MPrompt: A Pretraining-Prompting Scheme for Enhanced Fewshot Subgraph Classification

by

Muhua Xu

B.S. Computer Science and Engineering and Mathematics, MIT, 2024

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2024

Authored by:	Muhua Xu Department of Electrical Engineering and Computer Science May 10, 2024
Certified by:	Arvind Professor in Electrical Engineering and Computer Science, Thesis Supervisor
Certified by:	Jie Chen Senior Research Scientist, Thesis Supervisor
Certified by:	Xuhao Chen Research Scientist, Thesis Supervisor
Accepted by:	Katrina LaCurts Chair, Master of Engineering Thesis Committee

MPrompt: A Pretraining-Prompting Scheme for Enhanced Fewshot Subgraph Classification

by

Muhua Xu

Submitted to the Department of Electrical Engineering and Computer Science
on May 10, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

ABSTRACT

Motivated by the significant progress in NLP prompt learning, there have been great research interests recently in adopting the prompting mechanism for graph machine learning. Despite the prior success of prompting methods applied in node-level and graph-level learning tasks, subgraph-level tasks are highly underexplored, and the potential of prompting remains unclear. This thesis fills this gap by exploring the prompting mechanism for *subgraph classification*, which is a much more challenging task as it requires understanding both global and local graph structures.

In this work, we build upon state-of-the-art self-supervised graph learning models to develop a subgraph-specific prompting scheme ***Membership Prompt (MPrompt)*** based on traditional graph neural networks (GNN). Our proposed prompting scheme relies on ***node membership*** knowledge to help GNN distinguish between border and local connections, which increases its expressive power while maintaining the prompt's independence from any specific dataset or model architecture. Additionally, we also present Subgraph Reconstructive Pretraining (SRP) which can provide MPrompt with better structural embeddings during pretraining.

Experiments are conducted on both synthetic and real-world datasets, including protein function prediction and social network analysis. Our method demonstrated performance improvement under few-shot experiment setting and maintained comparable performance in full-shot settings while requiring less computation.

Thesis supervisor: Arvind
Title: Professor in Electrical Engineering and Computer Science

Thesis supervisor: Jie Chen
Title: Senior Research Scientist

Thesis supervisor: Xuhao Chen
Title: Research Scientist

Acknowledgments

I would like to express my deep gratitude to my advisor Professor Arvind and mentors, Dr. Jie Chen and Dr. Xuhao Chen for their guidance throughout my research journey, and to the entire research group — Locke Cai, Yitan Zhu, Michael Hadjiivanov, and Kiwhan Song — for their hard work and insightful discussions that significantly advanced this research.

Additionally, I would like to thank my family and friends for providing a supportive environment.

The success of this work would not have been possible without the combined efforts and faith of all these amazing people.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In contrast to text or images, which may be arranged sequentially or into grids, graph-structured data is represented in the form of node objects interconnected with edges. To effectively extract information from this intricate data structure, modern approaches to graph analysis typically rely on graph neural networks (GNNs) to create representations by aggregating neighborhood node information for graph deep learning.

Among the different tasks to be solved in graph deep learning, subgraph classification is one of the most challenging and understudied tasks among them. This is because the algorithm needs to understand not only the structure of the subgraph but also its connection to the background graph. Additional modifications are required to properly capture the distinct topology of subgraphs, as opposed to graph-level activities, which lack finer local structure, and node-level tasks, which might be detrimental to grasping an overarching view of the graphs. Subgraph tasks are essential for many real-world applications, such as finance and social networks, or scientific domains like biology and chemistry.

Recent advancements in GNNs have expanded their scope to address challenges associated with large-scale graphs featuring limited training labels, including issues like overfitting, domain shift, and unstable performance. In particular, graph-related tasks in scientific domains such as chemistry and biology require time-consuming and costly laboratory efforts for data

annotation. Moreover, the properties of training and testing graphs can vary considerably due to the nature of these tasks, complicating the task of achieving accurate model generalization. With inspiration adopted from natural language processing (NLP), researchers have shifted from the traditional supervised learning paradigm to a pre-training approach. This methodology involves training a generalized model on a self-supervised task before adapting it to a specific downstream task. Particularly in scenarios with a shortage of training labels, the "pre-training and prompting" approach emerges as a promising solution.

This thesis propose a specialized prompting methods with node membership knowledge dedicated to few-shot subgraph classification using GNN.

1.1 Graph Neural Network

Given that $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is a graph, with adjacency matrix $\mathbf{A}$ and input feature matrix $\mathbf{H}^0$ where each row of $\mathbf{H}^0$ is the feature vector of a node. For convenience, we denote individual feature vectors by using a lower-case letter, such as $\mathbf{h}_v^0$ for node v.

GNNs are used to learn a final feature vector for each vertex. The key idea is to construct a message passing framework, where each node obtains its representation by receiving and aggregating messages from its neighboring nodes recursively.

For a k-layer message-passing neural network (MPNN) [1] with input features $\mathbf{h}_v^0$, the resulting output feature vector would be $\mathbf{h}_v^k$ for each vertex v. At each time step t, we update the feature vector of each node as follows:

$$\mathbf{h}_v^t = \text{UPDATE}(\mathbf{h}_v^{t-1}, \text{AGGREGATE}(\mathbf{h}_u^{t-1} | u \in \mathcal{N}(v)))$$

where AGGREGATE is an aggregation function that combines all neighbors' features into a message feature (i.e., a weighted sum or mean function), and UPDATE controls how the message is used to update the current node feature (i.e., a simple addition or concatenation operation). A most basic aggregation step can be computed by a simple matrix multiplication

$\mathbf{H}^{i+1} = \mathbf{A}\mathbf{H}^{\mathbf{i}}\mathbf{W}$ involving the adjacency matrix $\mathbf{A}$, feature matrix $\mathbf{H}$, and a tunable weight matrix $\mathbf{W}$. The final output feature vector $\mathbf{h}_v^k$ can then be passed through simple feed-forward networks or multi-layer perceptrons (MLP) for downstream classification tasks. During training, we use back-propagation to update the weight matrices used in our AGGREGATE and UPDATE functions in each layer of the network, as well as the weights in our final MLP layer.

In the case of different tasks, such as graph-level prediction, a READOUT function is often adopted to aggregate all node feature information in the graph into a final representation.

$$\mathbf{h}_G = \text{READOUT}(\mathbf{h}_v^t | v \in \mathcal{V})$$

The simplest READOUT function can be summation or averaging. However, there are more complex choices available that are structural or attention-based.

Let $S \subset V$ be a subset of nodes. G_S is the subgraph of G induced by node set S. The specific task of subgraph classification of interest in this work can be formally defined as:

Given a background graph G and a collection of training subgraphs with labels, $\{(G_{S_i}, y_i) \mid G_{S_i} \subset G,\ y_i \in \texttt{Label},\ i \in \texttt{Train}\}$, predict the labels of subgraphs from the test set $\{G_{S_j} \mid G_{S_j} \subset G,\ j \in \texttt{Test}\}$. Each subgraph may be disconnected, and different subgraphs may overlap.

Based on the foundational concept of MPNN, many advancements have been made in recent research. One example is the Graph Attention Network (GAT) [2], which utilizes attention-based mechanisms to aggregate the features of neighboring nodes. Another example is the Graph Isomorphism Network (GIN) [3], which employs MLP in its UPDATE function because of its ability to represent the composition of functions. Other popular architectures include Graph Convolutional Network (GCN) [4] and Graph Sage (GSAGE) [5].

1.2 Pre-training and Prompting

To address these issues of label scarcity and domain transfer during downstream tasks, studies in various fields of deep learning have adopted the "pre-training and fine-tuning" paradigm. This approach involves initially pre-training the model using available unlabeled data and then transferring the general knowledge acquired during pre-training to a new domain or specific downstream task by fine-tuning the weights in the pre-trained model. In the setting of graph learning, typically we pre-train a GNN on some self-supervised tasks related to the overall graph structure, such as binary edge prediction. Afterward, we will tune only the final MLP layer for the specific task. These pre-training tasks, however, often aim to generate similar node features for connected nodes, even if they are unrelated to the goal of our downstream task and can cause poor performance during fine-tuning.

A promising solution developed from NLP to address this mismatch between pre-training and downstream goals is prompt tuning. Language prompts can either be a human-understandable piece of text or a tunable vector. A simple example would be adding the prompt "*Does [W1] refer to [W2]?* [MASK]" after the input text "Amy says she will major in Course 6." To convert a coreference resolution task into the known mask word prediction task used during pre-training. The final prompted input to the model would be:

Amy says she will major in Course 6. *Does She refer to Amy?* [MASK]

and the expected output would only be a single word token, *Yes* or *No*. Note that the prompt can be static and human-designed, like the example, or it can be a simple continuous and trainable vector that will be updated through back-propagation. While prompting is natural in NLP, the idea is harder in graph learning, as we need to consider both how to represent a prompt using nodes and edges and also how we can connect the prompt to our original graph. Formally, prompting in subgraph classification can be expressed as:

Given a GNN model $f(;\theta)$ pre-trained on graph G for some pre-training objective and a subgraph classification dataset S, we construct a new model $f^*(;\theta,\phi)$ based on f, parameterized by θ and prompt parameters ϕ. Then, prompt-tuning is done by optimizing for ϕ on S via the standard cross-entropy loss.

Chapter 2

Background and Related Work

2.1 Supervised Subgraph Classification Algorithms

The state-of-the-art subgraph classification methods mostly rely on supervised graph learning techniques. In the pre-GNN era, subgraph structures were learned through embeddings generated by language models using random walk samples within the subgraph [6]. However, these methods exclusively focus on the topology of subgraphs, neglecting both node features and connections to the broader graph. With the emergence of GNNs for both graph-level and node-level tasks, some simple methods for transferring to subgraph tasks include applying a graph-level GNN to the subgraph or pooling node features. Yet, these approaches either ignore border connections or lack detailed local structures. To address these challenges, cutting-edge solutions involve implementing MPNNs on the entire graph, with tailored adjustments made to the message aggregation at each layer.

