Edge Prompt Tuning for Graph Neural Networks

Xingbo Fu
University of Virginia
Charlottesville, Virginia, USA
xf3av@virginia.edu

Yinhan He
University of Virginia
Charlottesville, Virginia, USA
nee7ne@virginia.edu

Jundong Li
University of Virginia
Charlottesville, Virginia, USA
jundong@virginia.edu

Abstract

Pre-training powerful Graph Neural Networks (GNNs) with unlabeled graph data in a self-supervised manner has emerged as a prominent technique in recent years. However, inevitable objective gaps often exist between pre-training and downstream tasks. To bridge this gap, graph prompt tuning techniques design and learn graph prompts by manipulating input graphs or reframing downstream tasks as pre-training tasks without fine-tuning the pre-trained GNN models. While recent graph prompt tuning methods have proven effective in adapting pre-trained GNN models for downstream tasks, they overlook the crucial role of edges in graph prompt design, which can significantly affect the quality of graph representations for downstream tasks. In this study, we propose EdgePrompt, a simple yet effective graph prompt tuning method from the perspective of edges. Unlike previous studies that design prompt vectors on node features, EdgePrompt manipulates input graphs by learning additional prompt vectors for edges and incorporates the edge prompts through message passing in the pre-trained GNN models to better embed graph structural information for downstream tasks. Extensive experiments on three graph datasets under two pre-training strategies demonstrate the superiority of our proposed method against four baselines.

CCS Concepts

• **Computing methodologies** → **Machine learning**.

Keywords

Graph neural networks, prompt tuning

ACM Reference Format:

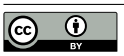
Xingbo Fu, Yinhan He, and Jundong Li. 2025. Edge Prompt Tuning for Graph Neural Networks. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Graph Neural Networks (GNNs) [1, 4, 7, 14, 20] are often trained for specific downstream tasks in an end-to-end manner. Nevertheless, the end-to-end manner for training powerful GNN models usually encounters significant challenges in practical deployments [2, 6, 8, 11]. First, annotating a sufficient number of labels for graph data is typically time-consuming and resource-intensive in the real world. Second, well-trained GNN models cannot be well generalized to other tasks, even on the same graph data [18]. To grapple with these critical challenges, applying pre-training techniques on graph data has become increasingly prevalent. Numerous recent studies have focused on designing effective pre-training strategies for training powerful GNN models without using any label information from downstream tasks [5, 6, 15–17, 19, 23]. The philosophy behind these pre-training strategies is to first train a GNN model on pre-training tasks via self-supervised learning and subsequently transfer the pre-trained GNN model to specific downstream tasks. Generally, there exists inevitable objective gaps between pre-training and the downstream tasks. To bridge the objective gap between pre-training and downstream tasks, one prevalent solution to adapt the pre-trained GNN model for downstream tasks is graph prompt tuning. Typically, graph prompt tuning keeps the pre-trained GNN model frozen and trains graph prompts for downstream tasks [2, 8, 9, 11, 12, 24, 25].

While recent graph prompt tuning methods show great prowess in adapting pre-trained GNN models for various downstream tasks, the existing methods still have several fundamental limitations. First, a few studies [11, 26] design graph prompt tuning methods based on specific pre-training strategies, which hinders their application to off-the-shelf pre-trained GNN models. Second, the important dependency information carried by graph structures is ignored in the existing studies [2, 8, 12]. They focus on designing and learning graph prompts primarily by applying them to node features or node representations. In this scenario, graph prompts are unable to enhance pre-trained GNN models in capturing complex graph structural information for downstream tasks.

