Read "Graph Contrastive Learning with Augmentations" first

Graph Contrastive Learning Automated

Yuning You [1] **Tianlong Chen** [2] **Yang Shen** [1] **Zhangyang Wang** [2]

Abstract

Self-supervised learning on graph-structured data has drawn recent interest for learning generalizable, transferable and robust representations from unlabeled graphs. Among many, graph contrastive learning (GraphCL) has emerged with promising representation learning performance. Unfortunately, unlike its counterpart on image data, the effectiveness of GraphCL hinges on ad-hoc data augmentations, which have to be manually picked per dataset, by either rules of thumb or trial-and-errors, owing to the diverse nature of graph data. That significantly limits the more general applicability of GraphCL. Aiming to fill in this crucial gap, this paper proposes a unified bi-level optimization framework to **automatically**, **adaptively** and **dynamically** select data augmentations when performing GraphCL on specific graph data. The general framework, dubbed **JO**int **A**ugmentation **O**ptimization (**JOAO**), is instantiated as min-max optimization. The selections of augmentations made by JOAO are shown to be in general aligned with previous "best practices" observed from handcrafted tuning: yet now being automated, more flexible and versatile. Moreover, we propose a new augmentation-aware projection head mechanism, which will route output features through different projection heads corresponding to different augmentations chosen at each training step. Extensive experiments demonstrate that JOAO performs on par with or sometimes better than the state-of-the-art competitors including GraphCL, on multiple graph datasets of various scales and types, yet without resorting to any laborious dataset-specific tuning on augmentation selection. We release the code at `https://github.com/Shen-Lab/GraphCL_Automated`.

[1]Texas A&M University [2]The University of Texas at Austin. Correspondence to: Yang Shen <yshen@tamu.edu>, Zhangyang Wang <atlaswang@utexas.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

1. Introduction

Self-supervised learning on graph-structured data has raised significant interests recently (Hu et al., 2019; You et al., 2020b; Jin et al., 2020; Hu et al., 2020b; Hwang et al., 2020; Manessi & Rozza, 2020; Zhu et al., 2020a; Peng et al., 2020a; Rong et al., 2020; Jin et al., 2021b; Roy et al., 2021; Huang et al., 2021). Among many others, graph contrastive learning methods extend the contrastive learning idea (He et al., 2020; Chen et al., 2020c), originally developed in the computer vision domain, to learn generalizable, transferable and robust representations from unlabeled graph data (Veličković et al., 2018; Sun et al., 2019; You et al., 2020a; Qiu et al., 2020; Hassani & Khasahmadi, 2020; Zhu et al., 2020b;c; Chen et al., 2020b;a; Ren et al., 2019; Park et al., 2020; Peng et al., 2020b; Jin et al., 2021a; Wang & Liu, 2021). For comprehensive reviews of the topic, please refer to (Xie et al., 2021; Liu et al., 2021).

Nevertheless, unlike images, graph datasets are abstractions of diverse nature (e.g. citation networks, social networks, and biomedical networks at various levels ranging from molecules to healthcare systems (Li et al., 2021)). Such a unique diversity challenge was not fully addressed by prior graph self-supervised learning approaches (Hu et al., 2019; You et al., 2020a;b). For example, the state-of-the-art graph contrastive learning framework, GraphCL (You et al., 2020a), constructs specific contrastive views of graph data via hand-picking ad-hoc augmentations for every dataset (You et al., 2020a; Zhao et al., 2020; Kong et al., 2020). The choice of augmentation follows empirical rules of thumb, typically summarized from many trial-and-error experiments *per dataset*. That seriously prohibits GraphCL and its variants from broader applicability, considering the tremendous heterogeneity of practical graph data. Moreover, even such trial-and-error selection of augmentations relies on a labeled validation set for downstream evaluation, which is not always available (Dwivedi et al., 2020; Hu et al., 2020a).

Contributions. Given a new and unseen graph dataset, can our graph contrastive learning methods automatically select their data augmentation, avoiding ad-hoc choices or tedious tuning? This paper targets at overcoming this crucial, unique, and inherent hurdle. We propose *joint augmentation optimization* (**JOAO**), a principled bi-level optimization

Two things:
1. Bilevel scheme to automatically learn which augmentations work best on a dataset
2. Use different projection heads for different augmentations

has this been done in CV before?

There is "AutoAugment" by google Brain that selects augmentations via RL but that is very expensive

otherwise we are not sure

which augmentations work well is highly dependent on the dataset

(but maybe NodeDrop and Subgraph are the most generally helpful ones)

framework that *learns to select* data augmentations *for the first time*. To highlight, the selection framework by JOAO is: (i) **automatic**, completely free of human labor of trial-and-error on augmentation choices; (ii) **adaptive**, generalizing smoothly to handling diverse graph data; and (iii) **dynamic**, allowing for augmentation types varying at different training stages. Compared to previous ad-hoc, per-dataset and pre-fixed augmentation selection, JOAO achieves an unprecedented degree of flexibility and ease of use. We summarize our contributions:

- Leveraging GraphCL (You et al., 2020a) as the baseline model, we introduce joint augmentation optimization (JOAO) as a plug-and-play framework. JOAO is the first to automate the augmentation selection when performing contrastive learning on specific graph data. It frees GraphCL from expensive trial-and-errors, or empirical ad-hoc rules, or any validation based on labeled data.
- JOAO can be formulated as a unified bi-level optimization framework, and be instantiated as *min-max optimization*. It takes inspirations from adversarial perturbations as data augmentations (Xie et al., 2020), and can be solved by an alternating gradient-descent algorithm.
- In accordance with diverse and dynamic augmentations enabled by JOAO, we design a new *augmentation-aware* projection head for graph contrastive learning. The rationale is to avoid too many complicated augmentations distorting the original data distribution. The idea is to keep one nonlinear projection head per augmentation pair, and each time using the single head corresponding to the augmentation currently selected by JOAO.
- Extensive experiments demonstrate that GraphCL with JOAO performs on par with or even sometimes better than state-of-the-art (SOTA) competitors, across multiple graph datasets of various types and scales, yet without resorting to tedious dataset-specific manual tuning or domain knowledge. We also show the augmentation selections made by JOAO are in general informed and often aligned with previous "best practices".

We leave two additional remarks: **(1)** JOAO is designed to be flexible and versatile. Although this paper mainly demonstrates JOAO on GraphCL, they are not tied with each other. The general optimization formulation of JOAO allows it to be easily integrated with other graph contrastive learning frameworks too. **(2)** JOAO is designed for automating the tedious and ad-hoc augmentation selection. It intends to *match* the state-of-the-art results achieved by exhaustive manual tuning, but not necessarily to surpass them all. To re-iterate, our aim is to scale up graph contrastive learning to numerous types and scales of graph data in the real world, via a hassle-free framework rather than tuning one by one.

2. Preliminaries and Notations

Graph neural networks (GNNs) have grown into powerful tools to model non-Euclidean graph-structured data arising from various fields (Xu et al., 2018; You et al., 2020c; You & Shen, 2020; Liu et al., 2020; Zhang et al., 2020). Let $G = \{V, E\}$[1] denote an undirected graph in the space $\mathcal{G}$ with V and E being the set of nodes and edges, respectively, and $X_v \in \mathcal{R}^D$ for $v \in V$ being node features. A GNN is defined as the mapping $f : \mathcal{G} \to \mathcal{R}^{D'}$ that encodes a sample graph G into an D'-dimensional vector.

Self-supervised learning on graphs is shown to learn more generalizable, transferable and robust graph representations, through exploiting vast unlabelled data (Jin et al., 2020; Hu et al., 2020b; Hwang et al., 2020; Manessi & Rozza, 2020; Zhu et al., 2020a; Peng et al., 2020a; Rong et al., 2020; Jin et al., 2021b; Xie et al., 2021; Roy et al., 2021; Huang et al., 2021; Li et al., 2021). However, earlier self-supervised tasks often need to be carefully designed with domain knowledge (You et al., 2020b; Hu et al., 2019) due to the intrinsic complicacy of graph datasets.

Graph contrastive learning recently emerges as a promising direction (Veličković et al., 2018; Sun et al., 2019; Qiu et al., 2020; Hassani & Khasahmadi, 2020; Zhu et al., 2020b;c; Chen et al., 2020b;a; Ren et al., 2019; Park et al., 2020; Peng et al., 2020b; Jin et al., 2021a; Wang & Liu, 2021). For example, the SOTA GraphCL framework (You et al., 2020a) enforces the perturbation invariance in GNNs through maximizing agreement between two augmented views of graphs: an overview is illustrated in Figure 1.

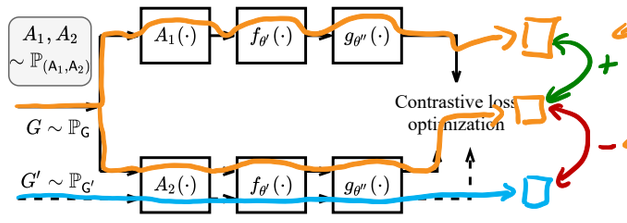

Figure 1: Overview of the GraphCL pipeline in (You et al., 2020a).

Specifically, we denote the input graph-structured sample G from certain empirical distribution $\mathbb{P}_{\mathsf{G}}$. Its samples two random augmentation operators $\mathsf{A}_1, \mathsf{A}_2$ from a given pool of augmentation types as $\mathcal{A} = \{\text{NodeDrop}, \text{Subgraph}, \text{EdgePert}, \text{AttrMask}, \text{Identical}\}$ (You et al., 2020a) and $A \in \mathcal{A} : \mathcal{G} \to \mathcal{G}$. GraphCL (You

[1] We use the sans-serif typeface to denote a random variable (e.g. G). The same letter in the italic font (e.g. G) denotes a sample, and the calligraphic font (e.g. $\mathcal{G}$) denotes the sample space.