SubGNN

Alsentzer et al. [7] were the first to formally propose the solution SubGNN, which not only aggregates information from neighborhood nodes but also from other sampled anchor patches in the entire graph to generate subgraph representations. The authors argued that subgraphs contain non-trivial internal topology, notions of position, and external connectivity that are

	Internal	Border
Position	Distance between Subgraph's components	Distance between Subgraph and rest of the Background Graph
Neighborhood	Identity of Subgraph's internal nodes	Identity of Subgraph's border nodes
Structure	Internal connectivity of Subgrpah	Border connectivity of Subgrpah

Table 2.1: SubGNN Subgraph Properties

essential for making classifications. Specifically, the paper identifies six essential properties that are crucial for creating subgraph representations, as shown in Table 2.1, where border nodes represent nodes within the k-hop neighborhood of any node in the subgraph. These properties are also what we focus on while designing our prompting methods.

During the message passing phase, SubGNN samples anchor patches for each of the six properties using a unique encoding and similarity function to compute the message. Messages are concatenated and aggregated across layers to produce the final subgraph representation that is used for classification.

Despite the success of SubGNN in over-performing traditional subgraph learning methods and simple node pooling, proving that plain GNN is not capable of capturing essential properties, the method bears a high overhead due to the need for pre-computing anchor patches and passing additional messages.

GLASS

In recognition of the shortcomings of SubGNN and the ability of plain GNN to capture topology and position through a simple BFS, a more simplified method of GNN with labeling tricks for Subgraph (GLASS) [8] is proposed as an improvement. GLASS argues that the complex properties defined in 2.1 can be easily learned through a plain neighborhood message passing by annotating whether the neighbor is within the subgraph or outside of it, representing internal versus border nodes.

Based on its superior performance and high efficiency, GLASS will serve as the primary supervised baseline method referenced in this thesis. Although much simpler than the orig-

inal SubGNN network, the additional MLP layers still provide a high level of reliability for labeled training samples.

2.2 Graph Pretraining Algorithms

Pretraining algorithms, often referred to as self-supervised learning [9], offer an alternative learning framework that diminishes reliance on manual annotations, thereby mitigating the limitations of supervised learning. This pretraining process entails training the model on a series of strategically designed tasks that automatically generate supervised labels directly from the graph without requiring additional human annotation.

However, this arbitrary pretrain goal could cause disparities between pretraining and subsequent tasks, which could undermine generalization and reduce pre-training's efficacy [10]. Therefore, selecting an appropriate pretraining objective is critical for optimal performance.

This work focuses on pretraining techniques that primarily use topology's structural information, since most subgraph datasets lack node or edge feature data. We specifically look at contrastive learning methods, which train encoders to maximize the agreement between positive pairs while minimizing the agreement between negative pairs sampled. [11]

The most straightforward example of contrastive learning is edge prediction [12], which aims to predict whether a link exists between two nodes in the graph. Positive pairs are two nodes connected by an edge, while negative pairs are nodes that are not connected. Negative edges are randomly sampled from the graph, with 10% of the edges as validation sets during training. This will prompt the graph to learn similar node features for connected nodes, thus learning local information.

Another class of contrastive learning is to construct negative pairs by perturbing certain aspects of the graph. Deep Graph Informax (DGI) [13] is the first self-supervised method for maximizing the mutual information between local node features and global graph information. The objective is to maximize the distance between positive sampled node features

and negative sampled corrupted node features compared to a summarized global graph representation. Implementation-wise, negative nodes are corrupted by shuffling. Graph Contrastive Learning (GraphCL)[14] expands on the concept of DGI by incorporating different augmentation strategies, such as node dropping, edge perturbation, attribute masking, and sampled subgraphs, to generate different augmented views for contrastive learning. As a result, GraphCL is able to minimize the distance between pairs of augmented graphs. SimGrace[15] is also an adaptation based on DGI that adds random noise perturbation to the model weights instead of the graph itself. This would prompt the pre-trained model to be more generalizable and robust.

Edge prediction concentrates on local linkages, and the remaining methods capture global insights, yet these existing pretraining techniques are generally devised for node-level or graph-level tasks, learning exclusively local or global patterns but not both. Such a limitation may pose challenges when adapting to subgraph prediction tasks that require an understanding of both the local interconnectivity within the subgraph and the broader context of the entire graph.

2.3 Graph Prompting Algorithms

The investigation into prompting algorithms gained attention following successes in various domains, including computer vision and natural language processing (NLP). Early studies delved into prompting methods in few-shot settings and examined their effectiveness in transitioning between node-level and graph-level tasks. However, none of these efforts delved into the complicated problem of subgraph classification.

GraphPrompt

GraphPrompt [16] stands out as one of the early pioneers embracing the pre-training and prompting framework in graph prediction. The paper introduces an innovative pre-training objective that utilizes k-hop neighbors as context for a node during edge prediction.

It presents two prompt types—element-wise multiplication and matrix linear transformation—both demonstrating improved performance, especially in few-shot scenarios. Additionally, the paper proposes the utilization of similarity measures between the final representation and a trainable virtual class prototype for label prediction, which closely resembles the pre-training edge prediction task. While demonstrating the potential of prompting methods, the work has limitations in terms of the tasks it supports and the pre-training schemes it utilizes.

Graph Prompt Feature

Recent developments in graph-prompting endeavors aim for a universal solution by employing pre-training strategies and downstream tasks that can be applied across different scenarios. Graph Prompt Feature (GPF) [17] emerges as one of the most effective prompting methods operating in the input feature space, with a theoretical guarantee of achieving robust performance. Results indicate that prompting methods can significantly outperform fine-tuning in both few-shot and full-shot scenarios. However, this generalized prompting scheme may prove to be less effective in subgraph tasks because it treats all nodes equally.

Chapter 3

Proposed Work

In this work, we propose MPrompt, a novel graph prompting method for subgraph classification that makes use of node membership to discriminate between nodes inside and outside the subgraph, and we also introduce Subgraph Reconstructive Pretraining (SRP), a subgraph-specific pretraining scheme.

3.1 MPrompt

Our methodology introduces a two-step process, beginning with the pre-training of a Graph Neural Network on the foundational edge prediction self-supervised task using only node features and the underlying structure of the graph. Following this, we refine the model by adjusting a specially designed prompt to guide the GNN's focus. In this prompting stage, we keep the original GNN weights frozen and train with actual subgraph labels. Figure 3.1 provides a visual summary of this process. Specifically, the node membership information is injected through our prompt.

We will now formally introduce our proposed algorithm, detailing its constituent components: Message Passing Neural Network (MPNN), Prompt, and Readout. Each component will be defined theoretically to establish a clear understanding of its role within the architecture. Additionally, we will also discuss the motivation behind the selection of this particular

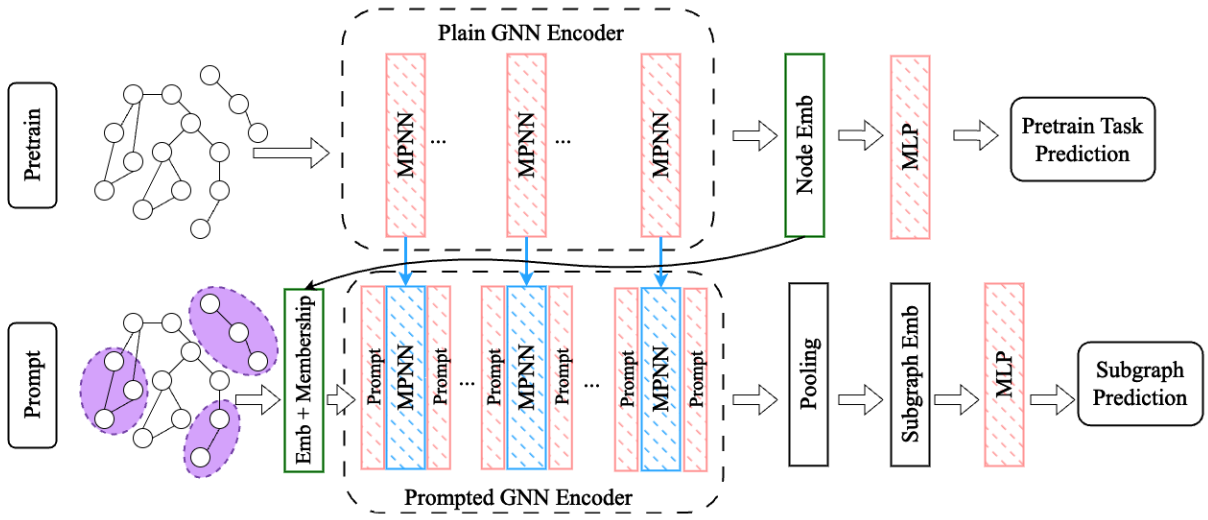


Figure 3.1: Overview of the proposed MPrompt with pretrain phase on top and prompt phase on the bottom. Blue indicate frozen weights while Red indicates activate tunable weights.

structure and how it contributes to the task of subgraph classification.

3.1.1 GNN architecture and Prompt

We first formally define the *plain* GNN backbone of our algorithm on a graph $G = (V, E)$ with node features H. As introduced, common GNN are composed of MPNN : $\mathbb{R}^{|V|\times d} \to \mathbb{R}^{|V|\times d}$ layers that operate on the node feature space where

$$\mathbf{h}_v^t = \text{MPNN}(\mathbf{h}_v^{t-1}; \theta_{MPNN}) = \text{UPDATE}(\mathbf{h}_v^{t-1}, \text{AGGREGATE}(\mathbf{h}_u^{t-1} | u \in \mathcal{N}(v)))$$

To increase the expressive power of our GNN, we further extend it to include two transformations $f_{post}, f_{pre} : \mathbb{R}^d \to \mathbb{R}^d$ that could be any arbitrary functions (e.g., linear layer) that work on individual node features. Thus, a message-passing layer g in our GNN would work as

$$g(;\theta) = f_{post}(;\theta_{post}) \circ \text{MPNN}(;\theta_{MPNN}) \circ f_{pre}(;\theta_{pre})$$

and the entire GNN model GNN : $\mathbb{R}^{|V|\times d} \to \mathbb{R}^{|V|\times d}$ with $\boldsymbol{l}$-layers would be

$$\text{GNN}(;\theta) = g_l(;\theta_l) \circ ... \circ g_2(;\theta_2) \circ g_1(;\theta_1)$$

which produces a final set of node features.

