Information Processing and Management 62 (2025) 104092

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/ipm

Information bottleneck-driven prompt on graphs for unifying downstream few-shot classification tasks

Xin Zhang [a,1], Wanyu Chen [c,1], Fei Cai [a,*], Jianming Zheng [b], Zhiqiang Pan [a], Yupu Guo [a], Honghui Chen [a]

[a] National Key Laboratory of Information Systems Engineering, National University of Defense Technology, Changsha, 410073, Hunan, China
[b] State Key Laboratory of Mathematical Engineering and Advanced Computing, Wuxi, 214000, Jiangsu, China
[c] College of Electronic Countermeasures, National University of Defense Technology, Hefei, 230000, Anhui, China

ARTICLE INFO

Keywords:
Graph Neural Network
Few-shot learning
Prompt
Information bottleneck
Diffusion

ABSTRACT

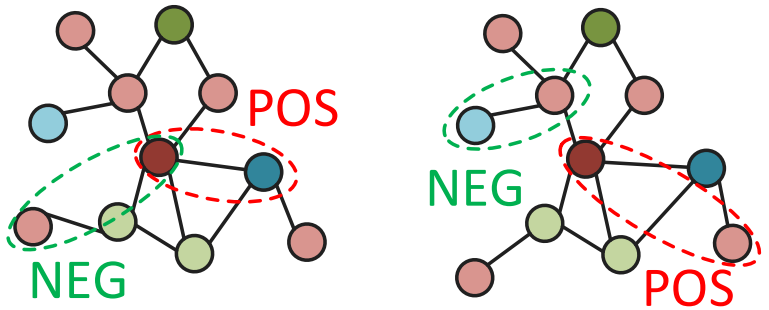
Inspired by the success of prompt in natural language processing, the graph prompt-based methods are proposed to solve the classification tasks under the conditions with limited instances. Recent graph prompt-based methods typically employ link prediction as the pre-training task, which brings a gap between the pre-training task and the downstream classification task, thus introducing irrelevant and noisy features to the downstream tasks. To tackle this issue, we propose a framework called Diffused Graph Prompt (Di-Graph), which consists of three major components, *i.e.*, the diffused pre-training, a graph prompt layer, and an information bottleneck optimizer. Specifically, the diffused pre-training aims to obtain the stable node features with a diffusion process, mitigating the gap between the pre-training tasks and the downstream tasks. The graph prompt layer enhances the pre-trained model to leverage its knowledge via capturing both the structural and node features in graphs. The information bottleneck optimizer helps the model discard redundant features by retaining the minimal sufficient statistic of the input data. Extensive experimental results on five public graph datasets demonstrate that our Di-Graph model surpasses the state-of-the-art model in terms of accuracy for both node and graph classification tasks. In particular, for graph-level tasks, Di-Graph achieves an average performance gain of 6.50% over the previous best model (GraphCL) on BZR dataset. For node-level tasks, Di-Graph achieves a 2.05% improvement over the best baseline (GraphCL) on PROTEINS dataset.

1. Introduction

Graph Neural Networks (GNNs) can effectively model non-Euclidean space data composed of interconnected nodes and edges. Upon that, GNNs are proficient in classifying the node and graph labels, as exemplified by their utility in analyzing social networks (Abdelrazek, Purificato, Boratto, & Luca, 2023; Liu, Cao, Yang, Liu, & Gao, 2022; Sun, Cheng, Liu, et al., 2023; Sun, Yin, et al., 2023), retrieving the scholar research field within citation networks (Huang, Lu, Liu, Cheng, & Bu, 2022; Yanardag & Vishwanathan, 2015), modeling the relationship between users and items in recommender systems (Liao, Deng, Wan, & Liu, 2022; Yao, Wang, Zhang, & Liang, 2023; Zhang, Yao, Sun, & Tay, 2019), and drug discovery (Du, Wang, Miao, & Zhang, 2021; Wang, Xiong, Wu, Sun, & Zhang, 2024). Current GNN-based classification approaches heavily depend on extensive labeled data to discern

* Corresponding author.
E-mail addresses: zhangxin16@nudt.edu.cn (X. Zhang), caifei08@nudt.edu.cn (F. Cai).
[1] Equal contributions.

https://doi.org/10.1016/j.ipm.2025.104092
Received 5 June 2024; Received in revised form 25 January 2025; Accepted 30 January 2025
Available online 13 February 2025



(a) The link prediction task in the pre-training phase.

(b) The downstream node classification task.

Fig. 1. Difference between the pre-training task and the downstream task.

the class-specific features (Chen, Jose, Yu, & Yuan, 2017; Ding et al., 2020; Zhou et al., 2019). Nonetheless, the acquisition of a substantial, well-labeled dataset is both time-consuming and labor-intensive.

In this light, researchers are striving to develop approaches tailored for few-shot scenarios, where the number of labeled samples is limited (Wang, Dong, Ding, Chen, & Li, 2023; Wang, Dong, Huang, Chen, & Li, 2022). The most seminal learning paradigm is the "pre-training, prompt and prediction" (Liu, Yu, Fang, & Zhang, 2023; Sun, Cheng, Li, & Liu, 2023; Sun, Zhou, He, Wang, & Wang, 2022), which constructs the diverse prompts to elicit knowledge from pre-trained language models, thereby enabling the completion of the downstream tasks. Liu, Yuan, et al. (2023) intuitively define a prompt as a type of clue provided to pre-trained language models, which enables the models to better understand human questions. Furthermore, from a technical perspective, the definition of a prompt is that it is a technical approach of leveraging the knowledge within pre-trained models by adding additional text to the input. Liu, Yuan, et al. (2023) Both definitions indicate that an appropriate pre-training strategy is important for prompt learning. This learning paradigm significantly reduces, if not obviates, the requirement for extensive labeled datasets (Liu, Ji, et al., 2022; Schick & Schütze, 2021; Wu et al., 2021). Notably, the state-of-the-art method, All-in-One (Sun, Cheng, Li, & Liu, 2023), utilizes the same prompt setting; however, its training mechanism is rooted in meta-learning (Finn, Abbeel, & Levine, 2017) and thus still necessitates a substantial amount of labeled data for training. In contrast, our work rigorously adheres to the true few-shot learning format (Perez, Kiela, & Cho, 2021), which fundamentally requires only a minimal number of labeled data.

Typically, in the prompt learning paradigm, the link prediction pre-training task is often used for node and graph classification (Liu, Yu, et al., 2023; Sun et al., 2022). However, we argue that employing such pre-training task may bring irrelevant and noisy information to the downstream classification tasks. The discrepancy may arise when the pre-trained models excel in the link prediction task by bringing the connected nodes close in the embedding space rather than the similar nodes. As shown in Fig. 1, in the link prediction task, the so-called sample bias does exist that models may select connected nodes of different classes as positive samples, while some negative nodes without connections are in fact semantically similar to each other. Even in some homogeneous graphs, links may appear between two nodes of different categories. For instance, graphs denote enzymes and nodes represent the secondary structure elements in ENZYMES dataset (Borgwardt et al., 2005; Schomburg et al., 2004). An edge connects two nodes if they are neighbors along the amino acid sequence or one of three nearest neighbors in space, which may belong to different categories (Morris et al., 2020). However, in the downstream classification task, the positive samples should belong to the same class. Consequently, a misalignment can occur between the pre-training task and the objectives of the downstream tasks, which leads to a negative transfer on downstream tasks. Additionally, the construction of prompts is another pivotal factor in prompt learning, as it dictates the extent to which knowledge can be elicited from a pre-trained model to facilitate the downstream tasks. Conventionally, the weight vectors are used as prompts and combined with node or graph embeddings through a Hadamard product to produce the task-specific representations (Liu, Yu, et al., 2023). However, such scheme overlooks the unique characteristics of graph data, causing the difficulty to effectively transfer the information learned from the pre-trained model to downstream tasks.

In summary, present methods mainly face the following issues:

- Traditional graph prompt learning frameworks rely on link prediction as the pretraining task, which may not align with the objectives of downstream classification tasks, potentially resulting in irrelevant or noisy features.
- The inherent scarcity of labeled samples in the pre-training phase leads to the knowledge gained during pretraining may not be fully applicable to downstream tasks.
- Furthermore, previous researches treat weighted vectors as prompts without considering the structural information of graph data, causing an inefficient transfer of knowledge from pretrained model to the downstream tasks.

Therefore, we propose the Diffused Graph Prompt (Di-Graph) to deal with the node and graph classification task in the true few-shot scenarios (Schick & Schütze, 2022). Initially, we introduce a diffusion-based strategy to sample nodes similar to the query nodes as positive instances. This strategy facilitates the generation of stable and robust node representations, thereby mitigating the

discrepancy between the pre-training and downstream tasks. Subsequently, to enhance the model's efficacy on handling the specific downstream tasks, we propose a graph convolution layer as the graph prompt. Following Liu, Yu, et al. (2023), we call our graph convolutional layer as *graph prompt layer*. This layer can leverage the knowledge acquired during the pre-training phase, capturing both the structural and node feature in graphs. Finally, to alleviate the impact of irrelevant and noisy features in the previously mentioned processes to generalize to downstream tasks, we devise an information-bottleneck-based loss function to optimize the prompt layer, retaining the minimal sufficient statistic of input data and discarding noisy information caused by the limited available samples.

We conduct extensive experiments on four public graph datasets for the node and graph classification tasks. The results show that Di-Graph can achieve a higher accuracy and better stability than the state-of-the-art models in the true few-shot scenarios. In summary, our main contributions are as follows:

- We introduce a diffusion-based sampling strategy for selecting appropriate instances for pre-training, which mitigates the gap between pre-training and downstream tasks by learning robust node representations.
- We employ a graph convolution layer as a prompt to guide the pre-trained model in completing various downstream tasks, thereby leveraging the structural information in graph.
- We design an information bottleneck-based optimizer as a flexible plug-in to the graph prompt-based models, to discard the irrelevant and redundant features captured in their pre-training stage.
- Extensive experimental results on five public graph datasets show the superiority of our model in the true few-shot scenarios for both the node and graph classification tasks.

2. Related work

2.1. Graph few-shot learning

Graph representation learning has garnered significant interest across various fields due to its exceptional ability to model and process graph data (Zhang, Ding, et al., 2022). For instance, Chen, Xi, et al. (2024) attempt to discover the influential nodes and Hao, Wang, Xiao, and Lin (2024) learn effective information from a multi-behavior network to improve the multi-behavior recommendation performance. Nonetheless, supervised graph representation learning often encounters the challenge of limited resources, as it typically relies on sparse labeled instances for learning. Furthermore, the process of data labeling is not only time-consuming but also requires substantial resources, which can encumber the conventional supervised learning approach. To address these issues, the concept of graph few-shot learning has emerged, aiming to mitigate the degradation in model performance that occurs under low-resource conditions. we primarily concentrate on node-level and graph-level few-shot learning techniques.

