Neural Graph Matching for Pre-training Graph Neural Networks (Supplementary Material)

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

Supplementary Material

The supplementary material is structured as follows. First, we provide preliminaries about general graph neural networks (GNNs), and then related work about graph neural networks and graph augmentation. After that, we provide the pre-training algorithm of GMPT-CL with approximate contrastive training. Last, we provide additional experiments and details.

S.1 GNN Preliminaries GNNs leverage graph structure as well as node and edge features to learn a representation vector $\boldsymbol{h}_v$ for each node $v \in G$, or the entire graph $\boldsymbol{h}_G$. Modern GNNs adopt a neighborhood aggregation strategy. Each node's representation is iteratively updated by aggregating representations of its neighboring nodes and edges [7]. After k iterations of aggregation, a node's representation captures its k-hop network neighborhood's structural features. Formally, the k-th layer of a GNN can be summarized by the following expression:

$$\boldsymbol{g}_v^{(k)} = \text{Aggregate}^{(k)}\left(\{\boldsymbol{h}_v^{(k-1)}, \boldsymbol{h}_u^{(k-1)}, \boldsymbol{e}_{uv}^{(k-1)} | u \in \mathcal{N}_v\}\right),$$
$$\boldsymbol{h}_v^{(k)} = \text{Update}^{(k)}(\boldsymbol{h}_v^{(k-1)}, \boldsymbol{g}_v^{(k)}),$$

where $\boldsymbol{h}_v^{(k)}$ is the representation of node v at the k-th layer/iteration, $\boldsymbol{e}_{uv}$ is the feature vector of edge between node u and v, $\mathcal{N}_v$ is the set of nodes adjacent to node v, and $\boldsymbol{h}_v^{(0)}$ is initialized as node v's node attributes $\boldsymbol{x}_v$. To obtain the representation of the entire graph, a permutation-invariant function READOUT pools node features after the last iteration K,

$$\boldsymbol{h}_G = \text{READOUT}\left(\{\boldsymbol{h}_v^{(K)} | v \in G\}\right).$$

S.2 Related Work Here we provide additional related work about graph neural networks and graph augmentation.

Graph neural networks. Recent years have witnessed remarkable progress of graph neural networks (GNNs) in characterizing graph-structured data [19]. To learn powerful representation of graph data, a surge of GNNs have been proposed, which roughly fall into two lines. The first line focuses on learning node representation in spectral domain. As a pioneer and representative work, GCN [11] simplifies the Chebyshev polynomial in graph filtering [1, 4] based on first-order approximation and naturally captures local structure by the graph Laplacian. Correspondingly, the other line aims at aggregating and updating local structural information in the spatial domain. Hence, a series of strategies have been proposed to guide the message passing in GNNs, including mean/max/LSTM aggregation [7], attention mechanism [18] and graph structure pooling [6]. Moreover, several recent efforts have been made for improving the training efficiency through layer sampling [2, 9] and graph partition [3, 22]. For a thorough review, please refer to the elaborate surveys [19, 23]. However, how to train GNNs in an effective and efficient way still remains greatly challenging, since domain-specific labels are always scarce during training.

Graph Augmentation. As graph-structured data are usually hard to obtain [8], recent years, several graph data augmentation approaches have been proposed to alleviate the data-scarce issue and for more robust GNN training. Graph augmentation researches mainly fall into two categories, topology-level augmentation, and feature-level augmentation. Topology-level augmentation aims to perturb edges and nodes to generate different graph structures [15, 24]. Feature-level augmentation updates node/edge features for augmented graph instances [12]. In self-supervised pre-training setting, GMPT-CL generates augmented graph views via heuristic topology-level strategies, such as node/edge perturbation and subgraph sampling.

[*]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.

Algorithm 1: The pre-training algorithm of **GMPT-CL** for one mini-batch of n graphs.

input: a mini-batch of n graphs $\{G_1, G_2, \ldots, G_n\}$

1 $\mathcal{S}_{view} \longleftarrow \emptyset$
2 **foreach** $G \in \{G_1, G_2, \ldots, G_n\}$ **do**
3 Generate two views $\tilde{G}_1$ and $\tilde{G}_2$
4 $\mathcal{S}_{view} \leftarrow \mathcal{S}_{view} \cup \{\tilde{G}_1, \tilde{G}_2\}$
5 $\mathcal{S}'_{view} \leftarrow q$ views randomly sampled from $\mathcal{S}_{view}$
6 **foreach** $\tilde{G}_1 \in \mathcal{S}'_{view}$ **do**
7 **foreach** $\tilde{G} \in \mathcal{S}_{view}$ **do**
8 **if** $\tilde{G}$ *is generated from the same graph as* $\tilde{G}_1$ **then**
9 $\tilde{G}_2 \leftarrow \tilde{G}$
10 Calculate $\ell_{1,2}$ with Eq. (3.4)
11 Backpropagate and accumulate the gradients
12 Update model parameters according to the accumulated gradients

