Bringing Your Own View: Graph Contrastive Learning without Prefabricated Data Augmentations

Yuning You  
Texas A&M University  
yuning.you@tamu.edu

Zhangyang Wang  
University of Texas at Austin  
atlaswang@utexas.edu

Tianlong Chen  
University of Texas at Austin  
tianlong.chen@utexas.edu

Yang Shen  
Texas A&M University  
yshen@tamu.edu

ABSTRACT
Self-supervision is recently surging at its new frontier of graph learning. It facilitates graph representations beneficial to downstream tasks; but its success could hinge on domain knowledge for handcraft or the often expensive trials and errors. Even its state-of-the-art representative, graph contrastive learning (GraphCL), is not completely free of those needs as GraphCL uses a prefabricated prior reflected by the ad-hoc manual selection of graph data augmentations. Our work aims at advancing GraphCL by answering the following questions: How to represent the space of graph augmented views? What principle can be relied upon to learn a prior in that space? And what framework can be constructed to learn the prior in tandem with contrastive learning? Accordingly, we have extended the prefabricated discrete prior in the augmentation set, to a learnable continuous prior in the parameter space of graph generators, assuming that graph priors per se, similar to the concept of image manifolds, can be learned by data generation. Furthermore, to form contrastive views without collapsing to trivial solutions due to the prior learnability, we have leveraged both principles of information minimization (InfoMin) and information bottleneck (InfoBN) to regularize the learned priors. Eventually, contrastive learning, InfoMin, and InfoBN are incorporated organically into one framework of bi-level optimization. Our principled and automated approach has proven to be competitive against the state-of-the-art graph self-supervision methods, including GraphCL, on benchmarks of small graphs; and shown even better generalizability on large-scale graphs, without resorting to human expertise or downstream validation. Our code is publicly released at https://github.com/Shen-Lab/GraphCL_Automated.

CCS CONCEPTS
• Computing methodologies → Unsupervised learning. Learning latent representations.  
Neural networks.

1 INTRODUCTION
Self-supervised learning on non-Euclidean structured data has recently intrigued vast interest, with the capability of learning generalizable, transferable and robust representations from unlabeled graph data [1–5]. Unlike images, speeches or natural languages, graph-structured data are not monomorphic but abstractions of diverse nature (e.g. social networks, polymers or power grids [6–9]). This unique heterogeneity challenge however has not been fully addressed in the previous self-supervised works. The success of existing approaches relies on carefully designed predictive pretext tasks with domain expertise (e.g. context prediction [3], meta-path extraction [10], graph completion [11], etc [12–14]), possessing the premise that the designated task is the generally effective prior across all datasets, while it is not always guaranteed especially in the mentioned diversity context. The recently emerged contrastive methods seem to be free from setting the pretext, which whilst exists in a disguised form: appropriate hand-crafted contrastive views are required to be constructed (e.g. global-local representations [15], diffusion matrices [16], r-ego networks [17], etc [18–20]) otherwise resulting in performance degrade [4, 15]. The state-of-the-art (SOTA) representative, graph contrastive learning (GraphCL) [4] even copes with this challenge with additional human labor: it contrasts on the augmented graphs via manually selecting and applying the prefabricated augmentation operations per dataset [4, 21–23], by either rule of thumb or trial-and-errors. Thus, it is more flexible in accordance with diverse graph datasets, although more expensive since the rules are derived by tedious tuning with downstream labels and built on top of a pool of prefabricated priors, i.e. off-the-shelf augmentations.

Our perspective to help close the gap is to turn the prefabricated self-supervised prior into a learnable one. Intuitively, the learnable prior, following the data-driven ethos, is more versatile compared with sticking to the unaltered one and less resource-demanding...
2 RELATED WORK

Graph contrastive learning. Contrastive learning on graph data is shown to be a promising technique in graph representation learning via exploiting abundant unlabelled data [4, 15, 18]. The state-of-the-art graph contrastive learning framework (GraphCL) [4] emphasizes the perturbation invariance in graph neural networks (GNNs) through maximizing agreement between two augmented graph views, with an overview in Figure 1.

Figure 1: Pipeline of Graph Contrastive Learning with augmentations (GraphCL).