In the context of few-shot node classification, Yao et al. (2020) introduce the concept of constructing a graph-structure prototype to capture the relational structure of training samples. This prototype is then integrated into a representation module, which allows for the inclusion of global information. Recognizing that nodes within a graph have varying degrees of importance, Ding et al. (2020) develop a node significance measure using a self-attention mechanism (Vaswani et al., 2017). This approach computes a weighted summation of training nodes, resulting in a refined prototype for each node category. After that, Lan et al. (2020) present an embedding transformation function that maps task-agnostic node embeddings to task-specific ones (Zhang, Ding, et al., 2022). Tan, Ding, Guo, and Liu (2022) enhance the computation of class prototypes by employing both node-level and task-level attention mechanisms, which are optimized through an incremental learning strategy. In a more recent development, Wang et al. (2023) propose a method to generate pseudo-labeled nodes using Poison Learning and mitigate the issue of over-fitting through the application of meta-learning techniques.

For graph classification, Chauhan, Nathani, and Kaul (2020) pioneered metric-based few-shot learning with "Super-Class", which employs the Graph Isomorphic Network (GIN) (Xu, Hu, Leskovec, & Jegelka, 2019) to encode graphs and groups them into corresponding super-classes by calculating prototypical graphs based on their spectral properties. Additionally, optimization-based methods utilize model-agnostic meta learning (MAML) to optimize few-shot graph classification models. In particular, Ma et al. (2020) introduce AS-MAML to generate graph embeddings by concatenating the mean and max pooling of all node embeddings. For molecular graph classification problems, Meta-MGNN (Guo et al., 2021) treats individual molecules as graphs and utilizes a graph-level GNN to learn their embeddings. It innovatively introduces task-specific weights, allowing MAML to discern variations in molecular properties and thereby enhancing model optimization performance. Pre-PAR (Wang, Abuduweili, Yao, & Dou, 2021) builds upon Meta-MGNN by modeling the relational structure among various molecular properties, enabling the efficient propagation of scarce labels among similar molecules. Peng, Mo, et al. (2024) propose a contrastive-based unsupervised graph representation learning framework termed UGRL to obtain the downstream-related graph embeddings.

It is worth noting that the majority of current research in graph few-shot learning adheres to a meta-learning paradigm. This approach generally necessitates a considerable volume of data for the base classes within the dataset to construct effective meta-tasks.

2.2. Graph prompt learning

Owing to the exponential increase in model parameters, the conventional "pre-training and fine-tuning" pipeline (Zhang, Chen, et al., 2022) has given way to a "pre-training, prompting, and prediction" process (Liu, Yuan, et al., 2023). Such paradigm enables a single pre-trained language model to address a wide range of downstream tasks in an unsupervised manner. For instance, Peng, Cai, et al. (2024) apply large language models to process multi-modal medical data, which obtains disease concepts with GPT-4 and constructs graph structures. Then they employ graph neural networks to extract relationships between patches, and finally make diagnoses of neurological diseases. The efficacy of this approach has been amply demonstrated by the successful deployment of large language models in various applications (Sun, Zhang, et al., 2023).

In the context of prompt learning within the graph domain, Fang, Zhang, Yang, and Wang (2022) propose the basic prompt as a learnable vector, which can be added to all node features. GraphPrompt (Liu, Yu, et al., 2023) employs a prompt token akin to prior work (Fang et al., 2022) and assumes the prompt resides in the hidden layer of the graph model. GPPT (Sun et al., 2022) views graph prompts as supplementary tokens that encapsulate both task-specific and structure-representative information, aligning node classification with link prediction. Sun, Cheng, Li, and Liu (2023) propose to leverage an additional subgraph randomly connected to the original input graph as the graph prompt, which can be optimized via efficient tuning. G-prompt (Duan, Liu, Chen, Chen, & Wu, 2024) generate graph-level sub-prompts by sampling and integrate them for each class separately to create a precise graph-level prompt for a specific class. They suggest that prompt tokens should share the same semantic domain as the original node features, facilitating the seamless manipulation of node features with these tokens. The essence of prompting is to recast downstream tasks in the mold of the pre-training task. Therefore, Sun, Zhang, et al. (2023) apply prompt tuning to share the same objective as the pre-training task. For example, GraphPrompt employs a similar loss function to learn prompts, while both GPPT and VNT (Tan, Guo, Ding, & Liu, 2023) utilize the same loss function for their respective link prediction and node classification tasks. In the realm of extended studies on graph prompts, researchers (Chen, Zhang, et al., 2024; Sun, Yin, et al., 2023) develop frameworks tailored to the time-sensitive dynamics present in temporal interaction graphs. Additionally, Zhao, Chen, Sun, Cheng, and Li (2024) incorporate external knowledge from diverse graph datasets to identify potential commonalities among them, thereby enhancing the few-shot learning capabilities. Moreover, Zi et al. (2024) create an integrated graph prompt benchmark, with the objective of assessing the performance, flexibility, and efficiency of various prompt methods across multiple datasets.

In contrast, our proposal is centered on the application of graph prompt learning in **true** few-shot learning scenarios, where only a limited amount of well-labeled data is available. Specifically, the **true** few-shot scenario means that the absolute number of labeled samples is very scarce, for instance, only 5 samples per class are available (Liu, Yu, et al., 2023; Liu, Yuan, et al., 2023), as opposed to a small proportion of labeled samples relative to the entire dataset (Sun et al., 2022). It is also different from the meta-learning paradigm-based few-shot learning methods (Sun, Cheng, Li, & Liu, 2023), where there are a large number of learnable samples from old classes to construct a large amount of few-shot meta tasks, aiming to help the model adapt to few-shot scenarios. We aim to guide the pre-trained model in utilizing its acquired knowledge to achieve a variety of downstream tasks through the use of prompts. Besides, we propose to leverage information bottleneck to filter out irrelevant information from the pre-trained model, which is not pertinent to the downstream tasks, thereby further enhancing the specificity of the embeddings for downstream tasks.

3. Preliminary

3.1. Graph description

A graph can be defined as a tuple $G = (V, A)$, where $V = \{v_1, v_2, \cdots, v_N\}$ denotes the set of vertices, $N = |V|$ represents the number of vertices, and $A = \{0, 1\}^{N \times N}$ is the adjacent matrix indicating the connection between vertices, *i.e.* structure of the graph. The graph link is represented by $A_{i,j} = 1$ if node v_i and v_j are connected, otherwise $A_{i,j} = 0$. Moreover, an attributed graph $G = (V, A, X)$ has a node feature matrix $X \in \mathbb{R}^{N \times F}$, where F denotes the number of dimension. Note that each node v_i is described via the vector $\mathbf{x}_i \in \mathbb{R}^F$, which is the ith row in matrix X.

3.2. Task definition

As for the graph classification task, we consider that there is a universal set of graphs $\mathcal{G}$, which can be divided into two subsets. The first one $\mathcal{G}_l$ is a set containing all the labeled graph samples as well as all the categories in $\mathcal{G}$, and the other one $\mathcal{G}_u$ contains the remaining graph samples without labels, *i.e.* $\mathcal{G} = \mathcal{G}_l \cup \mathcal{G}_u$ and $\emptyset = \mathcal{G}_l \cap \mathcal{G}_u$. We use Y_G to denote the label set of graphs $\mathcal{G}$. The goal of graph classification is to train a model that can accurately predict the labels of those remaining unlabeled samples in set $\mathcal{G}_u$. Furthermore, for few-shot graph classification, we consider that there are only K labeled samples for each category, known as K-shot classification. Formally, $\mathcal{G}_l = \{G_1^{(1)}, \ldots, G_1^{(K)}, \ldots, G_{|Y_G|}^{(1)}, \ldots, G_{|Y_G|}^{(K)}\}$, where $|Y_G|$ denotes the size of the graph label set Y_G.

Similarly, a node classification task is conducted exclusively on one single graph $G = (V, A, X)$, where both the labeled nodes set V_l and the unlabeled one V_u reside in the same graph G, *i.e.* $V = V_l \cup V_u$ and $\emptyset = V_l \cap V_u$. The label set of nodes is denoted by Y_V. The goal of node classification is to efficiently predict the labels of unlabeled nodes in set V_u using a trained model.

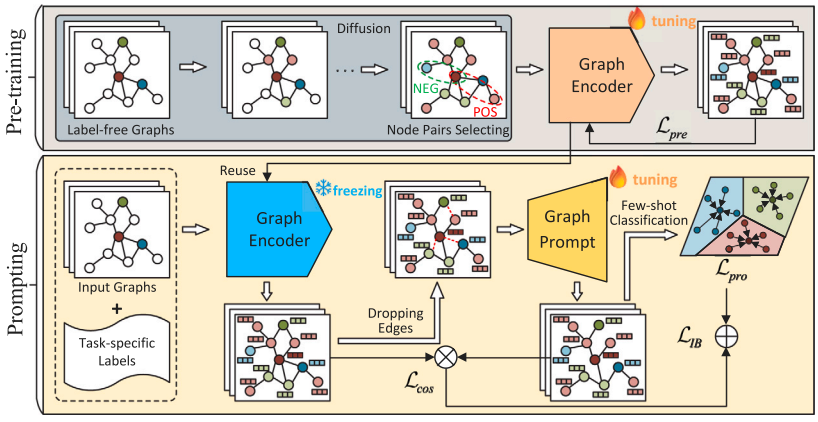


Fig. 2. The Di-Graph Framework.

4. Approach

In this section, we detail the Di-Graph framework, which consists of three main components, *i.e.*, diffused pre-training in Section 4.1, graph prompt tuning in Section 4.2 and information bottleneck optimizer in Section 4.3. The workflow of Di-Graph is plotted in Fig. 2. First, in the pre-training stage, a set of label-free graphs are fed into a diffusion function to obtain the node pairs for contrastive-based pre-training. Second, in the prompting stage, we feed the graphs into the pre-trained graph encoder for obtaining general embeddings of input graphs. Finally, to optimize the prompt layer, we compute the information bottleneck loss using both the general embeddings and the task-specific embeddings.

4.1. Diffused pre-training

In the pre-training stage, we attempt to narrow the gap between nodes in the same category and increase the distance between nodes with different categories. However, when the graph structure $A \in \mathbb{R}^{N\times N}$ and the feature matrix $X \in \mathbb{R}^{N\times F}$ are available but the node labels are unknown, we regard that nodes with high similarity have the label and those with low similarity have different labels.

Thus, the first step is selecting appropriate positive and negative instances for pre-training. Recent studies attempt to sample instances with an adjacency-based scheme, presenting a high sampling speed while a low quality due to the difference between the topological similarity and feature similarity (Liu, Yu, et al., 2023). Besides, some researchers calculate the similarity matrix for all nodes to select the nodes with the top-k highest similarity to a node as its positive samples. Such methods are time-consuming with high computational complexity especially on a large graph (Wang et al., 2020).

Thus, to striking a balance between sampling quality and spatial–temporal overhead, we sample the positive node instances in a label-free scenario with a diffusion-based strategy (Lee, Hyun, Lee, & Park, 2022), which can capture the remote information in graphs to enrich the samples for our pre-training task. Formally, given a graph G, we first feed it into a graph diffusion convolution function $GDC(\cdot)$ to obtain the diffused matrix S. Following the original graph diffusion (Klicpera, Weißenberger, & Günnemann, 2019), we utilize the personalized PageRank (Page, Brin, Motwani, & Winograd, 1999) as the diffusion algorithm:

$$S = GDC(G) = \sum_{t=0}^{\infty} \alpha(1-\alpha)^t T^t, \tag{1}$$

where $\alpha \in (0, 1)$ is the teleport probably (Chung, 2007), indicating the probability for a node to perform message passing through its connection. Each element s_{ij} in the diffusion matrix S represents the closeness of node v_i and v_j, and T is the symmetric transition matrix calculated by Eq. (2) with the *renormalization trick* (Kipf & Welling, 2017):

$$T = I_N + D^{-1/2} A D^{1/2}, \tag{2}$$

where A is the adjacent matrix of graph G, D is the diagonal matrix of node degrees, *i.e.* $D_{ii} = \sum_{j=1}^{N} A_{ij}$, and I_N is an identity matrix representing the self-loops in graph G. Be aware that diffusion the operation achieves a linear runtime $\mathcal{O}(N)$.