Although the significant role of edges in graph learning has been amplified by many studies [3, 10, 13, 21], unfortunately, none of the existing studies have exploited edges for graph prompt tuning. Naturally, we may ask a question: *how can we devise an edge-level graph prompt tuning method to effectively enhance the performance of a pre-trained GNN model for downstream tasks?* In this study, we aim to answer this question through a pioneering investigation into designing edge prompts for downstream tasks. To address the above issues, we propose a novel graph prompt tuning method named EdgePrompt purely from the perspective of edges, fundamentally differing from node-level prompt designs in the existing studies [2, 12]. The intuition of EdgePrompt is to manipulate the input graph by adding extra learnable prompt vectors to edges and thereby enhance the capability of pre-trained GNN models for downstream tasks. In EdgePrompt, all the edges in the input graph learn a shared prompt vector at each layer of the pre-trained GNN model. The edge prompts will be aggregated along with node representations during the forward pass of the message-passing mechanism. We conduct extensive experiments over three graph datasets under two pre-training strategies. The results validate the superiority of our proposed method compared with four baselines.



KDD '25, Toronto, ON, Canada

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 Preliminaries

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph where $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$ is the set of N nodes, and $\mathcal{E}$ is the edge set. $\mathbf{X} \in \mathbb{R}^{N \times D}$ denotes the node feature matrix where the i-th row $\mathbf{x}_i$ represents a D-dimensional feature vector of node $v_i \in \mathcal{V}$. The edges in $\mathcal{G}$ can also be represented by an adjacency matrix $\mathbf{A} \in \{0, 1\}^{N \times N}$ where each entry $a_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$, otherwise $a_{ij} = 0$. Specifically, GNN models aim to learn expressive node representations through the message-passing mechanism [4, 7, 14, 20] where the representation of a target node is iteratively updated by aggregating the representations of its neighboring nodes. Typically, a GNN model has two fundamental operators: AGG$(\cdot)$ extracting the neighboring information of the node, and COMB$(\cdot)$ integrating the previous representation of the node and its neighbors. Mathematically, the l-th layer of an L-layer GNN model f updates the representation of node $v_i \in \mathcal{V}$ by

$$\mathbf{h}_i^{(l)} = \text{COMB}^{(l)}(\mathbf{h}_i^{(l-1)}, \text{AGG}^{(l)}(\{\mathbf{h}_j^{(l-1)} : v_j \in \mathcal{N}(v_i)\})), \quad (1)$$

where $\mathbf{h}_i^{(l)} \in \mathbb{R}^{D_l}$ denotes the D_l-dimensional representation of node v_i at the l-th layer, and $\mathcal{N}(v_i)$ denotes the neighbors of node v_i. $\mathbf{h}_i^{(0)} \in \mathbb{R}^D$ is initialized with node v_i's feature $\mathbf{x}_i$. The final node representation $\mathbf{h}_i^{(L)}$ after the L-th layer of the GNN model can be subsequently used for various downstream tasks (e.g., node classification) with a trainable classifier g.

3 Methodology

3.1 Problem Setting

We consider a GNN model pre-trained by a pre-training task. We aim to adapt the pre-trained GNN model to a downstream task on a graph dataset through graph prompt tuning while keeping its parameters frozen. Specifically, given a pre-trained GNN model f, the goal is to transform the input graph $\mathcal{G}$ to a prompted graph $\mathcal{G}' = \mathcal{T}(\mathcal{G})$ with learnable prompts and obtain expressive node representations on $\mathcal{G}'$ by f for a specific downstream task. Here, $\mathcal{T}$ is a graph transformation to obtain $\mathcal{G}'$ by adding prompts to $\mathcal{G}$. The key problem in graph prompt tuning is to design and learn suitable graph prompts to benefit downstream tasks.

