Neural Graph Matching for Pre-training Graph Neural Networks

Yupeng Hou [*†‡] Binbin Hu [§] Wayne Xin Zhao [†‡¶] Zhiqiang Zhang [§]
Jun Zhou [§] Ji-Rong Wen [†‡‖]

Abstract

Recently, graph neural networks (GNNs) have been shown powerful capacity at modeling structural data. However, when adapted to downstream tasks, it usually requires abundant task-specific labeled data, which can be extremely scarce in practice. A promising solution to data scarcity is to pre-train a transferable and expressive GNN model on large amounts of unlabeled graphs or coarse-grained labeled graphs. Then the pre-trained GNN is fine-tuned on downstream datasets with task-specific fine-grained labels.

In this paper, we present a novel **G**raph **M**atching based GNN **P**re-**T**raining framework, called **GMPT**. Focusing on a pair of graphs, we propose to learn structural correspondences between them via neural graph matching, consisting of both intra-graph message passing and inter-graph message passing. In this way, we can learn adaptive representations for a given graph when paired with different graphs, and both node- and graph-level characteristics are naturally considered in a single pre-training task. The proposed method can be applied to fully self-supervised pre-training and coarse-grained supervised pre-training. We further propose an approximate contrastive training strategy to significantly reduce time/memory consumption. Extensive experiments on multi-domain, out-of-distribution benchmarks have demonstrated the effectiveness of our approach. The code is available at: `https://github.com/RUCAIBox/GMPT`.

1 Introduction

In the past few years, graph neural networks (GNNs) have emerged as a powerful technical approach for graph representation learning [9, 3]. By leveraging graph structure as well as node and edge features, GNNs can effectively learn low-dimensional representation vectors for each node or the entire graph. However, to apply GNNs to downstream applications, it usually requires abundant task-specific labeled data, which can be extremely scarce in practice. To alleviate the data scarcity issue [16], pre-training GNNs [6, 7] has been proposed as a promising solution. It first learns a transferable and expressive GNN on a large number of unlabeled graphs or coarse-grained labeled graphs. Then, the pre-trained GNN is fine-tuned on downstream datasets with task-specific labels.

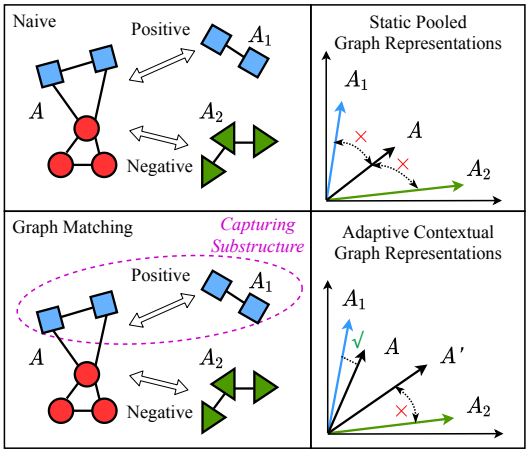

Figure 1: An example of neural graph matching and comparison with existing studies of static graph representations.

For GNN pre-training, existing studies mostly focus on the design of suitable tasks, such as graph structure reconstruction [3, 6, 7], mutual information maximization [21, 28, 17] and properties prediction [6]. Generally, these tasks can be classified into two main categories: (1) *node-level tasks* utilize node representations to predict the localized properties in the graph (*e.g.,* link prediction); (2) *graph-level tasks* focus on the entire graph and learn graph representations when designing the globalized optimization goal (*e.g.,* graph's property prediction).

Given the two kinds of GNN pre-training tasks, it is essential to combine node- and graph-level optimization goals [6], since they capture the graph characteristics in different views. Existing approaches either adopt a two-stage approach arranging multi-level pre-training tasks sequentially [6], or frame them in a multi-task

[*]Work done during internship at Ant Group.
[†]Gaoling School of Artificial Intelligence, Renmin University of China. {houyupeng,jrwen}@ruc.edu.cn, batmanfly@gmail.com
[‡]Beijing Key Laboratory of Big Data Management and Analysis Methods.
[§]Ant Group. {bin.hbb,lingyao.zzq,jun.zhoujun}@antfin.com
[¶]Corresponding author.
[‖]School of Information, Renmin University of China.

Copyright © 2022 by SIAM
Unauthorized reproduction of this article is prohibited.

learning manner [15]. In this way, each individual pre-training task is not aware of all the optimization goals at different levels, which might result in locally optimal representations *w.r.t.* some specific level (*e.g.,* node- or graph-level). Ideally, a good pre-training task can capture node- and graph-level characteristics simultaneously in order to derive more comprehensive node (and graph) representations.

For this purpose, we attempt to design new GNN pre-training tasks that are able to learn node- and graph-level graph semantics *in one single pre-training task*. Our solution is based on *neural graph matching* [23, 13, 26], a neural approach to learning structural correspondence among graphs. We present an illustrative example of our idea in Figure 1. At each time, a pair of two associated graphs (*e.g.,* with the same labels or augmented graphs) are given, and we evaluate whether the two graphs have similar structural properties. As a major advantage, neural graph matching naturally combines node-level correspondence (*e.g.,* v_1 to v_2) and graph-level properties (*e.g.,* whether containing shared substructure) when establishing their correspondence. That is the major reason why we adopt it as the GNN pre-training task. Another merit of this approach is that a graph will correspond to different representations when paired with different graphs. As shown in this example, we will derive different representations for graph A when paired with graph A_1 or A_2, since neural graph matching will enforce one graph to refer to another graph's information when learning graph representations. Therefore, we call the learned representations *adaptive graph representations*. As a comparison, existing graph-level pre-training tasks usually adopt static graph representations.