To obtain the positive sample v_q^{pos} of query node v_q, we select the index of the maximum value in row q as the positive node index, since the nodes with the high diffusion scores have a high probability of sharing the same label with the query node (Lee et al., 2022):

$$v_q^{pos} = \underset{m\in\{1,2,\ldots,N\}}{argmax} (S_{q,m}), \tag{3}$$

where $S_{q,m}$ denotes the value of mth element in the qth row in S.

For the negative instance v_q^{neg} of v_q, Liu, Yu, et al. (2023) employ the structure-based method for sampling. Thus, if a node has link to all node in a graph and without self loop, it will regard itself as a negative sample, which is clearly unreasonable. To tackle this issue, we add a self loop for each node and prioritize the sampling from nodes that are not connected to v_q with exceptions that v_q has links to all node in G, *i.e.* $A[:,q] = \mathbf{1} \in \mathbb{R}^N$. Otherwise, we directly regard the node with the minimum similarity as the negative sample:

$$v_q^{neg} = \begin{cases} v_m & , A_{m,q} = 0 \\ \underset{m \in \{1,2,\ldots,N\}}{argmax} \ (score(v_q, v_m)) & , A[:,q] = \mathbf{1} \in \mathbb{R}^N, \end{cases} \tag{4}$$

where $score(v_q, v_m)$ is a score function utilized for measuring the feature similarity between v_q and v_m based on their embedding $\mathbf{x}$, and v_m is a node in V:

$$score(v_q, v_m) = \frac{\mathbf{x}_q \cdot \mathbf{x}_m^\top}{\|\mathbf{x}_q\| \|\mathbf{x}_m\|}, \tag{5}$$

where $\|\cdot\|$ denotes the Euclidean norm. We feed the graph G into a GNN encoder to obtain the transformed graph G' with the transformed feature matrix X' for the lth layer in the GNN encoder:

$$\mathbf{x}_j^{(l)} = \text{UPDATE}\left(\mathbf{x}_j^{(l-1)}, \text{AGGREGATE}\left(\left\{\mathbf{x}_i^{(l-1)} | A_{ij} = 1\right\}\right)\right), \tag{6}$$

where UPDATE$(\cdot)$ represents a node feature updater, AGGREGATE$(\cdot)$ serves as an aggregation function of the neighborhood information. In this paper, the *sum* function is selected to aggregate. Note that the output of the last layer is considered as the feature of node in G', *i.e.*, $G' = (V, A, X')$.

Finally, we define a pre-training loss function via contrasting the similarity between query nodes and their corresponding positive as well as negative instances based on X':

$$\mathcal{L}_{pre} = -\sum_{v_q \in V} \log \frac{\exp(score(v_q, v_q^{pos}))}{\exp(score(v_q, v_q^{pos})) + \exp(score(v_q, v_q^{neg}))}. \tag{7}$$

Such pre-training loss aims to optimize the GNN encoder via conducting contrastive learning on a set of graphs without any label information. The optimized GNN encoder is frozen and applied to handle the downstream tasks.

4.2. Graph prompt tuning

However, because of the structural difference between the graph data and text data, the original format of the language prompt cannot adjust to graph prompt directly. Specifically, one obvious difference is that the natural language text mainly consists of one-dimensional, sequential structure, whereas the graph data embodies multi-dimensional, non-linear networked structure. From this perspective, the trainable text prompts guide the language models to generate the task-specific word vectors. Accordingly, we propose to employ a trainable graph convolutional layer $h_\phi(\cdot)$ to prompt the graph models in generating the task-specific graph vectors:

$$h_\phi(G') = \sigma(D'^{-1/2} A' D'^{1/2} X' W) \tag{8}$$

where G' is the output of the pre-trained GNN, W denotes the trainable weight matrix of $h_\phi(\cdot)$, ϕ denotes the parameters, and $\sigma(\cdot)$ is an activation function such as ReLU$(\cdot)$ or sigmoid$(\cdot)$. Note that $A' = A + I_N$ is the adjacent matrix added with self-connections, *i.e.*, $D'_{ii} = \sum_{j=1}^{N} A'_{ij}$. Via such message passing mechanism, $h_\phi(G')$ can utilize the structure for obtaining the feature vectors that aggregate the topological information from the graph. Then we describe the training of two classification tasks respectively.

4.2.1. Node classification

Following the description in Section 3, the node set V of graph G' can be divided into a k-shot labeled set V_l and an unlabeled one V_u, *i.e.*, $V = \{V_l, V_u\}$, where $V_l = \{v_1^{(1)}, \ldots, v_1^{(K)}, \ldots, v_{|Y_V|}^{(1)}, \ldots, v_{|Y_V|}^{(K)}\}$. Note that Y_V is the label set of nodes and its size is represented via $|Y_V|$. For the node classification task, we utilize the vectors in $\tilde{X}_{nt}$ as the node features, which are obtained by the node-level specific prompt $h_\phi^{node}(\cdot)$. To adjust the few-shot scenario, we introduce the metric-based learning via computing the distance between the query node v_q and the prototype c_{y_v} of class y_v, where $y_v \in Y_V$. Be aware that the prototype is a virtual node utilized as the class-level representation. The label of the query node v_q predicted to be the class of the prototype that is closest to it:

$$\hat{y}_q = \underset{y_v \in Y_V}{argmin} \ (dist(v_q, c_{y_v})), \tag{9}$$

where $dist(\cdot,\cdot)$ describes the distance between two nodes. For convenience, we define that $dist(\cdot,\cdot) = 1 - score(\cdot,\cdot)$ directly. To obtain the prototype embedding of class y_v, we use the mean average of these labeled node representations in class y_v as its prototype embedding $\tilde{\mathbf{x}}_{y_v}^c$:

$$\tilde{\mathbf{x}}_{y_v}^c = \frac{1}{K} \sum_{k=1}^{K} \tilde{\mathbf{x}}_{y_v}^{(k)}, y_v \in Y_V, \tag{10}$$

where $\tilde{\mathbf{x}}_{y_v}^{(k)}$ is a row vector in $\tilde{X}_{nt}$ produced by $h_{\phi}^{node}(G')$. Thus the loss for training the node classification task can be defined as $\mathcal{L}_{pro}^{v}$:

$$\mathcal{L}_{pro}^{v} = -\sum_{v_q \in V} \log \frac{\exp(score(v_q, c_{y_v}^{*}))}{\sum_{y_v \in Y_V} \exp(score(v_q, c_{y_v}))}, \tag{11}$$

where $c_{y_v}^{*}$ is the ground truth prototype of query node v_q.

4.2.2. Graph classification

For the graph classification task, we apply a graph-level specific prompt $h_{\phi}^{graph}(\cdot)$, where its structure is the same as $h_{\phi}^{node}(\cdot)$ defined in Section 4.2.1. Thus we can acquire the corresponding feature matrix $\tilde{X}_{gt}$.

Different from representing a node via a row in the feature matrix $\tilde{X}_{nt}$ in Section 4.2.1, we obtain the embedding of a query graph G_q by leveraging the whole matrix $\tilde{X}_{gt}$:

$$\tilde{\mathbf{x}}_q^{g} = Readout(\{\tilde{\mathbf{x}}_v | v \in V_q\}), \tag{12}$$

where V_q is the node set of query graph $\tilde{G}_q$, and $\tilde{\mathbf{x}}_v$ denotes the corresponding node feature in $\tilde{X}_{gt}$. $Readout(\cdot)$ is an integrating operation. Specifically, in this paper, we achieve it by summing up the representations of all nodes in $\tilde{G}_q$, *i.e.*, $\tilde{\mathbf{x}}_q^{g} = \sum_{v \in V_q} \tilde{\mathbf{x}}_v$. Similar to the node classification task, the graph prototype embedding of the graph label y_g can be obtained via:

$$\tilde{\mathbf{x}}_{y_g}^{c} = \frac{1}{K} \sum_{k=1}^{K} \tilde{\mathbf{x}}_{y_g}^{(k)}, y_g \in Y_G, \tag{13}$$

where Y_G is the graph label set, and $\tilde{\mathbf{x}}_{y_g}^{(k)}$ is a vector of a graph labeled as y_g. Moreover, the loss of the graph classification task $\mathcal{L}_{pro}^{g}$ can be calculated by:

$$\mathcal{L}_{pro}^{g} = -\sum_{\tilde{G}_q \in \mathcal{G}_u} \log \frac{\exp(score(\tilde{G}_q, c_{y_g}^{*}))}{\sum_{y_g \in Y_G} \exp(score(\tilde{G}_q, c_{y_g}))}, \tag{14}$$

where $c_{y_g}^{*}$ is the ground truth prototype of query graph $\tilde{G}_q$. Note that although the general form of prompt in Di-Graph is similar to GraphPrompt (Liu, Yu, et al., 2023), Di-Graph focuses on modeling and utilizing the structure features of graphs via the connections between the node embeddings. For GraphPrompt (Liu, Yu, et al., 2023), it computes the Hadamard Product of the graph embeddings and the weight vectors, neglecting the relations between nodes and resulting in the loss of information.

4.3. Information bottleneck optimizer

As a significant amount of noisy features may be introduced by the pre-training task and the limited labeled samples, the graph prompt module cannot transfer the information of pre-trained model directly. That is, solely relying on Eqs. (11) and (14) is insufficient for completing the downstream tasks in few-shot scenarios. Therefore, we attempt to mitigate the affect caused by irrelevant features by minimizing the information bottleneck (IB):

$$IB(\mathcal{D}, Y, Z) = [-I(Y; Z) + \beta I(\mathcal{D}; Z)], \tag{15}$$

where $\mathcal{D}$ is the original input, Y denotes the label set of queries, Z denotes the embeddings to be optimized, and β is a positive real number to balance these two terms. In this paper, $\mathcal{D}$ denotes the graph set $\mathcal{G}$ in the graph classification task, while in the node-level tasks, it denotes a single graph G. $I(\cdot;\cdot)$ represents the mutual information between two variables. The information bottleneck aims to learn the minimal sufficient representations for a given task by maximizing the mutual information between the embedding of input graphs and the target labels. Additionally, it simultaneously constrains the mutual information between the embeddings of input graphs and the original features of input graphs (Wu, Ren, Li, & Leskovec, 2020).