3.2 Edge Prompt Design

Considering the dependencies between nodes in graph data, we design learnable prompt vectors on edges and manipulate the input graph to a prompted one with the edge prompts; therefore, the pre-trained GNN model can generate expressive node representations on the prompted graph for the downstream task. More concretely, for each edge $(v_i, v_j) \in \mathcal{E}$, we aim to learn a prompt vector $\mathbf{e}_{ij}^{(l)} \in \mathbb{R}^{D_{l-1}}$ on it at the l-th layer of the pre-trained GNN model. Typically, this prompt vector can be regarded as the learnable properties of edges. As discussed previously, one critical challenge arises here: many popular GNN models do not accommodate edge attributes during the message-passing mechanism. Therefore, they are unable to absorb $\mathbf{e}_{ij}^{(l)}$ into node representations. To overcome this issue, we propose to aggregate the prompt vector along with node representations through the message-passing mechanism during the forward pass at each layer of the pre-trained GNN model. Specifically, to compute $\mathbf{h}_i^{(l)}$ of each node v_i at the l-th layer, the GNN model will aggregate not only $\mathbf{h}_j^{(l-1)}$ from its neighboring node $v_j \in \mathcal{N}(v_i)$ but also $\mathbf{e}_{ij}^{(l)}$ associated with edge (v_i, v_j). Mathematically, we can reformulate Equation (1) with the edge prompt vector at the l-th layer of the pre-trained GNN model by

$$\mathbf{h}_i^{(l)} = \text{COMB}^{(l)}(\mathbf{h}_i^{(l-1)}, \text{AGG}^{(l)}(\{\mathbf{h}_j^{(l-1)} + \mathbf{e}_{ij}^{(l)} : v_j \in \mathcal{N}(v_i)\})). \quad (2)$$

To obtain the prompt vector, one simple yet effective way is to learn a global prompt vector shared by all the edges. Let $\mathbf{p}^{(l)} \in \mathbb{R}^{D_{l-1}}$ denote the global prompt vector at the l-th layer of the pre-trained GNN model. For each edge (v_i, v_j), the prompt vector at the l-th layer will be $\mathbf{e}_{ij}^{(l)} = \mathbf{p}^{(l)}$.

With the learnable edge prompts, we can obtain more suitable node representations $\mathbf{h}_i^{(L)}$ for node $v_i \in \mathcal{V}$ by the pre-trained GNN model for node classification. Given the labeled node set $\mathcal{V}_L \in \mathcal{V}$, we optimize our edge prompts and a classifier g by

$$\min_{g, \{\mathcal{P}^{(1)}, \cdots, \mathcal{P}^{(L)}, \mathbf{W}^{(1)}, \cdots, \mathbf{W}^{(L)}\}} \frac{1}{|\mathcal{V}_L|} \sum_{v_i \in \mathcal{V}_L} \ell(g(f(\mathcal{G}')_i), y_i), \quad (3)$$

where y_i is the ground-truth label of node $v_i \in \mathcal{V}_L$, and ℓ is the downstream task loss, i.e., the cross-entropy loss for classification.

Table 1: Accuracy on 5-shot node classification tasks.

Pre-training	Tuning	Cora	CiteSeer	Pubmed
GraphCL	Classifier Only	$53.05_{\pm 4.76}$	$38.62_{\pm 3.43}$	$64.28_{\pm 4.51}$
	GPPT	$50.96_{\pm 6.67}$	$39.50_{\pm 1.67}$	$60.47_{\pm 4.75}$
	GraphPrompt	$55.71_{\pm 4.62}$	$40.81_{\pm 2.11}$	$63.47_{\pm 2.23}$
	ALL-in-one	$38.00_{\pm 4.17}$	$40.27_{\pm 2.09}$	$58.61_{\pm 3.49}$
	EdgePrompt	$58.60_{\pm 4.46}$	$43.31_{\pm 3.23}$	$67.76_{\pm 3.01}$
SimGRACE	Classifier Only	$52.27_{\pm 2.74}$	$40.45_{\pm 3.55}$	$56.72_{\pm 3.80}$
	GPPT	$52.07_{\pm 7.65}$	$40.25_{\pm 3.29}$	$58.65_{\pm 5.12}$
	GraphPrompt	$51.42_{\pm 2.80}$	$41.74_{\pm 2.22}$	$55.98_{\pm 2.94}$
	ALL-in-one	$34.64_{\pm 4.06}$	$38.95_{\pm 2.35}$	$54.18_{\pm 4.70}$
	EdgePrompt	$58.37_{\pm 4.51}$	$43.94_{\pm 4.15}$	$61.10_{\pm 3.69}$