Now, with the background graph G, the l-layer GNN composed of our massage passing layers g, we additionally define $\text{PROMPT}(\mathbf{h_v};\phi_P) : \mathbb{R}^d \to \mathbb{R}^d$ to be the base prompt function with trainable parameters ϕ_P that we can apply to any node feature $\mathbf{h_v} \in \mathbb{R}^d$ for any vertex $v \in V$. This PROMPT layer can also be added to both before and after each individual massage passing layer, which gives us a prompted message passing layer $g^* : \mathbb{R}^{|V|\times d} \to \mathbb{R}^{|V|\times d}$ as

$$g^*(;\theta;\phi) = \text{PROMPT}_{post}(;\phi_{P_{post}}) \circ f_{post}(;\theta_{post}) \circ \text{MPNN}(;\theta_{MPNN})$$

$$\circ f_{pre}(;\theta_{pre}) \circ \text{PROMPT}_{pre}(;\phi_{P_{pre}})$$

and a prompted GNN^* to be

$$\text{GNN}^*(;\theta) = g_l^*(;\theta_l,\phi_l) \circ ... g_2^*(;\theta_2,\phi_2) \circ g_1^*(;\theta_1,\phi_1)$$

The exact function of PROMPT can take any form, but in this work, we primarily focus on two types of PROMPT functions, MLP and Addition.

The MLP PROMPT takes the form of

$$\text{PROMPT}(\mathbf{h_v};\phi) = \text{MLP}_k(\mathbf{h_v};\phi)$$

where MLP_k is a k-layer Multilayer Perceptron (MLP), and the Addition PROMPT is

$$\text{PROMPT}(\mathbf{h_v};\phi) = \mathbf{h_v} + \phi$$

which does element-wise addition.

We particularly chose to let the PROMPT function operate on node features, as it is the

most compatible with our backbone GNN architecture. This choice is further justified by the fact that most subgraph classification datasets lack edge features, making node-focused modifications more relevant. To evaluate the flexibility of our approach, we experimented with both MLP and Addition PROMPT structures. These structures were chosen because they offer varying degrees of freedom, which could potentially influence performance in few-shot learning scenarios.

3.1.2 Node Membership

The main improvement made to increase the expressing power of PROMPT for subgraph tasks is by introducing node membership, in which we define $\mathrm{P}(, \mathbf{m}; \phi)$:

$$\mathrm{P}(\mathbf{h_v}, \mathbf{m}; \phi) = \begin{cases} \mathrm{PROMPT}(h_v; \phi_1) & \text{if } \mathbf{m}_v = 1 \\ \mathrm{PROMPT}(h_v; \phi_0) & \text{if } \mathbf{m}_v = 0 \end{cases}$$

where $\phi = \phi_1 \cup \phi_0$ and $\mathbf{m} \in (\mathbb{Z}_2)^{|V|}$ is a *labeling* vector that denotes whether a vertex belongs in a subgraph.

During training and inference, for a batch of subgraphs $\{G_{S_1}, G_{S_2}, ..., G_{S_m}\}$, $\mathbf{m}$ can be computed as

$$\mathbf{m}_v = \begin{cases} 1 & \text{if } v \in \bigcup_{i=1}^{m} V_{S_i} \\ 0 & \text{if } v \notin \bigcup_{i=1}^{m} V_{S_i} \end{cases} \quad \text{for all v } \in V.$$

The idea of node membership is inspired by the "zero-one labeling trick" first introduced in GLASS [8], which claims that our GNNs are not enough to capture subgraph topology with regard to the background graph as they only gather information about the rooted BFS tree at each node.

As exemplified in Figure 2, given a graph G with uniform node features, the information being passed through our massage passing layer would be identical for subgraphs S and S', as

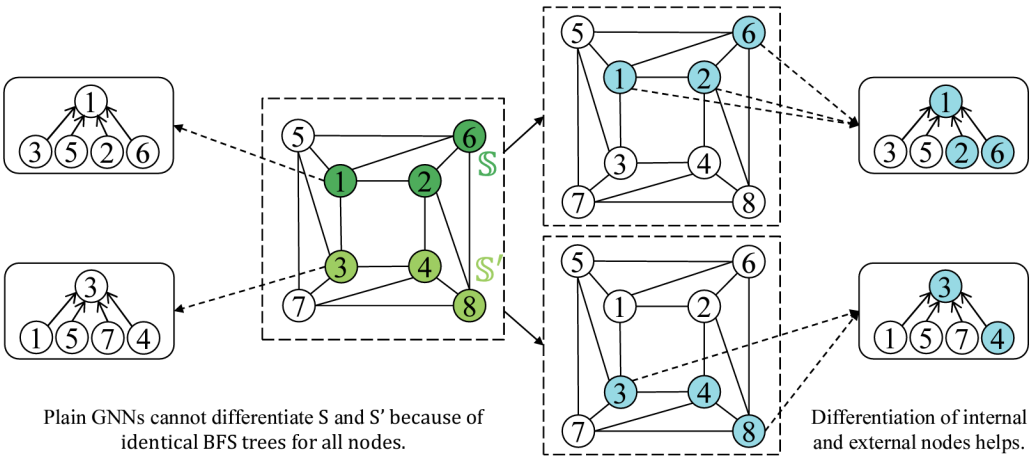


Figure 3.2: Example of failure to distinguish subgraphs without Node Membership [8]

they have the exact same BFS tree structure at each node. All nodes 1, 2, 3, 4, 6, and 8 have four neighbors. However, these subgraphs are non-isomorphic and should be distinguishable. This gap of information occurs because we lack information about the relationship between the subgraph and the background graph. By simply knowing the membership of the node, we can now generate different embeddings for S and S, as all nodes in S have two neighbors inside and two outside, while in S, 3 and 8, there is one neighbor inside and three outside.

3.1.3 Alternative Prompting Methods

In addition to the primary MLP and Addition PROMPTs, we have investigated various PROMPT variants tailored for few-shot subgraph classification.

Low-Rank Adaptation (Lora) PROMPT

A significant challenge in few-shot learning is the risk of overfitting; models often reach 100% accuracy on their small training sets without effectively learning the underlying patterns. To address this, we adopted a strategy to reduce the number of trainable parameters while preserving the model's original expressive capabilities. Drawing inspiration from techniques used in large language models, we implemented a Low-Rank Adaptation (LoRA)

approach [18]. This method employs rank decomposition matrices to approximate the MLP prompts, specifically approximating the original weight matrix $W \in \mathbb{R}^{d\times k}$ with

$$W = BA \ , \ B \in \mathbb{R}^{d\times r}, a \in \mathbb{R}^{r\times k}, r << min(d, k).$$

In practice, we often have $r = \frac{d}{4}$ and $d = k$ since the MLP layer transforms the original features without changing their dimensions. This would reduce the number of parameters by half. Following the original paper, A is initialized by a random Gaussian initialization and B is initialized to 0, which kept $W = BA = 0$ at the start.

Mixture of Expert (MOE) PROMPT

A well-known architecture that dramatically increases model capacity with little computing overhead is the mixture-of-experts (MoE) architecture [19], which has recently shown considerable success in deep learning. This model essentially replaces a single layer with a routing mechanism that, during both training and inference, allocates distinct samples to different specialized layers (i.e., experts). Because of the expert layers' non-linear capabilities, empirical evidence [20] demonstrates that this strategy performs better than the single-layer approach.

Formally, a k expert MOE layer replaces the normal function $f : \mathbb{R}^{|V|\times d} \to \mathbb{R}^{|V|\times d}$ with a routing function $r : \mathbb{R}^{|V|\times d} \to \mathbb{R}^{|V|\times kd}$ and the expert functions $e : \mathbb{R}^{|V|\times kd} \to \mathbb{R}^{|V|\times d}$, therefore we have

$$f(\mathbf{h_v}) = r(\mathbf{h_v}) \cdot e(\mathbf{h_v})$$

where $\cdot$ symbolizes matrix multiplication. Since we allocate a probability to each expert instead of selecting one, this method functions more like ensembling. The decision is made to ensure that the model is not skewed toward selecting a single expert.

Virtual Node PROMPT

Beyond merely focusing on node feature space, we also want to capture the distinctive attributes of graph datasets by integrating a PROMPT into the graph structure itself. Drawing inspiration from a fundamental graph classification method, which represents variable-sized graphs with a fixed-size vector through a virtual node summarizing the graph's latent characteristics [21], we introduce a PROMPT vector as an additional feature of a virtual node linked to the subgraph. By "linking," we imply creating undirected edges that connect this virtual node to every node within the subgraph, which fits the undirected nature of all subgraph datasets. This learnable feature vector is then propagated across the subgraph during the message-passing phase of our GNN architecture while it aggregates all information about a node's neighbor.

There are several important considerations in the implementation of the Virtual Node PROMPT. We want to maintain a consistent PROMPT across all subgraphs, so a single virtual node feature vector would be connected to all subgraphs. This approach introduces complexities during implementation, as it requires either expanding the dataset with multiple virtual nodes sharing identical features or frequently connecting and disconnecting the virtual node to and from the target subgraph. For simplicity in our code, we chose the latter approach. This is facilitated by employing the Node Membership concept, where we add an edge between the virtual node and all other nodes with $\mathbf{m}_v = 1$. However, a significant issue arises from this method: it creates unwanted connections between nodes in different subgraphs via the virtual node. To mitigate this, we limit our training to a batch size of 1, ensuring isolation between subgraphs during processing.