S.3 Approximate Contrastive Training Algorithm Algorithm 1 shows the pre-training algorithm of **GMPT-CL** with approximate contrastive training. Line $1 \sim 4$ show the graph augmentation progress (Section 3.2.1), line 5 shows the sampling of approximate contrastive training. In line $7 \sim 9$, we build the positive pair of views in Eq. (3.4), and in line $11 \sim 12$, we show the gradient accumulation and parameter optimization.

S.4 Proofs

S.4.1 Proof of Lemma 3.1

Proof. The contrastive loss $\ell_{i,j}$ in Eq. (3.4) can be rewritten for a batch of graphs (*e.g.,* n views),

$$\begin{aligned}\ell =& -\frac{1}{n}\sum_{i}^{n}\log\frac{\exp\left(s_{i,j}/\tau\right)}{\sum_{k\neq i}\exp\left(s_{i,k}/\tau\right)},\\ =&\mathbb{E}\left(-\frac{1}{n}\sum_{i}^{n}\log\frac{\exp\left(f_{i,j}\right)}{\sum_{k\neq i}\exp\left(f_{i,k}\right)}\right),\\ =&\mathbb{E}\left(\frac{1}{n}\sum_{i}^{n}\log\frac{\sum_{k\neq i}\exp\left(f_{i,k}\right)}{\exp\left(f_{i,j}\right)}\right),\end{aligned}$$

(S.1)

$$=\mathbb{E}\left(\frac{1}{n}\sum_{i}^{n}\log\sum_{k\neq i,j}^{n}\exp\left(f_{i,k}-f_{i,j}\right)\right)+\log\left(n-2\right),$$

where the expectation $\mathbb{E}$ is taken over N independent views $\{(G_i, G_j)\}$ from the joint distribution $p(G_i, G_j)$, and $f_{i,j} = s_{i,j}/\tau$ is a learnable graph matching-based score function. Thus, Eq. (S.1) fits into the formulation of the InfoNCE loss [17, 16]. Minimizing Eq. (3.4) results in a tighter lower bound of the mutual information. ∎

S.4.2 Proof of Lemma 3.2

Proof. We sample q views to contrast with all the n views. The loss can be rewritten in an expectation form,

$$\begin{aligned}\ell' =& -\frac{1}{q}\sum_{i\sim U(1,n)}^{q}\log\frac{\exp(s_{i,j}/\tau)}{\sum_{k\neq i}\exp\left(s_{i,k}/\tau\right)},\\ =&\mathbb{E}\left(-\frac{1}{q}\sum_{i\sim U(1,n)}^{q}\log\frac{\exp(f_{i,j})}{\sum_{k\neq i}\exp\left(f_{i,k}\right)}\right),\\ =&\mathbb{E}\left(-\mathbb{E}_{i\sim U(1,n)}\log\frac{\exp(f_{i,j})}{\sum_{k\neq i}\exp\left(f_{i,k}\right)}\right),\end{aligned}$$

(S.2)

$$\begin{aligned}=&\mathbb{E}\left(\mathbb{E}_{i\sim U(1,n)}\log\frac{\sum_{k\neq i}\exp\left(f_{i,k}\right)}{\exp(f_{i,j})}\right),\\ =&\mathbb{E}\left(\frac{1}{n}\sum_{i}^{n}\log\frac{\sum_{k\neq i}\exp\left(f_{i,k}\right)}{\exp(f_{i,j})}\right),\end{aligned}$$

(S.3)

$$=\mathbb{E}\left(\frac{1}{n}\sum_{i}^{n}\log\sum_{k\neq i,j}^{n}\exp\left(f_{i,k}-f_{i,j}\right)\right)+\log\left(n-2\right),$$

where we can find that Eq. (S.3) is the same as Eq. (S.1). The key point is that, by uniform sampling, the expectation of the contrastive loss of one single sample is equal to the averaged contrastive loss of the whole batch as Eq. (S.2). ∎

S.5 Additional Experiments and Details

S.5.1 Datasets Below are the descriptions of the datasets:

− **Bio** [8] is a biology dataset for protein function prediction, where nodes are proteins, and different edges denote types of relationship exists between a pair of proteins. To evaluate pre-training methods' transferability and out-of-distribution generalization, we split downstream dataset by *species*, which means the pre-trained model will be tested on new species.