Specifically, the first term of Eq. (15) represents the mutual information between task-specific labels and graph embeddings. According to the relationship between the mutual information and entropy, we can formulate $I(Y; Z)$ as:

$$I(Y; Z) = H(Y) - H(Y|Z), \tag{16}$$

where $H(Y)$ denotes the entropy of Y. Note that when Y is a single dataset, its entropy can be regarded as a fixed value, thus we focus on the entropy between the task-specific labels and graph embeddings, *i.e.*, $H(Y|Z)$. Formally, following the definition of entropy, $H(Y|Z)$ can be represented as:

$$\begin{aligned} H(Y|Z) &= -\sum_{i=1}^{|Y|} \sum_{j=1}^{K} p(y_i, z_j) \log \frac{p(y_i, z_j)}{p(z_j)} \\ &= -\sum_{i=1}^{|Y|} \sum_{j=1}^{K} p(z_j|y_i) p(y_i) \log p(y_i|z_j). \end{aligned} \tag{17}$$

For a classification task, $p(z_j|y_i)$ can be calculated through an indicator function, *i.e.*, $p(z_j|y_i) = \mathbb{1}(z_j, y_i)$, where $p(z_j|y_i) = 1$ if y_i is the ground truth label of z_j, otherwise $p(z_j|y_i) = 0$. Note that the number of available instances for each category is K, thus we regard $p(y_i)$ is a uniform distribution. Therefore, the loss function $\mathcal{L}_{pro}$, *i.e.*, Eq. (11) for node classification and Eq. (14) for graph classification, refers to the first term $I(Y; Z)$ in Eq. (15).

For the second term in Eq. (15), *i.e.*, $I(\mathcal{D}; Z)$, it can be formulated in the form of an expectation as:

$$I(\mathcal{D}; Z) = E\left(\log \frac{p(Z|D)}{p(Z)}\right). \tag{18}$$

However, it is impractical to directly perform calculations using the above formula because we do not have access to $p(Z)$. Thus, we introduce an auxiliary distribution $r(Z)$ to decompose $p(Z)$ for the purpose of constructing a variational upper bound of Eq. (18):

$$\begin{aligned} E\left(\log \frac{p(Z|D)}{p(Z)}\right) &= E\left(\log \frac{p(Z|D)}{r(Z)} \frac{r(Z)}{p(Z)}\right) \\ &= E\left(\log \frac{p(Z|D)}{r(Z)}\right) - E\left(\log \frac{p(Z)}{r(Z)}\right) \\ &\leq E\left(\log \frac{p(Z|D)}{r(Z)}\right) \\ &= D_{KL}\left(p(Z|D) \| r(Z)\right), \end{aligned} \tag{19}$$

where $D_{KL}\left(p(Z|D) \| r(Z)\right)$ is the constructed variational upper bound, and $D_{KL}(\cdot \parallel \cdot)$ denotes the KL-divergence. Thus, the process of minimizing $I(\mathcal{D}; Z)$ can be transformed into minimizing its variational upper bound. In detail, we utilize the pre-trained GNN for modeling the conditional distribution $p(Z|D)$, and $r(Z)$ can be instantiated as any GNN model (Wang et al., 2023). In this paper, to reduce the number of parameters, we instantiate $r(Z)$ with the graph prompt layer $h_\phi(\cdot)$. Moreover, in order to eliminate the dependence of $r(Z)$ on original input, we randomly drop a certain proportion of edges in graphs before feeding them into the prompt layer. This allows us to approximately consider the perturbed graph embeddings output by the graph prompt layer as being independent of the input.

Note that to simplify the computation of the variational upper bound defined by Eq. (19), we transform it into calculating the cosine similarity between the embeddings obtained by the pre-trained GNN as well as the graph prompt layer (Wang et al., 2023):

$$D_{KL}\left(p(Z|D) \| r(Z)\right) \rightarrow \mathcal{L}_{cos} = -\frac{\tilde{\mathbf{x}} \cdot \mathbf{x}'^{\top}}{\|\tilde{\mathbf{x}}\| \|\mathbf{x}'\|}, \tag{20}$$

where $\mathbf{x}'$ denotes the output of the pre-trained model, and $\tilde{\mathbf{x}}$ denotes the output of graph prompt $h_\phi(\cdot)$. Therefore, our final downstream task loss is the information bottleneck, *i.e.*, Eq. (15), which can be represented as $\mathcal{L}_{IB}$:

$$IB(\mathcal{D}, Y, Z) \rightarrow \mathcal{L}_{IB} = \mathcal{L}_{pro} + \beta \mathcal{L}_{cos}, \tag{21}$$

Thus, the parameters of the graph prompt layer as well as the graph features can be learned with $\mathcal{L}_{IB}$.

5. Experiments

In this section, we first introduce the datasets in Section 5.1. Then, we list the research questions and related experiments configurations in Section 5.2. Finally, we describe the competitive baselines and our proposed models in 5.3.

5.1. Evaluation datasets

We employ four public benchmark datasets for our experiments. Table 1 lists the statistics of these datasets, all of which can be obtained conveniently[2]:[3]:

- **ENZYMES** (Borgwardt et al., 2005; Schomburg et al., 2004) is a bioinformatics dataset containing 600 enzymes collected from the BRENDA enzyme database. These samples include 6 categories of graphs, with an equal number of samples for each category, amounting to 100 instances per class.
- **PROTEINS** (Dobson & Doig, 2003) is a protein graphs dataset including the amino acid sequence, conformation, structure, and features such as active sites of proteins.
- **COX2** (Sutherland, O'Brien, & Weaver, 2003) is a dataset of molecular structures including 467 cyclooxygenase-2 inhibitors, each of which is denoted as a graph.
- **BZR** (Sutherland et al., 2003) is a dataset collecting 405 ligands for benzodiazepine receptor, each of which is represented by a graph. The whole dataset is divided into 2 categories.
- **PubMed** (Yang, Cohen, & Salakhutdinov, 2016) is a citation network dataset that contains more than 10,000 scientific publications on diabetes and more than 40,000 links between them from the PubMed database.

Note that the last column of Table 1 represents the task on the corresponding dataset, *i.e.*, "N" denotes node classification and "G" denotes graph classification.

Moreover, following Liu, Yu, et al. (2023), we conduct node classification on ENZYMES and PROTEINS. To ensure that each node category appears in a single graph and that there are sufficient nodes of each class within a single graph for both training and test, we perform filtering on the two datasets. Thus, the amount of graphs used for node classification in ENZYMES and PROTEINS are 53 and 404, respectively. In addition, for the graph classification task, relevant experiments are conducted on all four datasets.

[2] https://chrsmrrs.github.io/datasets/docs/datasets/
[3] https://github.com/kimiyoung/planetoid

Table 1
Statistics of the datasets used in our experiments.

Dataset	Graphs	Graph classes	Avg. nodes	Avg. edges	Node features	Node classes	Task
ENZYMES	600	6	39.06	72.82	18	3	N&G
PROTEINS	1113	2	41.22	43.45	1	3	N&G
COX2	467	2	32.63	62.14	3	/	G
BZR	405	2	35.75	38.36	3	/	G
PubMed	1	–	19,717	88,648	500	3	N

5.2. Research questions and configurations

5.2.1. Research questions

We examine the effectiveness of our solutions and concentrate on the following research questions to guide our experiments:

(RQ1) Does our proposal improve the performance of both node and graph classification in the few-shot scenarios compared to the baselines?
(RQ2) Among our proposed components, which one contributes the most to the performance of various downstream tasks?
(RQ3) How is the performance of discussed models for few-shot downstream tasks affected by different drop rates of edges in our information bottlenecks? Does our proposal present the robustness against the drop edges?
(RQ4) How is the performance of involved approaches for few-shot node and graph classification affected by the number of available instances?
(RQ5) What are the effects of different classification heads? Which is better, prototype network or linear classifier?

5.2.2. Model configuration

Following the description of tasks in Section 3.2, we mainly focus on the two types of downstream few-shot tasks, *i.e.*, the node and graph classification.

We pre-train the GIN model (Xu et al., 2019) as the backbone to embed the graphs into vectors. Moreover, for models that require pre-training such as GraphPrompt and our Di-Graph, we leverage all the content except for the labels in datasets for guiding the models to capture common knowledge in 400 epochs. We set the batch size to 600 for datasets ENZYMES, COX2, and BZR. Due to memory constraints, the batch size of dataset PROTEINES is set to 256. We adopt the ADAM (Kingma & Ba, 2015) optimizer with a step-decay learning rate for training in all cases. Because of the limitation of learnable samples in few-shot scenario, we set the epochs to 30 for the downstream tasks.

We tune the hyper-parameters of all models by grid search with the validation set, especially for determining the trade-off parameter β in Eq. (21), and we find that it works well when $\beta = 0.4$. For evaluation, we randomly select samples for 100 times on each dataset to conduct a series of few-shot training and testing for the downstream few-shot tasks. Finally, we compute the average accuracy across these results to mitigate randomness. All experiments can be conducted on a computer equipped with an NVIDIA RTX3090 GPU and 64 GB of memory, utilizing PyTorch version 1.13.1 and DGL version 0.9.1.

5.3. Model summary

We evaluate Di-Graph against the typical and state-of-the-art solutions from three main types:

(1) End-to-End approaches

- **GCN** (Kipf & Welling, 2017) employs a neighborhood aggregation strategy based on mean-pooling to gather messages from neighboring nodes in order to learn node representations effectively.
- **GAT** (Velickovic et al., 2018) also relies on neighborhood aggregation for end-to-end node representation learning, but with the capability of assigning distinct weights to neighbors in order to recalibrate their individual contributions.
- **GIN** (Xu et al., 2019) effectively captures structural information through a Weisfeiler-Lehman-like aggregation process, improving upon previous GNN architectures.
- **GraphSAGE** (Hamilton, Ying, & Leskovec, 2017) is a scalable inductive framework for generating node embeddings by sampling and aggregating neighborhood information in large graphs, enabling effective predictions for unseen nodes.

(2) Pre-training approaches

- **GraphCL** (You et al., 2020) leverages various graph augmentation techniques to exploit the structural information in graphs and is designed to maximize the consistency between different augmentations during the process of graph pre-training.
- **InfoGraph** (Sun, Hoffmann, Verma, & Tang, 2020) obtains a graph-level representation, which maximize the mutual information between the graph-level representation and substructure representations at diversity scales. Without loss of fairness, we discuss its performance only on graph-level tasks.

Table 2
Graph classification performance in terms of Accuracy (%) for 5-shot few-shot tasks. The results produced by the best baseline and the best performer in each column are underlined and boldfaced, respectively.

Training schemes	Methods	Graph Classification			
		ENZYMES	PROTEINS	COX2	BZR
Supervised	**GCN**	23.37 ± 7.19	58.56 ± 14.32	50.61 ± 10.55	59.17 ± 10.75
	GAT	24.04 ± 6.53	60.02 ± 12.82	51.66 ± 11.75	61.26 ± 10.30
	GIN	22.79 ± 5.12	59.92 ± 8.14	49.44 ± 5.60	55.72 ± 6.07
	GraphSAGE	23.24 ± 6.30	55.61 ± 14.26	51.54 ± 9.94	59.96 ± 9.98
Pre-train	**GraphCL**	20.69 ± 5.33	63.93 ± 8.02	56.39 ± 10.91	66.94 ± 9.81
	InfoGraph	23.68 ± 3.13	61.98 ± 10.01	53.71 ± 9.47	59.06 ± 10.32
Prompt	**GraphCL + GPF**	22.13 ± 4.04	59.82 ± 3.05	**58.48 ± 8.30**	63.05 ± 10.79
	GraphPrompt	22.14 ± 7.35	52.44 ± 13.50	50.55 ± 10.69	53.02 ± 5.23
	Di-Graph	**25.47 ± 6.62**	**67.14 ± 10.39**	**58.48 ± 10.29**	**73.44 ± 10.34**

Table 3
Node classification performance in terms of Accuracy (%) for 5-shot few-shot tasks. The results produced by the best baseline and the best performer in each column are underlined and boldfaced, respectively.

