GraphTOP: Graph Topology-Oriented Prompting for Graph Neural Networks

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

Zhenyu Lei
University of Virginia
Charlottesville, VA, USA
vjd5zr@virginia.edu

Zihan Chen
University of Virginia
Charlottesville, VA, USA
brf3rx@virginia.edu

Binchi Zhang
University of Virginia
Charlottesville, VA, USA
epb6gw@virginia.edu

Chuxu Zhang
University of Connecticut
Storrs, CT, USA
chuxu.zhang@uconn.edu

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

Abstract

Graph Neural Networks (GNNs) have revolutionized the field of graph learning by learning expressive graph representations from massive graph data. As a common pattern to train powerful GNNs, the "pre-training, adaptation" scheme first pre-trains GNNs over unlabeled graph data and subsequently adapts them to specific downstream tasks. In the adaptation phase, graph prompting is an effective strategy that modifies input graph data with learnable prompts while keeping pre-trained GNN models frozen. Typically, existing graph prompting studies mainly focus on *feature-oriented* methods that apply graph prompts to node features or hidden representations. However, these studies often achieve suboptimal performance, as they consistently overlook the potential of *topology-oriented* prompting, which adapts pre-trained GNNs by modifying the graph topology. In this study, we conduct a pioneering investigation of graph prompting in terms of graph topology. We propose the first **Graph T**opology-**O**riented **P**rompting (GraphTOP) framework to effectively adapt pre-trained GNN models for downstream tasks. More specifically, we reformulate topology-oriented prompting as an edge rewiring problem within multi-hop local subgraphs and relax it into the continuous probability space through reparameterization while ensuring tight relaxation and preserving graph sparsity. Extensive experiments on five graph datasets under four pre-training strategies demonstrate that our proposed GraphTOP outshines six baselines on multiple node classification datasets. Our code is available at `https://github.com/xbfu/GraphTOP`.

1 Introduction

Graphs are ubiquitous in a wide range of real-world scenarios, such as social networks [43, 61], knowledge graphs [41], traffic networks [32], and healthcare [3, 7]. To gain deep insights from tremendous graph data, numerous graph learning models have been developed in recent years. Among these efforts, Graph Neural Networks (GNNs) [1, 11, 23, 33, 38, 42, 46] are a prevalent tool for modeling graph data and have shown great prowess in different graph-related downstream tasks, including node classification [28, 56], link prediction [16, 62], and graph classification [25, 63]. Traditionally, GNN models are trained in a supervised manner. However, the supervised manner relies heavily on sufficient labeled graph data, which may be infeasible in the real world. Furthermore, the trained GNN models cannot be well generalized to other downstream tasks, even on the same graph data. These two critical issues hinder further deployments of GNN models in practice.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

To address the above issues, the "pre-training, adaptation" scheme has been widely adopted by a cornucopia of studies [5, 15, 27, 35, 36, 52, 57, 8]. Typically, they first train GNN models on pre-training tasks in an unsupervised manner, followed by adapting the pre-trained GNN models to specific downstream tasks. For instance, a GNN model can be pre-trained via link prediction and later adapted for node classification as the downstream task. During the adaptation phase, the goal is to bridge the objective gap between pre-training and downstream tasks. Inspired by recent prompt tuning approaches in natural language processing [21, 58] and computer vision [18, 48], graph prompting [9] has become an effective adaptation strategy to adapt pre-trained GNN models for downstream tasks by modifying the input graph data with trainable prompt vectors while keeping the pre-trained GNN models frozen. Generally, the existing graph prompting methods are *feature-oriented* — they design and learn graph prompts mainly by applying them to node features [5, 36] or hidden representations [27, 51, 52].

While the feature-oriented design is intuitive in graph prompting methods, they mostly overlook graph topology — another essential component in graphs that fundamentally distinguishes graph data from image data and text data — when designing graph prompts. Notably, graph representations are not only dependent on feature information but also determined by graph topology. Numerous efforts have demonstrated the significant impact of graph structures on graph-related tasks, particularly node classification [6, 20, 26, 37, 55]. Unfortunately, the potential of *topology-oriented* prompting for pre-trained GNN models remains unexplored in existing studies. Therefore, it is natural to pose the question: *How should we design a topology-oriented prompting framework to effectively adapt a pre-trained GNN model for downstream tasks?*

To answer this question, we conduct the pioneering investigation of graph prompting in terms of graph topology. We propose GraphTOP — the first **Graph T**opology-**O**riented **P**rompting framework to adapt pre-trained GNN models for downstream tasks, particularly for node classification. In GraphTOP, we formulate topology-oriented prompting as an edge rewiring problem and relax it into the continuous probability space via reparameterization. To ensure computational feasibility in practice, we propose subgraph-constrained topology-oriented prompting by restricting edge rewiring between each target node and other nodes within its multi-hop subgraphs. In the end, we design the optimization objective of GraphTOP to ensure tight relaxation and maintain graph sparsity. Our theoretical analysis indicates that topology-oriented prompting can effectively enhance the pre-trained GNN models for node classification. We conduct comprehensive experiments over five datasets under four pre-training strategies to evaluate the performance of our proposed method. The experimental results validate the superiority of GraphTOP against six baselines.

We summarize the main contributions as follows:

- **Problem formulation.** We formulate and conduct an initial investigation of graph prompting from the perspective of graph topology.
- **Algorithmic design.** We propose GraphTOP, the first topology-oriented prompting framework to adapt pre-trained GNN models by modifying graph topology of the input graph.
- **Theoretical analysis.** We provide theoretical analysis to support that our design of topology-oriented prompting in GraphTOP can effectively enhance pre-trained GNN models for node classification.
- **Experimental evaluation.** We conduct comprehensive experiments on five public datasets under four pre-training strategies. Experimental results validate the effectiveness of our proposed GraphTOP against six baselines for node classification.

2 Related works

2.1 Graph pre-training

Graph pre-training aims to train powerful GNN models over unlabeled graph data in a self-supervised fashion [44]. Generally, graph pre-training methods can be roughly categorized into two types: contrast-based methods and generation-based methods. Contrast-based methods [34, 39, 45, 49, 54] are developed based on the concept of mutual information (MI) maximization, where the estimated MI between different views of similar objects is maximized. For example, DGI [39] maximizes MI between the local and global instance views. SimGRACE [45] constructs contrastive views from

the perturbed version of GNN models. Generation-based methods [12, 13, 14, 19, 27, 33, 35] focus on reconstructing specific information from graph data, such as graph structure and node features. For instance, GraphMAE [12] pre-trains GNN models by reconstructing masked node features. Meanwhile, previous graph prompting studies [27, 35] also adopt link prediction as the pre-training strategy.

2.2 Graph prompting

Graph prompting adapts pre-trained GNN models to close the objective gap between pre-training and downstream tasks. The intuition is to modify the input graph with learnable prompt vectors for downstream tasks without tuning the pre-trained GNN models. For example, GPPT [35] pre-trains a GNN model via link prediction and adapts it for node classification as the downstream task. It bridges the objective gap between link prediction and node classification by converting node classification to link prediction. GraphPrompt [27] and its variant GraphPrompt+ [50] design prompt vectors as feature weights and apply them to node (hidden) representations. GPF-plus [5] mainly focuses on graph classification as the downstream task and learns to manipulate the input graph by adding extra learnable prompt vectors to node features. All-in-one [36] unifies various downstream tasks as graph-level tasks and similarly designs prompt vectors to modify node features. MultiGPrompt [52] instead inserts prompt vectors into node representations at each hidden layer. ProNoG [51] investigates prompt design for non-homophilic graphs and learns prompt vectors as feature weights based on subgraph representations. While the above graph prompting methods have demonstrated remarkable performance in adapting pre-trained GNN models, they are largely feature-oriented and consistently overlook the potential of topology-oriented prompting design.

3 Preliminaries

3.1 Graph neural networks

An attributed graph can be denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ is the set of n nodes and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the edge set. It can also be represented as $\mathcal{G} = (\mathbf{A}, \mathbf{X})$. Here, $\mathbf{A} \in \{0,1\}^{n \times n}$ denotes the adjacency matrix where $a_{ij} = 1$ iff $(v_i, v_j) \in \mathcal{E}$. $\mathbf{X} \in \mathbb{R}^{n \times d_x}$ denotes the feature matrix where the i-th row $\boldsymbol{x}_i \in \mathbb{R}^{d_x}$ is the d_x-dimensional feature vector of node $v_i \in \mathcal{V}$. $\mathcal{N}(v_i)$ denotes the set of node v_i's neighboring nodes. GNN models leverage node features and graph structure to learn a d_h-dimensional representation vector $\boldsymbol{h}_i \in \mathbb{R}^{d_h}$ for each target node $v_i \in \mathcal{V}$. Generally, GNN models follow the message-passing mechanism, in which the representation of node v_i is iteratively updated by aggregating the representations from its neighboring nodes. More concretely, a GNN model f updates the representation of node $v_i \in \mathcal{V}$ at layer l by

$$\boldsymbol{m}_i^{(l)} = \texttt{AGGREGATE}^{(l)}\left(\left\{\boldsymbol{h}_j^{(l-1)} : v_j \in \mathcal{N}(v_i)\right\}\right), \ \boldsymbol{h}_i^{(l)} = \texttt{COMBINE}^{(l)}\left(\boldsymbol{h}_i^{(l-1)}, \boldsymbol{m}_i^{(l)}\right), \tag{1}$$

where $\boldsymbol{h}_i^{(l)} \in \mathbb{R}^{d_l}$ denotes the d_l-dimensional representation of node v_i at layer l, and we initialize $\boldsymbol{h}_i^{(0)} = \boldsymbol{x}_i$. $\texttt{AGGREGATE}^{(l)}(\cdot)$ represents the aggregation operation extracting the neighboring information of node v_i, and $\texttt{COMBINE}^{(l)}(\cdot)$ represents the combination operation integrating the previous representation of node v_i and its neighboring information at layer l. The ultimate representation $\boldsymbol{h}_i$ of node v_i after the final layer of the GNN model can be used for diverse downstream tasks. In this study, we focus on node-level downstream tasks, particularly on node classification, i.e., predicting the class label y_i of each target node v_i.