− **Chem** [8] is a chemistry dataset for molecular property prediction, where nodes and edges represent atoms and chemical bonds, respectively. As above, downstream dataset is split by *scaffold* (molecular substructure).

Note that in Bio, graph instances are subgraphs sampled from the original giant network via random walk [8], while in Chem, graphs are individual molec-

ular. By sampling a node (called *central node*) and its neighborhood as a subgraph, tasks defined on the central node can be viewed as graph classification tasks, *e.g.,* domain classification on citation networks [11, 18, 13]. In this case, labels of the sampled subgraph are actually the central node's labels.

To evaluate the generalization of the pre-trained models on out-of-distribution prediction tasks, we split the downstream dataset by *species* for Bio and *scaffold* for Chem following [8]. We strictly adopt the same way of splitting and pre-processing of these benchmarks as previous work. Please refer to [8] for more details about the benchmarks.

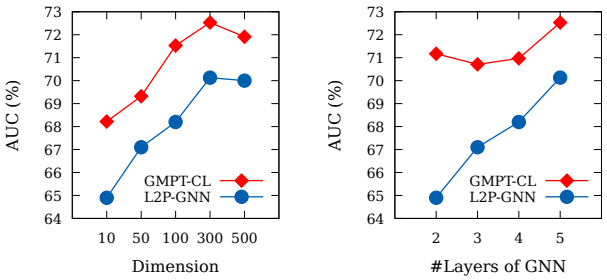
S.5.2 Detailed Parameter Settings For all the GNN architectures used in our experiments, we adopt the same hyper-parameters as following: 300 dimensional hidden units, 5 GNN layers ($K = 5$), average pooling for the READOUT function[1].

We use PyTorch [14] and PyTorch Geometric [5] for all of our implementations. Models are trained with Adam optimizer [10]. For self-supervised pre-training of GMPT, we use a batch size of 16 for 5 epochs on Bio datasets, while a batch size of 32 for 10 epochs on Chem datasets, which have fewer nodes per graph. We sample $q = 4$ views per batch in approximate contrastive training (Section 3.2.3). As suggested in the literature [21], we use edge perturbation for graph augmentation on Bio, as well as node drop and subgraph graph augmentation on Chem. For supervised pre-training, we use the same hyper-parameters as PropPred [8].

S.5.3 Performance on subtasks of Chem Table 1 presents the performance comparison with GIN in self-supervised pre-training setting over 8 downstream datasets of Chem. Generally, we can see that non of the compared pre-training methods can achieve the best performance over all the downstream tasks of Chem. We notice that GMPT-CL gains the best macro-average results (71.5%) compared to all the baselines, although it cannot achieve the best performance over all the 8 downstream subtasks.

S.5.4 Parameter Tuning In this part, we examine the robustness of our method and perform a detailed parameter analysis. For simplicity, we only report the performance of self-supervised pre-training setting on Bio dataset, and incorporate the best baseline L2P-GNN in Table 2 as a comparison.

Varying the Dimension d**.** Here we let graph matching module of the proposed method GMPT have the same dimension d as the pre-trained GNN. We vary d in $\{10, 50, 100, 300, 500\}$. As shown in Figure 5(a), our method constantly outperforms the best baseline, and achieves the best performance when $d = 300$. Overall, the performance is stable around $d \geq 300$, and the performance decreases sharply when $d < 100$.



(a) (b)

Figure 5: Tuning of different hyper-parameters of the GNN pre-trained with GMPT-CL on Bio dataset.

Varing the number of GNN layers L**.** The number of GNN layers L is also an essential hyper-parameter for the design of GNNs [20]. Here we vary L in the range 2 and 5. As shown in Figure 5(b), when $L = 5$, our method reaches the best performance. When $L \leq 4$, our method has a relatively stable result and marginally outperforms the best baseline.

[1] Except L2P-GNN, which requires a learnable pooling layer.

References

[1] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, *Spectral networks and locally connected networks on graphs*, arXiv preprint arXiv:1312.6203, (2013).

[2] J. Chen, T. Ma, and C. Xiao, *Fastgcn: fast learning with graph convolutional networks via importance sampling*, arXiv preprint arXiv:1801.10247, (2018).