Training schemes	Methods	Node classification		
		ENZYMES (5-shot)	PROTEINS (5-shot)	PubMed (1-shot)
Supervised	**GCN**	60.98 ± 17.10	55.70 ± 11.49	33.05 ± 13.31
	GAT	60.51 ± 16.68	55.37 ± 16.14	35.42 ± 8.34
	GIN	61.91 ± 15.46	59.63 ± 12.35	37.57 ± 9.19
	GraphSAGE	59.79 ± 17.26	55.98 ± 11.53	34.62 ± 10.14
Pre-train	**GraphCL**	62.22 ± 11.39	61.38 ± 11.20	**39.96 ± 9.81**
Prompt	**GPPT**	58.73 ± 12.31	60.71 ± 14.11	30.10 ± 11.29
	GraphCL + GPF	52.79 ± 5.67	59.45 ± 10.68	33.20 ± 7.92
	GraphPrompt	61.36 ± 12.11	60.57 ± 10.35	31.96 ± 9.03
	Di-Graph	**64.27 ± 10.34**	**62.77 ± 11.59**	39.60 ± 7.32

(3) Graph Prompt approaches

- **GPPT** (Sun et al., 2022) employs the link prediction task for pre-training, and designs a trainable prompt for the node classification task, which is aligned to a link prediction task. Without loss of fairness, we discuss its performance only on node-level tasks.
- **GPF** (Fang, Zhang, Yang, Wang, & Chen, 2023) is a universal prompt-based method for pre-trained GNN models, which operates on the input graph's feature space and can theoretically obtain an equivalent effect to any form of prompting function. Following Zi et al. (2024), we adapt the method for both graph and node classification tasks.
- **GraphPrompt** (Liu, Yu, et al., 2023) transforms link prediction pre-training task into similarity computation and employs a weight vector as the graph prompt for node and graph classification.

Note that the meta learning-based methods (Sun, Cheng, Li, & Liu, 2023) are not discussed in our experiments, as they typically require a large amount of labeled data in their base class for meta training. Our proposal only leverage label-free graphs for pre-training. In addition, we do not discuss the GPPT (Sun et al., 2022) model as well. As it can only solve the node classification task while our model can be applied to the node and graph classification tasks.

5.4. Overall performance (RQ1)

To address RQ1, we evaluate the few-shot graph and node classification capabilities of our proposed method as well as the baseline models within a 5-shot setting. The classification performance in terms of accuracy of all discussed models is detailed in Tables 2 and 3.

Turning to the baselines, Table 2 shows that GraphCL consistently achieves a higher accuracy compared to all competitive baseline methods. Classical GNN models, *e.g.*, GAT (Velickovic et al., 2018) and GIN (Xu et al., 2019), present a better performance for the graph-level task on the ENZYMES dataset and for the node-level task on the PROTEINS dataset, respectively. For graph classification, GraphCL notably improves the accuracy over the standard GNN baselines, except for the GAT method on the ENZYMES dataset. For instance, GraphCL achieves improvements of 3.81%, 4.73% and 5.68% in terms of accuracy compared to GAT on PROTEINS, COX2 and BZR, respectively.

This indicates that augmented contrastive learning is crucial for few-shot graph classification. Across four datasets, all models discussed achieve the lowest accuracy on the ENZYMES dataset, which comprises six categories of graphs. This suggests that an

Table 4
Ablation studies of Di-Graph for the 5-shot graph classification tasks. The biggest drop in each column is appended with ▼.

	Graph Classification			
	ENZYMES	PROTEINS	COX2	BZR
Di-Graph w/o diffusion based sampling	22.38 ± 6.12 (−3.09%)	63.21 ± 11.97 (−3.93%)	50.14 ± 10.91 (−8.34%)	65.97 ± 9.20 (−7.47%)
Di-Graph w/o information bottleneck	24.51 ± 6.75 (−0.96%)	62.03 ± 11.60 (−5.11%)	49.79 ± 10.84 (−8.69%)	67.99 ± 11.59 (−5.45%)
Di-Graph w/o message passing	22.75 ± 6.96 (−2.72%)	57.47 ± 7.58 (−9.67%)	50.35 ± 5.83 (−8.13%)	68.97 ± 13.13 (−4.47%)
Di-Graph w/o graph prompt	21.75 ± 7.65▼ (−3.72%)	52.55 ± 6.08▼ (−14.59%)	49.65 ± 5.09▼ (−8.83%)	54.19 ± 6.69▼ (−19.25%)
Di-Graph	**25.47 ± 6.62**	**67.14 ± 10.39**	**58.48 ± 10.29**	**73.44 ± 10.34**

increased number of label categories leads to challenges for classification, especially in the context of limited samples. For the node classification task, GraphCL also maintains its superiority on ENZYMES, PROTEINS, and PubMed datasets. It even presents the best performance on PubMed with 1-shot setting. On the other hand, GraphCL + GPF only obtains the best performance on the COX2 dataset. This phenomenon is primarily due to the fact that the graph-level contrastive learning and data augmentation are effective in helping the model capture the global characteristics of the entire graph and the local information within node neighborhoods. Moreover, the design of the prompt is also crucial as it determines whether the model can harness the capabilities acquired during the pre-training phase. An inappropriate prompt not only fails to effectively leverage the model's performance but can also interfere with the abilities of the pre-trained model.

Next, we zoom in on the comparison between the baselines and our Di-Graph. It is obvious that Di-Graph consistently outperforms the competitive baselines on all datasets, yielding the best results for both the graph and node-level tasks. For the graph-level tasks, Di-Graph achieves relative improvements of 1.43%, 3.21%, and 6.5% over the most competitive baselines on the ENZYMES, PROTEINS, and BZR datasets,respectively. On the other hand, GraphPrompt, another prompt-based method, exhibits notably different performance from our Di-Graph model across various datasets. This is probably because that GraphPrompt chooses to pre-train their backbone GNN with link prediction tasks. Such format tends to cluster topologically similar nodes together in the embedding space, rather than ensuring that nodes with similar features are close to each other. Additionally, although the performance of GraphCL + GPF on COX2 is almost identical to our proposal, it underperforms our proposal on several other datasets. Even though Fang et al. (2023) theoretically proves that GPF can be equivalent to any prompt method, achieving the desired performance, it still requires a significant amount of work on tuning hyperparameters.

For the node-level task, while the advantage of our model is somewhat reduced, it still maintains its leadership among most of discussed models. Specifically, Di-Graph achieves improvements of 1.05% and 1.39% in terms of accuracy compared to the best baselines on ENZYMES and PROTEINS, respectively. Besides, it also outperforms the best graph prompt baseline GraphCL + GPF on the large-scale dataset PubMed under 1-shot setting, indicating the advantages of our proposals. In summary, the improvement of our Di-Graph over the baselines can be attributed to three aspects. The first one is the diffused pre-training technique. By searching for a similar node from an augmented graph via diffused conclusion, such technique avoids directly sampling topologically similar node. Consequently, compared to GraphPrompt method, Di-Graph reduces the discrepancy between the pre-training task and the actual downstream tasks, facilitating the application of features acquired during pre-training to downstream tasks. The second aspect is graph prompt tuning, which guides the pre-trained backbone GNN to utilize the information captured during pre-training stage. The third aspect is the information bottleneck optimizer. By introducing the information bottleneck mechanism, the prompt tuning stage mitigates the impact of irrelevant features captured during pre-training.

5.5. Ablation study (RQ2)

In order to better understand the contribution of different components to the classification performance, we conduct ablation studies with our Di-Graph across both the graph-level and node-level tasks. In these studies, we remove or replace some specific modules or mechanisms to assess their influence on Di-Graph, denoted by the notation "w/o". Specifically, "w/o diffusion pre-training" indicates that only the link prediction task from the baseline models is used as the pre-training task, thereby isolating the contribution of the diffusion selection mechanism for study; "w/o information bottleneck" indicates that the optimization target is changed from minimizing the information bottleneck to the widely used cross-entropy classification loss, lacking the comparison between the data with randomly dropped edges and the metadata. Additionally, since that the message passing mechanism is one of the key components of a GNN layer. Thus, "'w/o message passing'" indicates that the graph prompt layer is replaced with a fully connected layer, which means that the capability of the prompt layer to capture graph structural information is removed. This ignores the relationships between nodes and treats the node feature matrix as a regular matrix for processing. Furthermore, "w/o graph prompt" signifies the complete removal of the graph prompt mechanism, and the ablated Di-Graph can only rely on the pre-trained model and classifier to classify the input samples. The ablation results are detailed in Tables 4 and 5.

As shown in Table 4, "w/o graph prompt" causes the most significant performance degradation for the graph-level tasks. Specifically, this removal leads to relative decreases in terms of accuracy of 3.72%, 14.59%, 8.83% and even 19.25% across the

Table 5
Ablation studies of Di-Graph for the node classification tasks. The biggest drop in each column is appended with ▼.

	Node Classification		
	ENZYMES (5-shot)	PROTEINS (5-shot)	PubMed (1-shot)
Di-Graph w/o diffusion based sampling	61.41 ± 8.90 (−2.86%)	58.04 ± 9.47▼ (−4.73%)	38.22 ± 7.29 (−1.38%)
Di-Graph w/o information bottleneck	61.03 ± 8.31▼ (−3.24%)	58.69 ± 10.73 (−4.08%)	37.62 ± 5.61 (−1.98%)
Di-Graph w/o message passing	62.42 ± 10.47 (−1.85%)	60.56 ± 10.35 (−2.21%)	35.19 ± 9.73 (−4.41%)
Di-Graph w/o graph prompt	61.19 ± 11.74 (−3.08%)	58.97 ± 11.18 (−3.80%)	33.06 ± 8.14▼ (−5.54%)
Di-Graph	**64.27 ± 10.34**	**62.77 ± 11.59**	**39.60 ± 7.32**

datasets. This is mainly because that the removal of graph prompt eliminates the only trainable portion of Di-Graph. Consequently, its performance for the downstream tasks becomes entirely dependent on the effectiveness of the pre-training phase.

To further investigate the critical role of each component during the optimization process, we conduct experiments by separately removing the information bottleneck and message passing mechanisms.

For the graph-level tasks, the experimental results shown in Table 4 demonstrate that the removal of both the information bottleneck and message passing components has an obvious impact on the performance of Di-Graph. In particular, on the ENZYMES and PROTEINS datasets, "Di-Graph w/o message passing" results in a more severe performance degradation than removing the information bottleneck. This suggests that the graph convolution layer, which incorporates the message passing mechanism, is instrumental in harnessing the structural information encoded in the graph adjacency matrix. Consequently, despite having slightly more parameters than linear layers to optimize, graph convolution layers still outperform linear layers in scenarios where only a limited number of samples are available. On the other hand, on the COX2 and BZR datasets, "Di-Graph w/o information bottleneck" has a more pronounced impact than "Di-Graph w/o message passing".

