Learning to Pre-train Graph Neural Networks

Yuanfu Lu[1], Xunqiang Jiang[1], Yuan Fang[2], Chuan Shi[1]

[1]Beijing University of Posts and Telecommunications, Beijing, China

[2]Singapore Management University, Singapore

Background

Code and datasets can be found in https://yuanfulu.github.io

GNNs

- node-level representation

$$\begin{aligned}\mathbf{h}_v^l =& \Psi(\psi; \mathcal{A}, \mathcal{X}, \mathcal{Z})^l \\ =& \textsc{Update}(\mathbf{h}_v^{l-1}, \\ & \textsc{Aggregate}(\{(\mathbf{h}_v^{l-1}, \mathbf{h}_u^{l-1}, \mathbf{z}_{uv}) : u \in \mathcal{N}_v\}))\end{aligned}$$

- graph-level representation

$$\mathbf{h}_\mathcal{G} = \Omega(\omega; \mathbf{H}^l) = \textsc{Readout}(\{\mathbf{h}_v^l | v \in \mathcal{V}\})$$

Pre-train GNNs

- Pre-training

on a large graph-structured dataset (e.g., multiple small graphs or a large-scale graph)

$$\theta_0 = \arg\min_\theta \mathcal{L}^{pre}(f_\theta; \mathcal{D}^{pre})$$

- Fine-tuning

on downstream tasks

$$\theta_1 = \theta_0 - \eta \nabla_{\theta_0} \mathcal{L}^{fine}(f_{\theta_0}; \mathcal{D}^{tr})$$

θ_0 is pre-trained without accommodating the adaptation in fine-tuning

A gap between pre-training and fine-tuning!

L2PGNN: Learn to Pre-train GNNs

C1: How to narrow the gap caused by different optimization objectives?

- *Existing methods fall into a two-step paradigm with a gap*
- *Solution: learn to pre-train (meta learning)*

C2: How to simultaneously preserve node- and graph-level information?

- *SOTAs either only consider the node level or require supervision for graph-level pre-training*
- *Solution: intrinsic self-supervision*

Task Construction

- the pre-training data

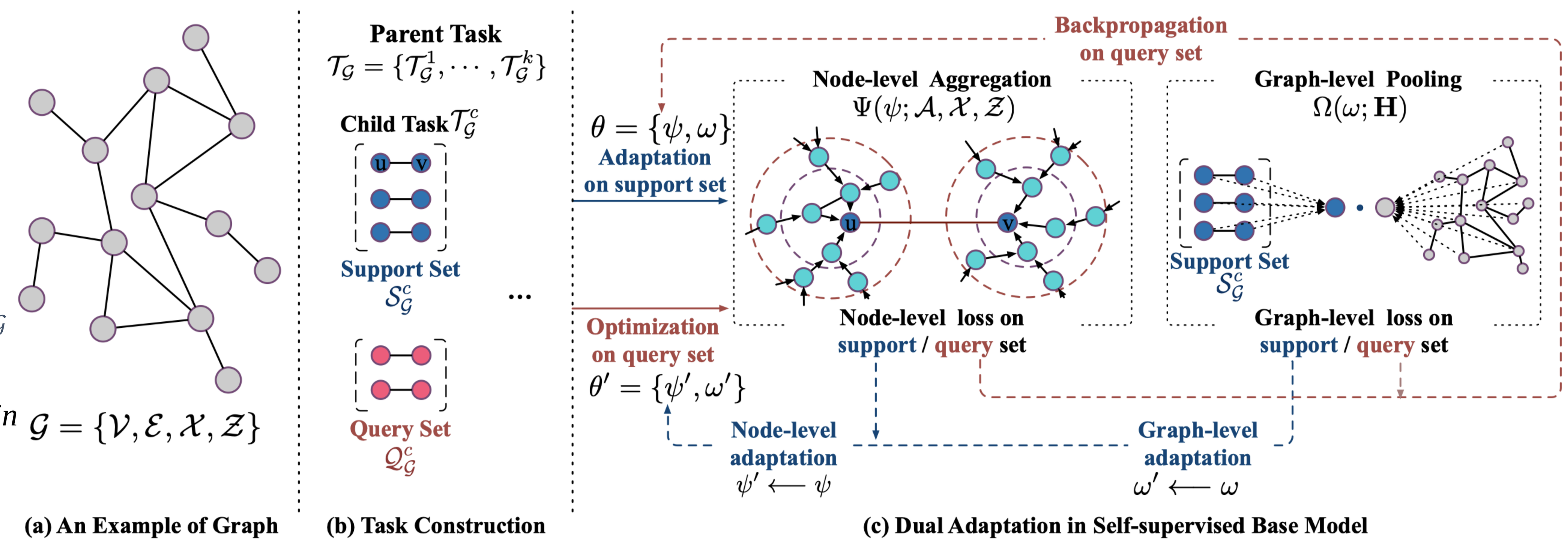
$$\mathcal{D}^{pre} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_N\}$$