The goal is not to beat GraphCL, but to match it without manually searching for the right augmentations.

Is this similar to the FLAG paper?

representations come from the same graph
⇒ increase similarity

representations come from two different graphs
⇒ decrease similarity

The used augmentations:
1. Drop Nodes: NodeDrop
2. Sample Subgraph: Subgraph
3. Add or drop Edges: EdgePert
4. Set attributes to 0: AttrMask

et al., 2020a) optimizes the following loss:

$$\min_\theta \mathcal{L}(\mathsf{G}, \mathsf{A}_1, \mathsf{A}_2, \theta)$$
$$= \min_\theta \Big\{ (-\mathbb{E}_{\mathbb{P}_\mathsf{G} \times \mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)}} \mathrm{sim}(\overbrace{\mathsf{T}_{\theta,1}(\mathsf{G}), \mathsf{T}_{\theta,2}(\mathsf{G})}^{\text{Positive pairs}}) \quad (1)$$
$$+ \mathbb{E}_{\mathbb{P}_\mathsf{G} \times \mathbb{P}_{\mathsf{A}_1}} \log(\mathbb{E}_{\mathbb{P}_{\mathsf{G}'} \times \mathbb{P}_{\mathsf{A}_2}} \exp(\mathrm{sim}(\underbrace{\mathsf{T}_{\theta,1}(\mathsf{G}), \mathsf{T}_{\theta,2}(\mathsf{G}')}_{\text{Negative pairs}}))) \Big\},$$

sample different graph G'

where $\mathsf{T}_{\theta,i} = \mathsf{A}_i \circ f_{\theta'} \circ g_{\theta''}$ $(i = 1, 2)$ is parameterized by $\theta = \{\theta', \theta''\}$, and $f_{\theta'} : \mathcal{G} \to \mathcal{R}^{D'}, g_{\theta''} : \mathcal{R}^{D'} \to \mathcal{R}^{D''}$ are the shared-weight GNN and projection head, respectively, $\mathrm{sim}(u, v) = \frac{u^\top v}{\|u\|\|v\|}$ is the cosine similarity function, $\mathbb{P}_{\mathsf{G}'} = \mathbb{P}_\mathsf{G}$ acts as the negative sampling distribution, and $\mathbb{P}_{\mathsf{A}_1}$ and $\mathbb{P}_{\mathsf{A}_2}$ are the marginal distributions. After the contrastive pre-training, the pre-trained $f_{\theta'^*}$ can be further leveraged for various downstream task fine-tuning.

In the current GraphCL framework, $(\mathsf{A}_1, \mathsf{A}_2)$ are selected by hand and pre-fixed for each dataset. In other words, $\mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)}$ is a Dirac distribution with the only spike at the selected augmentation pair. Yet given new graph data, how to select $(\mathsf{A}_1, \mathsf{A}_2)$ relies on no more than loose heuristics.

3. Methodology

3.1. JOAO: The Unified Framework

One clear limitation in (1) is that one needs to pre-define the sampling distribution $\mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)}$ based on prior rules, and only a Dirac distribution (i.e., only one pair for each dataset) was explored. Rather, we propose to dynamically and automatically learn to optimize $\mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)}$ when performing GraphCL (1), via the following bi-level optimization framework:

outer objective

$$\min_\theta \quad \mathcal{L}(\mathsf{G}, \mathsf{A}_1, \mathsf{A}_2, \theta),$$
$$\text{s.t.} \quad \mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)} \in \arg\min_{\mathbb{P}_{(\mathsf{A}'_1,\mathsf{A}'_2)}} \mathcal{D}(\mathsf{G}, \mathsf{A}'_1, \mathsf{A}'_2, \theta), \quad (2)$$

inner objective

We refer to (2) as joint augmentation optimization (JOAO), where the upper-level objective $\mathcal{L}$ is the same as the GraphCL objective (or the objective of any other graph contrastive learning approach), and the lower-level objective $\mathcal{D}$ optimizes the sampling distribution $\mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)}$ jointly for augmentation-pair selections. Notice that JOAO (2) only exploits the signals from the self-supervised training itself, without accessing downstream labeled data for evaluation.

3.2. Instantiation of JOAO as Min-Max Optimization

Motivated from adversarial training (Wang et al., 2019; Xie et al., 2020), we follow the same philosophy to always exploit the most challenging data augmentation of the current loss, hence instantiating the general JOAO framework as a concrete min-max optimization form:

$$\min_\theta \quad \mathcal{L}(\mathsf{G}, \mathsf{A}_1, \mathsf{A}_2, \theta),$$
$$\text{s.t.} \quad \mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)} \in \arg\max_{\mathbb{P}_{(\mathsf{A}'_1,\mathsf{A}'_2)}} \Big\{ \mathcal{L}(\mathsf{G}, \mathsf{A}'_1, \mathsf{A}'_2, \theta)$$
$$- \frac{\gamma}{2} \mathrm{dist}(\mathbb{P}_{(\mathsf{A}'_1,\mathsf{A}'_2)}, \mathbb{P}_{\text{prior}}) \Big\}, \quad (3)$$

Our goal is to always find the most difficald augmentation
⇒ try all augmentations ⇒ choose the one that produces the highest loss

just a regularization term to encourage more diverse augmentations, because $\mathbb{P}_{prior}$ is uniform

where $\gamma \in \mathcal{R}_{\geq 0}$, $\mathbb{P}_{\text{prior}}$ is the prior distribution on all possible augmentations, and $\mathrm{dist} : \mathcal{P} \times \mathcal{P} \to \mathcal{R}_{\geq 0}$ is a distance function between the sampling and the prior distribution ($\mathcal{P}$ is the probability simplex). Thereby, JOAO's formulation aligns with the idea of model-based adversarial training (Robey et al., 2020), where adversarial training is known to boost generalization, robustness and transferability (Robey et al., 2020; Wang et al., 2019).

In this work, we choose $\mathbb{P}_{\text{prior}}$ as the uniform distribution to promote diversity in the selections, following a common principle of maximum entropy (Guiasu & Shenitzer, 1985) in Bayesian learning. No additional information is assumed about the dataset or the augmentation pool. In practice, it encourageds more diverse augmentation selections rather than collapsing to few. Comparison between the formulations with and without the prior is shown in Table S5 of Appendix E. We use a squared Euclidean distance for $\mathrm{dist}(\cdot, \cdot)$. Accordingly, we have $\mathrm{dist}(\mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)}, \mathbb{P}_{\text{prior}}) = \sum_{i=1}^{|\mathcal{A}|} \sum_{j=1}^{|\mathcal{A}|} (p_{ij} - \frac{1}{|\mathcal{A}|^2})^2$ where the probability $p_{ij} = \mathrm{Prob}(\mathsf{A}_1 = A^i, \mathsf{A}_2 = A^j)$.

We will next present how to optimize (3). Following (Wang et al., 2019), we adopt the alternating gradient descent algorithm (AGD), alternating between upper-level minimization and lower-level maximization, as outlined in Algorithm 1.

Algorithm 1 AGD for optimization (3)

Input: initial parameter $\theta^{(0)}$, sampling distribution $\mathbb{P}^{(0)}_{(\mathsf{A}_1,\mathsf{A}_2)}$, optimization step N.
for $n = 1$ **to** N **do**
  1. Upper-level minimization: fix $\mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)} = \mathbb{P}^{(n-1)}_{(\mathsf{A}_1,\mathsf{A}_2)}$, and call equation (4) to update $\theta^{(n)}$.
  2. Lower-level maximization: fix $\theta = \theta^{(n)}$, and call equation (9) to update $\mathbb{P}^{(n)}_{(\mathsf{A}_1,\mathsf{A}_2)}$.
end for
Return: Optimized parameter $\theta^{(N)}$.

Upper-level minimization. The upper-level minimization w.r.t. θ follows the conventional gradient descent procedure as in the GraphCL optimization (1) given the sampling distribution $\mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)}$, represented as:

$$\theta^{(n)} = \theta^{(n-1)} - \alpha' \nabla_\theta \mathcal{L}(\mathsf{G}, \mathsf{A}_1, \mathsf{A}_2, \theta), \quad (4)$$

where $\alpha' \in \mathcal{R}_{>0}$ is the learning rate.