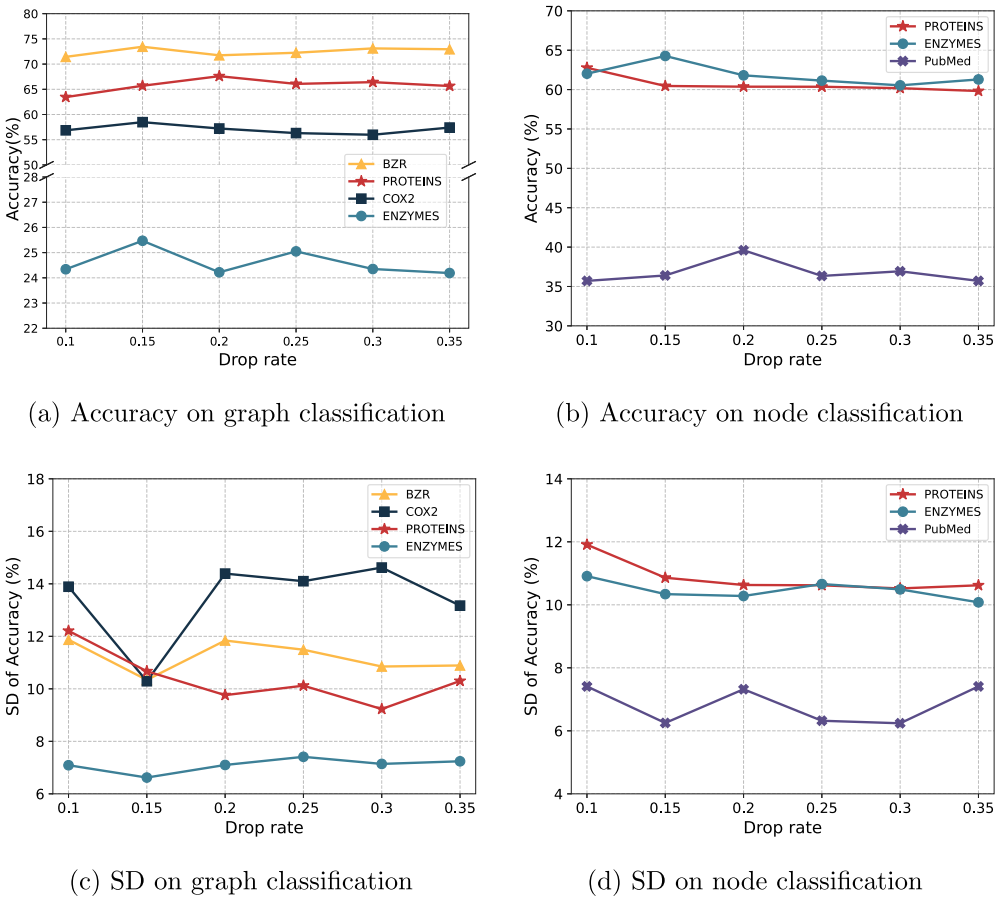
For the node-level tasks, the removal of the component of information bottleneck even results in the most substantial performance decline on the ENZYMES dataset. This can be attributed to the fact that the information bottleneck aims to make the perturbed node features resemble those of the original graph as output by the pre-trained model, thereby mitigating the impact of the perturbations. Thus, information bottleneck effectively retains those features that are robust to graph structure, thus reducing the sensitivity of the graph prompt module to the graph's structure. Additionally, the message passing mechanism shows its importance on PubMed under 1-shot setting. Such phenomenon indicates that on a large-scale graph, when only an extremely limited number of samples are known, the mechanism of message passing between nodes becomes more important than the information bottleneck. The role of the latter is primarily to enhance the robustness of the method.

5.6. Information bottleneck analysis (RQ3)

As previously discussed in Section 4.3, to eliminate the dependence of graph prompt layer on the original input, a certain proportion of edges in the original graphs are dropped randomly. Therefore, to answer RQ3 and assess the robustness of our Di-Graph framework, we conduct additional experiments at both the graph and node levels using six uniformly spaced values chosen from the interval $[0.1, 0.35]$.

For the graph-level tasks, when considering the overall performance, the classification performance in terms of accuracy exhibits a slight fluctuating trend across different datasets as the drop proportion increases. Such phenomenon indicates that our proposal is robust in general and demonstrates a relatively low sensitivity to the changes in the drop proportion. It is worth notifying that, for most of the datasets, a peak in terms of accuracy is observed at a drop proportion of 0.15 before decreasing as the proportion continues to increase. We consider that this phenomenon occurs because the degree of perturbation does not cause a significant damage to the inherent characteristics of the graph. In fact, it enhances the model's generalization capabilities. In addition, the standard deviation of the accuracy also displays a modest fluctuating pattern in response to the variations in the drop ratio. Similar to the trend in terms of accuracy, the standard deviation also reaches its minimum at a drop ratio of 0.15, which is particularly pronounced on the COX2 dataset.

For the node-level tasks, the classification accuracy of Di-Graph exhibits a slight downward trend as the drop proportion increases from 0.15, with higher accuracy values achieved at lower drop proportion. Additionally, for a large-scale graph, the classification accuracy achieve a peak when the drop rate is 0.2, indicating that the proportion of randomly dropped edges needs to be selected with an appropriate value, which is around 0.1 to 0.2 for most datasets to achieve good performance. The more edges that are dropped, the greater the impact on the information transfer mechanism, and the more interference the model will experience. Conversely, if too few edges are dropped, the data will be less perturbed. The model will not be able to learn more essential node representations from the perturbations. As shown in Figs. 3(c) and 3(d), from the perspective of standard deviation, the trend remains largely stable as the drop proportion increases, indicating that perturbations to the graph structure do not significantly impact the robustness of Di-Graph. Although there is a downward performance trend, even the lowest accuracy within the examined



(a) Accuracy on graph classification (b) Accuracy on node classification

(c) SD on graph classification (d) SD on node classification

Fig. 3. Graph and node classification performance in terms of accuracy and its standard deviation (SD) for 5-shot tasks under dropping different proportion of edges.

interval are higher than those obtained when no drop proportion is applied, *i.e.*, when removing the information bottleneck shown in Tables 4 and 5.

In summary, this comparative advantage further validates the effectiveness of the information bottleneck module for enhancing the performance of Di-Graph.

5.7. Impact of shot number (RQ4)

To address RQ4 and assess the impact of number of shots under few-shot conditions on the accuracy, we conduct experiments on the ENZYMES and PROTEINS datasets, selecting different types of baseline models, specifically GIN and GraphPrompt. We evaluated the performance of models in few-shot scenarios by conducting the graph classification tasks with shot numbers of 1, 2, 3, 4, 5, and 10, as well as the node classification tasks with shot numbers of 1, 2, 3, 4, and 5. The experimental results are presented in Fig. 4.

Overall, for all models examined, their accuracy tends to increase as the number of available samples increases. It is obvious that Di-Graph maintains its superiority for the graph-level tasks. Consistently as a prompt-based method, Di-Graph demonstrates greater gains when the number of samples increases than GraphPrompt. Specifically, as shown in Figs. 4(a) and 4(c), when only one shot per class is available, both models exhibit similar performance. However, when the number of samples per class is increased to five, Di-Graph outperform GraphPrompt by approximately 3%. Furthermore, the accuracy of GraphPrompt for 4-shot node classification task on PROTEINS is worse for the 3-shot task, indicating that its weight-based prompt has a limited capability to leverage more samples. The superiority of Di-Graph can be attributed to the fact that the graph convolutional layers used in our model have a strong inductive capability for graph data compared to weight vectors. Additionally, the information bottleneck optimizer filters out irrelevant features obtained during the pre-training phase. Such capability enables Di-graph to capture valuable information from just a limited number of samples and thereby swiftly enhances its performance. For the node-level tasks, Figs. 4(b) and 4(d) shows the similar trend to that of the graph-level tasks.

In summary, Di-Graph outperforms the involved baselines in node and graph classification tasks under few-shot conditions, which substantiates its capability to effectively capture both the global and local features. This ensures its excellence in performing two

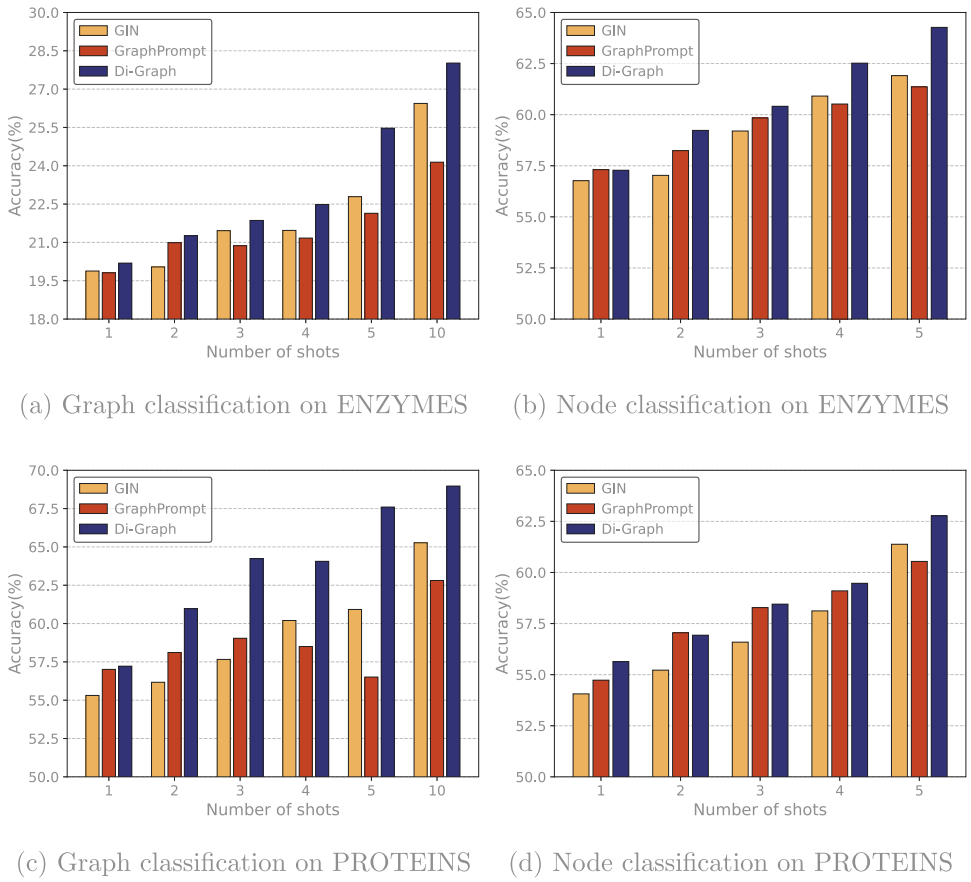


Fig. 4. Performance of Di-Graph with different number of samples on few-shot graph and node classification tasks.

distinct downstream tasks: the node-level classification and graph-level classification.