3.1.4 Read Out

Lastly, we also implement various read-out functions to obtain subgraph representations from its final node features. The most basic functions include summation and average pooling of node features:

$$\text{ADD POOL:} h_{G_{S_1}} = \sum_{v \in V_{S_i}} h_v \text{ , MEAN POOL:} h_{G_{S_1}} = \sum_{v \in V_{S_i}} \frac{h_v}{|V_{S_i}|}$$

More advanced functions include the degree pool, which calculates a weighted summation based on node degree,

$$\text{DEG POOL:} h_{G_{S_1}} = \sum_{v \in V_{S_i}} \frac{h_v \cdot Deg(v)}{\sum_{v \in V_{S_i}} Deg(v)}$$

and the attention pool, which trains an attention layer for each node.

The effectiveness of different pooling methods can vary significantly across datasets due to their underlying characteristics. For instance, additive pooling may be more sensitive to subgraph size, whereas attention-based pooling offers greater expressiveness by treating each node distinctly. This distinction highlights the importance of selecting an appropriate pooling strategy in our training process.

3.2 Subgraph Reconstructive Pretraining (SRP)

While exploring various prompt methods, we discovered that there are limitations in our pretraining objective, which is mostly designed for node-level or graph-level activities. Identifying this gap, we suggest a new pretraining strategy built around the idea of efficient graph reconstruction. This approach seeks to improve the model's resistance to vertex removals and its capacity to extract and integrate complicated structural information.

Drawing inspiration from the concept of *k-reconstruction of graphs* [22] or the ability to reconstruct the original graph from its induced k-vertex subgraphs that is proved to enhance the expressive power of GNNs. We transformed this approach into a pretraining objective that not only boosts the expressiveness of the GNN but also aligns closely with the downstream task of subgraph classification. This alignment ensures that the pretraining phase effectively prepares the GNN for specific challenges encountered in subgraph classification,

thereby improving overall model performance.

Formally, the k-reconstruction of graphs aims to be able to reconstruct the feature of a graph G

$$h(G) = f(CONCAT(\{h_{G_{S_i}} : S_i \in G, |V_{S_i}| = k\}))$$

where $f : \mathbb{R}^{|V| \times d} \to \mathbb{R}^{|V| \times d}$ is some function that transforms in the feature space and CONCAT denotes the row-wise concatenation of a set of vectors of the same shape in some arbitrary order.

In our subgraph reconstructive pretraining (SRP), we follow contrastive learning, which requires a positive pair and a negative pair. We construct a triplet by randomly sampling two subgraphs S and S' from the base graph G and a set of k vertex subgraphs induced from S as S_i. The idea is that the reconstruction of a S's k-induced subgraphs should be more similar to the embedding of S than to the embedding of S'. Formally, the positive pair is

$$h(G_S), f(CONCAT(\{h_{G_{S_i}} : S_i \in G_S, |V_{S_i}| = k\}))$$

and the negative pair is

$$h(G_{S'}), f(CONCAT(\{h_{G_{S_i}} : S_i \in G_S, |V_{S_i}| = k\})).$$

where the embeddings are produced from the GNNs defined. Similarity between embeddings is simply defined as the cosine distance between the two vectors. Visually, the process is shown in Figure 3.3.

In actual implementation, we sample subgraphs by BFS from random nodes, limiting the maximum size. This allows us to efficiently get subgraphs that are mostly connected, which preserves local information. k is chosen to be $n - 2$, where $n = |V_S|$ is the size of the subgraph and $|S_i| = n$ of those induced k-subgraphs are randomly selected. This is chosen based on the theoretical argument [22] that there exists a hierarchy of expressive power that

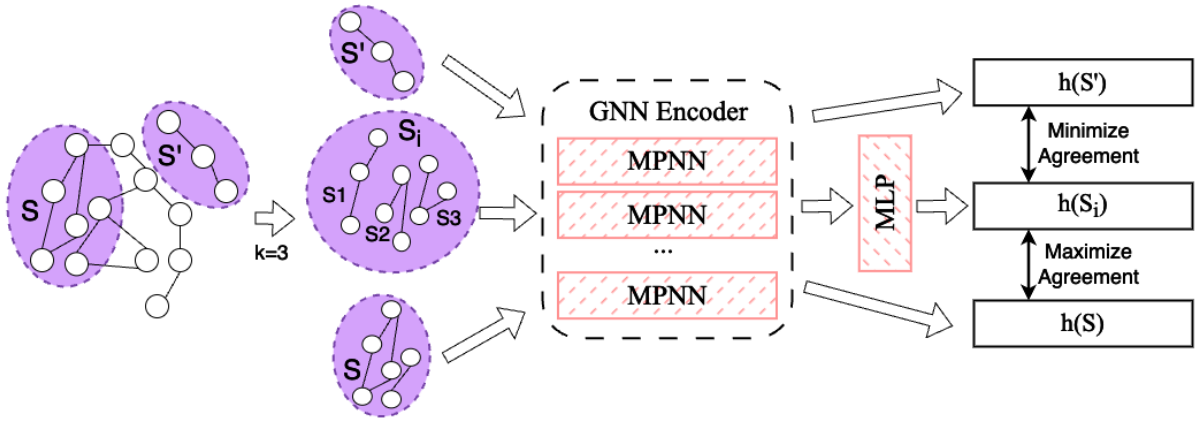


Figure 3.3: Overview of the proposed Subgraph Reconstructive Pretraining with k = 3

is monotonically increasing on k. Also, for more efficient training, we divide the sampled subgraphs into batches, and all positive and negative pairs are constructed within the batch. We kept the batch shuffled between each epoch to avoid any unwanted biases.

Chapter 4

Dataset And Experiment Setup

4.1 Datasets

The choice of datasets includes both synthetic datasets created for specific subgraph properties and real-world datasets. Synthetic datasets are adopted from the foundational SubGNN paper [7], which includes the following four distinct datasets:

- Density: defined as $D = \frac{2|E|}{|V| \cdot |V-1|}$ where $|E|$ is the number of edges and $|V|$ is the number of vertices of the subgraph.
- Cut Ratio: defined $CR = \frac{|BE|}{|V||G \backslash V|}$ where $|BE|$ is the number of border edges connecting nodes inside and outside of the subgraph, and $|G \backslash V|$ is the number of nodes in the rest of the graph G.
- Coreness: defined as the average k-coreness of all nodes in the subgraph [23].
- Component: defined as the number of connected components in the subgraph.

Performance on synthetic datasets can help us understand which properties our algorithm failed to learn and make adjustments accordingly. Besides synthetic datasets, we also utilize real-world datasets in the realm of biology and social networks to evaluate the effectiveness of our model in comprehending real-world information.

- ppi_bp [24]: human protein-protein interaction (PPI) network. The nodes in the network represent human proteins, while the edges represent physical interactions between the proteins. Subgraphs are collections of proteins involved in the same biological process, which can be categorized into different categories like metabolism or development.

- em_user: A co-occurrence network is created with nodes representing each workout, and edges are established between workouts completed by multiple users concurrently. Subgraphs are selected by conducting a random walk on a user's workout history and labeled based on the user's gender.

- hpo_metab [25]: a knowledge graph containing genotype and phenotypic data related to rare diseases. Subgraphs are groups of phenotypes classified by the subcategory of metabolic disorders that are connected to a rare monogenic disease.

- hpo_neuro [25]: Same base graph as hpo_metab, but focused on neurological disorders.

Some basic statistics of the data are shown in Table 4.1. We observed that this selection of datasets was composed of various sizes, densities, and numbers of classes, which would test the robustness of our algorithm.

Table 4.1: Summary of dataset statistics

	Background # Nodes	Background # Edges	Subgraph Ave # nodes	# Subgraphs	# Classes
DENSITY	5,000	29.521	20	250	3
CUT RATIO	5,000	83,969	20	250	3
CORENESS	5,000	118,785	20	221	3
COMPONENT	19,555	43,701	65.04	250	2
PPI-BP	17,080	316,95	10.20	1,591	6
HPO-METAB	14,587	3,238,174	14.44	2,400	6
HPO-NEURO	14,587	3,238,174	14.84	4,000	10
EM-USER	57,333	4,573,417	155.42	324	2

The distribution of label classes across each dataset is depicted in Figure 4.1. Although the synthetic datasets and most real-world datasets exhibit a balanced class distribution,

PPI_PI is notably characterized by a skewed class distribution. It is also important to mention that the class percentages for HPO_NEURO do not sum to 100% because it is a multi-label dataset, allowing each subgraph to be associated with multiple classes. Understanding the class distribution is crucial, particularly for few-shot learning, where the number of training samples per class is restricted—such as in 5-shot learning, which involves selecting five subgraphs from each class for our training set. Consequently, few-shot learning could introduce biases to an unbalanced dataset.