- A task involving a graph

$$\mathcal{T}_\mathcal{G} = (\mathcal{S}_\mathcal{G}, \mathcal{Q}_\mathcal{G})$$

  - *gradient descent w.r.t. the loss on* $\mathcal{S}_\mathcal{G}$
  - *optimize the performance on* $\mathcal{Q}_\mathcal{G}$
  - *simulating the training and testing in the fine-tuning step*

$\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{Z}\}$



(a) An Example of Graph (b) Task Construction (c) Dual Adaptation in Self-supervised Base Model

Self-supervised Base Model

- node-level aggregation

$$\mathcal{L}^{node}(\psi; \mathcal{S}_\mathcal{G}^c) = \sum_{(u,v) \in \mathcal{S}_\mathcal{G}^c} - \ln(\sigma(\mathbf{h}_u^\top \mathbf{h}_v)) - \ln(\sigma(-\mathbf{h}_u^\top \mathbf{h}_{v'}))$$

- graph-level pooling

$$\mathcal{L}^{graph}(\omega; \mathcal{S}_\mathcal{G}) = \sum_{c=1}^{k} - \log(\sigma(\mathbf{h}_{\mathcal{S}_\mathcal{G}^c}^\top \mathbf{h}_\mathcal{G})) - \log(\sigma(-\mathbf{h}_{\mathcal{S}_\mathcal{G}^c}^\top \mathbf{h}_{\mathcal{G}'}))$$

$$\mathcal{L}_{\mathcal{T}_\mathcal{G}}(\theta; \mathcal{S}_\mathcal{G}) = \mathcal{L}^{graph}(\omega; \mathcal{S}_\mathcal{G}) + \frac{1}{k}\sum_{c=1}^{k} \mathcal{L}^{node}(\psi; \mathcal{S}_\mathcal{G}^c)$$

Dual Adaptation

- Node-level adaptation $\psi' = \psi - \alpha \frac{\partial \sum_{c=1}^{k} \mathcal{L}^{node}(\psi; \mathcal{S}_\mathcal{G}^c)}{\partial \psi}$
- Graph-level adaptation $\omega' = \omega - \beta \frac{\partial \mathcal{L}^{graph}(\omega; \mathcal{S}_\mathcal{G})}{\partial \omega}$
- Global optimization $\theta \leftarrow \theta - \gamma \frac{\partial \sum_{\mathcal{G} \in \mathcal{D}^{pre}} \mathcal{L}_{\mathcal{T}_\mathcal{G}}(\theta'; \mathcal{Q}_\mathcal{G})}{\partial \theta}$

Experiments & Analysis

Datasets

A new dataset for pre-training GNNs

Dataset	Biology	PreDBLP
#subgraphs	394,925	1,054,309
#labels	40	6
#subgraphs for pre-training	306,925	794,862
#subgraphs for fine-tuning	88,000	299,447

Baselines

- EdgePred
- DGI
- ContextPred
- AttrMasking

GNN Architectures

- GCN, GraphSAGE, GAT, GIN

Table 2: Experimental results (mean ± std in percent) of different pre-training strategies w.r.t. various GNN architectures. The improvements are relative to the respective GNN without pre-training.

Model	Biology				PreDBLP			
	GCN	GraphSAGE	GAT	GIN	GCN	GraphSAGE	GAT	GIN
No pre-train	63.22±1.06	65.72±1.23	68.21±1.26	64.82±1.21	62.18±0.43	61.03±0.65	59.63±2.32	69.01±0.23
EdgePred	64.72±1.06	67.39±1.54	67.37±1.31	65.93±1.65	65.44±0.42	63.60±0.21	55.56±1.67	69.43±0.07
DGI	64.33±1.14	66.69±0.88	68.37±0.54	65.16±1.24	65.57±0.36	63.34±0.73	61.30±2.17	69.34±0.09
ContextPred	64.56±1.36	66.31±0.94	66.89±1.98	65.99±1.22	66.11±0.16	62.55±0.11	58.44±1.18	69.37±0.21
AttrMasking	64.35±1.23	64.32±0.78	67.72±1.16	65.72±1.31	65.49±0.52	62.35±0.58	53.34±4.77	68.61±0.16
L2P-GNN (Improv.)	**66.48**±1.59 (5.16%)	**69.89**±1.63 (6.35%)	**69.15**±1.86 (1.38%)	**70.13**±0.95 (8.19%)	**66.58**±0.28 (7.08%)	**65.84**±0.37 (7.88%)	**62.24**±1.89 (4.38 %)	**70.79**±0.17 (2.58%)