4 Experiments

4.1 Experimental Setup

We evaluate the effectiveness of our proposed method on node classification over three public graph datasets, including Cora [22], CiteSeer [22], and Pubmed [22]. We consider two pre-training strategies in our experiments, i.e., GraphCL [23] and SimGRACE [19]. We evaluate our proposed method against three state-of-the-art graph prompt tuning methods in our experiments, including GPPT [11], GraphPrompt [8], and All-in-one [12]. In addition, we also report the performance of solely training classifiers without any prompts (named as *Classifier Only*) in our experiments. We use a 2-layer GCN [7] as the backbone for node classification. We use the 5-shot setting for node classification.

4.2 Results

Table 1 reports the results of our method and four baselines on 5-shot node classification tasks over three datasets. Our method can consistently achieve the best performance among graph prompt tuning methods across different pre-training strategies.

References

[1] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *International conference on machine learning*.

[2] Taoran Fang, Yunchao Zhang, Yang Yang, Chunping Wang, and Lei Chen. 2023. Universal prompt tuning for graph neural networks. *Advances in Neural Information Processing Systems* (2023).

[3] Liyu Gong and Qiang Cheng. 2019. Exploiting edge features for graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.

[4] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*.

[5] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[6] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*.

[7] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

[8] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. 2023. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM Web Conference 2023*.

[9] Yihong Ma, Ning Yan, Jiayu Li, Masood Mortazavi, and Nitesh V Chawla. 2024. Hetgpt: Harnessing the power of prompt tuning in pre-trained heterogeneous graph neural networks. In *Proceedings of the ACM on Web Conference 2024*.

[10] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *15th International Conference on Extended Semantic Web Conference*.

[11] 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*.

[12] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. 2023. All in one: Multi-task prompting for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[13] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*.

[14] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

[15] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax. In *International Conference on Learning Representations*.

[16] Guancheng Wan, Yijun Tian, Wenke Huang, Nitesh V Chawla, and Mang Ye. 2024. S3GCL: Spectral, Swift, Spatial Graph Contrastive Learning. In *Forty-first International Conference on Machine Learning*.

[17] Zehong Wang, Donghua Yu, Shigen Shen, Shichao Zhang, Huawen Liu, Shuang Yao, and Maozu Guo. 2024. Select Your Own Counterparts: Self-Supervised Graph Contrastive Learning With Positive Sampling. *IEEE Transactions on Neural Networks and Learning Systems* (2024).

[18] Zehong Wang, Zheyuan Zhang, Nitesh V Chawla, Chuxu Zhang, and Yanfang Ye. 2024. GFT: Graph Foundation Model with Transferable Tree Vocabulary. *Advances in neural information processing systems* (2024).

[19] Jun Xia, Lirong Wu, Jintao Chen, Bozhen Hu, and Stan Z Li. 2022. Simgrace: A simple framework for graph contrastive learning without data augmentation. In *Proceedings of the ACM Web Conference 2022*.

[20] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*.

[21] Yulei Yang and Dongsheng Li. 2020. Nenn: Incorporate node and edge features in graph neural networks. In *Asian conference on machine learning*.

[22] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*.

[23] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in neural information processing systems* (2020).

[24] Xingtong Yu, Yuan Fang, Zemin Liu, and Xinming Zhang. 2024. Hgprompt: Bridging homogeneous and heterogeneous graphs for few-shot prompt learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[25] Xingtong Yu, Zhenghao Liu, Yuan Fang, Zemin Liu, Sihong Chen, and Xinming Zhang. 2024. Generalized graph prompt: Toward a unification of pre-training and downstream tasks on graphs. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[26] Xingtong Yu, Chang Zhou, Yuan Fang, and Xinming Zhang. 2024. MultiGPrompt for multi-task pre-training and prompting on graphs. In *Proceedings of the ACM on Web Conference 2024*.