3.2 Graph prompting

In this study, we mainly center on graph prompting for node-level downstream tasks, e.g., node classification. Given a GNN model pre-trained by a pre-training task, graph prompting aims to adapt the pre-trained GNN model for node classification by learning a graph transformation with trainable prompts to modify the input graph without tuning the pre-trained GNN model. Let f_{θ^*} denote the pre-trained GNN model with parameters θ^*. Given the input graph $\mathcal{G}$ with a labeled node list $\mathcal{V}_L \subset \mathcal{V}$, the graph transformation t_ϕ transforms it to a prompted graph $\tilde{\mathcal{G}} = t_\phi(\mathcal{G})$ where ϕ contains learnable prompt parameters. By tuning the prompt parameters in ϕ, the pre-trained GNN model f_{θ^*} can generate suitable node representations for node classification through a trainable

classifier g parameterized by ω. Mathematically, we can train ϕ and ω by optimizing the empirical loss minimization problem of graph prompting defined as

$$\min_{\phi,\omega} \mathcal{L}_P(\phi,\omega) = \frac{1}{|\mathcal{V}_L|} \sum_{v_i \in \mathcal{V}_L} \ell\left(g_\omega\left(\left[f_{\theta^*}\left(\tilde{\mathcal{G}}\right)\right]_i\right), y_i\right), \tag{2}$$

where $\ell(\cdot,\cdot)$ denotes the cross-entropy loss for node classification. The main goal of graph prompting is to figure out the optimal graph transformation t_ϕ for downstream tasks.

4 Methodology

In this section, we present GraphTOP — a graph topology-oriented prompting framework that adapts pre-trained GNN models for node classification by learning to modify graph topology while keeping the pre-trained GNN models frozen. We begin by formulating topology-oriented prompting as an edge rewiring problem. Then, we relax this problem into the continuous probability space via reparameterization and optimize the probabilities through a shared trainable projector. To reduce complexity, we restrict the edge rewiring problem to multi-hop local subgraphs. Finally, we present the complete optimization objective in GraphTOP, designed to ensure tight relaxation and preserve graph sparsity.

4.1 Topology-oriented prompting

Graph prompting learns to transform the input graph $\mathcal{G} = (\mathbf{A}, \mathbf{X})$ to a prompted one $\tilde{\mathcal{G}}$. Ideally, the prompted graph $\tilde{\mathcal{G}}$ can generate informative node representations through the pre-trained GNN model f_{θ^*} and is more suitable for downstream tasks, such as node classification. Unlike previous feature-oriented graph prompting studies [5, 36] that aim to transform the input graph with learnable prompt vectors applied to node features (i.e., $\mathbf{X}$), topology-oriented prompting designs learnable prompts to manipulate graph topology (i.e., $\mathbf{A}$). As a result, we will obtain the prompted graph $\tilde{\mathcal{G}} = (\mathbf{S}, \mathbf{X})$ with the prompted graph topology $\mathbf{S} \in \{0,1\}^{n \times n}$.

To achieve this, we formulate topology-oriented prompting as an edge rewiring problem. Specifically, given each pair of nodes $v_i \in \mathcal{V}$ and $v_j \in \mathcal{V}$ ($v_i \neq v_j$), the goal is to obtain a learnable binary edge selector $s_{ij} \in \mathbf{S}$ that determines whether an edge exists between them. In other words, edge (v_i, v_j) belongs to the prompted edge set iff $s_{ij} = 1$. Mathematically, we can reformulate the problem of graph prompting in Equation (2) as

$$\min_{\mathbf{S},\omega} \mathcal{L}_P(\mathbf{S},\omega) = \frac{1}{|\mathcal{V}_L|} \sum_{v_i \in \mathcal{V}_L} \ell\left(g_\omega\left(\left[f_{\theta^*}(\mathbf{S},\mathbf{X})\right]_i\right), y_i\right), \text{ s.t. } \mathbf{S} \in \{0,1\}^{n \times n}. \tag{3}$$

4.2 Prompt reparameterization

Solving the problem in Equation (3) is challenging since the loss function is discrete with respect to binary edge selectors in $\mathbf{S}$, making it intractable and difficult to apply in practice. In this study, we propose to address this issue through edge rewiring reparameterization. When we treat the edge selectors in $\mathbf{S}$ as binary random variables and reparameterize the problem in Equation (3) with respect to their distributions, it can be relaxed into an expected loss minimization problem over the classifier weight and probability spaces, which is continuous. More concretely, we treat each edge selector $s_{ij} \in \mathbf{S}$ as a Bernoulli random variable following the Bernoulli distribution with probability p_{ij}. During the forward pass, each edge selector s_{ij} is sampled from the Bernoulli distribution with probability p_{ij} to be 1 and $1 - p_{ij}$ to be 0. Let $\mathbf{P}$ represent the probability matrix. The problem of topology-oriented prompting in Equation (3) can be relaxed into the expected loss minimization problem as

$$\min_{\mathbf{P},\omega} \mathbb{E}_{\mathbf{S} \sim \texttt{Bern}(\mathbf{P})} \left[\mathcal{L}_P(\mathbf{S},\omega)\right], \text{ s.t. } \mathbf{P} \in [0,1]^{n \times n}. \tag{4}$$

Therefore, we reformulate the edge rewiring problem as a continuous problem in Equation (4).

However, solving the above problem via gradient descent remains challenging. The difficulty lies in computing the gradient of the expected loss with respect to the probability matrix $\mathbf{P}$, as Bernoulli sampling is non-differentiable. To make the probability matrix trainable, we propose to reparameterize the Bernoulli sampling in graph prompting via Gumbel-Softmax reparameterization [17, 30, 59, 60]. Before introducing this, we first present the following theorem.

Theorem 1. *Given two random variables G_1 and G_2 that follow the Gumbel distribution* $\texttt{Gumbel}(0,1)$*, for any probability $p_{ij} \in \mathbf{P}$, we have*

$$\texttt{Pr}\left(G_1 - G_2 + \log\left(\frac{p_{ij}}{1-p_{ij}}\right) \geq 0\right) = p_{ij}. \tag{5}$$

The proof of Theorem 1 is provided in Appendix A. Recall that the probability that each edge selector s_{ij} is equal to 1 is also p_{ij}. Therefore, we can rewrite the expected loss function in Equation (4) as

$$\mathbb{E}_{\mathbf{G}_1,\mathbf{G}_2}\left[\mathcal{L}_P\left(\mathbb{1}\left(\mathbf{G}_1 - \mathbf{G}_2 + \log\left(\frac{\mathbf{P}}{\mathbf{1}_{n\times n} - \mathbf{P}}\right) \geq 0\right), \omega\right)\right], \tag{6}$$

where $\mathbb{1}(\cdot)$ is the indicator function, and $\mathbf{1}_{n\times n}$ is an n-by-n all-ones matrix. $\mathbf{G}_1 \in \mathbb{R}^{n\times n}$ and $\mathbf{G}_2 \in \mathbb{R}^{n\times n}$ are two random matrices where each entry follows the Gumbel distribution $\texttt{Gumbel}(0,1)$.

Nevertheless, the rewritten expected loss is discrete due to the indicator function. A common solution to this issue is to approximate the expected loss by replacing the indicator function with a sigmoid function. Specifically, the expected loss can be approximated as

$$\mathbb{E}_{\mathbf{G}_1,\mathbf{G}_2}\left[\mathcal{L}_P\left(\sigma\left(\frac{\mathbf{G}_1 - \mathbf{G}_2 + \log\left(\frac{\mathbf{P}}{\mathbf{1}_{n\times n} - \mathbf{P}}\right)}{\tau}\right), \omega\right)\right], \tag{7}$$

where $\sigma(\cdot)$ is the element-wise sigmoid function, and τ is the temperature annealing parameter that decreases linearly, facilitating the transition from probabilistic approximations to near-deterministic outputs during training.

