush Rai. 2020. Graph Representation Learning via Ladder Gamma Variational Autoencoders. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020. AAAI Press, 5604–5611.
[27] Timo Schick, Helmut Schmid, and Hinrich Schütze. 2020. Automatically Identifying Words That Can Serve as Labels for Few-Shot Text Classification. In Proceedings of the 28th International Conference on Computational Linguistics, COLING. International Committee on Computational Linguistics, 5569–5578.
[28] Timo Schick and Hinrich Schütze. 2021. It’s Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners. In Proceedings of the 2021 Conference of the NAACL-HLT. Association for Computational Linguistics, 2339–2352.
[29] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of Graph Neural Network Evaluation. Relational Representation Learning Workshop, NeurIPS 2018 (2018).
[30] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP. Association for Computational Linguistics, 4222–4235.
[31] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. arXiv preprint arXiv:2010.15980 (2020).
[32] Arjun Subramonian. 2021. MOTIF-Driven Contrastive Learning of Graph Representations. In Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI. AAAI Press, 15980–15981.
[33] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph Attention Networks. CoRR abs/1710.10903 (2017).
[34] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax. ICLR (Poster) 2, 3 (2019), 4.
[35] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal Adversarial Triggers for Attacking and Analyzing NLP. In Proceedings of the EMNLP-IJCNLP 2019. Association for Computational Linguistics, 2153–2162.
[36] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 839–848.
[37] Jie Wang, Jiye Liang, Kaixuan Yao, Jianqing Liang, and Dianhui Wang. 2022. Graph convolutional autoencoders with co-learning of graph structure and node attributes. Pattern Recognit. 121 (2022), 108215.
[38] Junshan Wang, Guojie Song, Yi Wu, and Liang Wang. 2020. Streaming Graph Neural Networks via Continual Learning. In CIKM ’20: The 29th ACM International Conference on Information and Knowledge Management. ACM, 1515–1524.
[39] Ruijie Wang, Yuchen Yan, Jialu Wang, Yuting Jia, Ye Zhang, Weinan Zhang, and Xinbing Wang. 2018. Acekg: A large-scale knowledge graph for academic data mining. In Proceedings of the 27th ACM international conference on information and knowledge management. 1487–1490.
[40] Zhengyang Wang, Meng Liu, Youzhi Luo, Zhao Xu, Yaochen Xie, Limei Wang, Lei Cai, Qi Qi, Zhuoning Yuan, Tianbao Yang, et al. 2022. Advanced graph and sequence neural networks for molecular property prediction and drug discovery. Bioinformatics 38, 9 (2022), 2579–2586.
[41] Ruohan Zhan, Konstantina Christakopoulou, Ya Le, Jayden Ooi, Martin Mladenov, Alex Beutel, Craig Boutilier, Ed Chi, and Minmin Chen. 2021. Towards Content Provider Aware Recommender Systems: A Simulation Study on the Interplay between User and Provider Utilities. In WWW ’21: The Web Conference. ACM / IW3C2, 3872–3883.
[42] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. Advances in neural information processing systems 31 (2018).
[43] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. 2020. Revisiting graph neural networks for link prediction. (2020).
[44] Fan Zhou, Chengtai Cao, Kunpeng Zhang, Goce Trajcevski, Ting Zhong, and Ji Geng. 2019. Meta-gnn: On few-shot node classification in graph meta-learning. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management. 2357–2360.
[45] Huachi Zhou, Qiaoyu Tan, Xiao Huang, Kaixiong Zhou, and Xiaoling Wang. 2021. Temporal Augmented Graph Neural Networks for Session-Based Recommendations. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. 1798–1802.
[46] Kaixiong Zhou, Xiao Huang, Yuening Li, Daochen Zha, Rui Chen, and Xia Hu. 2020. Towards deeper graph neural networks with differentiable group normalization. Advances in Neural Information Processing Systems 33 (2020), 4917–4928.
[47] Kaixiong Zhou, Xiao Huang, Daochen Zha, Rui Chen, Li Li, Soo-Hyun Choi, and Xia Hu. 2021. Dirichlet energy constrained learning for deep graph neural networks. Advances in Neural Information Processing Systems 34 (2021).
[48] Kaixiong Zhou, Qingquan Song, Xiao Huang, and Xia Hu. 2019. Auto-gnn: Neural architecture search of graph neural networks. arXiv preprint arXiv:1909.03184 (2019).
[49] Kaixiong Zhou, Qingquan Song, Xiao Huang, Daochen Zha, Na Zou, and Xia Hu. 2021. Multi-channel graph neural networks. In Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence. 1352–1358.