Mathematically, we denote the graph-structured input \( G \in \mathcal{G} \) sampled from certain empirical distribution \( \P_G \), that \( \mathcal{G} = (V, E) \) is an undirected graph with \( V \), \( E \) denoting the set of nodes and edges, and \( X_u \in \mathbb{R}^{D_V} \) for \( u \in V \), \( X_e \in \mathbb{R}^{D_E} \) for \( e \in E \) denoting node and edge features, respectively. GraphCL samples two augmentation operators \( A_1, A_2 \sim \mathbb{P}_{A_1, A_2} \) from a given augmentation pool \( \mathcal{A} = \{ \text{NodeDrop}, \text{Subgraph}, \text{EdgePerm}, \text{AttrMask}, \text{Identical} \} \) [4, 21–23] that \( A \in \mathcal{A} : G \rightarrow \tilde{G} \) is the stochastic augmentation function [36], and optimizes the following contrastive loss:

\[
\min_\theta \mathbb{E}_{G_{-},L_{CL}}(G, A_1, A_2, \theta)
\]

where \( \mathbb{E}_{G_{-}}(\cdot) \) denotes a sample and that in the calligraphic font (e.g. \( \mathcal{G} \)) is the set of all graphs. The contrastive loss is defined as the expected value over both positive and negative pairs:

\[
\begin{align*}
\min_\theta & \mathbb{E}_{G_{-},L_{CL}}(G, A_1, A_2, \theta) \\
&= \min_\theta \mathbb{E}_{G_{-}} \left\{ -\mathbb{E}_{P-A_1} \log \left( \mathbb{P}(A_1 G, A_2 G) \left( 1 - \mathbb{E}^{P_{A_1}} \left( \mathbb{P}_{A_2} \left( \sim \mathbb{P}(A_2 G | A_1 G) \right) \right) \right) \right) \\
&+ \mathbb{E}_{A_1} \log \left( \mathbb{P}_{A_1} \left( \mathbb{P}_{A_2} \left( \sim \mathbb{P}(A_2 G | A_1 G) \right) \right) \right) \right\},
\end{align*}
\]

where \( T_{\theta, i} = A_1 \circ f_{\theta} \circ h_{\theta'} (i = 1, 2) \) is parameterized by \( \theta = (\theta', \theta''') \), \( f_{\theta} : \mathcal{G} \rightarrow \mathbb{R}^{D_{F}} \) is the projection head, \( h_{\theta'} : \mathbb{R}^{D_{F}} \rightarrow \mathbb{R}^{D_{T}} \) is the cosine similarity function, \( \mathbb{P}_{G_{-}} = \mathbb{P}_{G_{}} \) is the negative sampling distribution, and \( \mathbb{P}_{A_1} \) and \( \mathbb{P}_{A_2} \) are the marginal distributions. Here \( \mathbb{P}(A_1 A_2) \) is heuristically pre-defined per dataset [4]. After contrastive pre-training, the pre-trained \( f_{\theta'} \) can be further leveraged for downstream fine-tuning.

Learnable prior. The learned priors were explored in video generation, compressed sensing (CS) and Bayesian deep learning [24, 25, 28]. SVG-LP [24] utilizes a recurrent neural network to model the temporally dependent priors, interpreted as an uncertainty predictor in generating video frames; a branch of approaches in CS [25, 28, 37] shows that deep generative models can be used as extraordinary priors for images, with significantly fewer measurements compared to Lasso for a given reconstruction error; and there are plenty of prior designs for Gaussian processes, variational autoencoders and Bayesian neural networks [38]. Nevertheless, it has not been explored for graph data yet.