To this end, in this paper, we propose a novel **G**raph **M**atching based GNN **P**re-**T**raining method, named as **GMPT**. The key contribution lies in a neural graph matching module, where we pair two associated graphs as input at each time. To learn structural correspondences, we perform intra-graph as well as inter-graph message passing. In this way, the representations of a given graph are learned by referring to another paired graph, which derives adaptive graph representations. Such a method can capture both node- and graph-level characteristics when learning the graph representations. The proposed method can be applied to both fully self-supervised and coarse-grained supervised pre-training settings. In self-supervised setting, GNNs are optimized by a graph matching-based contrastive loss. To accelerate the learning of graph pairs during pre-training, we further proposed an *approximate contrastive training* strategy to significantly reduce the time/memory consumption, without loss of accuracy. While in supervised setting, we design different supervised tasks according to different coarse-grained labels.

In summarization, we design a new GNN pre-training task based on neural graph matching, devoted to adaptive graph representation learning by modeling both node- and graph-level characteristics in a single pre-training task. We also propose an approximate contrastive training strategy to reduce the time/memory consumption. Extensive experiments on public out-of-distribution benchmarks from multiple domains on various GNN architectures have demonstrated the effectiveness of our approach.

2 Related Work

In this section, we review the most related work about pre-training graph neural networks and graph matching.

2.1 Pre-training Graph Neural Networks

Though graph neural networks are powerful tools to characterize graph-structured data, they heavily rely on fine-grained domain-specific labels while training, which is usually scarce and difficult to obtain. To alleviate the above issues, pre-training for graph neural networks has drawn much attention recently, which empowers GNNs to capture the structural and semantic information of the input unlabeled graphs (or with few coarse-grained labels), followed by several fine-tuning steps on the downstream tasks of interest. Obviously, developing effective supervised (self-supervised) signals to guide GNNs to exploit structural and semantic properties on original graphs is at the heart. Generally, existing designed supervised signals can be classified into two main categories. The first is called *node-level tasks*, which aims at predicting localized properties utilizing node representations, such as graph structure reconstruction [3, 6, 7, 15, 11], localized attribute prediction [6, 7] and node representation recovery [4]. Another is called *graph-level tasks*, which defines globalized optimization goal for the entire graph, such as graph property prediction [6, 12] and mutual information maximization [21, 28, 17, 15, 27]. Our proposed framework differs from the above approaches in the following two aspects: learning node- and graph-level graph semantics *in one single pre-training task* and *adaptive graph representations*.

2.2 Graph Matching

Graph matching refers to establishing node correspondences between two (or among multiple) graphs [1], such that the similarity between the matched graphs is maximized. Some researches focus on the accuracy of the node correspondence, and regard graph matching as a quadratic assignment programming (QAP) prob-

Copyright © 2022 by SIAM
Unauthorized reproduction of this article is prohibited.

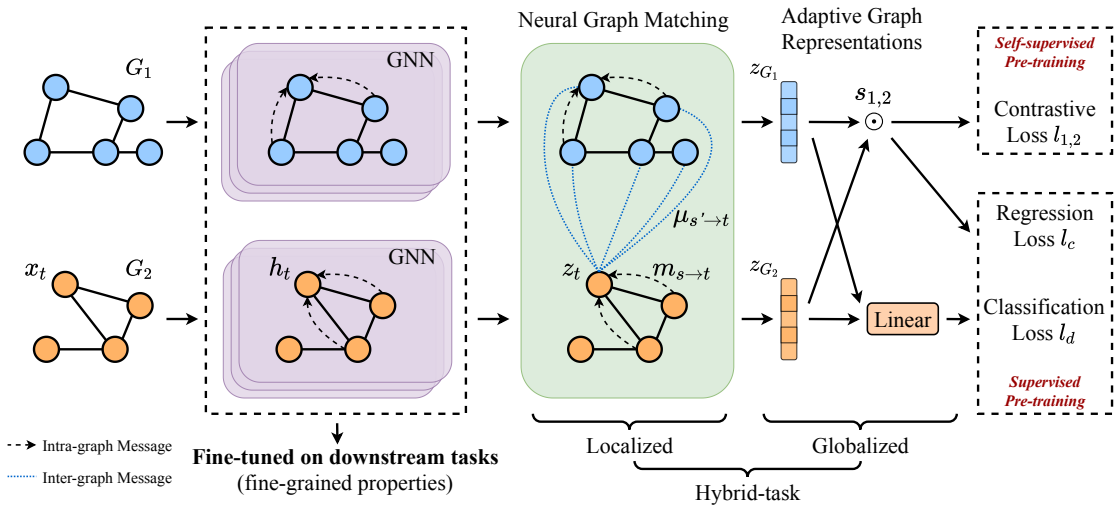

Figure 2: Overall framework of our proposed graph matching-based GNN pre-training methods.

lem [14], which is NP-complete. Thus, researchers mainly employ approximate techniques to seek inexact solutions, such as spectral approximation [10], double-stochastic approximation [2], and learning-based approaches [1, 29, 23]. While others care about the similarity calculation between graphs. Early efforts are mainly based on heuristic rules, such as minimal graph edit distance [24] and graph kernel methods [8, 22]. With the development of GNNs, recent work leverage message passing techniques to explore neural-based graph matching [13, 26]. In this work, we apply neural graph matching for GNN pre-training, to learn adaptive graph representations and encourage GNNs to integrate localized and globalized domain-specific features.