Have you actually found that the augmentations with the biggest loss yield the best down-stream performance in, for instance, GraphCL?
No easy answer for this currently but empirically we see that it works

Lower-level maximization. Since it is not intuitive to directly calculate the gradient of the lower-level objective w.r.t. $\mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)}$, we first rewrite the contrastive loss in (1) as:

$$\mathcal{L}(\mathsf{G},\mathsf{A}_1,\mathsf{A}_2,\theta)=\sum_{i=1}^{|\mathcal{A}|}\sum_{j=1}^{|\mathcal{A}|}\overbrace{p_{ij}}^{\text{Targeted}}\Big\{-\mathbb{E}_{\mathbb{P}_\mathsf{G}}\text{sim}(T_\theta^i(\mathsf{G}),T_\theta^j(\mathsf{G}))$$
$$+\mathbb{E}_{\mathbb{P}_\mathsf{G}}\log(\sum_{j'=1}^{|\mathcal{A}|}\underbrace{p_{j'}}_{\text{Undesired}}\mathbb{E}_{\mathbb{P}_{\mathsf{G}'}}\exp(\text{sim}(T_\theta^i(\mathsf{G}),T_\theta^{j'}(\mathsf{G}'))))\Big\}, \tag{5}$$

where $T_\theta^i=A^i\circ f_{\theta'}\circ g_{\theta''},\ (i=1,...,5)$, and the marginal probabilities $p_{j'}=p_j=\text{Prob}(\mathsf{A}_2=A^j)$. In the equation (5), we expand the expectation on augmentations $\mathsf{A}_1,\mathsf{A}_2$ into the form of weighted summation related to p_{ij} in order to calculate the gradient. However, within the expectation on G of the negative pair term there is the marginal probabilities $p_{j'}$ entangled, and therefore we make the following numerical approximation for the lower bound of the negative pair term to disentangle p_{ij} in the equation (5):

Jensen's inequality →
$$\begin{aligned}&\mathbb{E}_{\mathbb{P}_\mathsf{G}\times\mathbb{P}_{\mathsf{A}_1}}\log(\mathbb{E}_{\mathbb{P}_{\mathsf{G}'}\times\mathbb{P}_{\mathsf{A}_2}}\exp(\text{sim}(\mathsf{T}_{\theta,1}(\mathsf{G}),\mathsf{T}_{\theta,2}(\mathsf{G}'))))\\ \geq\ &\mathbb{E}_{\mathbb{P}_\mathsf{G}\times\mathbb{P}_{\mathsf{A}_1}\times\mathbb{P}_{\mathsf{A}_2}}\log(\mathbb{E}_{\mathbb{P}_{\mathsf{G}'}}\exp(\text{sim}(\mathsf{T}_{\theta,1}(\mathsf{G}),\mathsf{T}_{\theta,2}(\mathsf{G}'))))\\ \approx\ &\mathbb{E}_{\mathbb{P}_\mathsf{G}\times\mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_1)}}\log(\mathbb{E}_{\mathbb{P}_{\mathsf{G}'}}\exp(\text{sim}(\mathsf{T}_{\theta,1}(\mathsf{G}),\mathsf{T}_{\theta,2}(\mathsf{G}')))),\end{aligned} \tag{6}$$

where the first inequality comes from Jensen's inequality, and the second approximation is numerical. It results in the approximated contrastive loss:

$$\begin{aligned}\mathcal{L}(\mathsf{G},\mathsf{A}_1,\mathsf{A}_2,\theta)&\approx\sum_{i=1}^{|\mathcal{A}|}\sum_{j=1}^{|\mathcal{A}|}\overbrace{p_{ij}}^{\text{Targeted}}\ell(\mathsf{G},A^i,A^j,\theta)\\ &=\sum_{i=1}^{|\mathcal{A}|}\sum_{j=1}^{|\mathcal{A}|}p_{ij}\Big\{-\mathbb{E}_{\mathbb{P}_\mathsf{G}}\text{sim}(T_\theta^i(\mathsf{G}),T_\theta^j(\mathsf{G}))\\ &\quad+\mathbb{E}_{\mathbb{P}_\mathsf{G}}\log(\mathbb{E}_{\mathbb{P}_{\mathsf{G}'}}\exp(\text{sim}(T_\theta^i(\mathsf{G}),T_\theta^j(\mathsf{G}'))))\Big\}.\end{aligned} \tag{7}$$

Through approximating the contrastive loss, the lower-level maximization in the optimization (3) is rewritten as:

$$\mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)}\in\arg\max_{\boldsymbol{p}\in\mathcal{P},\boldsymbol{p}=[p_{ij}],i,j=1,...,|\mathcal{A}|}\{\psi(\boldsymbol{p})\},$$
$$\psi(\boldsymbol{p})=\sum_{i=1}^{|\mathcal{A}|}\sum_{j=1}^{|\mathcal{A}|}p_{ij}\ell(\mathsf{G},A^i,A^j,\theta)-\frac{\gamma}{2}\sum_{i=1}^{|\mathcal{A}|}\sum_{j=1}^{|\mathcal{A}|}(p_{ij}-\frac{1}{|\mathcal{A}|^2})^2, \tag{8}$$

find objective of lower level optimization

where $\psi(\boldsymbol{p})$ is a strongly-concave function w.r.t. $\boldsymbol{p}$ in the probability simplex $\mathcal{P}$. Thus, a projected gradient descent (Wang et al., 2019; Boyd et al., 2004) is performed to update the sampling distribution $\mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)}$ for selecting augmentation pairs, expressed as:

$$\boldsymbol{b}=\boldsymbol{p}^{(n-1)}+\alpha''\nabla_{\boldsymbol{p}}\psi(\boldsymbol{p}^{(n-1)}),\boldsymbol{p}^{(n)}=(\boldsymbol{b}-\mu\mathbf{1})_+, \tag{9}$$

calculate the usual gradient

project onto probability simplex to obtain new $p^{(n)}$

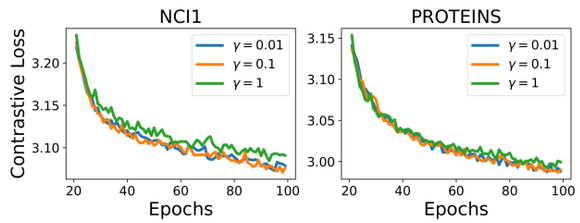

Figure 2: Empirical training curves of AGD in JOAO on datasets NCI1 and PROTEINS with different γ values.

where $\alpha''\in\mathcal{R}_{>0}$ is the learning rate, μ is the root of the equation $\mathbf{1}^\mathsf{T}(\boldsymbol{b}-\mu\mathbf{1})=1$, and $(\cdot)_+$ is the element-wise non-negative operator. μ can be efficiently found via the bi-jection method (Wang et al., 2019; Boyd et al., 2004).

Even though an optimizer with theoretical guarantee of convergence for non-convex non-concave min-max problems remains an open challenge, we acknowledge that AGD is an approximation of solving the bi-level optimization (3) precisely, which typically costs Bayesian optimization (Srinivas et al., 2010; Snoek et al., 2012), automatic differentiation (Luketina et al., 2016; Franceschi et al., 2017; Baydin et al., 2017; Shaban et al., 2019), or first-order techniques based on some inner-loop approximated solution (Maclaurin et al., 2015; Pedregosa, 2016; Gould et al., 2016). As most of them suffer from high time or space complexity, AGD was adopted as an approximated heuristic mainly for saving computational overhead. It showed some level of empirical convergence as seen in Figure 2.

the same

3.2.1. Sanity Check: JOAO Recovers Augmentation-Pairs Aligned with Previous "Best Practices"

passes

How reasonable are the JOAO-selected augmentation pairs per dataset? This section pass JOAO through a sanity check, by comparing its selections with the previous trial-and-error findings by manually and exhaustively combining different augmentations (using downstream labels for validation) (You et al., 2020a).

Table 1: Datasets statistics.

Datasets	Category	Graph Num.	Avg. Node	Avg. Degree
NCI1	Biochemical Molecules	4110	29.87	1.08
PROTEINS	Biochemical Molecules	1113	39.06	1.86
COLLAB	Social Networks	5000	74.49	32.99
RDT-B	Social Networks	2000	429.63	1.15

To examine such alignment, we visualize in the top row of Figure 3 the JOAO-optimized sampling distributions $\mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)}$, and in the bottom row the GraphCL's manual trial-and-error results over various augmentation pairs, for four different datasets (data statistics in Table 1). Please refer to the caption on Figure 3 how to interpret the percentage numbers in the top and bottom rows respectively. Overall, we observe a decent extent of alignments between the two rows'

Can you explain the optimization a bit more?
How can we update the parameters of P after making a discrete choice of augmentation?

2 level optimization
1. Update network weights to minimize contrastive loss for the sampled augmentations

$L(G, A^i, A^j, \theta)$

2. Update augmentation probabilities P with loss $\psi(p)$ that we want to maximize

number of augmentations

$$\psi(p)=\sum_{i=1}^{|\mathcal{A}|}\sum_{j=1}^{|\mathcal{A}|}p_{ij}\,L(G,A^i,A^j,\theta)-\frac{\gamma}{2}\sum_{i=1}^{|\mathcal{A}|}\sum_{j=1}^{|\mathcal{A}|}\left(p_{ij}-\frac{1}{|\mathcal{A}|^2}\right)^2$$

regularization

Sum of all possible losses weighted by the probabilities p_{ij} with which we choose them