We offer our first attempt to incorporate the learnable prior with graph neural network, further exploiting the power of abundant data under the instructive precedent assumption, and (ii) we learn in adaptivity and automation, that not only requires little human effort to prefabricate priors, i.e. augmentation functions, but also learns such knowledge during self-supervision in a data-driven, flexible and principled fashion. This is crucial for better generalizability in accordance with the graph polymorphism challenge, which is achieved in a data-driven, flexible and principled fashion. We hereby highlight our contributions, that (i) we make the first attempt to incorporate the learnable prior with graph neural network, further exploiting the power of abundant data under the instructive precedent assumption, and (ii) we learn in adaptivity and automation, that not only requires little human effort to prefabricate priors, i.e. augmentation functions, but also learns such knowledge during self-supervision in a data-driven, flexible and principled fashion. This is crucial for better generalizability in accordance with the graph polymorphism challenge, which is achieved in a data-driven, flexible and principled fashion.
Graph generative model. Generative models for graph data owns a longstanding history, with applications including molecular generation, anomaly detection and recommendation [26]. We here focus on recent learning-based generative models [14, 27, 39] conditioning on the input graph \( G \), that the \( \phi \)-parameterized stochastic generation function is defined as \( g_\phi : \mathcal{G} \to \mathcal{G} \) with the objective \( L_{\text{Gen}}(G, \phi) \) to optimize.

3 METHODOLOGY

3.1 Graph Generative Model as Learnable Priors

Formulating the prior in GraphCL. We start with illustrating how we define the prior in our problem before detailing the proposed techniques. We treat the prior in graph self-supervised learning as the inductive bias in belief beneficial to the downstream performance [3, 4], which we aim to enforce in our model via self-supervision. Thus, such a prior is usually incorporated in the pretext tasks of the predictive methods, or reflected in the constructed augmentations/views of the contrastive-learning methods. In GraphCL [4], it is reflected in the choice of the graph augmentation operators [4, 21–23]. For instance graph augmentation using node dropping encodes the prior that missing a random vertex does not alter downstream semantics.

Following this line of thoughts, we formally define the prior in GraphCL, as the stochastic mapping \( \phi \) between graph manifolds that \( \phi : \mathcal{G} \to \mathcal{G} \). Notice that this definition is flexible that can be easily extended to other contrastive-learning methods (e.g. concatenating \( m \) with another mapping \( m' : \mathcal{G} \to \mathcal{R}^D \) for methods demanding contrastive views in the vector space [15, 18]).

For clarification, the learned prior in GraphCL is different from that in Bayesian deep learning [38], since it is defined for the irregular and discrete space of the graph manifold rather than the regular Euclidean space. On the other hand, it is similar to a branch of works of the learned prior in compressed sensing leveraging generative models [25, 28, 37], with the belief that the inductive bias encoded in the architecture/parameters provides beneficial prior knowledge.

Based on the above definition of priors in GraphCL, we can interpret its choice of augmentation functions as a kind of prior selection [40, 41]. Selecting a proper prior (for a specific dataset) was crucial for the performance of previous graph-learning methods [3, 4]. However, current prior selection was either burdened with tedious trial-and-error or restricted to a pool of prefabricated views, which limits its applicability and potential.

A framework to incorporate a graph generative model-learned prior in contrastive learning. To close the aforementioned gaps in graph contrastive learning, we first propose the novel extension from the prefabricated discrete prior to a learnable continuous prior parameterized by a neural network (specifically, a graph generative model). Furthermore, the learned prior adaptively and dynamically evolves during contrastive training, which is detailed in the reward.

The recent rise of learning-based graph generative models [14, 27, 39] offers a smooth solution to parameterize a graph prior, where such a \( \phi \)-parameterized graph generator \( g_\phi \), well-capturing graph distributions, is already a well-defined stochastic function between graph manifolds. In this study we choose the widely-used variational graph auto-encoder (VGAE) [14] as the generative model with the random-walk sampler [27]. In addition to VGAE, other generators can also conveniently be the plug-and-play component in our framework (3).

A straightforward bi-level optimization form for GraphCL with the learned prior is thus written as:

\[
\begin{align*}
\min_{\theta} & \mathbb{E}_{P_c} L_{\text{CL}}(G, \phi_1, \phi_2, \theta), \\
\text{s.t.} & \phi_1, \phi_2 \in \arg \min_{\phi_1', \phi_2'} \mathbb{E}_{P_c} \left\{ L_{\text{Gen}}(G, \phi_1') + L_{\text{Gen}}(G, \phi_2') \right\},
\end{align*}
\]