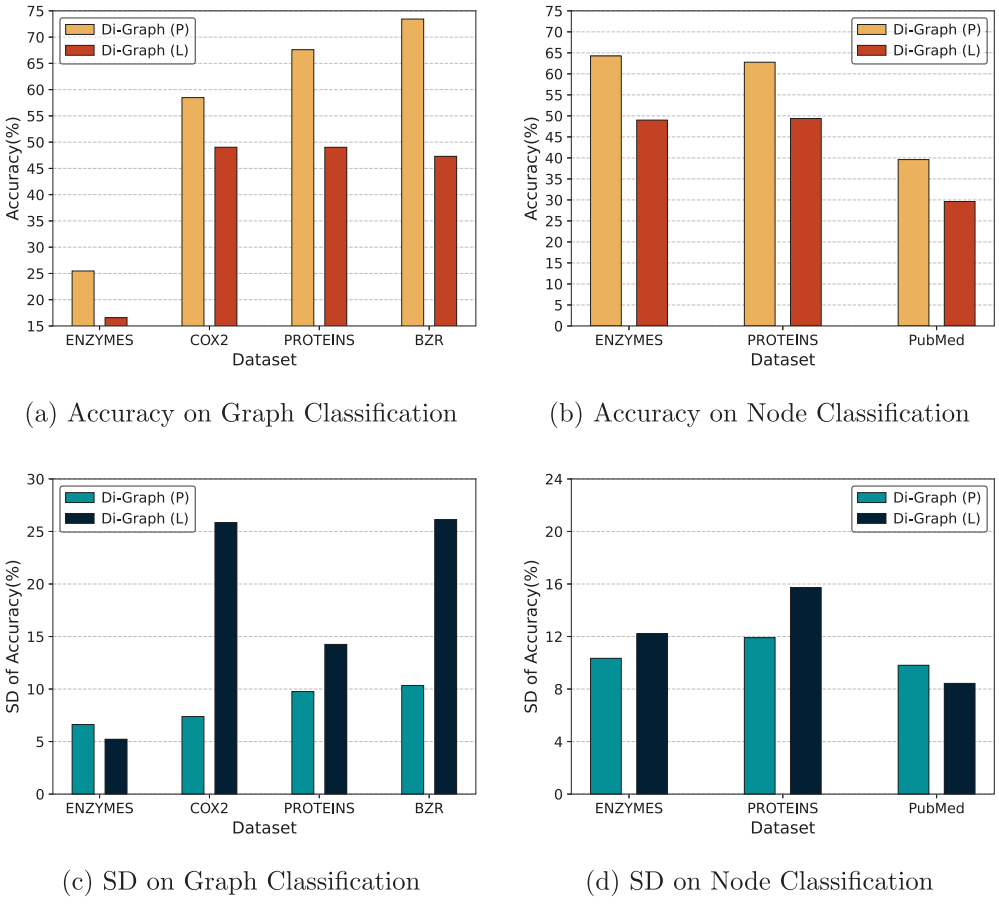
5.8. Classification head analysis (RQ5)

To address RQ5 and assess the impact of different classification heads, we conduct experiments with prototype network and a simple MLP linear layer as the classifier of Di-Graph. Specifically, we discuss the performance and robustness of different classifiers on the 5-shot graph classification and node classification tasks, respectively.

For the graph-level tasks, it is obviously that prototype network is more appropriate to serve as a classifier in few-shot scenario. As it shown in Figs. 5(a) and 5(c), Di-Graph with the prototype network, *i.e.*, Di-Graph(P) demonstrates its superiority on the whole five datasets. It outperforms the model variant with a linear classifier on all five datasets termed Di-Graph(L). To be specific, Di-Graph(P) obviously exceeds Di-Graph(L) in terms of accuracy, especially on the BZR dataset. Additionally, Di-Graph(P) shows obviously better stability than Di-Graph(L) because it has a lower standard deviation, which illustrates smaller performance fluctuations.

For node-level tasks, the performance comparison between Di-Graph(P) and Di-Graph(L) exhibits a similar trend to what they exhibit in graph-level tasks. In detail, Di-Graph(P) has not only higher accuracy but also lower standard deviation than Di-Graph(L) on both ENZYMES and PROTEINS datasets, showing its effectiveness and robustness, respectively. On the other hand, Di-Graph(L) achieves the lower standard deviation on PubMed under 1-shot setting. Since each class prototype under 1-shot condition is referred to only one sample, the model's performance depends more on the pre-training strategy than on the classifier's parameters. For the linear classifier, its optimization is driven by extremely limited number of samples. Therefore, it can only capture the local features of one class, resulting in overfitting and low accuracy.

Based on the experimental results, we consider that the prototypical network's superior performance stems from its metric-based learning approach, which classifies by calculating distances in embedding space instead of introducing any additional parameters. However, optimizing a linear classifier requires a large number of samples, which is not applicable in the experimental setting of this paper, namely the true few-shot scenario.



(a) Accuracy on Graph Classification

(b) Accuracy on Node Classification

(c) SD on Graph Classification

(d) SD on Node Classification

Fig. 5. Performance and robustness of Di-Graph with different classifiers on few-shot graph and node classification tasks.

5.9. Case study

To demonstrate how diffusion pre-training operates, we conducted a case study. Specifically, we selected a graph from the ENZYMES dataset, which comprises 84 nodes and 152 edges, for our experiment. The initial graph structure is depicted in Fig. 6(a), with each color indicating a different class of nodes. The figure reveals several isolated nodes with no connections. When a link prediction task is employed for pre-training on this graph, these isolated nodes will end up with only self-loops. Furthermore, **node 13** and **node 80** will be considered as positive examples, but they do not belong to the same category.

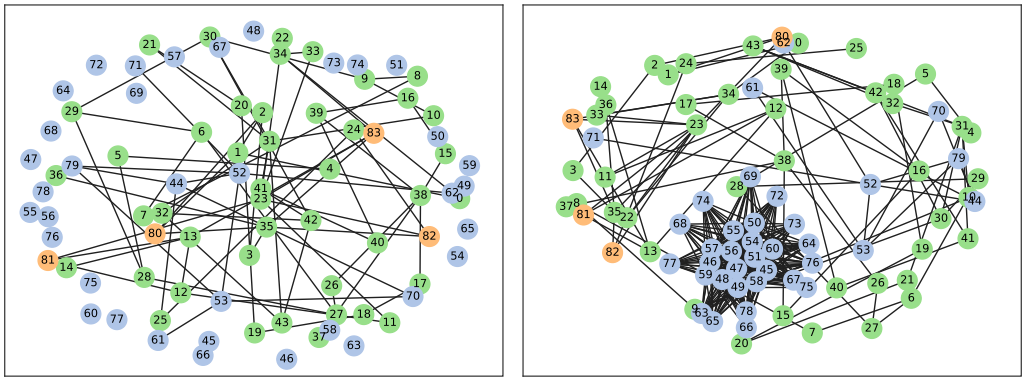
Fig. 6(b) illustrates that, using the diffusion strategy, the originally isolated nodes are now connected to nodes of the same category, forming edges that were not present in the original graph. As a result, each node is able to find its own positive example. In this way, the model can achieve better pre-training results based solely on the topological structure in an unsupervised manner.

6. Theoretical and practical implications

6.1. Theoretical implications

Diffusion-based Pretraining: Current methods often use link-prediction-based pretraining to improve the initialization parameters of models for downstream tasks. However, this approach may not align with the goals of specific downstream tasks, as link prediction focuses on bringing connected nodes closer in the embedding space, rather than grouping nodes by feature similarity. Our goal is to learn node embeddings that are more suitable for classification tasks, even without labeled data during pretraining. To achieve this, we employ a diffusion-based pretraining strategy that samples nodes similar to the query node, producing embeddings that are better suitable for classification tasks.

Graph Prompt Layer: There is no widely recognized definition of the form of graph prompts. Liu, Yu, et al. (2023) employ a linear layer as a prompt while Sun, Cheng, Li, and Liu (2023) use extra nodes as prompt tokens. However, the linear layer cannot capture the structural information of the graph input and the embeddings of extra nodes should be trained by meta-learning



(a) The original graph (b) Graph with diffusion strategy

Fig. 6. Case study for diffusion.

paradigm, which is different from the **true** few-shot scenarios discussed in our paper following Liu, Yu, et al. (2023). Our goal is to tackle the node-level and graph-level classification tasks using the same pretrained backbone in true few-shot settings. To this end, we introduce a graph convolutional layer designed to capture both the structural and node features of the graph data simultaneously.

Information Bottleneck Optimizer: Due to the discrepancy between pre-training and downstream classification tasks, the embeddings produced by the pre-training backbone often contain irrelevant and noisy information for downstream tasks. In this paper, we aim to preserve features of the embeddings that are closely related to the downstream task while discarding those that are not. To achieve this, we introduce an information bottleneck mechanism to reduce the impact of irrelevant and noisy features on the **true** few-shot learning optimization process. Consequently, the model can use the refined embeddings to outperform baseline models, even with a limited number of samples.

6.2. Practical implications

This paper constructs a framework capable of performing classification tasks on graph data at both the node and graph levels using a pretrained model in true few-shot scenarios. The proposed framework has several practical implications, particularly in analyzing the real-world complex networks. As the scale of real-world networks grows rapidly, for instance, with the addition of new users in social networks and new authors or papers in citation networks, the categorical labeling of this new data often fails to meet the requirements of models used for network analysis. Our model can derive stable node features, and the graph prompt layer completes specific tasks by capturing both structural and node features. Additionally, the information bottleneck optimizer guides the model in retaining features relevant to the downstream tasks.

7. Conclusion and future work

We introduce a Diffused Graph Prompt framework (Di-Graph) to unify the few-shot graph-level and node-level classification task with "pre-training, prompt tuning, and prediction" paradigm. In detail, Di-Graph leverages diffused pre-training for capturing the feature from label-free graphs, graph prompt tuning for unifying the graph-level and node-level tasks. In addition, an information bottleneck optimizer is utilized for filtering the irrelevant features captured during pre-training phase. Experimental results for the graph and node classification tasks on several datasets show the advantages of our proposal on improving the performance in true few-shot scenario.

As for future work, we plan to investigate how to deal with the few-shot graph tasks under an inductive setting. Furthermore, we plan to address the problem of cross-scale graph learning, which works in the form of training on a small-scale graph first and then applying the model to large-scale graphs. In addition to the node classification and graph classification, link prediction and clustering are also classic graph tasks. In future, we will expand the domain of prompts, not only constructing prompts suitable for the classification tasks but also attempting to build prompts for tasks such as link prediction and clustering. For instance, we consider that link prediction can be regarded as a binary classification task, *i.e.*, predicting whether a link exists between a node pair can be equivalent to determining the category ("True" or "False") of this node pair. On the other hand, an appropriate prompt for downstream task is necessary to leverage the capability of the model.

CRediT authorship contribution statement

Xin Zhang: Writing – original draft, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Wanyu Chen:** Writing – review & editing, Validation, Formal analysis. **Fei Cai:** Writing – review & editing, Supervision, Project administration, Investigation. **Jianming Zheng:** Visualization, Validation, Conceptualization. **Zhiqiang Pan:** Validation, Investigation, Formal analysis, Conceptualization. **Yupu Guo:** Visualization, Investigation, Data curation. **Honghui Chen:** Supervision, Resources, Investigation.