probabilities should be close to uniform

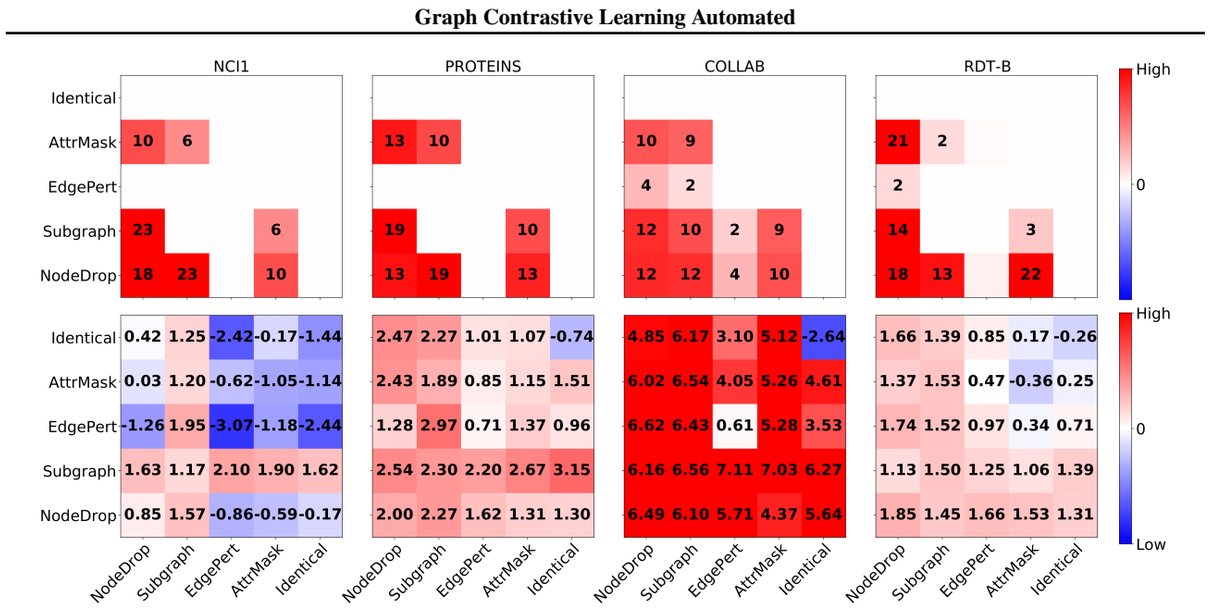

Figure 3: Top row: sampling distributions (%, defined as the percentage of this specific augmentation pair being selected during the entire training process) for augmentation pairs selected by JOAO on four different datasets (NCI1, PROTEINS, COLLAB, and RDT-B). **Bottom row**: GraphCL performance gains (classification accuracy %, see (You et al., 2020a) for the detailed setting) when exhaustively trying every possible augmentation pair. **Note that** the percentage numbers in the first and second rows have different meanings and are not apple-to-apple comparable; however, the overall alignments between the two rows' trends and high-value locations indicate that, if an augmentation pair was manually verified to yield better GraphCL results, it is also more likely to be selected by JOAO. Warmer (colder) colors indicate higher (lower) values, and white marks 0.

trends and especially the high-value locations, indicating that: if an augmentation pair was manually verified to yield better GraphCL results, it is also more likely to be selected by JOAO. More specifically we can see: (1) augmentation pairs containing EdgePert and AttrMask are more likely to be selected for biochemical molecules and denser graphs, respectively; (2) NodeDrop and Subgraph are generally adopted on all four datasets; and (3) the augmentation pairs of two identy transformations are completely abandoned by JOAO, and those pairs with more diverse transformation types are more desired. All those observations are well aligned with the "rules of thumb" summarized in (You et al., 2020a). More discussions are provided in Appendix D.

Therefore, the selections of augmentations made by JOAO are shown to be generally consistent with previous "best practices" observed from manual tuning – yet now being fully automated, flexible, versatile. It is also achieved without using any downstream task label, while (You et al., 2020a) would hinge on a labeled set to compare two augmentations by their downstream performance.

3.3. Augmentation-Aware Multi-Projection Heads: Addressing A New Challenge from JOAO

JOAO conveys the blessing of diverse and dynamic augmentations that are selected automatically during each GraphCL training, which may yield more robust and invariant features. However, that blessing could also bring up a new challenge: compared to one fixed augmentation pair throughout training, those varying and more aggressive augmentations can distort the training distribution more (Lee et al., 2020; Jun et al., 2020). Even mild augmentations, such as adding/dropping nodes or edges, could result in graphs very unlikely under the original distribution. Models trained with these augmentations may fit the original distribution poorly.

To address this challenge arising from using JOAO, we introduce multiple projection heads and an augmentation-aware selection scheme into GraphCL, as glimpsed in Figure 4 (see Figure S2 in Appendix D for a schematic diagram). Specifically, we construct $|\mathcal{A}|$ projection heads each of which corresponds to one augmentation type ($|\mathcal{A}|$ denotes the cardinality of the augmentation pool). Then during training, once an augmentation is sampled, it will only go through and update its corresponding projection head. The main idea is to explicitly disentangle the distorted feature distributions caused by various augmentation pairs, and each time we only use the one head corresponding to the augmentation currently selected by JOAO.

In mathematical forms, we route the output features from f through the projection head sampled from $\mathbb{P}_{(g_{\Theta_1''}, g_{\Theta_2''})}$ at each training step, where $\mathbb{P}_{(g_{\Theta_1''}, g_{\Theta_2''})} = \mathbb{P}_{(\mathsf{A}_1, \mathsf{A}_2)}$, and Θ_1'', Θ_2'' denote the head parameters, resulting in $\mathsf{T}_{\theta,i} = \mathsf{A}_i \circ f_{\theta'} \circ g_{\Theta_i''}$, $(i = 1, 2)$. Denot-

← Which augmentations JOAO selects

⇒ mostly NodeDrop and Subgraph are best

← Performance of GraphCL with the different augmentations

Now we have different augmentations in the same run and therefore use different projection heads for each augmentation

However!:

The projection head is also random just with the same probability as the augmentations.

Using one fixed projection head for each augmentation pair does not really give different performance.

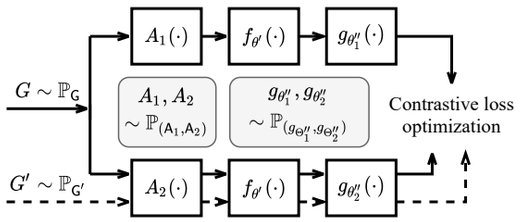

Figure 4: An overview of GraphCL with multiple augmentation-aware projection heads where $\mathbb{P}_{(g_{\Theta_1''}, g_{\Theta_2''})} = \mathbb{P}_{(A_1,A_2)}$.

ing $\mathcal{L}_{v2}(G, A_1, A_2, \theta', \Theta_1'', \Theta_2'') = \mathbb{E}_{\mathbb{P}_{(g_{\Theta_1''}, g_{\Theta_2''})}} \mathcal{L}(G, A_1, A_2, \{\theta', (\Theta_1'', \Theta_2'')\})$, we could then integrate the augmentation-aware projection head mechanism into the JOAO framework, referred to as **JOAOv2**:

$$\begin{aligned} \min_{\theta} \quad & \mathcal{L}_{v2}(G, A_1, A_2, \theta', \Theta_1'', \Theta_2''), \\ \text{s.t.} \quad & \mathbb{P}_{(A_1,A_2)} \in \arg\max_{\mathbb{P}_{(A_1',A_2')}} \Big\{ \mathcal{L}_{v2}(G, A_1, A_2, \theta', \Theta_1'', \Theta_2'') \\ & - \frac{\gamma}{2} \operatorname{dist}(\mathbb{P}_{(A_1',A_2')}, \mathbb{P}_{\text{prior}}) \Big\}, \\ & \mathbb{P}_{(g_{\Theta_1''}, g_{\Theta_2''})} = \mathbb{P}_{(A_1,A_2)}. \end{aligned} \tag{10}$$

Algorithm 1 could be easily adapted to solve (10) (see Algorithm S1 of Appendix A).

Our preliminary experiments in Table 2 show that, without bells and whistles, augmentation-aware projection heads improve the performance upon JOAO under different augmentation strengths. That aligns with the observations in (Lee et al., 2020; Jun et al., 2020), showing that disentangling augmented and original feature distributions could have the model benefit more from stronger augmentations.

Table 2: Experiments with JOAO and JOAOv2 without explicit hyper-parameter tuning under different augmentation strengths on NCI1 and PROTEINS. A.S. is short for augmentation strength.

Datasets	A.S.	JOAO	JOAOv2
NCI1	0.2	61.77±1.61	62.52±1.16
	0.25	60.95±0.55	61.67±0.72
PROTEINS	0.2	71.45±0.89	71.66±1.10
	0.25	71.61±1.65	73.01±1.02

We also plot the learned $\mathbb{P}_{(A_1,A_2)}$ in Figure S1 of Appendix D, where we can observe an ever stronger alignment than presented in Figure 3.

4. Experiments

In this section, we evaluate our proposed methods, JOAO and JOAOv2, against state-of-the-art (SOTA) *competitors* including self-supervised approaches heuristically designed with domain knowledge, and graph contrastive learning (GraphCL) with pre-defined rules for augmentation selection, using the *scenarios* of datasets originated from diverse sources, and datasets on specific bioinformatics domains. An summary of main results can be found in Table 3.

Table 3: Summary of JOAO performance.

	v.s. GraphCL	v.s. Heuristic methods
Across diverse fields	Comparable	Better
On specific domains	Better	Worse

4.1. Datasets and Experiment Settings

Datasets. We use datasets of diverse nature from the benchmark TUDataset (Morris et al., 2020), including graph data for small molecules & proteins (Riesen & Bunke, 2008; Dobson & Doig, 2003), computer vision (Nene et al., 1996) and various relation networks (Yanardag & Vishwanathan, 2015; Rozemberczki et al., 2020) of diverse statistics (see Table S1 of Appendix B), under semi-supervised and unsupervised learning. Additionally we gather domain-specific bioinformatics datasets from the benchmark (Hu et al., 2019) of relatively similar statistics (see Table S2 of Appendix B), under transfer-learning tasks for predicting molecules' chemical property or proteins' biological function. Lastly we take two large-scale benchmark datasets, ogbg-ppa & ogbg-code from Open Graph Benchmark (OGB) (Hu et al., 2020a) (see Table S3 of Appendix B for statistics) to evaluate scalability under semi-supervised learning.