Performance Comparison

- **6.27% and 3.52%** improvements compared to the best baseline
- **8.19% and 7.88%** gains relative to non-pretrained models
- **negative transfer** harms the generalization of the pre-trained GNNs

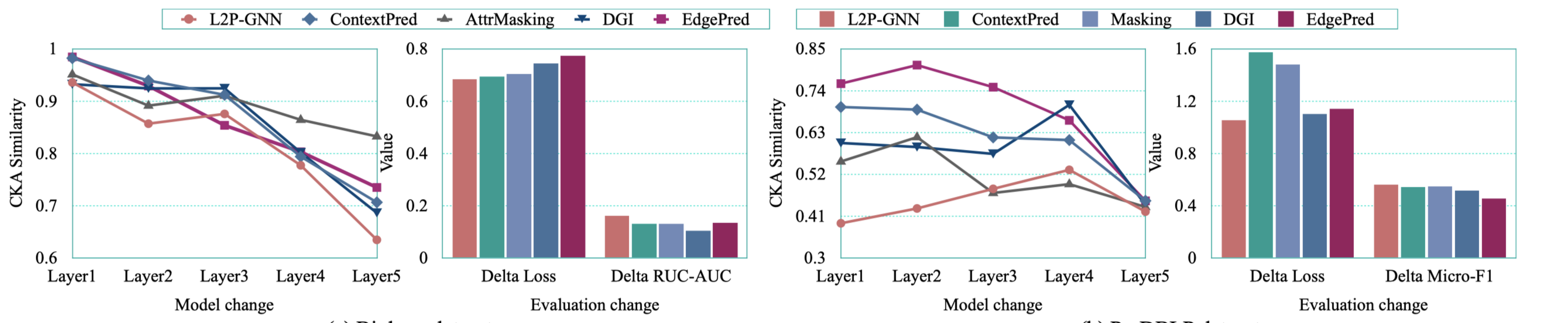
Comparative Analysis

- Centered Kernel Alignment (CKA) similarity between the parameters

Smaller similarity, larger changes of model parameters

- changes in loss and performance (delta loss and RUC-AUC/Micro-F1)

Smaller change, more easily achieve the optimal point



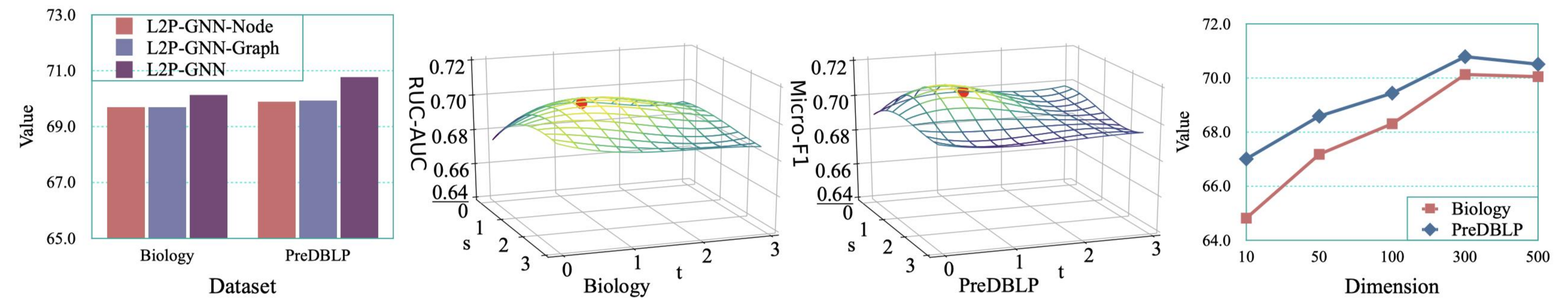
(a) Biology dataset (b) PreDBLP dataset

Ablation Study

- L2P-GNN-Node with only node-level adaptation
- L2P-GNN-Graph with only graph-level adaptation

Parameter Analysis

- the number of node- and graph-level adaptation steps (s, t)
- the dimension of node representations



(a) Ablation study. (b) Node- and graph-level adaptation steps (s, t). (c) Dimension analysis.

Conclusions

- **Problem:** ***There exists a divergence between the pre-training and fine-tuning objectives, resulting in suboptimal pre-trained GNN models***
- **Solution:** ***A self-supervised pretraining strategy for GNNs, L2P-GNN, which attempts to learn how to fine-tune in the pre-training process in the form of transferable prior knowledge***
- **Dataset:** ***A new large-scale graph structured data for pre-training GNNs***