Acknowledgments

This research was partially supported by the National Natural Science Foundation of China under No.: 62302511 and No.: 62372458, Basic Strengthening Program (Young Elite Scientists Sponsorship Program) under No.: 2023-JCJQ-QT-048, Scientific Research Project of National University of Defense Technology under No.: ZK22-11, and the PostGraduate Scientific Research Innovation Project of Hunan Province under No.: CX20230076. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

Data availability

Data will be made available on request.

References

Abdelrazek, M., Purificato, E., Boratto, L., & Luca, E. W. D. (2023). Fairup: A framework for fairness analysis of graph neural network-based user profiling models. In *SIGIR* (pp. 3165–3169).

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S. V. N., Smola, A. J., et al. (2005). Protein function prediction via graph kernels. *Bioinformatics*, *21*(1), 47–56.

Chauhan, J., Nathani, D., & Kaul, M. (2020). Few-shot learning on graphs via super-Classes based on graph spectral measures. In *ICLR*.

Chen, L., Jose, J. M., Yu, H., & Yuan, F. (2017). A semantic graph-based approach for mining common topics from multiple asynchronous text streams. In *WWW* (pp. 1201–1209).

Chen, L., Xi, Y., Dong, L., Zhao, M., Li, C., Liu, X., et al. (2024). Identifying influential nodes in complex networks via Transformer. *Information Processing & Management*, *61*(5), Article 103775.

Chen, X., Zhang, S., Xiong, Y., Wu, X., Zhang, J., Sun, X., et al. (2024). Prompt learning on temporal interaction graphs. CoRR abs/2402.06326 arXiv:2402.06326.

Chung, F. (2007). The heat kernel as the pagerank of a graph. *Proceedings of the National Academy of Sciences*, *104*(50), 19735–19740.

Ding, K., Wang, J., Li, J., Shu, K., Liu, C., et al. (2020). Graph prototypical networks for few-shot learning on attributed networks. In *CIKM* (pp. 295–304).

Dobson, P. D., & Doig, A. J. (2003). Distinguishing enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology*, *330*(4), 771–783.

Du, J., Wang, S., Miao, H., & Zhang, J. (2021). Multi-channel pooling graph neural networks. In *IJCAI* (pp. 1442–1448).

Duan, Y., Liu, J., Chen, S., Chen, L., & Wu, J. (2024). G-Prompt: Graphon-based Prompt Tuning for graph classification. *Information Processing & Management*, *61*(3), Article 103639.

Fang, T., Zhang, Y., Yang, Y., & Wang, C. (2022). Prompt tuning for graph neural networks. CoRR abs/2209.15240 arXiv:2209.15240.

Fang, T., Zhang, Y., Yang, Y., Wang, C., & Chen, L. (2023). Universal prompt tuning for graph neural networks. In *neurIPS* (pp. 52464–52489).

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML* (pp. 1126–1135).

Guo, Z., Zhang, C., Yu, W., Herr, J., Wiest, O., et al. (2021). Few-shot graph learning for molecular property prediction. In *WWW* (pp. 2559–2567).

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *NeurIPS* (pp. 1024–1034).

Hao, Q., Wang, C., Xiao, Y., & Lin, H. (2024). Simplices-based higher-order enhancement graph neural network for multi-behavior recommendation. *Information Processing & Management*, *61*(5), Article 103790.

Huang, Y., Lu, W., Liu, J., Cheng, Q., & Bu, Y. (2022). Towards transdisciplinary impact of scientific publications: A longitudinal, comprehensive, and large-scale analysis on Microsoft Academic Graph. *Information Processing & Management*, *59*(2), Article 102859.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR (poster)*.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *ICLR (poster)*.

Klicpera, J., Weißenberger, S., & Günnemann, S. (2019). Diffusion improves graph learning. In *NeurIPS* (pp. 13333–13345).

Lan, L., Wang, P., Du, X., Song, K., Tao, J., & Guan, X. (2020). Node classification on graphs with few-shot novel labels via meta transformed network embedding. In *NeurIPS* (pp. 16520–16531).

Lee, N., Hyun, D., Lee, J., & Park, C. (2022). Relational self-supervised learning on graphs. In *CIKM* (pp. 1054–1063).

Liao, G., Deng, X., Wan, C., & Liu, X. (2022). Group event recommendation based on graph multi-head attention network combining explicit and implicit information. *Information Processing & Management*, *59*(2), Article 102797.

Liu, W., Cao, J., Yang, Y., Liu, B., & Gao, Q. (2022). Category-universal witness discovery with attention mechanism in social network. *Information Processing & Management*, *59*(4), Article 102947.

Liu, X., Ji, K., Fu, Y., Tam, W., Du, Z., et al. (2022). P-Tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *ACL no. 2* (pp. 61–68).

Liu, Z., Yu, X., Fang, Y., & Zhang, X. (2023). GraphPrompt: Unifying pre-training and downstream tasks for graph neural networks. In *WWW* (pp. 417–428).

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., et al. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, *55*(9), 195:1–195:35.

Ma, N., Bu, J., Yang, J., Zhang, Z., Yao, C., et al. (2020). Adaptive-step graph meta-learner for few-shot graph classification. In *CIKM* (pp. 1055–1064).

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., & Neumann, M. (2020). TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 workshop*.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web. In *The web conference*.

Peng, L., Cai, S., Wu, Z., Shang, H., Zhu, X., & Li, X. (2024). MMGPL: Multimodal medical data analysis with graph prompt learning. *Medical Image Analysis*, *97*, Article 103225.

Peng, L., Mo, Y., Xu, J., Shen, J., Shi, X., Li, X., et al. (2024). GRLC: graph representation learning with constraints. *IEEE Transcations of Neural Networks and Learning Systems*, *35*(6), 8609–8622.

Perez, E., Kiela, D., & Cho, K. (2021). True few-shot learning with language models. In *NeurIPS* (pp. 11054–11070).

Schick, T., & Schütze, H. (2021). Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL* (pp. 255–269).

Schick, T., & Schütze, H. (2022). True few-shot learning with prompts - A real-world perspective. *Transcations of the Association for Computational Linguistics*, *10*, 716–731.

Schomburg, I., Chang, A., Ebeling, C., Gremse, M., Heldt, C., Huhn, G., et al. (2004). BRENDA, the enzyme database: updates and major new developments. *Nucleic Acids Research*, *32*(1), D431–D433.

Sun, X., Cheng, H., Li, J., & Liu, B. (2023). All in one: Multi-task prompting for graph neural networks. In *KDD* (pp. 2120–2131).

Sun, X., Cheng, H., Liu, B., Li, J., Chen, H., et al. (2023). Self-supervised hypergraph representation learning for sociological analysis. *IEEE Transcations on Knowledge and Data Engineering*, *35*(11), 11860–11871.

Sun, F., Hoffmann, J., Verma, V., & Tang, J. (2020). InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *ICLR*.
Sun, X., Yin, H., Liu, B., Meng, Q., et al. (2023). Structure learning via meta-hyperedge for dynamic rumor detection. *IEEE Transcations on Knowledge and Data Engineering*, *35*(9), 9128–9139.
Sun, X., Zhang, J., Wu, X., Cheng, H., Xiong, Y., & Li, J. (2023). Graph prompt learning: A comprehensive survey and beyond. CoRR abs/2311.16534 arXiv:2311.16534.
Sun, M., Zhou, K., He, X., Wang, Y., & Wang, X. (2022). GPPT: graph pre-training and prompt tuning to generalize graph neural networks. In *KDD* (pp. 1717–1727).
Sutherland, J. J., O'Brien, L. A., & Weaver, D. F. (2003). Spline-fitting with a genetic algorithm: A method for developing classification structure activity relationships. *Journal of Chemical Information and Computer Sciences*, *43*(6), 1906–1915.
Tan, Z., Ding, K., Guo, R., & Liu, H. (2022). Graph few-shot class-incremental learning. In *WSDM* (pp. 987–996).
Tan, Z., Guo, R., Ding, K., & Liu, H. (2023). Virtual node tuning for few-shot node classification. In *KDD* (pp. 2177–2188).
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., et al. (2017). Attention is all you need. In *NIPS* (pp. 5998–6008).
Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., et al. (2018). Graph attention networks. In *ICLR (poster)*.
Wang, Y., Abuduweili, A., Yao, Q., & Dou, D. (2021). Property-aware relation networks for few-shot molecular property prediction. In *NeurIPS* (pp. 17441–17454).
Wang, S., Dong, Y., Ding, K., Chen, C., & Li, J. (2023). Few-shot node classification with extremely weak supervision. In *WSDM* (pp. 276–284).
Wang, S., Dong, Y., Huang, X., Chen, C., & Li, J. (2022). FAITH: few-shot graph classification with hierarchical task graphs. In *IJCAI* (pp. 2284–2290).
Wang, Y., Xiong, Y., Wu, X., Sun, X., & Zhang, J. (2024). Advanced drug interaction event prediction. CoRR abs/2402.11472 arXiv:2402.11472.
Wang, X., Zhu, M., Bo, D., Cui, P., Shi, C., & Pei, J. (2020). AM-GCN: adaptive multi-channel graph convolutional networks. In *KDD* (pp. 1243–1253).
Wu, T., Ren, H., Li, P., & Leskovec, J. (2020). Graph information bottleneck. In *NeurIPS* (pp. 20437–20448).
Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., et al. (2021). Self-supervised graph learning for recommendation. In *SIGIR* (pp. 726–735).
Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks? In *ICLR*.
Yanardag, P., & Vishwanathan, S. V. N. (2015). Deep graph kernels. In *KDD* (pp. 1365–1374).
Yang, Z., Cohen, W. W., & Salakhutdinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. In *ICML* (pp. 40–48).
Yao, T., Wang, Y., Zhang, K., & Liang, S. (2023). Improving the expressiveness of K-hop message-passing GNNs by injecting contextualized substructure information. In *KDD* (pp. 3070–3081).
Yao, H., Zhang, C., Wei, Y., Jiang, M., Wang, S., et al. (2020). Graph few-shot learning via knowledge transfer. In *AAAI* (pp. 6656–6663).
You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., et al. (2020). Graph contrastive learning with augmentations. In *NeurIPS* (pp. 5812–5823).
Zhang, S., Chen, H., Yang, H., Sun, X., Yu, P. S., & Xu, G. (2022). Graph masked autoencoders with transformers. CoRR 2202.08391.
Zhang, C., Ding, K., Li, J., Zhang, X., Ye, Y., et al. (2022). Few-shot learning on graphs. In *IJCAI* (pp. 5662–5669).
Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys*, *52*(1), 5:1–5:38.
Zhao, H., Chen, A., Sun, X., Cheng, H., & Li, J. (2024). All in one and one for all: A simple yet effective method towards cross-domain graph pretraining. In *KDD* (pp. 4443–4454).
Zhou, F., Cao, C., Zhang, K., Trajcevski, G., Zhong, T., et al. (2019). Meta-GNN: On few-shot node classification in graph meta-learning. In *CIKM* (pp. 2357–2360).
Zi, C., Zhao, H., Sun, X., Lin, Y., Cheng, H., & Li, J. (2024). ProG: A graph prompt learning benchmark. CoRR abs/2406.05346 arXiv:2406.05346.