A DATASET STATISTICS

The dataset statistics for the concerned node classification are summarized in Table 4. We follow the official splitting of training/validation/testing for all the benchmark datasets. The datasets are public, and have been included in Torch Geometric.

Table 4: Statistics of datasets.

Dataset	#Nodes	#Edges	#Feature	#Labels
Cora	2,708	5,429	1433	7
Citeseer	3,327	4,732	3703	6
Pubmed	19,717	44,338	500	3
Reddit	232,965	11,606,919	602	41
CoraFull	19,793	126,84	126,84	70
Computer	13,752	491,722	767	10
Photo	7,650	238,163	745	8
ogbn-arxiv	169,343	1,166,243	128	40

B PSEUDO ALGORITHM

The pseudo algorithm of GPPT is given in Algorithm 1. Specifically, we use batch training to handle both of the small and large graphs. The batch sizes are 256, 256, 2048, 256, 4096, 2048, 4096, and 4096 for Cora, Citeseer, CoraFull, Pubmed, Ogbn-arxiv, AmazonCoBuy-Computer, AmazonCoBuyPhoto, and Reddit, respectively.

Algorithm 1 Training process of GPPT

```
Input: Input graph $\mathcal{G} = (X, A)$.
Output: The learned parameters $\theta, \phi$, task token embeddings $E^1, \cdots, E^m$.
1: Initialize GNN model $f_\theta$ with the pre-trained parameters $\theta'$.
2: Hyper-parameter: Cluster number $M$.
3: Split nodes into multiple non-overlapped clusters $\{\mathcal{G}_1, \cdots, \mathcal{G}_M\}$.
4: For each cluster $\mathcal{G}_m$ we train an independent task token embeddings $E^m = [e^m_1, \cdots, e^m_C]^\top \in \mathbb{R}^{C\times d}$.
5: Given task token $T_{\text{task}}(y_c)$ of node $v_i$ at cluster $m$, it is embeded by vector $e^m_c$
6: while not done do
7:     Sample bach of nodes $\mathcal{G}' \in \mathcal{G}$ to fin-tuning our model.
8:     for each node $n \in \mathcal{G}'$ do
9:         Construct the prompt $n' = f_{\text{prompt}}(n)$ to obtain token pair $[T_{\text{task}}(y_c), T_{\text{srt}}(n)]$, for $c = 1, \cdots, C$.
10:        We embed the token pair into continuous vectors.
11:    end for
12:    We then concatenate token pair vectors as input to the pre-trained projection head, and obtain the edge connection probability.
13:    Calculate cross entropy loss $\mathcal{L}^{\text{pre}}$ and orthogonality prompt constraint loss $\mathcal{L}_o$ by using Eq.(6).
14:    Update the parameters by minimizing the objective in Eq.(7).
15: end while
```

C HYPERPARAMETER STUDY OF CLUSTER NUMBER

The hyperparameter study reault on citation network datasets Citseer and Pubmed, which are shown in Figure 4. Similar to our previous analysis, without graph clustering via setting cluster number $M = 1$, the classification accuracy of GPPT is significantly damaged.

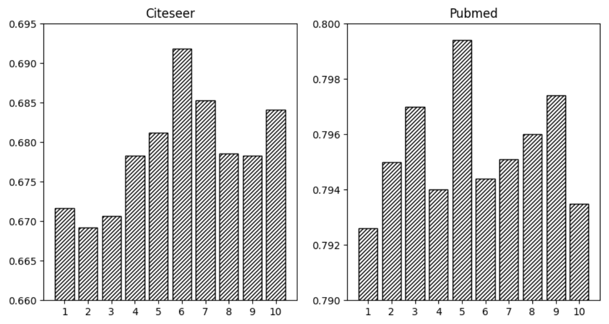

Figure 4: Left: Cluster number study on Citeseer. Right: Cluster number study on Pubmed.