The final task is the computation of $\mathbf{P}$. Instead of learning each individual probability in $\mathbf{P}$ as free parameters — which is usually hard to train, we propose to obtain the probabilities through a learnable projector shared by nodes. As the GNN model has been well pre-trained, we consider using the informative node representations from the pre-trained GNN model as the input of the projector. Given the representation matrix $\mathbf{H} \in \mathbb{R}^{n\times d_h}$ where the i-th row $\boldsymbol{h}_i$ is the representation vector of node v_i, we obtain $\mathbf{P}$ through a learnable projector m based on $\mathbf{H}$, i.e., $\mathbf{P} = m(\mathbf{H})$. Specifically, given two nodes v_i and v_j with their representations $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$, the projector computes the probability p_{ij} by

$$p_{ij} = \sigma\left(\mathbf{W}_2\left(\texttt{ReLU}\left(\mathbf{W}_1\left(\boldsymbol{h}_i + \boldsymbol{h}_j\right)\right)\right)\right), \tag{8}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are trainable weights in the projector m. Note that the projector will generate $p_{ij} = p_{ji}$, which is consistent with undirected graphs. The integration strategy of $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ in m can be altered (e.g., concatenation) to handle directed graphs. Therefore, the final problem of topology-oriented prompting can be formulated as

$$\min_{\phi,\omega} \mathcal{L}_P(\phi,\omega) = \frac{1}{|\mathcal{V}_L|}\sum_{v_i\in\mathcal{V}_L} \ell\left(g_\omega\left([f_{\theta^*}(\mathbf{S},\mathbf{X})]_i\right), y_i\right), \text{ where } \mathbf{S} = \sigma\left(\frac{\mathbf{g}_1 - \mathbf{g}_2 + \log\left(\frac{m_\phi(\mathbf{H})}{\mathbf{1}_{n\times n} - m_\phi(\mathbf{H})}\right)}{\tau}\right). \tag{9}$$

Here, $\phi = \{\mathbf{W}_1, \mathbf{W}_2\}$ represents the prompt parameters that we aim to learn. $\mathbf{g}_1 \in \mathbb{R}^{n\times n}$ and $\mathbf{g}_2 \in \mathbb{R}^{n\times n}$ are two matrices where each entry is sampled from $\texttt{Gumbel}(0,1)$ per iteration. Since the representation matrix $\mathbf{H}$ is generated by the pre-trained GNN model and fixed during training, we can compute it before the adaptation phase to avoid additional computational costs.

4.3 Subgraph-constrained topology-oriented prompting

Although the above formulation is feasible to solve, it still poses challenges in scalability. Since we need to learn an edge selector for each node pair, the number of learnable edge selectors grows overwhelmingly large as the number of nodes in the input graph increases. In GraphTOP, we propose subgraph-constrained topology-oriented prompting to reduce the complexity. In most GNN architectures, the representation of a target node through a pre-trained GNN model primarily depends on its local subgraph [23, 51]. Motivated by this, we consider restricting learning edge selectors for edge rewiring to a multi-hop local subgraph of each target node. More concretely, we first extract the ρ-hop subgraph $\mathcal{G}(v_i) = (\mathbf{A}(v_i), \mathbf{X}(v_i))$ for each target node v_i. Here, $\mathbf{X}(v_i)$ includes the feature vectors of node v_i and other nodes within ρ steps of node v_i, and $\mathbf{A}(v_i)$ indicates the connection between these nodes extracted from the original adjacency matrix $\mathbf{A}$. ρ is a pre-defined hyperparameter. Typically, we set $\rho = 2$ to balance efficiency and effectiveness. Then, the task is

to perform edge rewiring to obtain the prompted subgraph $\tilde{\mathcal{G}}(v_i) = (\mathbf{S}(v_i), \mathbf{X}(v_i))$ based on $\mathcal{G}(v_i)$. Nonetheless, the computation is still expensive if we consider edge connections between each pair of nodes in $\mathcal{G}(v_i)$. Theoretically, the computational cost is approximately $\mathcal{O}(D^{2\rho})$, where D represents the average node degree.

To further reduce the computational cost of edge rewiring, we propose to rewire edges only between the target node and other nodes in $\mathcal{G}(v_i)$. In this way, we only need to consider how the target node connects to other nodes while keeping other edges in the original subgraph $\mathcal{G}(v_i)$ intact. As a result, the computational cost will significantly decrease, reducing to $\mathcal{O}(D^{\rho})$. More specifically, For each pair of nodes v_j and v_k in the subgraph $\mathcal{G}(v_i)$, we can compute $s_{jk} \in \mathbf{S}(v_i)$ by

$$s_{jk} = \begin{cases} \sigma\left(\dfrac{g_1 - g_2 + \log\left(\frac{p_{jk}}{1-p_{jk}}\right)}{\tau}\right), & \text{if } v_i \in \{v_j, v_k\}, \\ a_{jk}, & \text{otherwise,} \end{cases} \tag{10}$$

where g_1 and g_2 are two values sampled from $\texttt{Gumbel}(0, 1)$. Therefore, the objective function in Equation (9) can be reformulated as

$$\mathcal{L}_P(\phi, \omega) = \frac{1}{|\mathcal{V}_L|} \sum_{v_i \in \mathcal{V}_L} \ell\left(g_\omega\left(\left[f_{\theta^*}\left(\mathbf{S}(v_i), \mathbf{X}(v_i)\right)\right]_i\right), y_i\right). \tag{11}$$

4.4 Prompt optimization

While the problem of topology-oriented prompting becomes solvable through the relaxation in Section 4.2, simply solving the problem in Equation (11) cannot guarantee the tightness of the relaxation. Typically, it is unlikely to obtain every probability p_{ij} converging to 0 or 1 after training. As a result, Bernoulli sampling may cause significant fluctuations in graph topology during inference, resulting in unstable classification performance. Therefore, the goal here is to encourage deterministic probabilities (i.e., either 0 or 1) during training. To achieve this, we introduce an extra entropy-based regularization term $\mathcal{L}_E(\phi)$ in the objective function. The intuition of this regularization term is to penalize high-entropy probabilities via entropy minimization [10]. Mathematically, the regularization term $\mathcal{L}_E(\phi)$ can be written as

$$\mathcal{L}_E(\phi) = \frac{1}{|\mathcal{V}_L|} \sum_{v_i \in \mathcal{V}_L} \left(\frac{1}{|\mathcal{N}_\rho(v_i)|} \sum_{v_j \in \mathcal{N}_\rho(v_i)} e\left(p_{ij}\right)\right), \text{ where } e\left(p_{ij}\right) = p_{ij}\log p_{ij} + (1-p_{ij})\log(1-p_{ij}). \tag{12}$$

Here, $\mathcal{N}_\rho(v_i)$ represents the set of nodes within node v_i's ρ-hop subgraph except node v_i. By minimizing $\mathcal{L}_E$, each probability p_{ij} will be likely to converge to 0 or 1 after training, leading to a deterministic graph topology for inference. As a result, the relaxation in Equation (4) becomes tight.

Meanwhile, the prompted graph may become overly dense, resulting in an almost fully connected graph topology. Such prompted graphs are often impractical for most applications and incur high computational costs [2, 20, 26]. Therefore, it is important to control how sparse the prompted graph is. To achieve this, we introduce another regularization term $\mathcal{L}_S(\phi)$ to constrain the size of connected edges in the prompted graph. More specifically, the regularization term $\mathcal{L}_S(\phi)$ can be formulated as

$$\mathcal{L}_S(\phi) = \frac{1}{|\mathcal{V}_L|} \sum_{v_i \in \mathcal{V}_L} \left| \frac{\sum_{v_j \in \mathcal{N}_\rho(v_i)} p_{ij}}{|\mathcal{N}_\rho(v_i)|} - \gamma \right|, \tag{13}$$

where γ is a hyperparameter to control the size of connected edges for each node. Finally, we can write the overall objective function as

$$\min_{\phi,\omega} \mathcal{L}_P(\phi, \omega) + \lambda_1 \mathcal{L}_E(\phi) + \lambda_2 \mathcal{L}_S(\phi), \tag{14}$$

where λ_1 and λ_2 are two hyperparameters to balance different loss terms.

5 Analysis of GraphTOP

In this section, we present a comprehensive analysis of our proposed framework. The overall algorithm of GraphTOP can be found in Appendix B.

5.1 Complexity analysis

Without loss of generality, we take a K-layer GCN model as an example for complexity analysis. When we set $\rho = 2$, the representation of each target node is computed based on its 2-hop local subgraph. When the average node degree is D, the expected number of nodes within its 2-hop local subgraph is $\mathcal{O}(D^2)$. Accordingly, the expected number of edges within its 2-hop local subgraph is $\mathcal{O}(D^3)$. We assume each layer l of the GCN model has the same hidden size as the feature matrix, i.e., $d_x = d_h = d_l$ for simplicity. In this case, the time complexity of the GCN model is $\mathcal{O}(KD^2d_l^2 + KD^3d_l)$. The extra cost in GraphTOP comes from the computation of the probabilities when generating the prompted graph. The time complexity for computing the probabilities is $\mathcal{O}(D^2d_l^2 + D^2d_l)$. Therefore, we can conclude that computing the probabilities in GraphTOP does not introduce significant additional computational costs compared with GNN models.

5.2 Theoretical analysis

In this subsection, we provide a theoretical analysis of how topology-oriented prompting in our GraphTOP benefits pre-trained GNN models for node classification. Following previous graph learning studies [29, 40], our analysis is similarly based on the contextual stochastic block model (CSBM) [4]. Given a random graph $\mathcal{G}$ generated by the CSBM with two node classes c_1 and c_2, node v_i has the feature vector $\boldsymbol{x}_i$ following a Gaussian distribution $\boldsymbol{x}_i \sim N(\boldsymbol{\mu}_1, \mathbf{I})$ if it is from class c_1; otherwise, $\boldsymbol{x}_i \sim N(\boldsymbol{\mu}_2, \mathbf{I})$. Here, we assume $\boldsymbol{\mu}_1 \neq \boldsymbol{\mu}_2$. The edges in $\mathcal{G}$ are generated following an intra-class probability $p > 0$ and an inter-class probability $q > 0$. In other words, each pair of nodes will be linked through an edge with probability p if they are from the same class; otherwise, the probability is q. We denote a random graph generated by the CSBM as $\mathcal{G} \sim \texttt{CSBM}(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, p, q)$.