where the upper-level objective is for contrastive training and the lower-level constraint is for the generator optimization. Compared to the formulation (1) with prefabricated graph views \( A \), the upper-level objective is now re-written with graph views from \( \phi \), that is, \( L_{\text{CL}}(G, \phi_1, \phi_2, \theta) = - \text{sim}(T_{\theta, \phi_1}(G), T_{\theta, \phi_2}(G)) + \log(\mathbb{E}_{P_c} \exp(\text{sim}(T_{\theta, \phi_1}(G), T_{\theta, \phi_2}(G)))) \), where \( T_{\theta, \phi_1} = g_{\phi_1} \circ f_{\theta} \circ h_{\theta} \) (i = 1, 2) and \( \theta \). The reward signal for graph generator. The straightforward formulation (2), although sensible, has no message passing from the upper- to lower-level optimization during training. In other words, the generative model is trained in (2) regardless of GraphCL, which makes the prior learning non-adaptive to contrastive learning and could lead to trivial solutions of priors and mode collapse of contrastive learning (e.g. two graph generators output the same distribution, resulting in an easy contrastive optimization [29, 30]). In order to propagate feedback to the lower-level generator optimization, we additionally give it a “reward” signal (as depicted in Figure 2) and reach a new formulation of bi-level optimization as

\[
\begin{align*}
\min_{\theta} & \mathbb{E}_{P_c} L_{\text{CL}}(G, \phi_1, \phi_2, \theta), \\
\text{s.t.} & \phi_1, \phi_2 \in \arg \min_{\phi_1', \phi_2'} \mathbb{E}_{P_c} \left\{ L_{\text{Gen}}(G, \phi_1') + L_{\text{Gen}}(G, \phi_2') \right\},
\end{align*}
\]

where the reward is in the simple form of \( r(G, \phi_1, \phi_2, \theta) = \begin{cases} 1, \quad \text{given some condition} \\ 1 - \frac{\text{distance}}{\delta}, \quad \text{otherwise} \end{cases} \), with the condition determined by certain principles and the reward weakens to \( 0 \) if the condition is not satisfied. Optimization (3) can be numerically solved by alternating gradient descent [42, 43]. The sensitivity analysis on \( \delta \) is conducted in Section 4.2.3.

We will discuss the specific instantiation of the principles next in Section 3.2. Without bells and whistles, preliminary results in Table 1 demonstrate the necessity of the principled reward for better performance.

Table 1: Preliminary experiments on learned prior (LP) w/ and w/o ogbg datasets. The principle is selected from [InfoMin, InfoBN] based on validation. Red fonts indicate the best performances.

| Methods      | ogbg-ppa | ogbg-code |
|--------------|----------|-----------|
| GraphCL      | 57.77±1.25 | 22.45±0.17 |
| LP w/o principle | 58.13±1.80 | 23.16±0.24 |
| LP w/ proper principle | **59.10±0.88** | **23.50±0.22** |
3.2 Principles for Learning Priors to Contrast

In this section we propose to adopt the principles of InfoMin and InfoBN and to incorporate them either or both, which guides the generator optimization to avoid collapsing to trivial solutions as discussed earlier. A schematic illustration is in Figure 3.

**Information minimization.** The principle of information minimization (InfoMin) [31, 42] encourages contrastive views to share less mutual information (MI) when maximizing the agreement between their latent representations. InfoMin thus can explicitly push generators to behave differently and avoid collapse. Intuitively the MI minimization can make the encoders throw away more irrelevant information (or nuisance factors) to facilitate downstream generalization. Since the (negative) contrastive loss (1) is commonly adopted to act as a numerical MI estimator of the Donsker–Varadhan representation between views [4, 31, 44, 45], we define the InfoMin reward function based on the estimated MI as

\[ r_{\text{InfoMin}}(G, \phi_1, \phi_2, \theta) = \begin{cases} 1, & \text{if } L_{\text{CL}}(G, \phi_1, \phi_2, \theta) > \text{threshold} \\ 0, & \text{otherwise} \end{cases} \]

where the threshold is treated as a hyper-parameter with sensitivity analysis in Section 4.2.3.