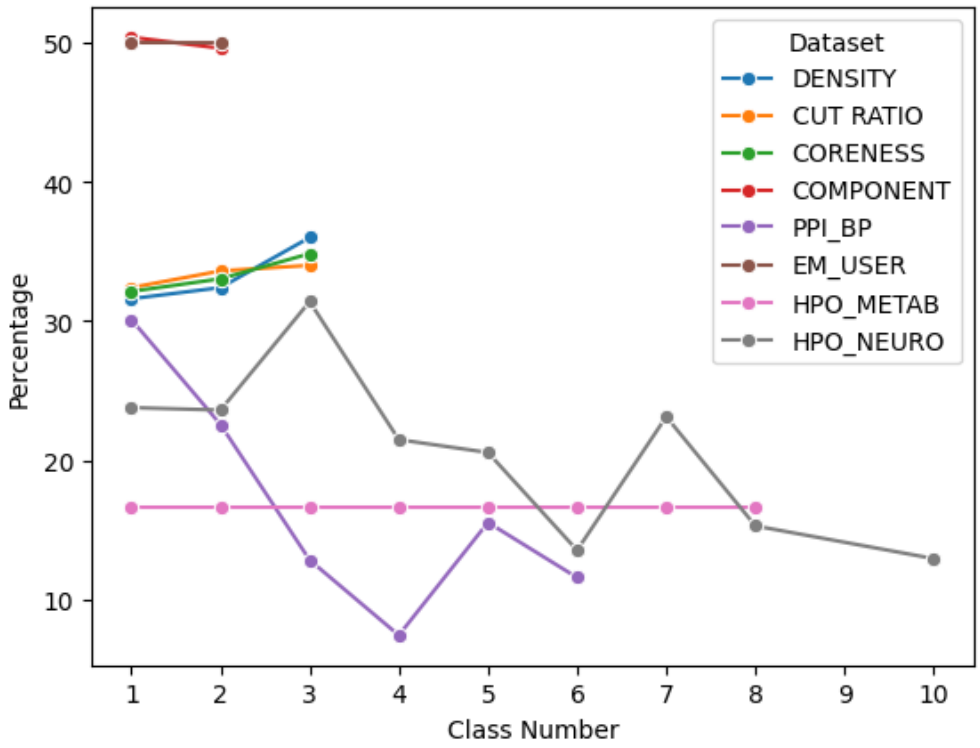


Figure 4.1: Dataset class distribution

4.2 Objective Function and Evaluation

To make the final prediction from the learned subgraph representation, we adopted two different inference algorithms. The first and most common approach is to input a subgraph graph representation into a MLP layer and output a class label, which is then put through

the negative log-likelihood function to get our loss.

However, due to the limitations of few-shot learning and the burden of additional parameters to train in the MLP layer, we also explore the prototype idea from GraphPrompt [16], where we create class prototypes for subgraph representations during the training phase. We calculate the label by comparing the similarity of the test subgraph representation to each class prototype. Formally, the definition is that for pairs of samples (s_i, y_i) in the training set, the loss of the prompt is defined as

$$L = -\sum_{(s_i,y_i)} \ln \frac{\exp(sim(s_i, p_{y_i}))}{\sum_{c\in Y} \exp(sim(s_i, p_c))}$$

where s_i is the subgraph representation output by the model, p_c is the class prototype for class $c \in Y$, and sim is a similarity function such as cosine.

To measure the performance of our classification, we sum over the confusion matrix of each class c to get both accuracy as

$$\text{ACC} = \frac{\sum_{c\in C}(TP_c + TN_c)}{\sum_{c\in C}(TP_c + TN_c + FP_c + FN_c)}$$

and micro-F1 score as

$$\text{micro-F1} = \frac{2 * precision * recall}{precision + recal} = \frac{\frac{2\cdot TP\cdot TP}{(TP+TN)(TP+FP)}}{\frac{TP}{TP+TN} + \frac{TP}{TP+FP}} = \frac{\sum_{c\in C} TP_c}{\sum_{c\in C} TP_c + \frac{1}{2}(FP_c + FN_c)}$$

Besides, there are complications in deciding the best epoch result. Existing literature often use methods like 1) the lowest training loss, which might be overfitting. 2) best validation accuracy, which assumes the validation set is available; and 3) set number of epochs, which can have an uncontrolled randomized effect. However, due to the high variance of few-shot training, we decided to choose a combined strategy where we pick a set number of epochs to stop at for each dataset based on the best average validation score for 10 runs. These combat both the high variance of performance and the cherry-picking results for the

specific training set.

4.3 Baseline Methods

To demonstrate the efficacy of our proposed MPrompt method, we conducted performance comparisons with several baseline methods commonly used in subgraph classification. Given that few-shot learning within subgraph classification remains relatively unexplored, all baseline methods utilized for comparison are traditionally designed for full-shot scenarios. This comparison provides insight into how well MPrompt adapts to the unique challenges of few-shot learning compared to standard approaches. We categorize our baseline methods along two dimensions: 1) whether it considers the subgraph as segregated from the background graph, and 2) whether the algorithm is primarily GNN-based, as shown in Table 4.2.

	Segregated	Connected
GNN based	GNN Seg, Virtual Node Seg	GLASS, SubGNN, GNN Plain
Not GNN based	Sub2Vec, Average Node Embedding	\

Table 4.2: Baseline Algorithms sorted into their respective categories

GNN-based algorithms that consider the entire background graph are the current state-of-the-art for subgraph classification, which includes GLASS, SubGNN, and GNN Plain, which are detaily discussed in previous sections.

- GLASS [8]: Primary baseline methods have the ability to capture complex subgraph properties through simple message passing and node annotating.
- SubGNN [7]: Augmented message passing with additional information from sampled anchor patches in the entire graph to get information on both internal and border graph properties. Due to its highly complex computation structure and the proof that GLASS is strictly superior in all datasets, we only consider it while comparing full-shot performance.

- GNN Plain [8]: GNN Plain is adopted from the baseline methods used in GLASS, which is a simple GNN with no specific augmentation made for subgraph classification. The final prediction is simply made by pooling all final node features output by the GNN.

On the other hand, we can also simply consider subgraph classification as graph classification by isolating the subgraph from the rest of the background graph, which gives us GNN Seg and Virtual Node Seg.

- GNN Seg [8]: GNN Seg is also adopted from the baseline methods used in GLASS, which have the same simple GNN backbone but treat each subgraph as an isolated graph from the background graph. Messages are only propagated within nodes inside the subgraph. The final subgraph feature is a pooling of all of its node features.

- Virtual Node Seg [26]: Virtual Node Seg is GNN Seg with a virtual node augmented to each isolated subgraph, which is a common practice in graph classification. This would be especially useful for the COMPONENT synthetic dataset, as the virtual node can connect different components of the subgraph.

Lastly, we also consider two traditional methods of graph representation learning that do not rely on a GNN backbone.

- Sub2Vec [6]: Obtain graph structure through a random walk-sampled path through a language model. We simply isolated the subgraphs as separate graphs in this algorithm without preserving the original node ID, as it only considers topological structures.

- Average Node Embedding (AVE): The most traditional baseline is one where we simply remove all edge information and pool all node features of the subgraph. However, as all subgraph datasets do not have node features to start with, we use pre-trained node features from our edge prediction GNN. Therefore, this baseline would be more powerful than normal average node embedding as it contains edge information implicitly.

4.4 Computation Setup

For our model, we utilized PyTorch and PyTorch Geometric as the primary libraries, running our experiments on two NVIDIA V100 GPUs, each equipped with 16GB of memory, to evaluate both performance and computational efficiency. The experiments were conducted on a Linux server configured with 32 CPU cores and a total of 50GB of RAM.

Chapter 5

Results

This chapter presents the performance of our proposed methods on the datasets in comparison to the baseline methods detailed in Chapter 4. Note that we separately present results for MPrompt and Subgraph Reconstructive Pretraining instead of combining them in the same pipeline.

5.1 MPrompt Results

Fewshot Construction Before presenting the results, we would like to clarify how we designed our few-shot experiment. Following traditional definitions, k-shot refers to having k samples of each class in the training set. However, one of our datasets, HPO_NEURO, is multi-labeled, which means that each subgraph can have multiple labels, which makes it almost impossible to sample exactly k subgraphs for each class, and the resulting training set can be significantly smaller. Thus, for HPO_NEURO, we define k-shot as $k\%$ of its original training set.

Implementation-wise, we first split all subgraphs into Train, Validation, and Test, and then randomly sample a few shot examples from the subgraphs. To eliminate unwanted bias, we resplit the dataset into 60%, 20%, and 20% on a set random seed that is used across testing of all methods.

None of the subgraph datasets provide node features; therefore, we initialized the node features to one feature for synthetics and node ID features for the real world before pretraining. The rationale is that synthetic datasets are usually smaller, and the properties we aim to predict do not require distinguishing each individual node.

5.1.1 Synthetic Dataset

We first present results on synthetic datasets, which clearly show the ability of our model to understand the topological properties of subgraphs.

Table 5.1: Comparison with other baseline strategies on synthetic datasets in a 10-shot setting. All datasets use the micro-F1 metric and standard deviation calculated with 10 runs.

	CUT RATIO	CORENESS	DENSITY	COMPONENT
MPrompt (add)	0.920 ± 0.041	**0.784 ± 0.033**	0.849 ± 0.027	0.865 ± 0.896
MPrompt (MLP)	**0.938 ± 0.026**	0.764 ± 0.072	0.869 ± 0.044	**1.000 ± 0.000**
AVE	0.811 ± 0.027	0.430 ± 0.085	0.584 ± 0.068	0.984 ± 0.047
AVE(Attention)	0.795 ± 0.148	0.430 ± 0.060	0.621 ± 0.046	0.927 ± 0.061
Virtual Node Seg	0.272 ± 0.079	0.364 ± 0.011	0.380 ± 0.035	**1.000 ± 0.000**
GNN_Seg	0.272 ± 0.079	0.364 ± 0.011	0.381 ± 0.054	**1.000 ± 0.000**
Sub2Vec	0.333 ± 0.049	0.477 ± 0.071	0.611 ± 0.061	0.602 ± 0.086
GNN_Plain	0.502 ± 0.191	0.388 ± 0.054	0.473 ± 0.0198	**1.000 ± 0.000**
GLASS	0.924 ± 0.017	0.760 ±0.042	**0.897 ± 0.030**	0.974 ± 0.042

Specifically, we are most interested in CUT RATIO and CORENESS, which are the two datasets that include both internal and border topology information, as described in the previous chapter. In contrast, DENSITY and COMPONENT can be successfully calculated just based on an isolated subgraph. This is also why we see a significant drop in performance for the segregated baseline methods (e.g., GNN Seg, Virtual Node Seg, and Sub2Vec) compared to the rest. For these two datasets, MPrompt is able to achieve superior performance, as bolded in Table 5.1.