3 The Proposed Method

In this section, we first introduce the notation and problem definition. Then we present the proposed graph matching based GNN pre-training method **GMPT** for both self-supervised setting and coarse-grained supervised setting. Our approach takes pairs of graphs as input. With a carefully designed cross-graph message passing mechanism, one graph can be adaptively encoded into different graph representations when paired with different graphs. Figure 2 presents the overall architecture of our proposed framework.

3.1 Notation and Problem Definition

A graph can be represented as $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{X}, \boldsymbol{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$ denotes the node set, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the edge set, $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}| \times d_v}$ and $\boldsymbol{E} \in \mathbb{R}^{|\mathcal{E}| \times d_e}$ represent the d_v- and d_e-dimensional attribute matrix for nodes and edges, respectively. Furthermore, each graph is possibly associated with some label y from a label set $\mathcal{Y}$. Given a set of graphs with labels $\{(G_1, y_1), (G_2, y_2), \ldots, (G_N, y_N)\}$, graph neural networks (GNNs) leverage graph structure as well as node and edge features to learn a representation vector for the entire graph $\boldsymbol{h}_G$, and further utilize $\boldsymbol{h}_G$ to predict the corresponding label y. A detailed preliminary of GNNs is provided in Supplementary Material S.1[1].

In this work, we focus on pre-training GNNs: GNNs that are initialized with pre-trained parameters are fine-tuned according to various downstream tasks. Given the defined graph learning task, we consider two kinds of GNN pre-training paradigms based on whether graph labels are used or not during pre-training: *self-supervised setting* (without graph labels) and *supervised setting* (with graph labels).

3.2 Self-supervised Pre-training

We have no available labeled data for pre-training in self-supervised setting. The pre-training task is to evaluate whether a pair of augmented graph views are generated based on the same graph. We adopt contrastive learning to construct the learning objective and name our approach in this setting as **GMPT-CL**.

3.2.1 Graph Representation Learning via Graph Matching We first present how to learn graph representations via graph matching.

Graph augmentation and encoding. Given a list of n graph examples, we first apply stochastic data

[1]Please refer to: `https://github.com/RUCAIBox/GMPT/blob/main/paper/supplementary_material_gmpt.pdf`

Copyright © 2022 by SIAM
Unauthorized reproduction of this article is prohibited.

augmentation to transform any given graph example into two correlated views randomly ($2n$ views in total). We consider various augmentation techniques, including node/edge perturbation [28], subgraph sampling [28], diffusion [5], and adaptive methods [30]. The selection of graph augmentation techniques depends on the actual data domain [28]. To pre-train an expressive GNN encoder, we consider whether a pair of graph views denoted by $\tilde{G}_1$ and $\tilde{G}_2$ are matched or not based on their graph representations. Specially, we first apply the GNN encoder to obtain node representations in the two graph views. Let $\boldsymbol{h}_s^{(1)}$ and $\boldsymbol{h}_t^{(2)}$ denote the representations of node s from $\tilde{G}_1$ and node t from $\tilde{G}_2$, respectively.

Neural graph matching. Following recent progress in neural graph matching [13, 23], we incorporate message passing within a graph (called *intra-graph message*) and between a pair of graphs (called *inter-graph messages*). Given a intra-graph node pair $\langle s, t\rangle$ and a inter-graph node pair $\langle s', t'\rangle$, we define the two kinds of message passing mechanisms formally as:

$$\boldsymbol{m}_{s\to t} = \mathrm{MSG}_{\mathrm{intra}}\left(\boldsymbol{h}_s^{(1)}, \boldsymbol{h}_t^{(1)}, \boldsymbol{e}_{st}\right), \tag{3.1}$$

$$\boldsymbol{\mu}_{s'\to t'} = \mathrm{MSG}_{\mathrm{inter}}\left(\boldsymbol{h}_{s'}^{(1)}, \boldsymbol{h}_{t'}^{(2)}\right), \tag{3.2}$$

where $\boldsymbol{m}_{s\to t}$ and $\boldsymbol{\mu}_{s'\to t'}$ are intra-graph and inter-graph messages, respectively. Intra-graph message passing can be defined in a similar way following standard GNN architectures, like GIN [25]. While for $\mathrm{MSG}_{\mathrm{inter}}$, we adopt a cross-graph attention mechanism as:

$$\boldsymbol{\mu}_{s'\to t'} = a_{s'\to t'} \cdot \boldsymbol{h}_{s'}^{(1)},$$

where $a_{s'\to t'} = \frac{\exp(\mathrm{sim}(\boldsymbol{h}_{s'}^{(1)}, \boldsymbol{h}_{t'}^{(2)}))}{\sum_{k\in\tilde{G}_2}\exp(\mathrm{sim}(\boldsymbol{h}_{s'}^{(1)}, \boldsymbol{h}_{k}^{(2)}))}$ and $\mathrm{sim}(\cdot)$ is a similarity function, such as dot product and cosine similarity. The above attention mechanism allows an adaptive message exchange between the paired graphs. Intuitively, messages passed between similar substructures of graphs will have higher attention weights. Here, we normalize the attention weights of messages from the same *source* node, which means $\sum_{t'} a_{s'\to t'} = 1$. Besides, it is also optional to normalize the attention weights of messages to the same *target* nodes ($\sum_{s'} a_{s'\to t'} = 1$).