**Information bottleneck.** The principle of information bottleneck (InfoBN) [32, 33, 46, 47] is originally proposed to discourage the representations from acquiring superfluous information that is irrelevant for predicting the target, aiming at better generalizability and robustness. In the GraphCL framework, we introduce InfoBN by diminishing the information overlap between each contrastive view and its latent representation, whereas the pair of contrastive views still maintain a certain level of agreement in the latent space (through the upper-level contrastive loss). Thereby, the InfoBN reward is expressed as

\[ r_{\text{InfoBN}}(G, \phi_1, \phi_2, \theta, \pi) = \begin{cases} 1, & \text{if } L_{\text{InfoBN}}(G, \phi_1, \phi_2, \theta, \pi) > \text{threshold} \\ 0, & \text{otherwise} \end{cases} \]

where an additional MI estimator between views and embeddings [45, 47] is formulated as:

\[
\begin{align*}
L_{\text{InfoBN}}(G, \phi_1, \phi_2, \theta, \pi) &= -\log(\mathbb{E}_{G'} \exp(\text{sim}(T_{\pi, \phi_1}(G), T_{\theta, \phi_2}(G')))) \\
&\quad + \log(\mathbb{E}_{G} \exp(\text{sim}(T_{\pi, \phi_1}(G), T_{\theta, \phi_2}(G)))) ,
\end{align*}
\]

which needed to be minimized w.r.t. \( \pi \) to achieve a precise estimation. In this way, GraphCL with the InfoBN-rewarded learned prior is written as:

\[
\begin{align*}
\min_{G} & \quad E_{G, \phi_1, \phi_2} L_{\text{CL}}(G, \phi_1, \phi_2, \theta) \\
\text{s.t.} & \quad \phi_1, \phi_2 \in \arg \min_{\phi_1', \phi_2'} E_{G, \phi_1', \phi_2'} r_{\text{InfoBN}}(G, \phi_1', \phi_2', \theta, \pi) \{ L_{\text{Gen}}(G, \phi_1') + L_{\text{InfoBN}}(G, \phi_1, \phi_2, \theta, \pi') \} \\
&\quad + L_{\text{Gen}}(G, \phi_2'), \quad \pi \in \arg \min_{G} E_{G, \phi_1, \phi_2} L_{\text{InfoBN}}(G, \phi_1, \phi_2, \theta, \pi') .
\end{align*}
\]

**Incorporating InfoMin with InfoBN.** We lastly incorporate InfoMin with InfoBN as Info(Min&BN) to explore whether the pro-per collaboration of both principles can outperform each individual. We construct the collaborated reward dependent on the weighted summation of the estimated MI between two contrastive views and that between views and their embeddings, formulated as

\[
\begin{align*}
r_{\text{Info(Min&BN)}}(G, \phi_1, \phi_2, \theta, \pi) &= \begin{cases} 1, & \text{if } r_{\text{CL}}(G, \phi_1, \phi_2, \theta) + (1-\gamma) L_{\text{InfoBN}}(G, \phi_1, \phi_2, \theta, \pi) > \text{threshold} \\ 0, & \text{otherwise} \end{cases} \]
\]

with \( \gamma \in [0, 1] \) as the major hyper-parameter tuned in Section 4.

4 EXPERIMENTS

We evaluate our proposed method, GraphCL with learned prior (LP) against state-of-the-art (SOTA) competitors under the settings
Table 2: Semi-supervised learning on small-scale benchmarks from TUDataset (the first four) and large-scale ones from OGB (the last two). Shown in red are the best three accuracies (%) for TUDataset and the best for ogbg-ppa and F1-score (%) for ogbg-code. The SOTA results compared here are as published under the same experimental setting (- indicates that results were not available in corresponding publications).