Learning protocols. Learning experiments are performed in three settings, following the same protocols as in SOTA work. (1) In semi-supervised learning (You et al., 2020a) on datasets without the explicit training/validation/test split, we perform pre-training with all data and did finetuning & evaluation with K folds where $K = \frac{1}{\text{label rate}}$; and on datasets with the train/validation/test split, we only perform pre-training with the training data, finetuning on the partial training data and evaluation on the validation/test sets. (2) In unsupervised representation learning (Sun et al., 2019), we pre-train using the whole dataset to learn graph embeddings and feed them into a downstream SVM classifier with 10-fold cross-validation. (3) In transfer learning (Hu et al., 2019), we pre-train on a larger dataset then finetune and evaluate on smaller datasets of the same category using the given training/validation/test split.

GNN architectures & augmentations. We adopt the same GNN architectures with default hyper-parameters as in the SOTA methods under individual experiment settings. Specifically, (1) in semi-supervised learning, ResGCN (Chen et al., 2019) is used with 5 layers and 128 hidden dimensions, (2) in unsupervised representation learning, GIN (Xu et al., 2018) is used with 3 layers and 32 hidden dimensions, and (3) in transfer learning and on large-scale OGB datasets, GIN is used with 5 layers and 300 hidden dimensions. Plus, we adopt the same graph data augmentations as in GraphCL

(You et al., 2020a) with the default augmentation strength 0.2. We tune the hyper-parameter γ controlling the trade-off in the optimization (3) in the range of $\{0.01, 0.1, 1\}$.

4.2. Compared Algorithms.

Training from scratch (with augmentations) and graph kernels. The naïve baseline training from the random initialization (with same augmentations as in GraphCL (You et al., 2020a)) is compared, as well as SOTA graph kernel methods including GL (Shervashidze et al., 2009), WL (Shervashidze et al., 2011) and DGK (Yanardag & Vishwanathan, 2015).

Heuristic self-supervised methods. Heuristic self-supervised methods are designed based on certain domain knowledge, which work well when such knowledge is available and benefits downstream tasks. The compared ones include: (1) edge-based reconstruction including GAE (Kipf & Welling, 2016), node2vec (Grover & Leskovec, 2016) and EdgePred (Hu et al., 2019), (2) vertex feature masking & recover, namely AttrMasking (Hu et al., 2019), (3) sub-structure information preserving such as sub2vec (Adhikari et al., 2018), graph2vec (Narayanan et al., 2017) and ContextPred (Hu et al., 2019), and (4) global-local representation consistency such as Infomax (Veličković et al., 2018) & InfoGraph (Sun et al., 2019). We adopt the default hyper-parameters published for these methods.

GraphCL with pre-fixed augmentation sampling rules. For constructing the sampling pool of augmentations, we follow the same rule as in (You et al., 2020a) that uses (1) NodeDrop and Subgraph for biochemical molecules, (2) all for dense relation networks, and (3) all except AttrMask for sparse relation networks. The exact augmentations for each dataset are shown in Table S4 of Appendix C.

4.3. Results

4.3.1. On Diverse Datasets from TUDataset

The results of semi-supervised learning & unsupervised representation learning on TUDataset are in Tables 4 & 5, respectively. Through comparisons between (1) JOAO and GraphCL, (2) JOAOv2 and JOAO, and (3) JOAOv2 and heuristic self-supervised methods, we have the following observations.

(i) With automated selection, JOAO is comparable to GraphCL with ad hoc rules from exhaustive manual tuning. With the automatic, adaptive, and dynamic augmentation selection procedure, JOAO performs comparably to GraphCL whose augmentations are based on empirical ad-hoc rules gained from expensive trial-and-errors on the same TUDatase. Specifically in semi-supervised learning (Table 4), JOAO matches or beats GraphCL in 7 out of 10 experiments, albeit with a slightly worse average rank. Similar observations were made in unsupervised learning (Table 5). These results echo our earlier results in Section 3.2.1 that automated selections of augmentation pairs made by JOAO were generally consistent with GraphCL's "best practices" observed from manual tuning.

(ii) Augmentation-aware projection heads provide further improvement upon JOAO. With augmentation-aware projection heads introduced into JOAO, JOAOv2 further improves the performance and sometimes even outperforms GraphCL with the pre-defined rules of thumb for augmentation selections. In semi-supervised learning (Table 4), JOAOv2 achieves the best average ranks of 2.0 and 2.8 under 1% and 10% label rate, respectively, and in unsupervised representation learning (Table 5) its average rank (2.8) is only edged by GraphCL (2.6). The performance acquired by JOAOv2 echoes our conjecture in sec. 3.3 that such explicit disentanglement of the distorted feature distributions caused by various augmentation pairs would reel in the benefits from stronger augmentations.

(iii) Across diverse datasets, JOAOv2 generally outperform heuristic self-supervised methods. On datasets originated from diverse sources, JOAOv2 generally outperforms heuristic self-supervised methods. Specifically in Table 4, JOAOv2 achieves no less than 0.3 average ranking gap with all heuristic self-supervised methods under 1% label rate, and 1.0 with all but Infomax under 10% label rate, which outperforms JOAO but still underperforms JOAOv2, and in Table 5 only InfoGraph outperforms JOAO but underperforms JOAOv2 where there is no less that 1.5 average ranking gap between others and JOAOv2. This in general meets our expectation that heuristic self-supervised methods can work well when guided by useful domain knowledge, which is hard to guarantee across diverse datasets. In contrast JOAOv2 can dynamically and automatically adapt augmentation selections during self-supervised training, exploiting the signals (knowledge) from data.

4.3.2. On Specific Bioinformatics Datasets.

The results of transfer learning on bioinfomatics datasets are in Table 6. Similarly, through comparisons among JOAO(v2), GraphCL and heuristic self-supervised methods, we make the following findings.

(iv) Without domain expertise incorporated, JOAOv2 underperforms some heuristic self-supervised methods in specific domains. Nevertheless, converse to that on diverse datasets, on the specific bioinformatics datasets, JOAOv2 underperforms some heuristic self-supervised methods designed with dataset-specific domain knowledge (Hu et al., 2019) even though it improves average rank against GraphCL and JOAO. As stated in Section 4.2, the specific domain knowledge encoded in the compared heuristically designed methods correlates with the downstream

What would you say in which domain GraphCL is the most valuable? Molecules?
We cannot really say something this specific but Yuning thinks GraphCL is more promising for graph level tasks than node level tasks

Table 4: Semi-supervised learning on TUDataset. Shown in **red** are the best accuracy (%) and those within the standard deviation of the best accuracy or the best average ranks. - indicates that label rate is too low for a given dataset size. L.R. and A.R. are short for label rate and average rank, respectively. The compared results except those for ContextPred are as published under the same experiment setting.

L.R.	Methods	NCI1	PROTEINS	DD	COLLAB	RDT-B	RDT-M5K	GITHUB	A.R.↓
1%	No pre-train.	60.72±0.45	-	-	57.46±0.25	-	-	54.25±0.22	7.6
	Augmentations	60.49±0.46	-	-	58.40±0.97	-	-	56.36±0.42	6.6
	GAE	61.63±0.84	-	-	63.20±0.67	-	-	59.44±0.44	4.0
	Infomax	**62.72**±0.65	-	-	61.70±0.77	-	-	58.99±0.50	3.3
	ContextPred	61.21±0.77	-	-	57.60±2.07	-	-	56.20±0.49	6.6
	GraphCL	**62.55**±0.86	-	-	**64.57**±1.15	-	-	58.56±0.59	**2.6**
	JOAO	61.97±0.72	-	-	**63.71**±0.84	-	-	60.35±0.24	3.0
	JOAOv2	**62.52**±1.16	-	-	**64.51**±2.21	-	-	**61.05**±0.31	**2.0**
10%	No pre-train.	73.72±0.24	70.40±1.54	73.56±0.41	73.71±0.27	86.63±0.27	51.33±0.44	60.87±0.17	7.0
	Augmentations	73.59±0.32	70.29±0.64	74.30±0.81	74.19±0.13	87.74±0.39	52.01±0.20	60.91±0.32	6.2
	GAE	74.36±0.24	70.51±0.17	74.54±0.68	75.09±0.19	87.69±0.40	**53.58**±0.13	63.89±0.52	4.5
	Infomax	**74.86**±0.26	72.27±0.40	**75.78**±0.34	73.76±0.29	88.66±0.95	**53.61**±0.31	65.21±0.88	3.0
	ContextPred	73.00±0.30	70.23±0.63	74.66±0.51	73.69±0.37	84.76±0.52	51.23±0.84	62.35±0.73	7.2
	GraphCL	**74.63**±0.25	**74.17**±0.34	**76.17**±1.37	74.23±0.21	**89.11**±0.19	52.55±0.45	65.81±0.79	**2.4**
	JOAO	74.48±0.27	72.13±0.92	**75.69**±0.67	**75.30**±0.32	88.14±0.25	52.83±0.54	65.00±0.30	3.5
	JOAOv2	**74.86**±0.39	73.31±0.48	**75.81**±0.73	**75.53**±0.18	88.79±0.65	52.71±0.28	**66.66**±0.60	**1.8**

Table 5: Unsupervised representation learning on TUDataset. **Red** numbers indicate the top-3 accuracy (%) or the top-2 average ranks. The compared results are from the published papers, and - indicates that results were not available in published papers. For MVGRL we report the numbers with the NT-Xent loss to be comparable with GraphCL.