Match enhanced graph representations. After passing intra-graph messages from nodes' neighbors (denoted by $\mathcal{N}_{\mathrm{intra}}$) and inter-graph messages from all the nodes of another graph (denoted by $\mathcal{N}_{\mathrm{inter}}$), we aggregate the messages together and update to obtain the contextual node features $\boldsymbol{Z}$. For a node t, we update its original representation $\boldsymbol{h}_t$ as:

$$\boldsymbol{z}_t = \mathrm{Update}\left(\boldsymbol{h}_t, \sum_{s\in\mathcal{N}_{\mathrm{intra}}} \boldsymbol{m}_{s\to t}, \sum_{s'\in\mathcal{N}_{\mathrm{inter}}} \boldsymbol{\mu}_{s'\to t}\right), \tag{3.3}$$

where we use the sum operation for aggregation. Finally, we obtain the entire graph's adaptive representation $\boldsymbol{z}_G$ by employing a permutation-invariant function READOUT to pool contextual node features:

$$\boldsymbol{z}_G = \mathrm{READOUT}\left(\{\boldsymbol{z}_v | v \in \mathcal{V}\}\right). \tag{3.4}$$

Note that when involved in different pairs, a given graph will correspond to different representations in our approach. It is a key merit for subsequent pre-training tasks, since it can adaptively capture structural correspondences instead of using static graph representations as in previous studies [6, 28].

3.2.2 Contrastive Learning with Adaptive Graph Representations Contrastive learning is a commonly used technique to learn with augmented graph views in pairs [28]. It aims to increase similarity scores for positive pairs and decrease similarity scores for negative pairs. However, existing graph contrastive learning methods mainly adopt static graph representations [28, 5, 30], where node-level interaction across graphs is not explicitly modeled in this process. As a comparison, given a pair of graph views, we first apply the neural graph matching technique (Section 3.2.1) to characterize inter-graph interaction, and then construct the contrastive loss based on the adaptive graph representations.

Formally, given a positive pair $(\tilde{G}_i, \tilde{G}_j)$, we firstly adaptively encode them into contextual graph representations $\boldsymbol{z}_{\tilde{G}_i}$ and $\boldsymbol{z}_{\tilde{G}_j}$ (Eqn. (3.4)), and then formalize the contrastive loss below:

$$\ell_{i,j} = -\log\frac{\exp\left(s_{i,j}/\tau\right)}{\sum_{k\neq i}\exp\left(s_{i,k}/\tau\right)}, \tag{3.5}$$

where $s_{i,j} = \mathrm{sim}(\boldsymbol{z}_{\tilde{G}_i}, \boldsymbol{z}_{\tilde{G}_j})$ and τ is a temperature parameter. In practice, we usually have a batch of graph views, and we enumerate all the possible pairs of graphs in a batch for this loss in the denominator of Eqn. (3.5).

Although the above contrastive loss is also defined at the graph level (whether two views are augmented from the same graph), the derived graph representations $\boldsymbol{z}_{\tilde{G}_i}$ and $\boldsymbol{z}_{\tilde{G}_j}$ are enhanced with inter-graph node interaction via neural graph matching. As such, optimizing $\ell_{i,j}$ will encourage GNNs to capture both node- and graph-level characteristics in graph representations.

Copyright © 2022 by SIAM
Unauthorized reproduction of this article is prohibited.

3.2.3 Approximate Contrastive Training A major issue with graph matching is it incurs a quadratic time and space cost in terms of the number of nodes. Here, we propose an approximate contrastive training strategy to improve algorithm efficiency.

Complexity analysis. We consider the setting with a mini-batch of n graphs. As mentioned before, we would generate $2n$ augmented graph views for graph matching and contrastive learning. Typically, for a mini-batch of n graphs, GMPT-CL considers $2n \times 2n$ times of graph comparisons (two augmented views each graph) in total. Each comparison performs a node-to-node similarity calculation (refer to Eqn. (3.2)), taking an additional cost of $O(m^2 \cdot d)$ time and space, where $m = \sum_{i=1}^{2n} |\mathcal{V}_i|$ denotes the total number of nodes in $2n$ graph views and d is the dimensionality of representation vectors.

Approximate calculation. In order to reduce time and memory consumption, the key idea is to perform an approximate calculation of the proposed contrastive loss (Eqn. (3.5)): we sample q out of $2n$ graph views to contrast with all the other views ($q \times 2n$ times comparisons totally). In this way, the additionally expected time complexity can be reduced to $O(\frac{q}{2n} \cdot m^2 \cdot d)$. For a further reduction of space complexity, we adopt the gradient accumulation technique. For each time of sampling, we perform $1 \times 2n$ times of comparison. After calculating the contrastive loss, the model backpropagates prediction error and calculates the gradients, but doesn't update model parameters immediately. Instead, the gradients are accumulated until all the q samples are calculated. In this way, the sampled q graphs only require an additional space complexity of $O\left(\frac{1}{2n} \cdot m^2 \cdot d\right)$.

Theoretical analysis. We provide a theoretical analysis to reveal the connection between GMPT-CL with approximate contrastive training and mutual information maximization. Firstly, we show that,

LEMMA 3.1. *Minimizing Eqn.* (3.5) *is equivalent to maximizing a lower bound of the mutual information between the latent representations of two views of graphs.*

Furthermore, we show that,

LEMMA 3.2. *Optimizing Eqn.* (3.5) *with approximate contrastive training algorithm has the same optimization lower bound as originally in expectation.*

Thus we can see that the proposed approximate contrastive training fits the formulation of the InfoNCE loss [19, 18] in expectation. The proofs are provided in Supplementary Material S.3.