We aim to analyze how our edge rewiring design in topology-oriented prompting improves linear separability under pre-trained GNN models. Here, we consider 2-layer linear GCN models [23] for simplicity. We are particularly interested in the expected Euclidean distance between node representations of the two classes after 2-layer GCN operations. Let $\texttt{Dist}'$ and $\texttt{Dist}$ denote the expected Euclidean distances with or without our edge rewiring design, respectively. In GraphTOP, we have the following theorem.

Theorem 2. *Given a pre-trained 2-layer linear GCN model f_{θ^*} and a random graph $\mathcal{G} \sim \texttt{CSBM}(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, p, q)$, when $p \neq q$, there always exists edge rewiring in the prompted graph by GraphTOP that satisfies*

$$\texttt{Dist}' = \frac{p+q}{|p-q|}\texttt{Dist} > \texttt{Dist}. \tag{15}$$

The proof of Theorem 2 can be found in Appendix C. Theorem 2 indicates that the expected distance between the two class centroids can be enlarged effectively when GraphTOP alters how every target node connects to the other nodes within its multi-hop local subgraph. Under this circumstance, the representations of nodes from the two classes are more likely to be correctly classified. Therefore, we conclude that our edge rewiring design in GraphTOP can theoretically enhance the classification performance of pre-trained GNN models.

6 Experiments

6.1 Experimental setup

Datasets We adopt five real-world graph datasets from various domains to evaluate the performance of our framework. These datasets include Cora [47], PubMed [47], Amazon [31], Minesweeper [31], and Flickr [53]. Detailed information about these datasets can be found in Appendix E.1.

Pre-training strategies To evaluate the compatibility of our framework with different pre-training strategies, we conduct experiments under four representative pre-training strategies. More specifically, we adopt GraphCL [49] and SimGRACE [45] for contrast-based methods. As for generation-based methods, we follow two previous studies — GPPT [35] and GraphPrompt [27] to pre-train GNN models via link prediction. We term them LP-GPPT and LP-GraphPrompt, respectively. More information about these pre-training strategies can be found in Appendix E.2.

Table 1: Accuracy on 5-shot node classification over five datasets. The best-performing method is **bolded**, and the runner-up is underlined.

Pre-training strategies	Graph prompting methods	Cora	PubMed	Amazon	Minesweeper	Flickr
GraphCL	Linear Probe	$55.69_{\pm 5.74}$	$67.30_{\pm 6.26}$	$23.19_{\pm 7.21}$	$67.59_{\pm 6.30}$	$29.31_{\pm 8.91}$
	GPPT	$61.50_{\pm 4.49}$	$65.75_{\pm 3.99}$	$24.27_{\pm 3.74}$	$65.44_{\pm 8.97}$	$24.64_{\pm 3.15}$
	ALL-in-one	$52.33_{\pm 4.55}$	$65.78_{\pm 8.65}$	$22.82_{\pm 6.09}$	$63.82_{\pm 8.63}$	$21.57_{\pm 4.64}$
	GraphPrompt	$62.12_{\pm 3.28}$	$67.01_{\pm 4.56}$	$21.71_{\pm 2.93}$	$61.19_{\pm 3.50}$	$21.92_{\pm 3.72}$
	GraphPrompt+	$58.91_{\pm 3.12}$	$66.26_{\pm 5.75}$	$23.83_{\pm 2.30}$	$61.64_{\pm 6.36}$	$24.43_{\pm 4.62}$
	ProNoG	$60.01_{\pm 7.03}$	$68.17_{\pm 4.82}$	$23.26_{\pm 2.42}$	$65.48_{\pm 3.40}$	$26.17_{\pm 5.18}$
	GraphTOP	$\mathbf{63.44_{\pm 4.21}}$	$\mathbf{68.28_{\pm 4.15}}$	$\mathbf{27.43_{\pm 7.02}}$	$\mathbf{68.25_{\pm 7.14}}$	$\mathbf{30.93_{\pm 9.07}}$
SimGRACE	Linear Probe	$40.68_{\pm 2.29}$	$54.59_{\pm 6.02}$	$24.58_{\pm 4.18}$	$60.58_{\pm 6.42}$	$26.78_{\pm 5.29}$
	GPPT	$44.83_{\pm 4.67}$	$52.25_{\pm 5.91}$	$24.27_{\pm 3.74}$	$59.62_{\pm 4.80}$	$22.11_{\pm 3.56}$
	ALL-in-one	$41.11_{\pm 4.92}$	$51.45_{\pm 4.73}$	$22.66_{\pm 3.55}$	$58.11_{\pm 3.82}$	$21.50_{\pm 4.49}$
	GraphPrompt	$47.02_{\pm 3.87}$	$55.74_{\pm 5.80}$	$21.24_{\pm 2.78}$	$58.72_{\pm 4.37}$	$19.72_{\pm 4.54}$
	GraphPrompt+	$\mathbf{51.26_{\pm 4.90}}$	$55.93_{\pm 6.98}$	$25.07_{\pm 1.71}$	$60.76_{\pm 6.75}$	$20.79_{\pm 5.65}$
	ProNoG	$42.44_{\pm 2.97}$	$55.11_{\pm 5.98}$	$22.53_{\pm 2.65}$	$\mathbf{63.03_{\pm 2.74}}$	$25.44_{\pm 4.45}$
	GraphTOP	$50.57_{\pm 2.91}$	$\mathbf{56.64_{\pm 5.42}}$	$\mathbf{25.67_{\pm 3.34}}$	$61.25_{\pm 5.08}$	$\mathbf{27.70_{\pm 5.69}}$
LP-GPPT	Linear Probe	$24.40_{\pm 2.83}$	$42.26_{\pm 5.09}$	$25.50_{\pm 4.11}$	$63.22_{\pm 8.16}$	$23.76_{\pm 4.30}$
	GPPT	$32.08_{\pm 7.66}$	$44.85_{\pm 4.73}$	$28.90_{\pm 3.50}$	$63.44_{\pm 8.28}$	$22.25_{\pm 4.41}$
	ALL-in-one	$26.67_{\pm 6.24}$	$41.11_{\pm 4.92}$	$24.49_{\pm 3.51}$	$59.97_{\pm 4.67}$	$18.09_{\pm 4.30}$
	GraphPrompt	$30.14_{\pm 2.01}$	$44.72_{\pm 6.68}$	$21.88_{\pm 3.56}$	$61.73_{\pm 6.35}$	$19.72_{\pm 1.76}$
	GraphPrompt+	$33.42_{\pm 2.91}$	$45.17_{\pm 6.91}$	$24.34_{\pm 1.72}$	$61.15_{\pm 3.37}$	$21.02_{\pm 4.22}$
	ProNoG	$33.71_{\pm 4.12}$	$46.07_{\pm 3.62}$	$21.39_{\pm 1.69}$	$66.11_{\pm 4.20}$	$24.08_{\pm 4.10}$
	GraphTOP	$\mathbf{33.97_{\pm 2.43}}$	$\mathbf{46.52_{\pm 6.18}}$	$\mathbf{32.41_{\pm 7.18}}$	$\mathbf{66.67_{\pm 3.83}}$	$\mathbf{25.95_{\pm 3.17}}$
LP-GraphPrompt	Linear Probe	$50.15_{\pm 5.88}$	$66.26_{\pm 5.69}$	$25.00_{\pm 6.78}$	$65.90_{\pm 7.36}$	$23.75_{\pm 3.26}$
	GPPT	$52.13_{\pm 7.15}$	$63.16_{\pm 8.25}$	$25.38_{\pm 5.78}$	$62.53_{\pm 8.91}$	$24.16_{\pm 3.88}$
	ALL-in-one	$49.42_{\pm 2.70}$	$64.73_{\pm 6.46}$	$21.37_{\pm 3.65}$	$58.17_{\pm 4.63}$	$22.10_{\pm 2.92}$
	GraphPrompt	$52.35_{\pm 4.82}$	$\mathbf{68.16_{\pm 8.23}}$	$22.76_{\pm 2.81}$	$58.01_{\pm 3.26}$	$21.15_{\pm 1.46}$
	GraphPrompt+	$52.19_{\pm 5.22}$	$62.19_{\pm 6.70}$	$24.44_{\pm 1.81}$	$61.78_{\pm 3.92}$	$21.48_{\pm 4.09}$
	ProNoG	$52.49_{\pm 5.43}$	$67.68_{\pm 5.02}$	$23.79_{\pm 2.04}$	$59.88_{\pm 8.50}$	$24.74_{\pm 1.10}$
	GraphTOP	$\mathbf{53.44_{\pm 4.72}}$	$68.14_{\pm 5.47}$	$\mathbf{27.07_{\pm 5.84}}$	$\mathbf{67.09_{\pm 8.45}}$	$\mathbf{25.03_{\pm 3.84}}$

Baselines We include five state-of-the-art graph prompting methods as the baselines of our experiments, including GPPT [35], All-in-one [36], GraphPrompt [27], GraphPrompt+ [50], and ProNoG [51]. Additionally, we also report the performance of tuning linear probes as the classifier based on node representations from pre-trained GNN models without any graph prompting methods (termed *Linear Probe*). More information on these baselines can be found in Appendix E.3.