The other two datasets, COMPONENT and DENSITY, have properties that solely care about the subgraph's internal structure and have no connection to the background graph. Consequently, we observe that GLASS may overfit the problem, whereas GNN Seg and Virtual Node Seg are able to obtain more consistent performance. Although it might seem that AVE's lack of edge information would prevent it from accurately learning COMPONENT, since our node features are pre-trained with edge prediction, there would be sufficient information to identify the link. We will not present performance data for DENSITY and COMPONENT for the remainder of this chapter, as these tasks are too simple to demonstrate our algorithms' effectiveness.

5.1.2 Real World Dataset

Next, we examine real-world dataset performance, which can help us better understand how our methods are able to extract real-world information. The values bolded in Table 5.2 represent the best performance, taking one standard deviation of the interval, because these datasets are generally significantly noisier. This allows us to make a fair comparison and go beyond the random sample bias, which has an impact on performance.

Our method achieves the best performance on PPI_BP, EM_USER, and HPO_METAB with significant performance increases for HPO_METAB and EM_USER. The lower performance on HPO_NEURO compared to GLASS happens as we take 10% of the training set (400 subgraphs), which is significantly larger than fewshot training in other datasets. As our method is specifically developed for few-shot learning, it is expected that it will not be exceptional on larger training sets.

Additionally, we see that the PPI_BP dataset exhibits varying performance, which can be attributed to its unbalanced class distribution, in which the first class represents roughly 30% of the dataset. We find that AVE performs substantially better than the other baselines except GLASS. This gap may be caused by this imbalance, but it may also indicate that the internal topology is less informative. In particular, PPI_BP subgraphs are considerably

Table 5.2: Comparison with other baseline strategies on real-world datasets in a 10-shot setting. All datasets use the micro-F1 metric and standard deviation calculated with 10 runs.

	PPI-BP	EM-USER	HPO-METAB	HPO-NEURO
MPrompt (add)	0.259 ± 0.036	0.548 ± 0.072	**0.399 ± 0.021**	0.370 ± 0.030
MPrompt (MLP)	**0.282 ± 0.033**	**0.683 ± 0.080**	**0.414 ± 0.019**	0.475 ± 0.013
AVE	**0.286 ± 0.044**	0.516 ± 0.081	0.244 ± 0.043	0.271 ± 0.018
AVE(Attention)	**0.287 ± 0.055**	0.529 ± 0.093	0.170 ± 0.052	0.168 ± 0.051
Sub2Vec	0.264 ± 0.019	0.533 ± 0.066	0.141 ± 0.050	0.229 ± 0.055
Virtual Node Seg	0.213 ± 0.045	0.474 ± 0.024	0.160 ± 0.014	0.000 ± 0.000
GNN Seg	0.200 ± 0.019	0.482 ± 0.019	0.179 ± 0.030	0.233 ± 0.069
GNN Plain	0.224 ± 0.025	**0.661 ± 0.068**	0.288 ± 0.017	0.501 ± 0.097
GLASS	**0.306 ± 0.025**	**0.662 ± 0.089**	0.365 ± 0.038	**0.517 ± 0.017**

smaller than those in HPO_METAB, which has dense local information.

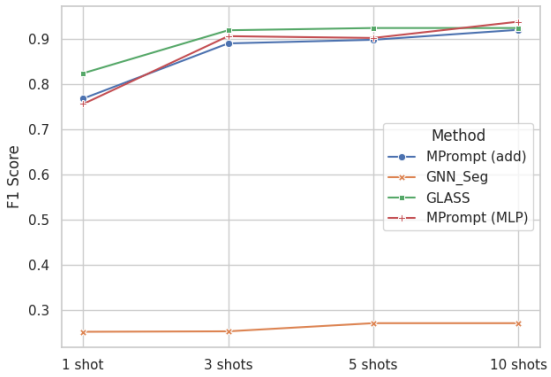
5.1.3 Ablation Study Results

We demonstrate the method's robustness to variations in the number of shots and the efficiency of our suggested Node Membership prompt adaptation through a series of ablation studies.

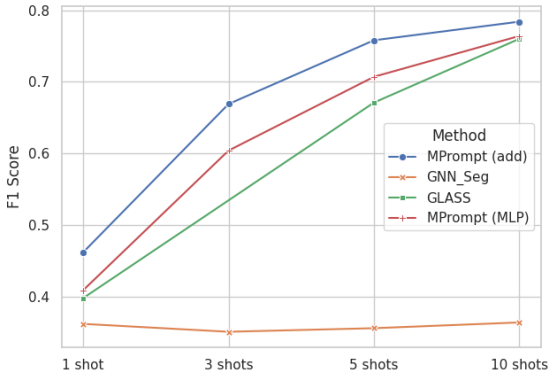
Impact of shots on Performance

A key factor influencing performance is undoubtedly the size of the training set. To verify the robustness of our algorithm, we conducted experiments with varying numbers of training examples, specifically 1, 3, 5, and 10 shots. Overall, MPrompt is able to maintain a steady lead in the majority of the datasets across shot counts. Lower performance on 1-shot scenarios can be caused by pure randomness in training set construction.

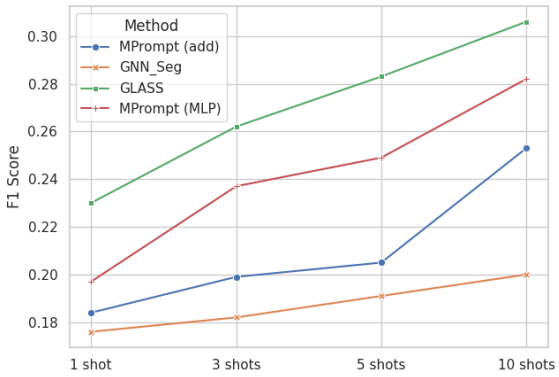
As depicted in Figure 5.1, we generally observed an increasing trend in performance with more training samples. However, an exception appeared with GNN Seg, where per-



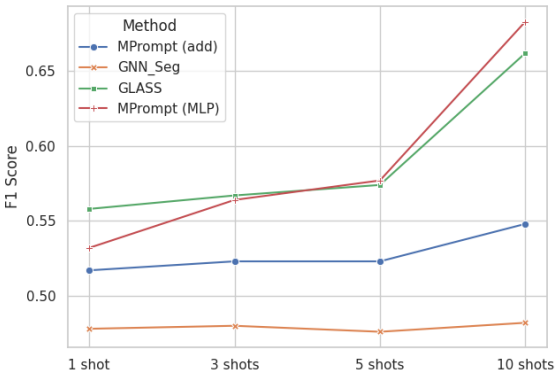
(a) CUT_RATIO



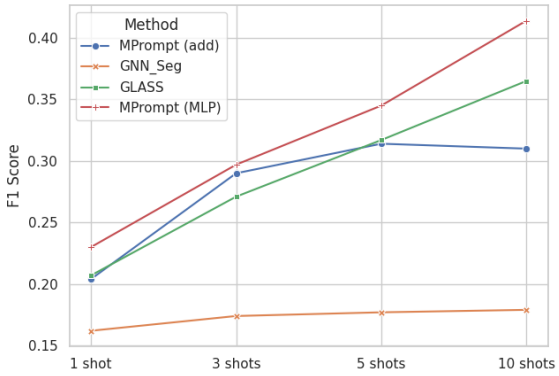
(b) CORENESS



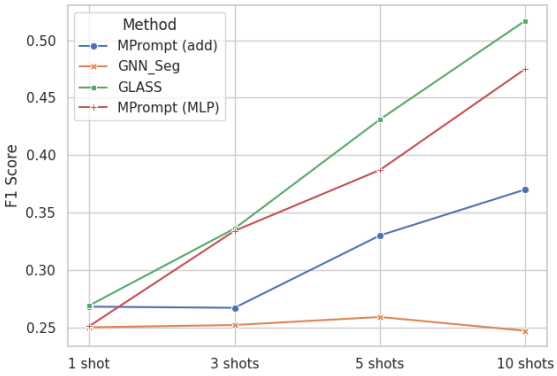
(c) PPI_BP



(d) EM_USER



(e) HPO_METAB



(f) HPO_NEURO

Figure 5.1: Ablation study on impact of shot count on performance. All datasets use the micro-F1 metric.

formance sometimes remained stagnant. This anomaly can be attributed to the fact that segregated graphs do not provide any information about their class (e.g., CORENESS and CUT RATIO). Therefore, additional training samples do not contribute additional useful information for these cases. GLASS in general has a steeper slope compared to MPrompt, as it contains more parameters and is more sensitive to changes in training size. Specifically, for CORENESS in Figure 5.1b, we see a conversion of performance across MPrompt and baseline methods when shot counts increase.

Standard deviations are omitted from the graph for better visualization.

Effectiveness of Node Membership

By contrasting MPrompt with Single Prompt—which essentially uses the same prompt for nodes inside and outside of the subgraph—we can see the effectiveness of the proposed node membership. Both MLP and addition single prompts are constructed for comparison. Table 5.3 confirms that, when we keep to the prior standards of incorporating confidence intervals for real-world datasets, MPrompt consistently outperforms Single Prompt across all datasets.

We observe that in single prompts, Addition outperforms MLP more frequently compared to when using MPrompt. As single prompt is incapable of distinguishing these properties in question for the task, adding more complexity by using MLP would just increase the probability of overfitting. This hypothesis is further proved by the fact that this gap between MLP and addition single prompt only exists in the synthetic datasets, which specifically require knowledge of node membership.

Table 5.3: Ablation study on the effectiveness of node membership in a 10-shot setting. All datasets use the micro-F1 metric and standard deviation calculated with 10 runs.