Empirically, experiments in Section 4.3 will show that performances of the fine-tuned GNNs on downstream tasks don't drop (even improve) with the proposed approximate contrastive training. The overall pre-training algorithm of GMPT-CL in one mini-batch is provided in Supplementary Material S.3.

3.3 Supervised Pre-training

Besides a large number of unlabeled graphs for self-supervised pre-training, sometimes we can obtain graphs with coarse-grained labels. Different from elaborately created fine-grained labels, coarse-grained labels have a relatively weak correlation with downstream task goals, but can be obtained in an easier way. For example, in molecular property prediction, we can easily collect the properties of molecules that have been experimentally measured so far [6].

In supervised pre-training setting, we pre-train GNNs on graphs with coarse-grained labels, and then the pre-trained GNNs are fine-tuned according to fine-grained labels in downstream tasks. Note that coarse-grained labels used for supervised pre-training are not the real labels of downstream tasks. Based on whether coarse-grained labels are continuous or discrete, we propose two variants of GMPT in supervised setting, named as **GMPT-Sup** and **GMPT-Sup**$_{++}$, respectively.

Continuous labels. Labels with continuous properties can be regarded as real-value vectors. We assume that similar pairs of graphs also correspond to similar labels. Based on this consideration, we propose **GMPT-Sup** to learn the similarities between graphs via graph matching module, and then minimize the difference between the learned similarity and the actual label similarity. Given a pair of graphs G_1 and G_2 and their coarse-grained labels y_1 and y_2, we firstly obtain their adaptive graph representations $\boldsymbol{z}_{G_1}$ and $\boldsymbol{z}_{G_2}$ as Eqn. (3.4). Then we define the loss function as $\ell_c = \text{MSE}(s_p, s_g)$, where $s_p = \text{sim}(\boldsymbol{y}_1, \boldsymbol{y}_2), s_g = \text{sim}(\boldsymbol{z}_{G_1}, \boldsymbol{z}_{G_2})$, and $\text{MSE}(\cdot, \cdot)$ denotes the standard mean squared error loss. This loss drives the representation similarity of two graphs to be close to their label similarity.

Discrete labels. For discrete labels, we do not enforce a direct comparison of the two graphs in a pair. Instead, we construct a classification-based approach **GMPT-Sup**$_{++}$ to associated graph representations with suitable coarse-grained labels. For a given pair of graphs G_1 and G_2 (encoded by graph matching module into $\boldsymbol{z}_{G_1}$ and $\boldsymbol{z}_{G_2}$), we jointly predict the corresponding coarse-grained labels y_1 and y_2. The loss for discrete labels ℓ_d can be calculated as,

$$\ell_d = \sum_{k=1,2} \text{BCE}(\boldsymbol{y}_k, \boldsymbol{W}_k \cdot \boldsymbol{z}_{G_k} + \boldsymbol{b}_k),$$

Copyright © 2022 by SIAM
Unauthorized reproduction of this article is prohibited.

Table 1: Statistics of the datasets. *PT* denotes Pre-Training and *FT* denotes Fine-Tuning. * denotes the total number of tasks for the eight downstream datasets.

Dataset	Bio	Chem
#(sub)graphs for self-supervised PT	307K	2,000K
#(sub)graphs for supervised PT/FT	88K	456K
#Coarse-grained labels for PT	5,000	1,310
#Downstream FT tasks	40	678*

where BCE$(\cdot,\cdot)$ denotes the popular binary cross-entropy loss, and $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ are vectorized representations of labels y_1 and y_2, respectively. Although the loss of the two graphs in a pair is calculated separately, their representations are obtained from the graph matching module (*e.g.,* cross-graph message passing in Eqn. (3.2)).

4 Experiments

In this section, we conduct extensive experiments to verify the effectiveness of our proposed methods in both self-supervised and supervised settings. Moreover, we also give an in-depth analysis of training strategy and transferability. Additional experiments on key parameters sensitivity analysis are provided in Supplementary Material S.5.4.

4.1 Experimental Setup

4.1.1 Datasets We conduct experiments on two public out-of-distribution (sub)graph classification benchmarks from different domains, namely Bio and Chem [6]. We strictly adopt the same way of splitting and pre-processing of these benchmarks as previous work [6]. Dataset statistics are summarized in Table 1. Detailed descriptions of the datasets are given in Supplementary Material S.5.1.

4.1.2 Baselines We compare our pre-training methods with the following representative GNN pre-training methods:
– **Infomax** [21] maximizes the mutual information between patch representations and corresponding high-level summaries of graphs.
– **EdgePred** [3] directly predicts the connectivity of node pairs, *a.k.a.,* link prediction task.
– **ContextPred** [6] uses subgraphs to predict their surrounding graph structures.
– **AttrMasking** [6] predicts nodes' or edges' attributes, which are randomly masked.
– **GraphCL** [28] contrast the static representations of augmented views and judge whether they are generated from the same graph.
– **L2P-GNN** [15] utilizes meta-learning to alleviate the divergence between multi-task pre-training and fine-tuning objectives.
– **PropPred** [6] predicts the coarse-grained labels of graphs in the pre-training datasets.

Among the baselines, PrepPred is designed for supervised pre-training setting, while others are targeted at self-supervised pre-training setting. Moreover, results of the non-pre-trained model are also reported.