Implementation details We use a 2-layer GCN [23] as the GNN model for each graph prompting method. The hidden size is 128. All the experimental results are based on the 5-shot setting. Each method is trained using the Adam optimizer [22] with a learning rate of 0.005. The number of epochs is set to 500 for graph prompting. We set $\gamma = 0.5$ in our experiments. We conduct a grid search for λ_1 and λ_2. The reported performance is the average result of five runs with different random seeds.

6.2 Effectiveness evaluation

We first evaluate the overall performance of our method and other baselines. Table 1 reports the average accuracies on 5-shot node classification over five datasets under four pre-training strategies. According to the results in the table, we first observe that Linear Probe can achieve competitive performance in some cases, although it only trains a classifier during the adaptation phase. For example, it outperforms several baselines on Flickr across four pre-training strategies. Additionally, we observe that ProNoG is a strong baseline compared to other graph prompting methods. It achieves the runner-up position in some cases, such as on four datasets under LP-GPPT. Finally, it is noteworthy that GraphTOP achieves the best performance in most cases compared to other baselines. More specifically, GraphTOP outperforms other baselines in 17 out of 20 experiments. It consistently achieves the best performance on Amazon and Flickr across all four pre-training strategies. These observations validate the effectiveness of GraphTOP in modifying graph topology to adapt pre-trained GNN models for node classification.

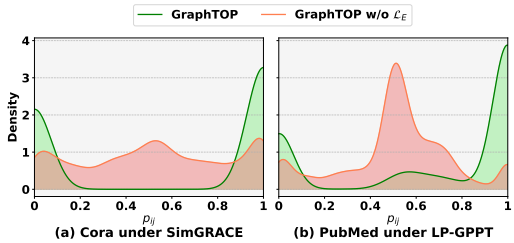

Figure 1: The distribution curves of p_{ij} by GraphTOP with and without $\mathcal{L}_E$.

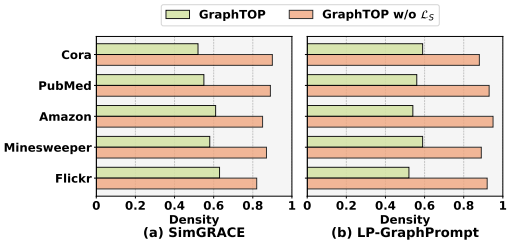

Figure 2: The average edge densities of target nodes by GraphTOP with and without $\mathcal{L}_S$.

6.3 Analysis of GraphTOP

Analysis of $\mathcal{L}_E$ The loss term $\mathcal{L}_E$ ensures tight relaxation of GraphTOP by encouraging each probability to converge to 0 or 1. To evaluate the effectiveness of $\mathcal{L}_E$, we conduct experiments on the probability distribution by GraphTOP and its variant without $\mathcal{L}_E$. Figure 1 illustrates the distribution curves of p_{ij} by GraphTOP with and without $\mathcal{L}_E$. Based on the distribution curves, we observe that the probabilities are not centered around 0 or 1 without $\mathcal{L}_E$. Instead, we may obtain many probabilities around 0.5. In this case, tight relaxation cannot be guaranteed in our reparameterization. However, when we keep $\mathcal{L}_E$ in the objective function, the probabilities are more likely to be close to 0 or 1, which validates the motivation of our design in GraphTOP.

Analysis of $\mathcal{L}_S$ In the objective function of GraphTOP, $\mathcal{L}_S$ is designed to control the number of neighboring nodes for each target node by restricting the sparsity of a target node to the threshold γ (we set $\gamma = 0.5$ in our experiments). Typically, the edge densities should be forced to γ by the loss term $\mathcal{L}_S$. To validate the effectiveness of $\mathcal{L}_S$, we conduct experiments evaluating the edge densities of target nodes when removing $\mathcal{L}_S$. Figure 2 illustrates the average edge densities of target nodes by GraphTOP with and without $\mathcal{L}_S$. From these bar figures, we observe that the average edge densities are consistently very high when we remove $\mathcal{L}_S$ from the objective function. It means that each target node connects to almost all other nodes within its local subgraph, leading to an overly dense graph topology. When we retain $\mathcal{L}_S$ in the objective function, we observe that the average densities by GraphTOP decrease significantly toward 0.5, thereby ensuring the sparsity of the prompted graph. Therefore, we can conclude that the loss term $\mathcal{L}_S$ can effectively avoid overly dense prompted graphs.

Efficiency analysis In Section 4.3, we restrict edge rewiring within the ρ-hop local subgraph of each target node. To further reduce the complexity, we propose to rewire edges only between each target node and other nodes within its multi-hop subgraph. To evaluate the efficiency improvement by our design in GraphTOP, we conduct experiments on the running time of GraphTOP and its variant with edge rewiring between each pair of nodes within a multi-hop subgraph (i.e., GraphTOP$_{\text{all_nodes}}$). Table 2 shows the running time of the two methods when $\rho = 2$ and $\rho = 3$. From the results in the table, we can observe that GraphTOP$_{\text{all_nodes}}$ requires much more time compared with GraphTOP. For instance, GraphTOP$_{\text{all_nodes}}$ needs 651.22 seconds to finish each experiment when $\rho = 2$, which is about $2.97\times$ longer than GraphTOP. Furthermore, when we set $\rho = 3$, GraphTOP$_{\text{all_nodes}}$ will be out of GPU memory on our server. In contrast, the running time of GraphTOP does not increase significantly. We can observe similar patterns on other datasets. These observations strongly support our design in GraphTOP by rewiring edges only between each target node with other nodes within its local subgraph.

Table 2: Running time (seconds) of GraphTOP and its variant over three datasets when $\rho = 2$ and $\rho = 3$ (OOM: out of GPU memory).

Dataset	ρ	GraphTOP	GraphTOP$_{\text{all_nodes}}$
Cora	2	36.52	101.70
	3	53.08	OOM
Amazon	2	218.61	651.22
	3	252.21	OOM
Minesweeper	2	77.51	113.37
	3	78.65	199.48

More experimental results Due to the page limit, more experimental results, including the influence of λ_1 and λ_2, analysis of GPU memory usage, and performance with different numbers of shots, are provided in Appendix F.

7 Conclusion

In this study, we conduct a pioneering investigation into graph prompting from the perspective of graph topology. We propose GraphTOP — the first graph topology-oriented prompting framework that adapts pre-trained GNN models by modifying graph topology for downstream tasks, particularly node classification. Extensive experiments over five graph datasets validate the effectiveness of GraphTOP against six baselines under four pre-training strategies.

Acknowledgments and Disclosure of Funding

This work is supported in part by the National Science Foundation (NSF) under grants IIS-2006844, IIS-2144209, IIS-2223769, IIS-2331315, CNS-2154962, BCS-2228534, and CMMI-2411248, the Office of Naval Research (ONR) under grant N000142412636, the Commonwealth Cyber Initiative (CCI) under grant VV-1Q25-004, and the iPRIME Fellowship Award at UVA.

References

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

[2] Yu Chen, Lingfei Wu, and Mohammed Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in neural information processing systems*, 2020.

[3] Limeng Cui, Haeseung Seo, Maryam Tabar, Fenglong Ma, Suhang Wang, and Dongwon Lee. Deterrent: Knowledge guided graph attention network for detecting healthcare misinformation. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020.

[4] Yash Deshpande, Subhabrata Sen, Andrea Montanari, and Elchanan Mossel. Contextual stochastic block models. *Advances in Neural Information Processing Systems*, 2018.

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

[6] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. Slaps: Self-supervision improves structure learning for graph neural networks. *Advances in Neural Information Processing Systems*, 2021.

[7] Xingbo Fu, Chen Chen, Yushun Dong, Anil Vullikanti, Eili Klein, Gregory Madden, and Jundong Li. Spatial-temporal networks for antibiogram pattern prediction. In *2023 IEEE 11th International Conference on Healthcare Informatics*, 2023.

[8] Xingbo Fu, Yinhan He, and Jundong Li. Edge prompt tuning for graph neural networks. In *International Conference on Learning Representations*, 2025.

[9] Xingbo Fu, Zehong Wang, Zihan Chen, Jiazheng Li, Yaochen Zhu, Zhenyu Lei, Cong Shen, Yanfang Ye, Chuxu Zhang, and Jundong Li. Graph prompting for graph learning models: Recent advances and future directions. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, 2025.

[10] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, 2004.

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

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

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

[14] Ziniu Hu, Changjun Fan, Ting Chen, Kai-Wei Chang, and Yizhou Sun. Pre-training graph neural networks for generic structural feature extraction. *arXiv preprint arXiv:1905.13728*, 2019.

[15] Renhong Huang, Jiarong Xu, Xin Jiang, Chenglu Pan, Zhiming Yang, Chunping Wang, and Yang Yang. Measuring task similarity and its implication in fine-tuning graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[16] EunJeong Hwang, Veronika Thost, Shib Sankar Dasgupta, and Tengfei Ma. An analysis of virtual nodes in graph neural networks for link prediction. In *The First Learning on Graphs Conference*, 2022.

[17] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

[18] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision*, 2022.

[19] Wei Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zitao Liu, and Jiliang Tang. Self-supervised learning on graphs: Deep insights and new direction. *arXiv preprint arXiv:2006.10141*, 2020.

[20] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020.

[21] Muhammad Uzair Khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. Maple: Multi-modal prompt learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19113–19122, 2023.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

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

[24] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 2018.

[25] Shuangli Li, Jingbo Zhou, Tong Xu, Dejing Dou, and Hui Xiong. Geomgcl: Geometric graph contrastive learning for molecular property prediction. In *Proceedings of the AAAI conference on artificial intelligence*, 2022.