[3] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, *Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks*, in SIGKDD, 2019, pp. 257–266.

[4] M. Defferrard, X. Bresson, and P. Vandergheynst, *Convolutional neural networks on graphs with fast localized spectral filtering*, arXiv preprint arXiv:1606.09375, (2016).

[5] M. Fey and J. E. Lenssen, *Fast graph representation learning with pytorch geometric*, CoRR, abs/1903.02428 (2019).

[6] H. Gao and S. Ji, *Graph u-nets*, in ICML, 2019, pp. 2083–2092.

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

Table 1: Evaluation in self-supervised setting on Chem. Here we test ROC-AUC (%) performance on molecular prediction benchmarks using different pre-training methods with GIN. The rightmost column averages the mean of test performance across the 8 datasets. **Bold** numbers indicate the best performance and underline numbers indicate the second-best performance.

Pre-training methods	Dataset								Average
	BBBP	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV	BACE	
–	65.8 ± 4.5	74.0 ± 0.8	63.4 ± 0.6	57.3 ± 1.6	58.0 ± 4.4	71.8 ± 2.5	75.3 ± 1.9	70.1 ± 5.4	67.0
Infomax	68.8 ± 0.8	75.3 ± 0.5	62.7 ± 0.4	58.4 ± 0.8	69.9 ± 3.0	75.3 ± 2.5	76.0 ± 0.7	75.9 ± 1.6	70.3
EdgePred	67.3 ± 2.4	76.0 ± 0.6	64.1 ± 0.6	60.4 ± 0.7	64.1 ± 3.7	74.1 ± 2.1	76.3 ± 1.0	79.9 ± 0.9	70.3
AttrMasking	64.3 ± 2.8	**76.7** ± 0.4	**64.2** ± 0.5	**61.0** ± 0.7	71.8 ± 4.1	74.7 ± 1.4	77.2 ± 1.1	79.3 ± 1.6	71.1
ContextPred	68.0 ± 2.0	75.7 ± 0.7	63.9 ± 0.6	60.9 ± 0.6	65.9 ± 3.8	75.8 ± 1.7	77.3 ± 1.0	79.6 ± 1.2	70.9
GraphCL	**69.7** ± 0.7	73.9 ± 0.7	62.4 ± 0.6	60.5 ± 0.9	**76.0** ± 2.7	69.8 ± 2.7	**78.5** ± 1.2	75.4 ± 1.4	70.8
L2P-GNN	66.4 ± 1.1	75.5 ± 0.4	63.1 ± 0.5	58.9 ± 0.8	66.2 ± 5.0	74.7 ± 1.8	77.3 ± 1.2	**81.1** ± 0.8	70.4
GMPT-CL	66.7 ± 2.0	74.6 ± 0.7	63.1 ± 0.4	60.1 ± 0.7	**76.0** ± 1.3	**76.6** ± 2.1	77.5 ± 1.2	77.0 ± 1.4	**71.5**

[8] 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.

[9] W. Huang, T. Zhang, Y. Rong, and J. Huang, *Adaptive sampling towards fast graph representation learning*, arXiv preprint arXiv:1809.05343, (2018).

[10] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in ICLR, 2015.

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

[12] K. Kong, G. Li, M. Ding, Z. Wu, C. Zhu, B. Ghanem, G. Taylor, and T. Goldstein, *FLAG: adversarial data augmentation for graph neural networks*, arXiv preprint arXiv:2010.09891, (2020).

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

[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, *Pytorch: An imperative style, high-performance deep learning library*, in NIPS, 2019, pp. 8024–8035.

[15] Y. Rong, W. Huang, T. Xu, and J. Huang, *Dropedge: Towards deep graph convolutional networks on node classification*, in ICLR, 2020.

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

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

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

[19] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, *A comprehensive survey on graph neural networks*, IEEE transactions on neural networks and learning systems, (2020).

[20] J. You, Z. Ying, and J. Leskovec, *Design space for graph neural networks*, in NIPS, 2020.

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

[22] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, *Graphsaint: Graph sampling based inductive learning method*, arXiv preprint arXiv:1907.04931, (2019).

[23] Z. Zhang, P. Cui, and W. Zhu, *Deep learning on graphs: A survey*, IEEE Transactions on Knowledge and Data Engineering, (2020).

[24] T. Zhao, Y. Liu, L. Neves, O. J. Woodford, M. Jiang, and N. Shah, *Data augmentation for graph neural networks*, in AAAI, 2021.