4.1.3 Parameter Settings To enhance the reproducibility, we elaborately present the implementation details as follows. Detailed hyper-parameters for GNN architecture and training are provided in Supplementary Material S.5.2.

GNN architecture. We mainly experiment on Graph Isomorphism Networks (GINs) [25], the most expressive GNN architecture for graph-level prediction tasks. We also experiment with other popular architectures: GCN [9], GraphSAGE [3] and GAT [20]. We select the same GNN hyper-parameters as previous works [6, 15]. For the proposed graph matching methods, we adopt dot production for the sim$(\cdot)$ function and select $\tau = 0.07$ in Eqn. (3.5). We utilize a multiple layer perceptron (MLP) as function of Update in Eqn. (3.3).

Pre-training and fine-tuning settings. Results of baselines on different datasets are directly taken if they have been reported literaturely. For the other method, we pre-train the models with a learning rate of 0.001, and fine-tune the GNNs with a learning rate tuned in $\{0.01, 0.001, 0.0001\}$ for all the methods. We report the ROC-AUC for both datasets. The downstream experiments are run with 10 random seeds, and we report the mean and standard deviation of the metrics.

4.2 Performance Comparison

4.2.1 Self-supervised Setting Table 2 presents the performance comparison in self-supervised pre-training setting between GMPT-CL and the baselines.

The proposed pre-training method GMPT-CL achieves the best performance 72.53% over all the compared methods on Bio. with the most expressive GNN architecture GIN. On Chem dataset, we also notice that GMPT-CL gains the best results (71.5%) compared to all the baselines.

Applying GMPT-CL on currently popular GNN architectures, as shown in Table 2, GMPT-CL achieves the best macro-average result (71.13%) over all the compared baseline methods. In particular, we can see

Copyright © 2022 by SIAM
Unauthorized reproduction of this article is prohibited.

Table 2: Evaluation in self-supervised setting. We test ROC-AUC (%) performance using different pre-training methods. Besides, performances with different GNN architecture on Bio are also presented. The macro-average results over all GNN architectures on Bio, and results of GIN over all subtasks on Chem are also reported. In L2P-GNN, GNN is fine-tuned with a parameterized global pooling layer, while others use average pooling.

Pre-training methods	**Bio**					**Chem**
	GCN	GraphSAGE	GAT	GIN	Average	
w/o pre-training	63.20 ± 1.00	65.70 ± 1.20	68.20 ± 1.10	64.80 ± 1.00	65.48	67.0
Infomax	62.83 ± 1.22	67.21 ± 1.84	66.94 ± 2.61	64.10 ± 1.50	65.27	70.3
EdgePred	63.18 ± 1.12	66.05 ± 0.78	65.72 ± 1.17	65.70 ± 1.30	65.16	70.3
ContextPred	62.81 ± 1.87	66.47 ± 1.27	67.86 ± 1.19	65.20 ± 1.60	65.59	71.1
AttrMasking	62.40 ± 1.35	63.32 ± 1.01	61.72 ± 2.70	64.40 ± 1.30	62.96	70.9
GraphCL	67.05 ± 1.16	**71.53** ± 0.46	65.68 ± 3.98	67.88 ± 0.85	68.04	70.8
L2P-GNN	66.48 ± 1.59	69.89 ± 1.63	69.15 ± 1.86	70.13 ± 0.95	68.91	70.4
GMPT-CL	**70.65** ± 0.53	70.29 ± 0.21	**71.07** ± 0.14	**72.53** ± 0.42	**71.13**	**71.5**

Table 3: Evaluation in supervised setting. We report ROC-AUC (%) performance on Bio and Chem using different supervised pre-training methods with GIN.

Pre-training methods	**Bio**	**Chem**
w/o pre-training	64.8 ± 1.0	67.0
PropPred	69.0 ± 2.4	70.0
GMPT-Sup	**70.84 ± 0.59**	–
GMPT-Sup$_{++}$	70.73 ± 0.42	**70.4**

that GMPT-CL is powerful even with less expressive GNN architectures like GCN or GAT, which brings 3.60% and 1.92% absolutely gains compared to the best baseline, respectively.

In sum, we make the following observations.

(1) On average, the proposed GMPT-CL yields the best performance on benchmarks of different domains (72.53% on Bio and 71.5% on Chem). As GMPT-CL is a hybrid-level pre-training task, it encourages GNNs to capture both localized and globalized domain-specific semantics. Graph matching module of GMPT-CL can generate adaptive graph representations, in which shared substructures are enhanced.

(2) Pre-training GNNs with a large amount of unlabeled data is clearly helpful to downstream tasks, as GMPT-CL brings 6.69% and 4.5% absolutely gains compared to non-pre-trained models on the macro-average results over datasets of two domains, respectively.

4.2.2 Supervised Setting Table 3 presents the performance comparison between GMPT-Sup, GMPT-Sup$_{++}$, and the baselines (i.e., non-pre-trained GIN and PropPred) in supervised pre-training setting. Though coarse-grained labels in Bio datasets are multi-hot vectors, we still view them as continuous properties in GMPT-Sup. While coarse-grained labels in Chem datasets contain plenty of missing values. As similarity over missing values is hard to define, GMPT-Sup is not a suitable method for labels with missing values, and we only report the result of GMPT-Sup$_{++}$ on Chem.