[26] Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, and Shirui Pan. Towards unsupervised deep graph structure learning. In *Proceedings of the ACM Web Conference 2022*, 2022.

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

[28] Sitao Luan, Chenqing Hua, Minkai Xu, Qincheng Lu, Jiaqi Zhu, Xiao-Wen Chang, Jie Fu, Jure Leskovec, and Doina Precup. When do graph neural networks help with node classification? investigating the homophily principle on node distinguishability. *Advances in Neural Information Processing Systems*, 2023.

[29] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? In *International Conference on Learning Representations*, 2022.

[30] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

[31] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: Are we really making progress? In *International Conference on Learning Representations*, 2023.

[32] Jianzhong Qi, Zhuowei Zhao, Egemen Tanin, Tingru Cui, Neema Nassir, and Majid Sarvi. A graph and attentive multi-path convolutional network for traffic prediction. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[33] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

[34] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2020.

[35] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. 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*, 2022.

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

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

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

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

[40] Junfu Wang, Yuanfang Guo, Liang Yang, and Yunhong Wang. Understanding heterophily for graph neural networks. In *International Conference on Machine Learning*, 2024.

[41] Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[42] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 2019.

[43] Xuemei Wei, Yezheng Liu, Jianshan Sun, Yuanchun Jiang, Qifeng Tang, and Kun Yuan. Dual subgraph-based graph neural network for friendship prediction in location-based social networks. *ACM Transactions on Knowledge Discovery from Data*, 2023.

[44] Lirong Wu, Haitao Lin, Cheng Tan, Zhangyang Gao, and Stan Z Li. Self-supervised learning on graphs: Contrastive, generative, or predictive. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

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

[46] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

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

[48] Seungryong Yoo, Eunji Kim, Dahuin Jung, Jungbeom Lee, and Sungroh Yoon. Improving visual prompt tuning for self-supervised vision transformers. In *International Conference on Machine Learning*, 2023.

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

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

[51] Xingtong Yu, Jie Zhang, Yuan Fang, and Renhe Jiang. Non-homophilic graph pre-training and prompt learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025.

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

[53] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020.

[54] Yifei Zhang, Hao Zhu, Zixing Song, Piotr Koniusz, and Irwin King. Spectral feature augmentation for graph contrastive learning and beyond. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[55] Zheng Zhang, Junxiang Wang, and Liang Zhao. Curriculum learning for graph neural networks: Which edges should we learn first. *Advances in Neural Information Processing Systems*, 2023.

[56] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. Imbalanced node classification with synthetic over-sampling. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[57] WANG Zhili, DI Shimin, CHEN Lei, and ZHOU Xiaofang. Search to fine-tune pre-trained graph neural networks for graph-level tasks. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 2024.

[58] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 2022.

[59] Xiao Zhou, Yong Lin, Weizhong Zhang, and Tong Zhang. Sparse invariant risk minimization. In *International Conference on Machine Learning*, 2022.

[60] Xiao Zhou, Weizhong Zhang, Hang Xu, and Tong Zhang. Effective sparsification of neural networks with global sparsity constraint. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[61] Zhilun Zhou, Yu Liu, Jingtao Ding, Depeng Jin, and Yong Li. Hierarchical knowledge graph learning enabled socioeconomic indicator prediction in location-based social network. In *Proceedings of the ACM Web Conference*, 2023.

[62] Jing Zhu, Yuhang Zhou, Vassilis N Ioannidis, Shengyi Qian, Wei Ai, Xiang Song, and Danai Koutra. Pitfalls in link prediction with graph neural networks: Understanding the impact of target-link inclusion & better practices. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 2024.

[63] Xiang Zhuang, Qiang Zhang, Keyan Ding, Yatao Bian, Xiao Wang, Jingsong Lv, Hongyang Chen, and Huajun Chen. Learning invariant molecular representation in latent discrete space. *Advances in Neural Information Processing Systems*, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper's contributions and scope are specified in the abstract and introduction, including problem formulation, algorithmic design, theoretical analysis, and experimental evaluation.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please check Section G.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Please check Section 5.2, Section A, and Section C.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Please check Section 6.1 and Section E.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Please check our code is available at `https://anonymous.4open.science/r/GraphTOP-9678`.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Please check Section 6.1 and Section E.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We reported the standard deviations in the experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please check Section E.4. We also evaluated time of execution and memory needed in the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We follow the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Please check Section H.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The data and related techniques are explicitly cited in the paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Proof of Theorem 1

Theorem 1. *Given two random variables G_1 and G_2 that follow the Gumbel distribution* $\texttt{Gumbel}(0,1)$*, for any probability $p_{ij} \in \mathbf{P}$, we have*

$$\texttt{Pr}\left(G_1 - G_2 + \log\left(\frac{p_{ij}}{1-p_{ij}}\right) \geq 0\right) = p_{ij}. \tag{16}$$

Proof. According to the definition of the Gumbel distribution, the probability density function of $\texttt{Gumbel}(\mu, 1)$ is

$$f(x;\mu) = e^{-(x-\mu)-e^{-(x-\mu)}}. \tag{17}$$

The cumulative distribution function of $\texttt{Gumbel}(\mu, 1)$ is

$$F(x;\mu) = e^{-e^{-(x-\mu)}}. \tag{18}$$

Obviously, we have

$$\texttt{Pr}\left(G_1 - G_2 + \log\left(\frac{p_{ij}}{1-p_{ij}}\right) \geq 0\right) = \texttt{Pr}\left(\left(\log\left(p_{ij}\right) + G_1\right) - \left(\log\left(1-p_{ij}\right) + G_2\right) \geq 0\right). \tag{19}$$

Let $x_1 = \log\left(p_{ij}\right) + G_1$, $x_2 = \log\left(1-p_{ij}\right) + G_2$. We know

$$x_1 \sim \texttt{Gumbel}(\log\left(p_{ij}\right), 1),\ x_2 \sim \texttt{Gumbel}(\log\left(1-p_{ij}\right), 1) \tag{20}$$

Then, we have

$$\begin{aligned}
&\texttt{Pr}\left(\left(\log\left(p_{ij}\right) + G_1\right) - \left(\log\left(1-p_{ij}\right) + G_2\right) \geq 0\right) \\
&= \texttt{Pr}(x_1 - x_2 \geq 0) \\
&= \int_{-\infty}^{+\infty}\int_{-\infty}^{x_1} f(x_2;\log\left(1-p_{ij}\right)) f(x_1;\log\left(p_{ij}\right)) dx_2 dx_1 \\
&= \int_{-\infty}^{+\infty} F(x_1;\log(1-p_{ij})) f(x_1;\log(p_{ij})) dx_1 \\
&= \int_{-\infty}^{+\infty} e^{-e^{-(x_1-\log(1-p_{ij}))}} \cdot e^{-(x_1-\log(p_{ij}))-e^{-(x_1-\log(p_{ij}))}} dx_1 \\
&= \int_{-\infty}^{+\infty} e^{-e^{-(x_1-\log(1-p_{ij}))}-(x_1-\log(p_{ij}))-e^{-(x_1-\log(p_{ij}))}} dx_1 \\
&= \int_{-\infty}^{+\infty} e^{-(x_1-\log(p_{ij}))-e^{-x_1}\cdot(e^{\log(1-p_{ij})}+e^{\log(p_{ij})})} dx_1 \\
&= \int_{-\infty}^{+\infty} e^{-(x_1-\log(p_{ij}))-e^{-x_1}} dx_1 \\
&= p_{ij}\int_{-\infty}^{+\infty} e^{-x_1-e^{-x_1}} dx_1 \\
&= p_{ij}\int_{-\infty}^{+\infty} f(x_1;0) dx_1 \\
&= p_{ij}
\end{aligned} \tag{21}$$

Therefore, we can conclude

$$\texttt{Pr}\left(G_1 - G_2 + \log\left(\frac{p_{ij}}{1-p_{ij}}\right) \geq 0\right) = p_{ij}. \tag{22}$$

□

Algorithm 1 GraphTOP

Input: Pre-trained GNN model f_{θ^*}, initial g_ω and t_ϕ, input graph $\mathcal{G}$, training node list $\mathcal{V}_L$, hyperparameters λ_1 and λ_2, learning rate η, training epochs E
Extract ρ-hop local subgraph $\mathcal{G}(v_i)$ for each node $v_i \in \mathcal{V}_L$
for $e = 1$ **to** E **do**
  Anneal temperature by $\tau = \max\left(0.97 \times (1 - e/E) + 0.03, 0.1\right)$
  for each v_i and its local subgraph $\mathcal{G}(v_i)$ **do**
    $\mathbf{S}(v_i) = \mathbf{A}(v_i)$
    for each node $v_j \in \mathcal{N}_\rho(v_i)$ **do**
      Compute $p_{ij} = \sigma\left(\mathbf{W}_2\left(\texttt{ReLU}\left(\mathbf{W}_1\left(\boldsymbol{h}_i + \boldsymbol{h}_j\right)\right)\right)\right)$
      Sample g_1 and g_2 from $\texttt{Gumbel}(0, 1)$
      $s_{ij} = \sigma\left(\dfrac{g_1 - g_2 + \log\left(\frac{p_{ij}}{1-p_{ij}}\right)}{\tau}\right)$
      Compute $e\left(p_{ij}\right) = p_{ij}\log p_{ij} + (1 - p_{ij})\log(1 - p_{ij})$
    end for
  end for
  Compute $\mathcal{L}_P(\phi, \omega) = \frac{1}{|\mathcal{V}_L|}\sum_{v_i \in \mathcal{V}_L} \ell\left(g_\omega\left(\left[f_{\theta^*}\left(\mathbf{S}(v_i), \mathbf{X}(v_i)\right)\right]_i\right), y_i\right)$
  Compute $\mathcal{L}_E(\phi) = \frac{1}{|\mathcal{V}_L|}\sum_{v_i \in \mathcal{V}_L}\left(\frac{1}{|\mathcal{N}_\rho(v_i)|}\sum_{v_j \in \mathcal{N}_\rho(v_i)} e\left(p_{ij}\right)\right)$
  Compute $\mathcal{L}_S(\phi) = \frac{1}{|\mathcal{V}_L|}\sum_{v_i \in \mathcal{V}_L}\left|\frac{\sum_{v_j \in \mathcal{N}_\rho(v_i)} p_{ij}}{|\mathcal{N}_\rho(v_i)|} - \gamma\right|$
  Update $\omega \leftarrow \omega - \eta\nabla\mathcal{L}_P(\phi, \omega)$
  Update $\phi \leftarrow \phi - \eta\nabla\left(\mathcal{L}_P(\phi, \omega) + \lambda_1\mathcal{L}_E(\phi) + \lambda_2\mathcal{L}_S(\phi)\right)$
end for