D FINE-TUNING EFFICIENCY

The comparisons of the training loss dynamics between GraphSAGE, Joint-Edge, Pre-Edge, and GPPT on Pubmed, CoraFull, Computer, Photo, ogbn-arxiv and Reddit in Figure 5. It is observed GPPT consistently shows the most efficient convergence, compared with the traditional training paradigms.

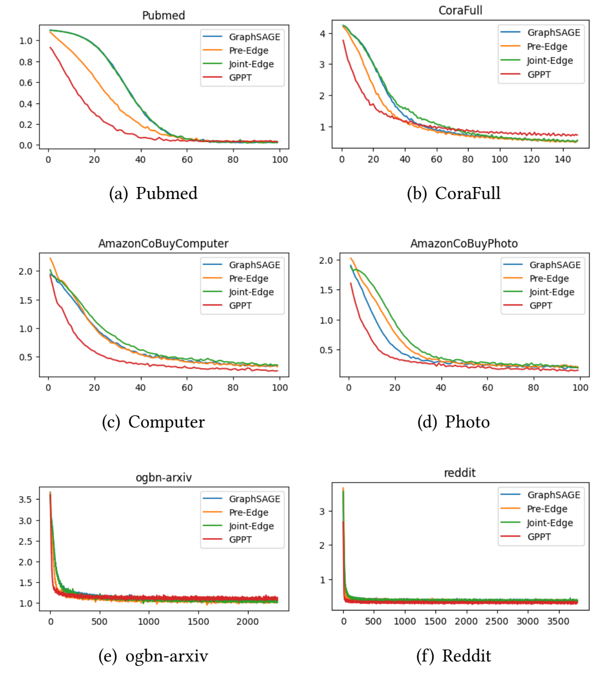

Figure 5: Training losses with epochs benchmark datasets.

Table 5: Test accuracy in percent in few-shot learning setting with masking ratio of 30%.

Training schemes	Methods	Data: **Cora**	**Citeseer**	**Pubmed**	**CoraFull**	**Computer**	**Photo**	**ogbn-arxiv**	**Reddit**	Imp(%)
Supervised	GraphSAGE	75.51±0.73	64.46±0.56	77.79±0.34	64.70±0.53	85.64±1.16	90.33±0.63	65.69±0.23	90.78±0.24	1.41
	GCN	74.04±0.30	56.05±0.56	78.12±0.64	60.48±0.59	84.58±0.39	90.47±0.98	63.18±0.38	87.47±0.24	5.50
	GAT	76.00±0.51	61.23±0.60	77.40±0.63	61.80±0.29	83.47±0.89	89.01±0.66	64.32±0.25	87.83±0.28	3.87
Joint	GAE	76.12±0.81	62.18±0.74	77.58±0.30	63.99±0.23	84.64±0.99	89.79±0.67	64.60±0.31	90.17±0.36	2.47
	Joint-Edge	76.34±0.44	63.14±0.64	78.58±0.19	62.59±0.95	85.24±0.98	89.16±0.40	66.01±0.21	88.16±0.10	2.38
	Joint-Infomax	76.32±0.56	62.30±0.91	77.76±0.33	63.41±0.96	85.28±0.64	89.96±0.81	65.68±0.34	91.18±0.42	2.03
Pre-train	Pre-Edge	76.32±0.32	63.71±0.46	79.30±0.48	64.37±0.84	85.98±0.47	90.49±1.19	65.99±0.69	91.00±0.50	1.09
	Pre-Infomax	72.03±1.16	61.41±0.93	78.10±0.54	61.53±0.33	85.33±0.35	90.56±0.92	65.02±0.63	88.15±0.79	3.78
Prompt	GPPT(w/o C)	76.28±0.44	64.00±0.70	79.68±0.15	65.07±0.89	85.93±0.78	90.80±0.15	64.69±0.42	90.98±0.24	1.06
	GPPT(w/o N)	76.11±0.44	63.27±0.32	79.60±0.68	64.89±0.41	87.12±0.62	91.10±0.86	**66.40**±0.30	**92.33**±0.21	0.55
	GPPT	**76.83**±0.26	**64.74**±0.17	**80.15**±0.34	**65.51**±0.24	**86.93**±0.12	**91.21**±0.65	66.18±0.33	92.25±0.35	-

Table 6: Test accuracy in percent in few-shot learning setting with masking ratio of 50%.