The compared baseline PropPred always outperforms the non-pre-trained method, reflecting that GNNs pre-trained on coarse-grained labels can characterize domain-specific semantics. Compared with PropPred, which encodes graphs into static representations with a graph-level pre-training task, the proposed hybrid-level methods GMPT-Sup and GMPT-Sup$_{++}$ generate adaptive graph representations (Section 3.2.1). Consequently, we can see that the proposed methods achieve the best performances on Bio and Chem, respectively, demonstrating the effectiveness of our pre-training methods. Besides, we notice that GMPT-Sup outperforms GMPT-Sup$_{++}$ slightly on Bio dataset. A possible reason is that directly predicting every single property of coarse-grained labels (as GMPT-Sup$_{++}$) may cause overfitting and limit the transferability of the pre-trained model, especially when coarse-grained labels lack precious domain-specific semantics.

4.3 Training Strategy Analysis

As mentioned above, we propose an approximate contrastive training strategy to reduce the time and space consumption of GMPT-CL and the corresponding theoretical analysis about time and memory complexity can be found in In Section 3.2.3. Here we conduct detailed experiments to show how the fine-tuned GNNs' performance is affected by the batch size n and number of sampled views q, and what's the actual running time/memory when the proposed approximate training

Copyright © 2022 by SIAM
Unauthorized reproduction of this article is prohibited.

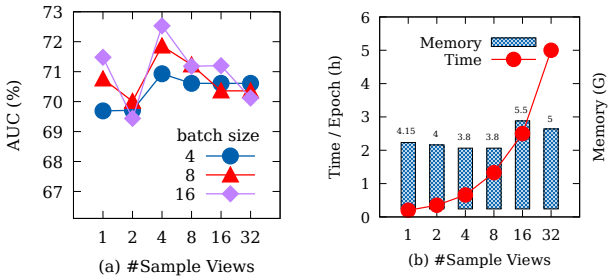

Figure 3: Tuning of approximate contrastive training with different numbers of sampled views on Bio dataset. (a) Performance with different batch sizes. (b) Time/Memory Consumption with batch size 32. Labels on the histogram indicate memory consumption.

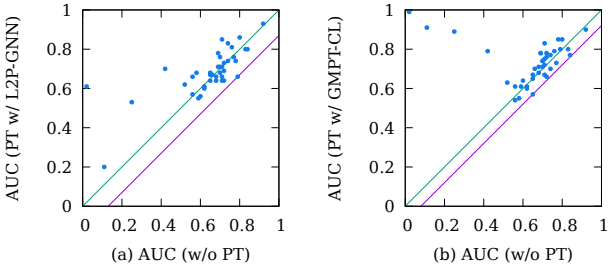

Figure 4: Analysis of transferability for (a) L2P-GNN and (b) GMPT-CL over 40 out-of-distribution subtasks of Bio. "PT" denotes as "pre-training". The green line indicates the borderline. The purple line indicates the worst negative transfer across the 40 subtasks.

strategy is applied.

4.3.1 Performance Comparison *w.r.t.* Batch Size and the Number of Sampled Views We pre-train GIN models in different batch sizes $n \in \{4, 8, 16\}$ and numbers of sampled views $q \in \{1, 2, 4, 8, 16, 32\}$, and compare the fine-tuned model's performance on downstream dataset of Bio. Note that for a certain batch size n, the maximum number of sampled views is $q = 2n$ (totally $2n$ views).

As Figure 3(a) shows, with different batch sizes and number of sampled views, our method can reach no less than 69% ROCAUC performance. We notice that applying approximate contrastive training usually improves the performance of GMPT-CL. Generally, GMPT-CL reaches the best performance when the number of sampled views $q = 4$. We speculate that the performance gain comes from the randomness introduced by contrastive approximate training. Besides, we find that even with a small sampling number $q = 1$, our method can still have competitive performances.

4.3.2 Time and Memory Consumption *w.r.t.* the Number of Sampled Views Here, we take an intuitive look at the actual time and space consumption of the proposed approximate contrastive training strategy. As Figure 3(b) shows, we find that as q decreases, the training time of GMPT-CL per epoch also decreases, which verifies that choosing a relatively small q can dramatically reduce the training time of GMPT-CL. Besides, we find that the memory consumption doesn't change a lot with different choices of q, verifying that the space complexity doesn't affect much by q.

In summary, we suggest adopting the proposed approximate contrastive training strategy when pre-training GNNs with GMPT-CL, which has been shown to reduce the time/space consumption and gain a slight performance increasing. For the choice of the number of sampled views q with the batch size n, we suggest to select a relative small q (i.e., $4 \leq q \leq n$).

4.4 Transferability Analysis

Out-of-distribution (OOD) problem widely exists in real-world applications, meaning that graphs in the training set are structurally very different from graphs in the test set [6]. Existing study shows that improperly designed GNN pre-training tasks may cause serious negative transfer. Thus, we analyze the proposed GMPT-CL and the best baseline L2P-GNN to analyze the transfer status over the individual subtasks of out-of-distribution datasets. The left and up area indicates positive transfer, and the right and bottom area indicates negative transfer. As shown in Figure 4, we can see that compared to the best baseline L2P-GNN, the proposed GMPT-CL has less negative transfer cases (12 *v.s.* 17), as well as a slighter negative transfer extent (-0.07 *v.s.* -0.13). Besides we can see that GNN pre-trained by GMPT-CL gets AUC result > 0.5 on all downstream subtasks of Bio. All the observations above show the good transferability of the proposed GMPT-CL method.