	CUT RATIO	CORENESS	PPI-BP	EM-USER	HPO-METAB	HPO-NEURO
MPrompt (add)	0.920 ± 0.041	**0.784 ± 0.033**	0.259 ± 0.036	0.548 ± 0.072	**0.399 ± 0.021**	0.370 ± 0.030
MPrompt (MLP)	**0.938 ± 0.026**	0.764 ± 0.072	**0.282 ± 0.033**	**0.683 ± 0.080**	**0.414 ± 0.019**	**0.475 ± 0.013**
Single prompt (add)	0.919 ± 0.031	0.782 ± 0.034	0.254 ± 0.045	0.559 ± 0.080	0.316 ± 0.050	0.443 ± 0.043
Single prompt (MLP)	0.872 ± 0.049	0.533 ± 0.068	0.269 ± 0.037	0.647 ± 0.077	0.406 ± 0.019	**0.502 ± 0.029**

Full-shot performance

Although MPrompt is not designed for full-shot tasks, it shows overall comparable and even improvement in performance compared to other supervised subgraph classification algorithms, as shown in Table 5.4. The unexpected improvements can be attributed to the fact that most of the information that is crucial for making classifications is learned during the pretraining phase, which decreases the need for complicated structures during downstream tuning. Segregated methods like Sub2Vec and GNN Seg also tend to have extremely low performance as they lack key connections to the background graph.

Table 5.4: Ablation study on full shot performance for synthetic and real-world datasets. All datasets use the micro-F1 metric and standard deviation calculated with 6 runs.

	CUT RATIO	CORENESS	PPI-BP	EM-USER	HPO-METAB	HPO-NEURO
MPrompt (add)	0.933 ± 0.009	0.770 ± 0.042	0.483 ± 0.001	0.747 ± 0.038	0.803 ± 0.015	0.530 ± 0.019
MPrompt (MLP)	**0.960 ± 0.000**	**0.815 ± 0.042**	0.493 ± 0.016	**0.818 ± 0.021**	**0.847 ± 0.005**	0.670 ± 0.005
Sub2Vec	0.365 ± 0.022	0.487 ± 0.0061	0.299 ± 0.026	0.614 ± 0.092	0.286 ± 0.014	0.231 ± 0.053
GNN Seg	0.274 ± 0.075	0.333 ± 0.048	0.323 ± 0.000	0.670 ± 0.028	0.193 ± 0.026	0.251 ± 0.076
GNN_Plain	0.499 ± 0.268	0.792 ± 0.052	0.485 ± 0.015	0.803 ± 0.012	0.797 ± 0.004	0.633 ± 0.007
SubGNN	0.520 ± 0.021	0.593 ± 0.093		0.773 ± 0.012	0.813 ± 0.010	0.724 ± 0.006
GLASS	0.933 ± 0.019	**0.815 ± 0.010**	**0.527 ± 0.009**	0.793 ± 0.007	0.806 ± 0.012	**0.758 ± 0.004**

Note that full-shot numbers differ from the original GLASS [8] and SubGNN [7] papers as we implemented a different random split of the dataset, a different process of picking the best epoch, and different pre-train node features. Since MPrompt is tested with these newly defined procedures, to maintain fair comparison, we also measure our baseline methods accordingly. SubGNN performance is not reported for PPI_BP as its preprocessing phase does not finish within 24 hours, which is cross-proven by other conference papers[27].

Effectiveness of Prompting

In order to demonstrate the advantages of prompting over direct GNN tuning, we contrast MPrompt with FINETUNE, which functions as an MLP prompt while the GNN weights are not frozen. As a result, during the training phase of FINETUNE, additional parameters would be updated. With the exception of the HPO_NEURO dataset, Table 5.5 shows that

MPrompt performs better than FINETUNE. This is expected given that FINETUNE would be more susceptible to overfitting when it had fewer training samples.

Table 5.5: Ablation study on the effectiveness of Prompting in a 10-shot setting. All datasets use the micro-F1 metric and standard deviation calculated with 10 runs.

	CUT RATIO	CORENESS	PPI-BP	EM-USER	HPO-METAB	HPO-NEURO
MPrompt (add)	0.920 ± 0.041	**0.784 ± 0.033**	0.259 ± 0.036	0.548 ± 0.072	**0.399 ± 0.021**	0.370 ± 0.030
MPrompt (MLP)	**0.938 ± 0.026**	0.764 ± 0.072	**0.282 ± 0.033**	**0.683 ± 0.080**	**0.414 ± 0.019**	0.475 ± 0.013
FINETUNE	0.908 ± 0.036	0.707 ± 0.066	**0.282 ± 0.031**	0.599 ± 0.081	0.387 ± 0.028	**0.522 ± 0.01**

5.2 Subgraph Reconstructive Pretraining Results

In this section, we present the results of our proposed Subgraph Reconstructive Pretraining (SRP) algorithm and compare it to the traditional edge prediction method. Due to computational, memory, and time constraints, we report performance only for CUT RATIO, CORENESS, PPI BP, and EM USER. The number of subgraphs sampled is proportional to the total available in the dataset, leading to significantly higher computational demands for both HPO datasets compared to others. Specifically, SRP (random) involves sampling five times the number of subgraphs present in the dataset for pretraining, with each subgraph containing up to 100 nodes. SRP (In dataset), on the other hand, utilizes all existing subgraphs directly for pretraining. It is important to note that this approach does not introduce any sampling bias, as subgraph labels are not used during the pretraining phase.

We first evaluate the performance of our generated node features using the AVE method, which serves as the most direct indicator of the effectiveness of pretraining. From Table 5.6, it is evident that our Subgraph Reconstructive Pretraining (SRP) significantly enhances performance on synthetic datasets, particularly when using SRP (In dataset). This improvement can be attributed to the uniform distribution of the properties of interest (e.g., for CUT RATIO, the classes are divided into subgraphs with high, medium, and low CUT

RATIO), ensuring that the sampled subgraphs closely align with the expected distribution. Conversely, SRP shows diminished performance on real-world datasets, where the subgraphs of interest typically exhibit specific characteristics that random sampling fails to capture. Moreover, using these specific subgraphs directly for pretraining does not sufficiently expose the algorithm to a broad spectrum of general subgraph knowledge, thereby limiting its performance.

Pretrain	**1 shot**	**3 shots**	**5 shots**	**10 shots**	**full shots**
			CUT RATIO		
SRP (In dataset)	0.851 ± 0.085	0.924 ± 0.022	0.932 ± 0.018	0.930 ± 0.031	0.947 ± 0.015
SRP (Random)	0.695 ± 0.126	0.819 ± 0.074	0.887 ± 0.053	0.941 ± 0.025	0.968 ± 0.0
Edge Prediction	0.565 ± 0.077	0.800 ± 0.076	0.869 ± 0.033	0.911 ± 0.027	0.926 ± 0.007
			CORENESS		
SRP (In dataset)	0.418 ± 0.058	0.441 ± 0.102	0.448 ± 0.073	0.445 ± 0.038	0.560 ± 0.008
SRP (Random)	0.362 ± 0.097	0.380 ± 0.064	0.407 ± 0.063	0.479 ± 0.058	0.577 ± 0.022
Edge Prediction	0.362 ± 0.080	0.391 ± 0.082	0.398 ± 0.092	0.430 ± 0.085	0.369 ± 0.046
			PPI BP		
SRP (In dataset)	0.165 ± 0.032	0.183 ± 0.028	0.199 ± 0.035	0.212 ± 0.028	0.392 ± 0.010
SRP (Random)	0.159 ± 0.032	0.172 ± 0.016	0.189 ± 0.033	0.175 ± 0.033	0.310 ± 0.011
Edge Prediction	0.195 ± 0.031	0.237 ± 0.034	0.235 ± 0.049	0.286 ± 0.044	0.542 ± 0.007
			EM USER		
SRP (In dataset)	0.515 ± 0.081	0.496 ± 0.093	0.447 ± 0.077	0.514 ± 0.051	0.544 ± 0.042
SRP (Random)	0.478 ± 0.080	0.484 ± 0.084	0.489 ± 0.064	0.492 ± 0.036	0.463 ± 0.082
Edge Prediction	0.520 ± 0.068	0.512 ± 0.065	0.514 ± 0.065	0.516 ± 0.081	0.752 ± 0.029

Table 5.6: Summary of results for different pretraining with AVE downstream algorithm.All datasets use the micro-F1 metric and standard deviation calculated with 10 runs.

We further assessed the quality of the features generated using more complex GNN-based methods in Table 5.7. We excluded MPrompt from this evaluation because the GNN model used during pretraining lacks relevant information for the downstream task, making

it impractical to retain the model weights frozen. Consequently, we focused our testing on GLASS and GNN Plain.

SRP demonstrates better performance compared to GNN Plain, benefiting from its simpler yet effective architecture relative to GLASS, which gains more on better initial node features. This improvement is particularly evident in situations with fewer training examples, where the GNN backbone has limited data to learn from. The fact that SRP achieves performance comparable to GLASS when using GNN Plain on synthetic datasets indicates that the node features generated by SRP possibly contain valuable node membership information. Moreover, SRP results are subject to increase as the model hyperparameters are not exhaustively searched through.

Overall, SRP seems to be a pretraining method that has huge potential in subgraph prediction tasks and also has the potential to generalize in other areas.