Methods	NCI1	PROTEINS	DD	MUTAG	COLLAB	RDT-B	RDT-M5K	IMDB-B	A.R.↓
GL	-	-	-	81.66±2.11	-	77.34±0.18	41.01±0.17	65.87±0.98	7.4
WL	**80.01**±0.50	72.92±0.56	-	80.72±3.00	-	68.82±0.41	46.06±0.21	**72.30**±3.44	5.7
DGK	**80.31**±0.46	73.30±0.82	-	87.44±2.72	-	78.04±0.39	41.27±0.18	66.96±0.56	4.9
node2vec	54.89±1.61	57.49±3.57	-	72.63±10.20	-	-	-	-	8.6
sub2vec	52.84±1.47	53.03±5.55	-	61.05±15.80	-	71.48±0.41	36.68±0.42	55.26±1.54	9.5
graph2vec	73.22±1.81	73.30±2.05	-	83.15±9.25	-	75.78±1.03	47.86±0.26	71.10±0.54	5.7
MVGRL	-	-	-	75.40±7.80	-	82.00±1.10	-	63.60±4.20	7.2
InfoGraph	76.20±1.06	**74.44**±0.31	72.85±1.78	**89.01**±1.13	**70.65**±1.13	82.50±1.42	53.46±1.03	**73.03**±0.87	3.0
GraphCL	77.87±0.41	**74.39**±0.45	**78.62**±0.40	86.80±1.34	**71.36**±1.15	**89.53**±0.84	**55.99**±0.28	**71.14**±0.44	**2.6**
JOAO	**78.07**±0.47	**74.55**±0.41	**77.32**±0.54	**87.35**±1.02	**69.50**±0.36	**85.29**±1.35	**55.74**±0.63	70.21±3.08	3.3
JOAOv2	78.36±0.53	74.07±1.10	**77.40**±1.15	**87.67**±0.79	69.33±0.34	**86.42**±1.45	**56.03**±0.27	70.83±0.25	**2.8**

datasets as shown in (Hu et al., 2019), which is not made available to benefit JOAOv2 that works well for a *general* dataset (as shown in Section 4.3.1), which may not suffice to capture the sophisticated domain expertise. Therefore, to make JOAOv2 even more competitive, domain knowledge can be introduced into the framework, for instance through proposing dataset-specific augmentation types and/or priors. We leave this to future work.

(v) With better generalizability, JOAOv2 outperforms GraphCL on unseen datasets. Different from results on diverse datasets from TUDataset, both JOAO and JOAOv2 outperform GraphCL with empirically pre-defined rules for augmentation selection on the unseen bioinfomatics datasets. Specifically in Table 6 JOAO improves the average rank by 0.1 and JOAOv2 did by 0.3 compared to GraphCL. Note that the sampling rules of GraphCL were empirically derived from TUDataset hence these rules are not necessarily effective for the previously-unseen bioinfomatics datasets. In contrast, JOAOv2 dynamically and automatically learns the sampling distributions during self-supervised training, possessing the better generalizability.

4.3.3. On Large-Scale OGB Datasets.

(vi) JOAOv2 scales up well for large datasets. Both JOAO and JOAOv2 scale up for large datasets at least as well as GraphCL does. In Table 7, for the ogbg-ppa dataset, JOAO improved even more significantly compared to GraphCL (by reference to earlier, smaller datasets), with $>$3.49% and $>$1.55% accuracy gains at 1% and 10% label rates, respectively.

4.4. Summary of Main Findings

We briefly summarize the aforementioned results as follows:

- Across datasets originated from diverse sources, JOAO with adaptive augmentation selection performs comparably to GraphCL, a strong baseline with exhaustively tuned augmentation rules by hand.
- With augmentation-aware projection heads, JOAOv2 further boosts the performance and sometimes even

Table 6: Transfer learning on bioinformatics datasets. **Red** numbers indicate the top-3 performances (AUC of ROC in %). Results for SOTA methods are as published.

Methods	BBBP	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV	BACE	PPI	A.R.↓
No pre-train.	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	64.8±1.0	6.6
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	64.1±1.5	5.3
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	**65.7**±1.3	3.8
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	**65.2**±1.6	3.1
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	64.4±1.3	3.4
GraphCL	**69.68**±0.67	73.87±0.66	62.40±0.57	**60.53**±0.88	**75.99**±2.65	69.80±2.66	**78.47**±1.22	75.38±1.44	**67.88**±0.85	4.6
JOAO	**70.22**±0.98	74.98±0.29	62.94±0.48	59.97±0.79	**81.32**±2.49	71.66±1.43	76.73±1.23	77.34±0.48	64.43±1.38	4.5
JOAOv2	**71.39**±0.92	74.27±0.62	63.16±0.45	60.49±0.74	**80.97**±1.64	73.67±1.00	**77.51**±1.17	75.49±1.27	63.94±1.59	4.3

Table 7: Semi-supervised learning on large-scale OGB datasets. **Red** numbers indicate the top-2 performances (accuracy in % on ogbg-ppa, F1 score in % on ogbg-code).

L.R.	Methods	ogbg-ppa	ogbg-code
1%	No pre-train.	16.04±0.74	6.06±0.01
	GraphCL	40.81±1.33	**7.66**±0.25
	JOAO	**47.19**±1.30	6.84±0.31
	JOAOv2	**44.30**±1.67	**7.74**±0.24
10%	No pre-train.	56.01±1.05	17.85±0.60
	GraphCL	57.77±1.25	**22.45**±0.17
	JOAO	**60.91**±0.83	22.06±0.30
	JOAOv2	**59.32**±1.11	**22.65**±0.22

outperforms GraphCL.

- On datasets from specific bioinformatics domains, JOAOv2 achieves better performance than GraphCL whose empirical rules were not derived from such data, indicating its better generalizability to unseen datasets.
- Both JOAO and JOAOv2 outperform heuristic self-supervised methods with few exceptions. They might be further enhanced by encoding domain knowledge.
- JOAOv2 scales up to large datasets as well as GraphCL does, sometimes with even more significant improvement compared with that for smaller datasets.

5. Conclusions & Discussions

In this paper, we propose a unified bi-level optimization framework to dynamically and automatically select augmentations in GraphCL, named JOint Augmentation Optimization (JOAO). The general framework is instantiated as min-max optimization, with empirical analysis showing that JOAO makes augmentation selections in general accordance with previous "best practices" from exhaustive hand tuning for every dataset. Furthermore, a new augmentation-aware projection head mechanism is proposed to overcome the potential training distribution distortion, resulting from the more aggressive and varying augmentations by JOAO. Experiments demonstrate that JOAO and its variant performs on par with and sometimes better than the state-of-the-art competitors including GraphCL on multiple graph datasets of various scales and types, yet without resorting to tedious dataset-specific manual tuning.

Although JOAO automates GraphCL in selecting augmentation pairs, it still relies on human prior knowledge in constructing and configuring the augmentation pool to select from. In this sense "full" automation is still desired and will be pursued in future work. Meanwhile, in parallel to the principled formulation of bi-level optimization, a meta-learning formulation can also be pursued.

References

Adhikari, B., Zhang, Y., Ramakrishnan, N., and Prakash, B. A. Sub2vec: Feature learning for subgraphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 170–182. Springer, 2018.

Baydin, A. G., Cornish, R., Rubio, D. M., Schmidt, M., and Wood, F. Online learning rate adaptation with hypergradient descent. *arXiv preprint arXiv:1703.04782*, 2017.

Boyd, S., Boyd, S. P., and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Chen, B., Zhang, J., Zhang, X., Tang, X., Cai, L., Chen, H., Li, C., Zhang, P., and Tang, J. COAD: Contrastive pre-training with adversarial fine-tuning for zero-shot expert linking. *arXiv preprint arXiv:2012.11336*, 2020a.

Chen, D., Lin, Y., Li, L., Li, X. R., Zhou, J., Sun, X., et al. Distance-wise graph contrastive learning. *arXiv preprint arXiv:2012.07437*, 2020b.

Chen, T., Bian, S., and Sun, Y. Are powerful graph neural nets necessary? a dissection on graph classification. *arXiv preprint arXiv:1905.04579*, 2019.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020c.

What does the augmentation distribution look like for molecules?
Similar to figure 3.

What is your reasoning why this works?
If we drop an atom, the molecule does not stay semantically the same unlike for images if we for example rotate them.
It depends on the downstream task if the augmentation leaves the molecule semantically the same.
For some tasks dropping some atoms does maybe not matter.

How did you get the idea?
Observations from GraphCL

What is that?

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

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Franceschi, L., Donini, M., Frasconi, P., and Pontil, M. Forward and reverse gradient-based hyperparameter optimization. In *International Conference on Machine Learning*, pp. 1165–1173. PMLR, 2017.

Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R. S., and Guo, E. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 2016.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

Guiasu, S. and Shenitzer, A. The principle of maximum entropy. *The mathematical intelligencer*, 7(1):42–48, 1985.

Hassani, K. and Khasahmadi, A. H. Contrastive multi-view representation learning on graphs. *arXiv preprint arXiv:2006.05582*, 2020.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020a.

Hu, Z., Dong, Y., Wang, K., Chang, K.-W., and Sun, Y. GPT-GNN: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1857–1867, 2020b.

Huang, T., Pei, Y., Menkovski, V., and Pechenizkiy, M. Hop-count based self-supervised anomaly detection on attributed networks. *arXiv preprint arXiv:2104.07917*, 2021.