5 Conclusion

In this work, we propose GMPT, a general graph matching-based GNN pre-training framework for both self-supervised pre-training and coarse-grained supervised pre-training. By structuralized neural graph matching module, we generate adaptive representations for the matched graphs, encouraging GNNs to learn both globalized and localized domain-specific semantics in a single pre-training task. We also propose ap-

Copyright © 2022 by SIAM
Unauthorized reproduction of this article is prohibited.

proximate contrastive training strategy, which significantly reduces the time/memory consumption brought by the graph matching module. Extensive experiments on multi-domain out-of-distribution benchmarks show the effectiveness and transferability of our method.

Besides GMPT, more hybrid-level GNN pre-training tasks can be explored in the future. In addition, we will also consider generalizing our framework to more complicated graph structures (e.g. dynamic graphs, knowledge graphs, and heterogeneous graphs).

Acknowledgement

This work was partially supported by the National Natural Science Foundation of China under Grant No. 61872369 and 61832017, Beijing Outstanding Young Scientist Program under Grant No. BJJWZYJH012019100020098, and CCF-Ant Group Research Fund. Xin Zhao is the corresponding author.

References

[1] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola, *Learning graph matching*, TPAMI, (2009).

[2] S. Gold and A. Rangarajan, *A graduated assignment algorithm for graph matching*, TPAMI, (1996).

[3] W. L. Hamilton, Z. Ying, and J. Leskovec, *Inductive representation learning on large graphs*, in NIPS, 2017, pp. 1024–1034.

[4] B. Hao, J. Zhang, H. Yin, C. Li, and H. Chen, *Pre-training graph neural networks for cold-start users and items representation*, in WSDM, 2021.

[5] K. Hassani and A. H. K. Ahmadi, *Contrastive multi-view representation learning on graphs*, in ICML, 2020.

[6] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. S. Pande, and J. Leskovec, *Strategies for pre-training graph neural networks*, in ICLR, 2020.

[7] Z. Hu, Y. Dong, K. Wang, K. Chang, and Y. Sun, *GPT-GNN: generative pre-training of graph neural networks*, in SIGKDD, 2020, pp. 1857–1867.

[8] H. Kashima, K. Tsuda, and A. Inokuchi, *Marginalized kernels between labeled graphs*, in ICML, T. Fawcett and N. Mishra, eds., 2003, pp. 321–328.

[9] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, in ICLR, 2017.

[10] M. Leordeanu and M. Hebert, *A spectral technique for correspondence problems using pairwise constraints*, in ICCV, 2005, pp. 1482–1489.

[11] F. Li, B. Yan, Q. Long, P. Wang, W. Lin, J. Xu, and B. Zheng, *Explicit semantic cross feature learning via pre-trained graph neural networks for CTR prediction*, in SIGIR, 2021.

[12] P. Li, J. Wang, Z. Li, Y. Qiao, X. Liu, F. Ma, P. Gao, S. Song, and G. Xie, *Pairwise half-graph discrimination: A simple graph-level self-supervised strategy for pre-training graph neural networks*, in IJCAI, 2021.

[13] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, *Graph matching networks for learning the similarity of graph structured objects*, in ICML, 2019.

[14] E. M. Loiola, N. M. M. de Abreu, P. O. B. Netto, P. Hahn, and T. M. Querido, *A survey for the quadratic assignment problem*, Eur. J. Oper. Res., 176 (2007), pp. 657–690.

[15] Y. Lu, X. Jiang, Y. Fang, and C. Shi, *Learning to pre-train graph neural networks*, in AAAI, 2021.

[16] S. J. Pan and Q. Yang, *A survey on transfer learning*, TKDE, (2010).

[17] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, *GCC: graph contrastive coding for graph neural network pre-training*, in SIGKDD, 2020, pp. 1150–1160.

[18] M. Tschannen, J. Djolonga, P. K. Rubenstein, S. Gelly, and M. Lucic, *On mutual information maximization for representation learning*, in ICLR, 2020.

[19] A. van den Oord, Y. Li, and O. Vinyals, *Representation learning with contrastive predictive coding*, CoRR, abs/1807.03748 (2018).

[20] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph attention networks*, in ICLR, 2018.

[21] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, *Deep graph infomax*, in ICLR, 2019.

[22] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, *Graph kernels*, Journal of Machine Learning Research, 11 (2010), pp. 1201–1242.

[23] R. Wang, J. Yan, and X. Yang, *Learning combinatorial embedding networks for deep graph matching*, in ICCV, 2019, pp. 3056–3065.

[24] P. Willett, J. M. Barnard, and G. M. Downs, *Chemical similarity searching*, J. Chem. Inf. Comput. Sci., 38 (1998), pp. 983–996.

[25] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, *How powerful are graph neural networks?*, in ICLR, 2019.

[26] K. Xu, L. Wang, M. Yu, Y. Feng, Y. Song, Z. Wang, and D. Yu, *Cross-lingual knowledge graph alignment via graph matching neural network*, in ACL, 2019, pp. 3156–3161.

[27] M. Xu, H. Wang, B. Ni, H. Guo, and J. Tang, *Self-supervised graph-level representation learning with local and global structure*, in ICML, 2021.

[28] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, *Graph contrastive learning with augmentations*, in NIPS, 2020.

[29] A. Zanfir and C. Sminchisescu, *Deep learning of graph matching*, in CVPR, 2018, pp. 2684–2693.

[30] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, *Graph contrastive learning with adaptive augmentation*, in WWW, 2021.

Copyright © 2022 by SIAM
Unauthorized reproduction of this article is prohibited.