Pretrain	Downstream	1 shot	3 shots	5 shots	10 shots	full shots
CUT RATIO						
SRP (Random)	GNN Plain	0.844 ± 0.069	0.898 ± 0.023	0.900 ± 0.032	0.906 ± 0.015	0.913 ± 0.009
SRP (Random)	GLASS	0.828 ± 0.136	0.895 ± 0.033	0.920 ± 0.018	0.918 ± 0.014	0.927 ± 0.009
edge prediction	GNN Plain	0.490 ± 0.148	0.488 ± 0.151	0.572 ± 0.179	0.502 ± 0.191	0.699 ± 0.268
edge prediction	GLASS	0.824 ± 0.083	0.919 ± 0.028	0.924 ± 0.019	0.924 ± 0.017	0.920 ± 0.0
CORENESS						
SRP (Random)	GNN Plain	0.396 ± 0.092	0.444 ± 0.046	0.444 ± 0.058	0.522 ± 0.045	0.467 ± 0.018
SRP (Random)	GLASS	0.402 ± 0.124	0.562 ± 0.084	0.676 ± 0.075	0.740 ± 0.054	0.778 ± 0.031
edge prediction	GNN Plain	0.362 ± 0.059	0.380 ± 0.032	0.384 ± 0.028	0.388 ± 0.054	0.392 ± 0.052
edge prediction	GLASS	0.398 ± 0.101	0.593 ± 0.125	0.671 ± 0.050	0.760 ± 0.042	0.815 ± 0.010
PPI BP						
SRP (Random)	GNN Plain	0.224 ± 0.042	0.246 ± 0.026	0.272 ± 0.024	0.318 ± 0.029	0.521 ± 0.017
SRP (Random)	GLASS	0.199 ± 0.024	0.220 ± 0.032	0.249 ± 0.032	0.281 ± 0.035	0.510 ± 0.017
edge prediction	GNN Plain	0.178 ± 0.032	0.177 ± 0.030	0.197 ± 0.033	0.224 ± 0.025	0.385 ± 0.015
edge prediction	GLASS	0.230 ± 0.035	0.262 ± 0.042	0.283 ± 0.032	0.306 ± 0.025	0.527 ± 0.009
EM USER						
SRP (Random)	GNN Plain	0.488 ± 0.029	0.532 ± 0.042	0.612 ± 0.059	0.678 ± 0.103	0.778 ± 0.068
SRP (Random)	GLASS	0.494 ± 0.030	0.527 ± 0.047	0.489 ± 0.068	0.516 ± 0.076	0.449 ± 0.073
edge prediction	GNN Plain	0.541 ± 0.065	0.568 ± 0.048	0.595 ± 0.077	0.661 ± 0.068	0.803 ± 0.012
edge prediction	GLASS	0.558 ± 0.059	0.567 ± 0.027	0.574 ± 0.073	0.662 ± 0.089	0.793 ± 0.007

Table 5.7: Summary of results for different pretraining with GNN-based downstream algorithm. All datasets use the micro-F1 metric and standard deviation calculated with 10 runs.

Chapter 6

Conclusion

In this thesis, we introduced MPrompt, a specialized prompting method that leverages node membership knowledge and is based on Graph Neural Networks (GNN) to enhance few-shot subgraph classification across both synthetic and real-world datasets. Furthermore, we investigated an innovative pretraining approach, Subgraph Reconstructive Pretraining (SRP), which has great potential for enhancing the node features generated.

This research contributes significantly to the field of machine learning by proposing an improved model architecture while also making substantial real-world impacts, such as predicting molecular properties, identifying diseases, or detecting financial fraud when applying the algorithm to real datasets.

We recognize that there are areas for improvement and opportunities for further development in this work. Due to computational resource limitations and time constraints, we were unable to conduct exhaustive tests and hyperparameter tuning for all proposed methods, particularly for SRP. Future research directions worth exploring include:

- Extension of Node Membership: Currently our Node Membership concept only distinguishes whether or not the node belongs in a subgraph, yet, nodes outside of the subgraph can be worth different importance. For example, future directions can consider labeling the nodes based on their distance to the subgraph.

- Pretrained Node Feature: The existing edge prediction node feature, even without considering SRP, is a rather basic method in comparison to the vast number of sophisticated graph representational learning techniques that exist in the field. We can anticipate greater performance improvements with a stronger pretrained node feature, particularly as prompting algorithms rely more on the initial node feature.

- GNN Architecture: For improved representational capacity, the underlying GNN used for the downstream job can be extended to more complex structures like GAT[2] or Graph Sage (GSAGE)[5]. Furthermore, we may think about incorporating specific techniques, such as Neighborhood Wasserstein Reconstruction [28], to improve few-shot subgraph representation learning by better understanding of structural information.

Appendix A

GLASS mixture implementation

The actual implementation of GLASS which uses different MLP layers to transform nodes in and out of the subgraph, uses a weighted mixture and skip layer to finally update the message to the original node feature.

```
# transform node features with different parameters individually.
x1 = self.activation(self.trans_fns[1](x_))
x0 = self.activation(self.trans_fns[0](x_))
# mix transformed feature.
x = torch.where(mask, self.z_ratio * x1 + (1 - self.z_ratio) * x0,
                        self.z_ratio * x0 + (1 - self.z_ratio) * x1)
# pass messages.
x = self.adj @ x
x = self.gn(x)
x = F.dropout(x, p=self.dropout, training=self.training)
x = torch.cat((x, x_), dim=-1)
# transform node features with different parameters individually.
x1 = self.comb_fns[1](x)
x0 = self.comb_fns[0](x)
# mix transformed feature.
x = torch.where(mask, self.z_ratio * x1 + (1 - self.z_ratio) * x0,
                self.z_ratio * x0 + (1 - self.z_ratio) * x1)
```

References

[1] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," 2017. arXiv: 1704.01212 [cs.LG].

[2] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," ***International Conference on Learning Representations***, 2018. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ.

[3] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" ***CoRR***, vol. abs/1810.00826, 2018. arXiv: 1810.00826. [Online]. Available: http://arxiv.org/abs/1810.00826.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017. arXiv: 1609.02907 [cs.LG].

[5] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2018. arXiv: 1706.02216 [cs.SI].

[6] B. Adhikari, Y. Zhang, N. Ramakrishnan, and B. A. Prakash, "Distributed representation of subgraphs," ***CoRR***, vol. abs/1702.06921, 2017. arXiv: 1702.06921. [Online]. Available: http://arxiv.org/abs/1702.06921.

[7] E. Alsentzer, S. G. Finlayson, M. M. Li, and M. Zitnik, "Subgraph neural networks," ***CoRR***, vol. abs/2006.10538, 2020. arXiv: 2006.10538. [Online]. Available: https://arxiv.org/abs/2006.10538.

[8] X. Wang and M. Zhang, "GLASS: GNN with labeling tricks for subgraph representation learning," in ***International Conference on Learning Representations***, 2022. [Online]. Available: https://openreview.net/forum?id=XLxhEjKNbXj.

[9] Y. Liu, S. Pan, M. Jin, C. Zhou, F. Xia, and P. S. Yu, "Graph self-supervised learning: A survey," ***CoRR***, vol. abs/2103.00111, 2021. arXiv: 2103.00111. [Online]. Available: https://arxiv.org/abs/2103.00111.

[10] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. S. Pande, and J. Leskovec, "Pre-training graph neural networks," ***CoRR***, vol. abs/1905.12265, 2019. arXiv: 1905.12265. [Online]. Available: http://arxiv.org/abs/1905.12265.

[11] Y. Zhu, Y. Xu, Q. Liu, and S. Wu, "An empirical study of graph contrastive learning," ***CoRR***, vol. abs/2109.01116, 2021. arXiv: 2109.01116. [Online]. Available: https://arxiv.org/abs/2109.01116.

[12] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," 2018. arXiv: 1802.09691 [cs.LG].

[13] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," 2018. arXiv: 1809.10341 [stat.ML].

[14] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," 2021. arXiv: 2010.13902 [cs.LG].

[15] J. Xia, L. Wu, J. Chen, B. Hu, and S. Z. Li, "Simgrace: A simple framework for graph contrastive learning without data augmentation," in ***Proceedings of the ACM Web Conference 2022***, ACM, Apr. 2022. DOI: 10.1145/3485447.3512156. [Online]. Available: http://dx.doi.org/10.1145/3485447.3512156.

[16] Z. Liu, X. Yu, Y. Fang, and X. Zhang, "Graphprompt: Unifying pre-training and downstream tasks for graph neural networks," 2023. arXiv: 2302.08043 [cs.LG].

[17] T. Fang, Y. Zhang, Y. Yang, C. Wang, and L. Chen, "Universal prompt tuning for graph neural networks," 2023. arXiv: 2209.15240 [cs.LG].

[18] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," ***CoRR***, vol. abs/2106.09685, 2021. arXiv: 2106.09685. [Online]. Available: https://arxiv.org/abs/2106.09685.

[19] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the em algorithm," ***Neural Computation***, vol. 6, no. 2, pp. 181–214, Mar. 1994. DOI: 10.1162/neco.1994.6.2.181.

[20] Z. Chen, Y. Deng, Y. Wu, Q. Gu, and Y. Li, "Towards understanding mixture of experts in deep learning," 2022. arXiv: 2208.02813 [cs.LG].

[21] T. Pham, T. Tran, K. H. Dam, and S. Venkatesh, "Graph classification via deep learning with virtual nodes," ***CoRR***, vol. abs/1708.04357, 2017. arXiv: 1708.04357. [Online]. Available: http://arxiv.org/abs/1708.04357.

[22] L. Cotta, C. Morris, and B. Ribeiro, "Reconstruction for powerful graph representations," ***CoRR***, vol. abs/2110.00577, 2021. arXiv: 2110.00577. [Online]. Available: https://arxiv.org/abs/2110.00577.

[23] V. Batagelj and M. Zaveršnik, "An o(m) algorithm for cores decomposition of networks," ***CoRR***, vol. cs.DS/0310049, Oct. 2003.

[24] M. Zitnik, R. Sosič, S. Maheshwari, and J. Leskovec, ***BioSNAP Datasets: Stanford biomedical network dataset collection***, http://snap.stanford.edu/biodata, Aug. 2018.

[25] S. Köhler, M. Gargano, N. Matentzoglu, ***et al.***, ***The human phenotype ontology in 2021***, en, Jan. 2021.

[26] K. Ishiguro, S.-i. Maeda, and M. Koyama, "Graph warp module: An auxiliary module for boosting the power of graph neural networks in molecular graph analysis," ***arXiv preprint arXiv:1902.01020***, 2019.

[27] C. Liu, Y. Yang, Z. Xie, H. Lu, and Y. Ding, "Position-aware subgraph neural networks with data-efficient learning," in ***Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining***, ser. WSDM '23, ACM, Feb. 2023. DOI: 10.1145/3539597.3570429. [Online]. Available: http://dx.doi.org/10.1145/3539597.3570429.

[28] M. Tang, C. Yang, and P. Li, ***Graph auto-encoder via neighborhood wasserstein reconstruction***, 2022. arXiv: 2202.09025 [cs.LG].