Hwang, D., Park, J., Kwon, S., Kim, K., Ha, J.-W., and Kim, H. J. Self-supervised auxiliary learning with meta-paths for heterogeneous graphs. *Advances in Neural Information Processing Systems*, 33, 2020.

Jin, M., Zheng, Y., Li, Y.-F., Gong, C., Zhou, C., and Pan, S. Multi-scale contrastive siamese networks for self-supervised graph representation learning. *arXiv preprint arXiv:2105.05682*, 2021a.

Jin, W., Derr, T., Liu, H., Wang, Y., Wang, S., Liu, Z., and Tang, J. Self-supervised learning on graphs: Deep insights and new direction. *arXiv preprint arXiv:2006.10141*, 2020.

Jin, W., Liu, X., Zhao, X., Ma, Y., Shah, N., and Tang, J. Automated self-supervised learning for graphs, 2021b.

Jun, H., Child, R., Chen, M., Schulman, J., Ramesh, A., Radford, A., and Sutskever, I. Distribution augmentation for generative modeling. In *International Conference on Machine Learning*, pp. 5006–5019. PMLR, 2020.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

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

Lee, H., Hwang, S. J., and Shin, J. Self-supervised label augmentation via input transformations. In *International Conference on Machine Learning*, pp. 5714–5724. PMLR, 2020.

Li, M. M., Huang, K., and Zitnik, M. Representation learning for networks in biology and medicine: Advancements, challenges, and opportunities. *arXiv preprint arXiv:2104.04883*, 2021.

Liu, M., Gao, H., and Ji, S. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 338–348, 2020.

Liu, Y., Pan, S., Jin, M., Zhou, C., Xia, F., and Yu, P. S. Graph self-supervised learning: A survey. *arXiv preprint arXiv:2103.00111*, 2021.

Luketina, J., Berglund, M., Greff, K., and Raiko, T. Scalable gradient-based tuning of continuous regularization hyperparameters. In *International conference on machine learning*, pp. 2952–2960. PMLR, 2016.

Maclaurin, D., Duvenaud, D., and Adams, R. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pp. 2113–2122. PMLR, 2015.

Manessi, F. and Rozza, A. Graph-based neural network models with multiple self-supervised auxiliary tasks. *arXiv preprint arXiv:2011.07267*, 2020.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., and Jaiswal, S. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.

Nene, S. A., Nayar, S. K., Murase, H., et al. Columbia object image library (coil-100). 1996.

Park, C., Kim, D., Han, J., and Yu, H. Unsupervised attributed multiplex network embedding. In *AAAI*, pp. 5371–5378, 2020.

Pedregosa, F. Hyperparameter optimization with approximate gradient. In *International conference on machine learning*, pp. 737–746. PMLR, 2016.

Peng, Z., Dong, Y., Luo, M., Wu, X.-M., and Zheng, Q. Self-supervised graph representation learning via global context prediction. *arXiv:2003.01604*, 2020a.

Peng, Z., Huang, W., Luo, M., Zheng, Q., Rong, Y., Xu, T., and Huang, J. Graph representation learning via graphical mutual information maximization. In *Proceedings of The Web Conference 2020*, pp. 259–270, 2020b.

Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., and Tang, J. GCC: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

Ren, Y., Liu, B., Huang, C., Dai, P., Bo, L., and Zhang, J. Heterogeneous deep graph infomax. *arXiv preprint arXiv:1911.08538*, 2019.

Riesen, K. and Bunke, H. Iam graph database repository for graph based pattern recognition and machine learning. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pp. 287–297. Springer, 2008.

Robey, A., Hassani, H., and Pappas, G. J. Model-based robust deep learning. *arXiv preprint arXiv:2005.10247*, 2020.

Rong, Y., Bian, Y., Xu, T., Xie, W., Wei, Y., Huang, W., and Huang, J. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems*, 33, 2020.

Roy, K. K., Roy, A., Rahman, A., Amin, M. A., and Ali, A. A. Node embedding using mutual information and self-supervision based bi-level aggregation. *arXiv preprint arXiv:2104.13014*, 2021.

Rozemberczki, B., Kiss, O., and Sarkar, R. An API oriented open-source python framework for unsupervised learning on graphs. *arXiv preprint arXiv:2003.04819*, 2020.

Shaban, A., Cheng, C.-A., Hatch, N., and Boots, B. Truncated back-propagation for bilevel optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1723–1732. PMLR, 2019.

Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., and Borgwardt, K. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pp. 488–495. PMLR, 2009.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 2012.

Srinivas, N., Krause, A., Kakade, S., and Seeger, M. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the International Conference on Machine Learning, 2010*, 2010.

Sun, F.-Y., Hoffmann, J., Verma, V., and Tang, J. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*, 2019.

Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

Wang, C. and Liu, Z. Learning graph representation by aggregating subgraphs via mutual information maximization. *arXiv preprint arXiv:2103.13125*, 2021.

Wang, J., Zhang, T., Liu, S., Chen, P.-Y., Xu, J., Fardad, M., and Li, B. Towards a unified min-max framework for adversarial exploration and robustness. *arXiv preprint arXiv:1906.03563*, 2019.

Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A. L., and Le, Q. V. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Xie, Y., Xu, Z., Zhang, J., Wang, Z., and Ji, S. Self-supervised learning of graph neural networks: A unified review. *arXiv preprint arXiv:2102.10757*, 2021.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Yanardag, P. and Vishwanathan, S. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1365–1374, 2015.

You, Y. and Shen, Y. Cross-modality protein embedding for compound-protein affinity and contact prediction. *arXiv preprint arXiv:2012.00651*, 2020.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33, 2020a.

You, Y., Chen, T., Wang, Z., and Shen, Y. When does self-supervision help graph convolutional networks? In *International Conference on Machine Learning*, pp. 10871–10880. PMLR, 2020b.

You, Y., Chen, T., Wang, Z., and Shen, Y. L^2-GCN: Layer-wise and learned efficient training of graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2127–2135, 2020c.

Zhang, H., Lin, S., Liu, W., Zhou, P., Tang, J., Liang, X., and Xing, E. P. Iterative graph self-distillation. *arXiv preprint arXiv:2010.12609*, 2020.

Zhao, T., Liu, Y., Neves, L., Woodford, O., Jiang, M., and Shah, N. Data augmentation for graph neural networks. *arXiv preprint arXiv:2006.06830*, 2020.

Zhu, Q., Du, B., and Yan, P. Self-supervised training of graph convolutional networks. *arXiv preprint arXiv:2006.02380*, 2020a.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020b.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. Graph contrastive learning with adaptive augmentation. *arXiv preprint arXiv:2010.14945*, 2020c.

Graph Contrastive Learning Automated
(Appendix)

Yuning You [1] **Tianlong Chen** [2] **Yang Shen** [1] **Zhangyang Wang** [2]

A. Alternating Gradient Descent for JOAOv2

We adapt alternating gradient descent (AGD) in Algorithm 1 to optimize (10) in main text, executed as Algorithm S1, with the following modified upper-level minimization and lower-level maximization.

Algorithm S1 AGD for optimization (10) in main text

Input: initial parameter $\theta^{(0)}$, sampling distribution $\mathbb{P}^{(0)}_{(\mathsf{A}_1,\mathsf{A}_2)}$, $\mathbb{P}^{(0)}_{(\Theta_1'',\Theta_2'')} = \mathbb{P}^{(0)}_{(\mathsf{A}_1,\mathsf{A}_2)}$, optimization step N.
for $n = 1$ **to** N **do**
1. Upper-level minimization: fix $\mathbb{P}^{(n-1)}_{(\mathsf{A}_1,\mathsf{A}_2)}, \mathbb{P}^{(n-1)}_{(\Theta_1'',\Theta_2'')}$, and call equation (1) to update $\theta^{(n)}$.
2. Lower-level maximization: fix $\theta^{(n)}$, call equation (3) to update $\mathbb{P}^{(n)}_{(\mathsf{A}_1,\mathsf{A}_2)}$, and set $\mathbb{P}^{(n)}_{(\Theta_1'',\Theta_2'')} = \mathbb{P}^{(n)}_{(\mathsf{A}_1,\mathsf{A}_2)}$.

end for
Return: Optimized parameter $\theta^{(N)}$.

Upper-level minimization. The upper-level minimization w.r.t. θ given the sampling distribution $\mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)}$, setting $\mathbb{P}_{(\Theta_1'',\Theta_2'')} = \mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)}$, is represented as:

$$\theta^{(n)} = \theta^{(n-1)} - \alpha' \nabla_\theta \mathcal{L}_{\text{v2}}(\mathsf{G}, \mathsf{A}_1, \mathsf{A}_2, \theta', \Theta_1'', \Theta_2''), \quad (1)$$

where $\alpha' \in \mathcal{R}_{>0}$ is the learning rate.