Training schemes	Methods	Data: **Cora**	**Citeseer**	**Pubmed**	**CoraFull**	**Computer**	**Photo**	**ogbn-arxiv**	**Reddit**	Imp(%)
Supervised	GraphSAGE	68.40±0.96	57.98±0.98	68.23±0.46	62.72±0.47	85.04±0.94	90.60±0.81	64.56±0.75	91.07±0.86	5.10
	GCN	71.78±0.50	52.15±0.27	65.05±0.15	61.17±0.53	83.54±0.45	89.92±0.28	64.19±0.59	87.50±0.29	7.90
	GAT	74.94±1.26	59.50±0.61	69.30±0.97	61.08±0.33	83.02±0.41	87.96±0.84	63.97±0.69	87.65±0.90	4.94
Joint	GAE	73.83±0.32	63.17±0.24	66.74±0.33	63.76±0.16	83.55±0.45	89.63±0.93	64.14±0.33	89.70±0.25	3.62
	Joint-Edge	74.75±0.87	62.76±1.15	69.03±0.58	62.72±0.72	84.18±0.61	89.19±0.71	65.14±0.58	88.00±0.22	3.32
	Joint-Infomax	73.74±0.32	63.37±0.28	68.30±0.24	62.55±0.88	84.33±0.58	89.54±0.55	65.01±0.50	90.91±0.96	3.08
Pre-train	Pre-Edge	76.38±0.89	65.49±0.90	71.29±0.66	62.79±0.66	85.33±0.89	90.04±0.69	64.86±0.67	90.91±0.36	1.41
	Pre-Infomax	68.91±0.89	63.40±0.89	68.98±0.36	60.18±0.49	84.24±0.97	89.83±1.18	64.56±0.94	87.59±0.73	4.91
Prompt	GPPT(w/o C)	76.84±0.68	64.79±1.03	71.78±0.32	63.43±0.36	86.08±0.83	90.30±0.43	65.38±0.82	90.81±0.92	1.02
	GPPT(w/o N)	76.96±0.45	65.63±0.52	71.00±0.50	63.38±0.75	**86.82**±0.68	90.80±0.45	65.47±0.52	**92.44**±0.23	0.57
	GPPT	**77.16**±1.35	**65.81**±0.97	**72.23**±1.22	**64.01**±0.73	86.80±0.26	**91.40**±0.92	**66.13**±0.44	92.13±0.57	-

Table 7: Test accuracy in percent for degree classification.

Training schemes	Methods	Data: **Cora**	**Citeseer**	**Pubmed**	Imp(%)
Super-vised	GraphSAGE	18.21±0.80	25.93±0.76	25.64±0.30	39.50
	GCN	13.64±0.58	33.42±0.44	23.23±0.86	46.08
	GAT	15.32±0.27	29.14±0.14	15.85±0.97	76.18
Joint	GAE	18.12±0.21	**37.41**±0.65	24.27±0.77	28.19
	Joint-Edge	14.91±0.79	29.99±0.86	35.12±0.44	24.97
	Joint-DGI	17.98±0.94	31.50±0.81	38.86±0.98	11.86
Pre-train	Pre-Edge	15.25±0.35	29.33±0.57	26.77±0.21	38.09
	Pre-DGI	15.03±0.75	28.87±0.57	27.21±0.48	38.47
Prompt	GPPT(w/o C)	15.39±0.23	30.12±0.74	27.84±0.13	34.49
	GPPT(w/o N)	15.54±0.75	36.97±0.23	**45.04**±0.62	6.19
	GPPT	**18.53**±0.43	37.34±0.15	44.29±0.42	-

E FEW-SHOT LEARNING WITH MASKING

The test accuracy obtained with 30% and 50% masking ratio is listed in Table 5 and Table 6. Similar to the masking ratio of 50%, GPPT consistently delivers the most outperforming results.

F NODE DEGREE PREDICTION

We adopt the node degree prediction as new downstream application, and test the generality of pre-trained models. We list the comparison results on the three citation networks in Table 7.

G ENVIRONMENT

All the experiments are implemented with PyTorch and graph package of Torch Geometric. They are tested on the machine with one NVIDIA3090 GPU.