B Overall Algorithm

The overall algorithm of GraphTOP is provided in Algorithm 1.

C Proof of Theorem 2

Theorem 2. *Given a pre-trained 2-layer linear GCN model f_{θ^*} and a random graph $\mathcal{G} \sim \texttt{CSBM}(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, p, q)$, when $p \neq q$, there always exists edge rewiring in the prompted graph by GraphTOP that satisfies*

$$\texttt{Dist}' = \frac{p+q}{|p-q|}\texttt{Dist}. \tag{23}$$

Proof. According to the CSBM, we suppose that the labels of a target node v_i's neighbors will be independently sampled from a neighborhood distribution $\mathcal{D}_{c_1} = [\frac{p}{p+q}, \frac{q}{p+q}]$ if node v_i is from class c_1 or $\mathcal{D}_{c_2} = [\frac{q}{p+q}, \frac{p}{p+q}]$ if node v_i is from class c_2 [29].

Without GraphTOP, the expected feature obtained from the first layer of the GCN operation will be

$$\mathbb{E}_{c_1}[\boldsymbol{h}^{(1)}] = \frac{p}{p+q}\cdot\boldsymbol{\mu}_1 + \frac{q}{p+q}\cdot\boldsymbol{\mu}_2 \tag{24}$$

for nodes from class c_1 and

$$\mathbb{E}_{c_2}[\boldsymbol{h}^{(1)}] = \frac{q}{p+q}\cdot\boldsymbol{\mu}_1 + \frac{p}{p+q}\cdot\boldsymbol{\mu}_2 \tag{25}$$

for nodes from class c_2. Here, we ignore the linear transformation in the weight matrices of the pre-trained GNN model, as it can be absorbed by the linear classifier.

Similarly, the expected feature obtained from the second layer of the GCN operation will be

$$\mathbb{E}_{c_1}[\boldsymbol{h}^{(2)}] = \frac{p}{p+q}\cdot\mathbb{E}_{c_1}[\boldsymbol{h}^{(1)}] + \frac{q}{p+q}\cdot\mathbb{E}_{c_2}[\boldsymbol{h}^{(1)}] \tag{26}$$

for nodes from class c_1 and

$$\mathbb{E}_{c_2}[\boldsymbol{h}^{(2)}] = \frac{q}{p+q} \cdot \mathbb{E}_{c_1}[\boldsymbol{h}^{(1)}] + \frac{p}{p+q} \cdot \mathbb{E}_{c_2}[\boldsymbol{h}^{(1)}] \tag{27}$$

for nodes from class c_2. When $p \neq q$, the nodes from the two classes are distinguishable from each other, i.e., $\mathbb{E}_{c_1}[\boldsymbol{h}^{(2)}] \neq \mathbb{E}_{c_2}[\boldsymbol{h}^{(2)}]$.

To evaluate the linear separability of linear classifiers, we calculate the expected distance $\texttt{Dist}$ between the two classes c_1 and c_2 by

$$\begin{aligned}
\texttt{Dist} &= \left\| \mathbb{E}_{c_1}[\boldsymbol{h}^{(2)}] - \mathbb{E}_{c_2}[\boldsymbol{h}^{(2)}] \right\| \\
&= \left\| \frac{p-q}{p+q} \cdot \mathbb{E}_{c_1}[\boldsymbol{h}^{(1)}] + \frac{q-p}{p+q} \cdot \mathbb{E}_{c_2}[\boldsymbol{h}^{(1)}] \right\| \\
&= \frac{|p-q|}{p+q} \cdot \left\| \mathbb{E}_{c_1}[\boldsymbol{h}^{(1)}] - \mathbb{E}_{c_2}[\boldsymbol{h}^{(1)}] \right\| \\
&= \frac{|p-q|}{p+q} \cdot \left\| \frac{p-q}{p+q} \cdot \boldsymbol{\mu}_1 + \frac{q-p}{p+q} \cdot \boldsymbol{\mu}_2 \right\| \\
&= \frac{(p-q)^2}{(p+q)^2} \cdot \|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\| .
\end{aligned} \tag{28}$$

Next, we consider the expected distance when rewiring edges with GraphTOP. Since GraphTOP only alters how every target node connects to the nodes within its 2-hop local subgraph, $\mathbb{E}_{c_1}[\boldsymbol{h}^{(1)}]$ and $\mathbb{E}_{c_2}[\boldsymbol{h}^{(1)}]$ are still the expected features for nodes from class c_1 and c_2, respectively. Let p' and q' denote intra-class and inter-class probabilities of edges between a target node and other nodes after rewiring edges with GraphTOP, respectively. Then, the new expected feature from the second layer of the GCN operation will be

$$\mathbb{E}'_{c_1}[\boldsymbol{h}^{(2)}] = \frac{p'}{p'+q'} \cdot \mathbb{E}_{c_1}[\boldsymbol{h}^{(1)}] + \frac{q'}{p'+q'} \cdot \mathbb{E}_{c_2}[\boldsymbol{h}^{(1)}] \tag{29}$$

for nodes from class c_1 and

$$\mathbb{E}'_{c_2}[\boldsymbol{h}^{(2)}] = \frac{q'}{p'+q'} \cdot \mathbb{E}_{c_1}[\boldsymbol{h}^{(1)}] + \frac{p'}{p'+q'} \cdot \mathbb{E}_{c_2}[\boldsymbol{h}^{(1)}] \tag{30}$$

for nodes from class c_2. In this case, the new expected distance after rewiring edges with GraphTOP will be

$$\begin{aligned}
\texttt{Dist}' &= \left\| \mathbb{E}'_{c_1}[\boldsymbol{h}^{(2)}] - \mathbb{E}'_{c_2}[\boldsymbol{h}^{(2)}] \right\| \\
&= \left\| \frac{p'-q'}{p'+q'} \cdot \mathbb{E}_{c_1}[\boldsymbol{h}^{(1)}] + \frac{q'-p'}{p'+q'} \cdot \mathbb{E}_{c_2}[\boldsymbol{h}^{(1)}] \right\| \\
&= \frac{|p'-q'|}{p'+q'} \cdot \left\| \mathbb{E}_{c_1}[\boldsymbol{h}^{(1)}] - \mathbb{E}_{c_2}[\boldsymbol{h}^{(1)}] \right\| \\
&= \frac{|p'-q'|}{p'+q'} \cdot \left\| \frac{p-q}{p+q} \cdot \boldsymbol{\mu}_1 + \frac{q-p}{p+q} \cdot \boldsymbol{\mu}_2 \right\| \\
&= \frac{|p'-q'|}{p'+q'} \cdot \frac{|p-q|}{p+q} \cdot \|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|
\end{aligned} \tag{31}$$

Then we have

$$\texttt{Dist}' = \frac{|p'-q'|}{p'+q'} \cdot \frac{p+q}{|p-q|} \texttt{Dist}. \tag{32}$$

With the loss term $\mathcal{L}_E$, each probability p_{ij} is forced to be 0 or 1. To ensure that the nodes from the two classes c_1 and c_2 both have the probability of p' to connect intra-class nodes, we may compute p_{ij} in Equation (8) to be 1 as long as v_i are v_j are from the same class; otherwise, $p_{ij} = 0$. In this case, we will have $p' = 1$ and $q' = 0$ for each target node in the prompted graph. Hence, we can get

$$\texttt{Dist}' = \frac{p+q}{|p-q|} \texttt{Dist}. \tag{33}$$

□

Table 3: Basic information and statistics of graph datasets adopted in our experiments.

Dataset	#(Nodes)	#(Edges)	#(Features)	Average degree	Homophily ratio	#(Classes)
Cora	2,708	10,556	1,433	3.90	0.810	7
PubMed	19,717	88,648	500	4.49	0.802	3
Amazon	24,492	93,050	300	3.80	0.380	5
Minesweeper	10,000	39,402	7	3.94	0.683	2
Flickr	89,250	899,756	500	10.08	0.319	7

D Extension to graph-level tasks

For graph-level tasks, such as graph classification, we can similarly model edge rewiring as Bernoulli random variables. Since graphs in graph-level tasks are typically not very large (usually less than 10K nodes per graph), we do not need to restrict topology-oriented prompting to a multi-hop subgraph but instead apply it to the whole graph. In this case, GraphTOP is simplified for graph-level tasks. We keep empirical evaluation of GraphTOPP on graph-level tasks as our future work.