Lower-level maximization. To calculate the gradient of the lower-level objective w.r.t. $\mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)}$, we make similar efforts to approximate the contrastive loss as:

$$\begin{aligned}
&\mathcal{L}_{\text{v2}}(\mathsf{G}, \mathsf{A}_1, \mathsf{A}_2, \theta', \Theta_1'', \Theta_2'') \\
\approx &\sum_{i=1}^{|\mathcal{A}|}\sum_{j=1}^{|\mathcal{A}|} p_{ij}\ell_{\text{v2}}(\mathsf{G}, A^i, A^j, \theta', \theta''^i, \theta''^j) \\
= &\sum_{i=1}^{|\mathcal{A}|}\sum_{j=1}^{|\mathcal{A}|} p_{ij}\Big\{ - \mathbb{E}_{\mathbb{P}_\mathsf{G}}\text{sim}(T^i_{\text{v2},\theta}(\mathsf{G}), T^j_{\text{v2},\theta}(\mathsf{G})) \\
&+ \mathbb{E}_{\mathbb{P}_\mathsf{G}}\log(\mathbb{E}_{\mathbb{P}_{\mathsf{G}'}}\exp(\text{sim}(T^i_{\text{v2},\theta}(\mathsf{G}), T^j_{\text{v2},\theta}(\mathsf{G}'))))\Big\}, \quad (2)
\end{aligned}$$

where $T^i_{\text{v2},\theta} = A^i \circ f_{\theta'} \circ g_{\theta''^i},\ i = 1, ..., 5$. Thus, projected gradient descent is performed as:

$$\boldsymbol{b} = \boldsymbol{p}^{(n-1)} + \alpha'' \nabla_{\boldsymbol{p}} \psi_{\text{v2}}(\boldsymbol{p}^{(n-1)}), \boldsymbol{p}^{(n)} = (\boldsymbol{b} - \mu\mathbf{1})_+, \quad (3)$$

where $\boldsymbol{p} = [p_{ij}], i, j = 1, ..., |\mathcal{A}|, \psi_{\text{v2}}(\boldsymbol{p}) = \sum_{i=1}^{|\mathcal{A}|}\sum_{j=1}^{|\mathcal{A}|} p_{ij}\ell_{\text{v2}}(\mathsf{G}, A^i, A^j, \theta', \theta''^i, \theta''^j) - \frac{\gamma}{2}\sum_{i=1}^{|\mathcal{A}|}\sum_{j=1}^{|\mathcal{A}|}(p_{ij} - \frac{1}{|\mathcal{A}|^2})^2$, $\alpha'' \in \mathcal{R}_{>0}$ is the learning rate, μ is the root of the equation $\mathbf{1}^\mathsf{T}(\boldsymbol{b} - \mu\mathbf{1}) = 1$, and $(\cdot)_+$ is the element-wise non-negative operator.

B. Dataset Statistics

Dataset statistics can be found in Table S1, S2 and S3.

Table S1: Statistics for datasets of diverse nature from the benchmark TUDataset.

Dataset	Graph Count	Avg. Node	Avg. Degree
NCI1	4,110	29.87	1.08
PROTEINS	1,113	39.06	1.86
DD	1,178	284.32	715.66
MUTAG	188	17.93	19.79
COLLAB	5,000	74.49	32.99
RDT-B	2,000	429.63	1.15
RDB-M	2,000	429.63	497.75
GITHUB	4,999	508.52	594.87
IMDB-B	1,000	19.77	96.53

[1]Texas A&M University [2]The University of Texas at Austin. Correspondence to: Yang Shen <yshen@tamu.edu>, Zhangyang Wang <atlaswang@utexas.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

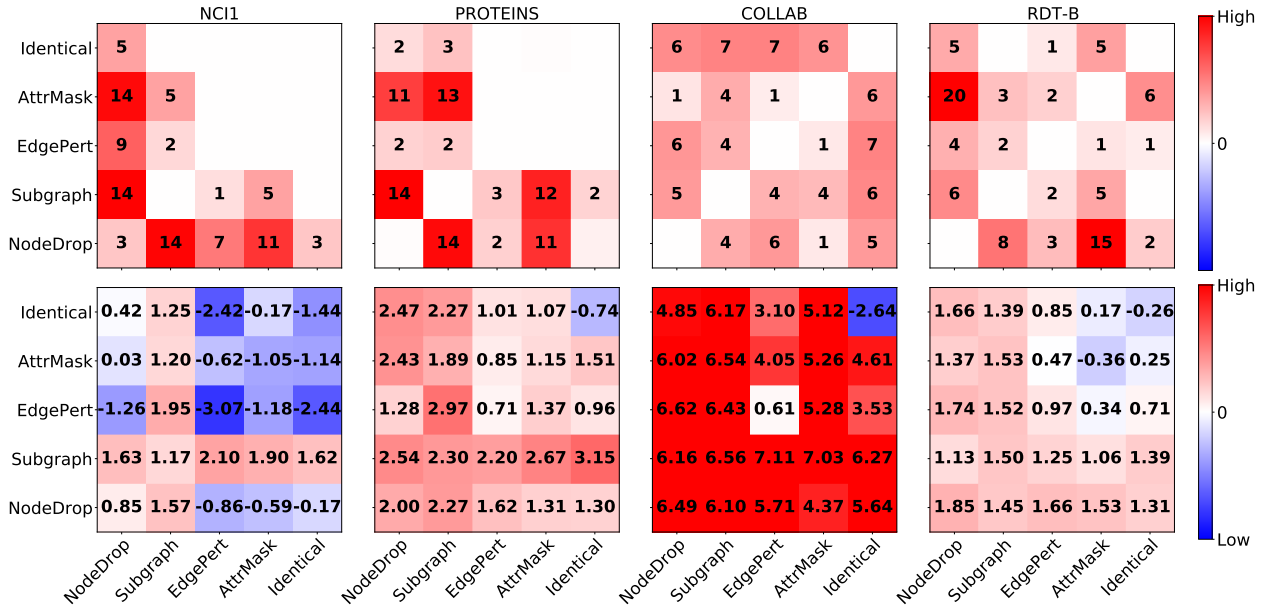

Figure S1: Top row: sampling distributions (%, defined as the percentage of this specific augmentation pair being selected during the entire training process) for augmentation pairs selected by JOAOv2 on four different datasets (NCI1, PROTEINS, COLLAB, and RDT-B). **Bottom row**: GraphCL performance gains when exhaustively trying every possible augmentation pair. Warmer (colder) colors indicate higher (lower) values, and white marks 0.

Table S2: Statistics for bioinformatics datasets.

Dataset	Graph Count	Avg. Node	Avg. Degree
BBBP	2,039	24.06	51.90
Tox21	7,831	18.57	38.58
ToxCast	8,576	18.78	38.52
SIDER	1,427	33.64	70.71
ClinTox	1,477	26.15	55.76
MUV	93,087	24.23	52.55
HIV	41,127	25.51	54.93
BACE	1,513	34.08	73.71
PPI	88,000	49.35	890.77

Table S3: Statistics for large-scale OGB datasets.

Dataset	Graph Count	Avg. Node	Avg. Degree
ogbg-ppa	158,110	243.4	2,266.1
ogbg-code	452,741	125.2	124.2

C. Augmentation Sampling Rules for GraphCL

GraphCL uniformly samples augmentations from a pre-defined pool. Augmentation pools for datasets are presented in Table S4.

Table S4: Pre-defined augmentation pools for different datasets.

Datasets	Augmentation Pools
NCI1	{NodeDrop, Subgraph}
PROTEINS	{NodeDrop, Subgraph}
DD	{NodeDrop, Subgraph}
MUTAG	{NodeDrop, Subgraph}
COLLAB	{NodeDrop, Subgraph, EdgePert, AttrMask}
RDT-B	{NodeDrop, Subgraph, EdgePert}
RDB-M	{NodeDrop, Subgraph, EdgePert}
GITHUB	{NodeDrop, Subgraph, EdgePert, AttrMask}
IMDB-B	{NodeDrop, Subgraph}
BBBP	{NodeDrop, Subgraph}
Tox21	{NodeDrop, Subgraph}
ToxCast	{NodeDrop, Subgraph}
SIDER	{NodeDrop, Subgraph}
ClinTox	{NodeDrop, Subgraph}
MUV	{NodeDrop, Subgraph}
HIV	{NodeDrop, Subgraph}
BACE	{NodeDrop, Subgraph}
PPI	{NodeDrop, Subgraph, EdgePert}
ogbg-ppa	{NodeDrop, Subgraph, EdgePert}
ogbg-code	{NodeDrop, Subgraph}

D. JOAOv2 Selected Augmentation-Pairs Alignment with Previous "Best Practices"

Schematic diagram of JOAOv2 is drawn in Figure S2. $\mathbb{P}_{(A_1,A_2)}$ optimized by JOAOv2 is plotted in the top row of Figure S1 along with GraphCL performance gains of different augmentation pairs in the bottom row, following

the same procedure as described in Sec. 3.2.1 of main text.

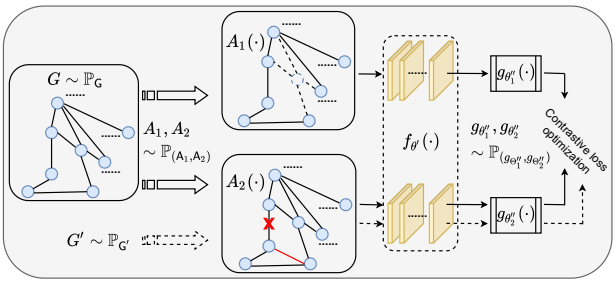

Figure S2: Schematic diagram of GraphCL with multiple augmentation-aware projection heads where $\mathbb{P}_{(g_{\Theta_1''}, g_{\Theta_2''})} = \mathbb{P}_{(\mathsf{A}_1,\mathsf{A}_2)}$.

E. Comparison between JOAO w/ and w/o the Prior

See Table S5 for comparison between JOAO w/ and w/ the prior in the semi-supervised learning setting.

Table S5: Semi-supervised performance (%) of JOAO w/ and w/o prior.

γ	w/o prior	w/ prior
NCI1	61.51±0.32	61.97±0.72
PROTEINS	71.78±0.70	72.13±0.92