E More details about experimental setup

E.1 Datasets

We use five real-world graph datasets to evaluate the performance of GraphTOP. The statistics of these datasets can be found in Table 3.

E.2 Pre-training strategies

We adopt four representative methods for pre-training GNN models in our experiments. These methods are also adopted by the previous graph prompting studies [27, 35, 36]. The details of these pre-training strategies are listed here.

- GraphCL [49] generates two perturbed views of a graph using node dropping and edge perturbation. A GNN model encodes both views into representations, which are then mapped to a latent space using a nonlinear projection head. The contrastive loss is applied to maximize agreement between the two views, optimizing both the GNN model and the projection head.
- SimGRACE [45] constructs a perturbed version of the GNN model by adding Gaussian noise to its parameters. Given an input graph, the perturbed and the original GNN models generate representations that form a positive pair for contrastive learning.
- LP-GPPT [35] randomly masks a subset of edges in the input graph. The model learns to predict whether a given pair of nodes is connected. Negative samples are generated by selecting node pairs that are not linked in the original graph.
- LP-GraphPrompt [27] samples a connected neighbor as one positive node and an unlinked node as one negative node for each target node. The training objective is to maximize the similarity between connected node pairs while minimizing the similarity between unconnected pairs.

We would like to emphasize that designing effective pre-training strategies for training powerful GNNs remains an ongoing challenge in graph learning. Many graph prompting methods [35, 27, 50] adopt link prediction as their pre-training strategies, while one recent study [51] discusses when to use link prediction or contrastive learning for pre-training. Although choosing a better pre-training strategy can yield more powerful GNN models, it is outside the scope of this study: we aim to obtain the best performance on downstream tasks given a pre-trained GNN model, regardless of its pre-training strategy.

E.3 Baselines

We use six SOTA baselines in our experiments. We provide the details of these baselines as follows.

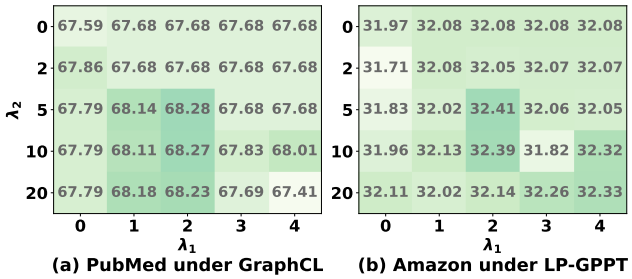

Figure 3: The average accuracies of GraphTOP with different values of λ_1 and λ_2.

- Linear Probe only trains a linear classifier during the adaptation phase without any graph prompting design.
- GPPT [35] pre-trains a GNN model via edge masking and link prediction, coupled with a task-specific prompt module that reparameterizes downstream node classification into edge likelihood estimation.
- All-in-one [36] reformulates node/edge tasks as graph tasks via multi-hop subgraphs, using learnable prompt graphs with token vectors, tunable token structures, and feature-similarity-weighted insertion patterns. It aligns downstream tasks with graph-level pre-training via episodic meta-learning over multi-task episodes.
- GraphPrompt [27] bridges the gap between pre-training and downstream tasks through subgraph similarity calculations as a unified template, with a learnable prompt updated during the adaptation phase to incorporate task-specific knowledge for tasks like node and graph classification.
- GraphPrompt+ [50] extends GraphPrompt by introducing prompt vectors within each layer of the pre-trained GNN model, effectively capturing hierarchical information beyond the readout layer to enhance adaptation.
- ProNoG [51]: ProNoG is a graph prompting framework for non-homophilic graphs. It employs a conditional network to generate node-specific prompts based on its multi-hop subgraph.

E.4 Hardware information

We run our experiments using a server equipped with 512 GB of memory, 128 AMD EPYC 7543 32-core CPUs, and 6 NVIDIA RTX A6000 GPUs, each of which has 48 GB of memory.

F More experimental results

F.1 Influence of λ_1 and λ_2

The two hyperparameters λ_1 and λ_2 balance different loss terms in the objective function of GraphTOP. To explore their influence on model utility, we conduct grid search for λ_1 and λ_2. Other robust optimization strategies, such as Bayesian Optimization and Hyperband [24], can also be used for hyperparameter selection. Figure 3 reports the accuracy results of GraphTOP with different values of λ_1 and λ_2 over PubMed under GraphCL and Amazon under LP-GPPT. According to the results, we observe that the performance of GraphTOP varies with different values of both λ_1 and λ_2. GraphTOP can obtain the best performance when we set $\lambda_1 = 2$ and $\lambda = 5$ or 10. Additionally, we also notice that the accuracies are not high but quite stable when $\lambda_2 = 0$. We conjecture that the probabilities are mostly driven toward values close to 1 by the loss term $\mathcal{L}_E$, leading to the same densely connected prompted graphs. When $\lambda_2 > 0$, however, the edge densities by GraphTOP are reduced. In this case, detrimental edges will be removed to enhance adaptation for pre-trained GNN models.

Table 4: GPU memory usage (GB) of GraphTOP and its variant when $\rho = 2$ and $\rho = 3$ (OOM: out of GPU memory).

Dataset	ρ	GraphTOP	GraphTOP$_{\text{all_nodes}}$
Cora	2	0.44	6.41
	3	0.69	OOM
PubMed	2	0.24	13.41
	3	2.33	OOM
Amazon	2	0.49	6.54
	3	0.58	OOM
Minesweeper	2	0.32	0.84
	3	0.36	2.79

Table 5: Accuracy with different numbers of shots. The best-performing method is **bolded**, and the runner-up is underlined.

3-shot			
Pre-training Strategies	**Graph Prompting Methods**	**Cora**	**Minesweeper**
GraphCL	Linear Probe	$51.76_{\pm 2.82}$	$\underline{63.93_{\pm 6.42}}$
	GPPT	$49.84_{\pm 3.51}$	$58.77_{\pm 6.21}$
	ALL-in-one	$47.69_{\pm 4.35}$	$52.40_{\pm 5.58}$
	GraphPrompt	$52.76_{\pm 4.94}$	$55.35_{\pm 8.01}$
	GraphPrompt+	$50.30_{\pm 4.16}$	$50.37_{\pm 6.99}$
	ProNoG	$\underline{52.94_{\pm 3.86}}$	$59.18_{\pm 7.93}$
	GraphTOP	$\mathbf{56.70_{\pm 5.70}}$	$\mathbf{64.70_{\pm 9.43}}$
10-shot			
Pre-training Strategies	**Graph Prompting Methods**	**Amazon**	**Minesweeper**
SimGRACE	Linear Probe	$\underline{25.36_{\pm 3.89}}$	$61.60_{\pm 3.93}$
	GPPT	$22.87_{\pm 6.08}$	$58.90_{\pm 4.70}$
	ALL-in-one	$18.36_{\pm 5.49}$	$57.64_{\pm 5.74}$
	GraphPrompt	$22.66_{\pm 2.00}$	$58.65_{\pm 5.34}$
	GraphPrompt+	$24.06_{\pm 2.47}$	$61.61_{\pm 5.64}$
	ProNoG	$22.88_{\pm 1.96}$	$\underline{62.01_{\pm 4.59}}$
	GraphTOP	$\mathbf{26.78_{\pm 4.48}}$	$\mathbf{63.20_{\pm 8.27}}$
20-shot			
Pre-training Strategies	**Graph Prompting Methods**	**PubMed**	**Amazon**
LP-GraphPrompt	Linear Probe	$73.38_{\pm 2.23}$	$28.50_{\pm 5.70}$
	GPPT	$75.92_{\pm 3.07}$	$27.54_{\pm 5.38}$
	ALL-in-one	$72.29_{\pm 5.42}$	$\underline{29.36_{\pm 4.91}}$
	GraphPrompt	$\underline{76.40_{\pm 1.81}}$	$25.00_{\pm 1.34}$
	GraphPrompt+	$70.77_{\pm 2.30}$	$26.41_{\pm 2.10}$
	ProNoG	$75.55_{\pm 1.69}$	$26.12_{\pm 1.99}$
	GraphTOP	$\mathbf{77.40_{\pm 5.06}}$	$\mathbf{32.40_{\pm 4.60}}$

F.2 Results of memory usage

Apart from time efficiency, we are also interested in GPU memory usage by GraphTOP and GraphTOP$_{\text{all_nodes}}$. Table 4 shows GPU memory usage by GraphTOP and GraphTOP$_{\text{all_nodes}}$ on four datasets. From the table, we notice that GraphTOP$_{\text{all_nodes}}$ requires more GPU memory resources compared with GraphTOP. The situation is even more pronounced than what we observed in Table 2. Considering this, our design for rewiring edges only between each target node and other nodes within its multi-hop local subgraph is very essential in GraphTOP.

F.3 Performance with different numbers of shots

We also conduct experiments with different numbers of shots. Table 5 shows the performance of GraphTOP and other baselines.

G Limitations

The theoretical analysis uses the CSBM [4] model to generate random graphs and analyze the linear separability of linear GCN models. Although it follows previous studies in graph learning, the nonlinear separability is also an important issue to explore.

H Broader impacts

This study will benefit many real-world applications related to graph prompting, such as anomaly detection and bad actor prediction.