| Methods   | COLLAB | RDT-B | RDT-MSK | GITHUB | ogbg-ppa | ogbg-code |
|-----------|--------|-------|---------|--------|----------|-----------|
| No pre-train | 73.71±0.27 | 86.63±0.27 | 51.33±0.44 | 60.87±0.17 | 56.01±1.05 | 17.85±0.60 |
| Augmentations | 74.19±0.13 | 87.74±0.39 | 52.01±0.20 | 60.91±0.32 | -         | -         |
| GAE       | 75.09±0.19 | 87.69±0.40 | 53.58±0.13 | 63.89±0.52 | -         | -         |
| Infomax   | 73.76±0.29 | 88.66±0.95 | 53.61±0.31 | 65.21±0.88 | -         | -         |
| ContextPred | 73.69±0.37 | 84.76±0.52 | 51.23±0.84 | 62.35±0.73 | -         | -         |
| GraphCL   | 74.23±0.21 | 89.11±0.19 | 52.55±0.45 | 65.81±0.79 | 57.77±1.25 | 22.45±0.17 |
| LP-InfoMin | 74.66±0.14 | 88.03±0.46 | 53.00±0.26 | 62.71±0.54 | 59.10±0.88 | 23.50±0.22 |
| LP-InfoBN  | 74.61±0.28 | 87.64±0.33 | 53.05±0.14 | 62.64±0.37 | 55.48±0.97 | 23.31±0.22 |
| LP-Info(Min&BN) | 74.84±0.31 | 87.81±0.45 | 53.32±0.23 | 63.11±0.33 | 57.31±0.99 | 23.61±0.27 |

4.1 Comparison with the State of the Art

4.1.1 Semi-Supervised Learning. Setup. Four small social-network benchmarks are collected from TUDataset [34] and two large-scale graph datasets, protein-protein interaction networks and code abstract syntax trees, are gathered from Open Graph Benchmark (OGB) [35]. (i) For TUDataset benchmarks without explicit training/validation/test split, we perform self-supervised pre-train-ing with all data and then supervised fine-tuning/evaluation with 10 folds, each of which contains 10% of data. This whole procedure is repeated for 5 times to report the average performance and the error bar. Residual graph convolutional network (ResGCN) [49] is used as the backbone architecture with 5 layers and 128 hidden dimensions following GraphCL. (ii) For OGB datasets with training/validation/test split, we perform self-supervised pre-training with all training data and supervised fine-tuning with 10% of them then evaluate on the validation/test sets, which is repeated for 10 times. Graph isomorphism network (GIN) [50] is used with 5 layers and 300 hidden dimensions following [35].

Results. Table 2 leads to the following observations.

(i) On small benchmarks, learned prior performs on par with GraphCL and other SOTA methods. With the parameterized prior that is adaptively, dynamically and in principle learned from data, GraphCL with learned priors achieves better performance on COLLAB (≥0.38%) and RDT-MSK (≥0.45%) compared to GraphCL with hand-crafted augmentations. It has slightly worse results than GraphCL on RDT-B and GITHUB but the best-performing version is still comparable with other SOTAs (ranked the 3rd and 4th out of 9, respectively). Similar observations are made in the classical small benchmarks (Appendix A). Note that GraphCL selects augmentations by exhaustively and manually tuning on TUDataset [4], whereas our models are completely free from such labor. Therefore the competitive performance of our methods demonstrates the effectiveness of the learnable prior. Interestingly, GA-E performs well for these 4 benchmarks, showing its strong capability of learning priors for small-scale and regularly-structured social networks.
(ii) On large-scale datasets, learned prior with appropriate principle(s) generalizes better versus pre-defined augmentations. Under the guidance of the proper principle, the learned prior leads to better generalizability on ogbg-ppa (+1.33%) and ogbg-code (+1.16%) against manually designed and tuned augmentations in GraphCL. This indicates that the data-driven learnable prior defined in a continuous space, rather than the manual selection from a discrete pool of pre-defined augmentations, can effectively evolve into the generalizable and scalable prior that benefits downstream performance, even for large-scale graphs.

(iii) Rule of choosing principle is dataset-dependent, and reconstruction quality is a potential indicator. Across different datasets, we observe that the benefit of various principles (InfomIn, InfobN, or both) to guide prior learning is dataset-dependent, which is intuitively reasonable given the enormously diverse nature of graph datasets. Toward a rule of choosing the proper principle again without relying on downstream validation, we find that the reconstruction quality of the graph generator can potentially be an indicator, which will be discussed in Section 4.2.1.

4.1.2 Transfer Learning. Setup. One pre-training dataset for molecules and eight fine-tuning/evaluation datasets are taken from ZINC15 [51] and MoleculeNet [3, 52], respectively. And one pre-training dataset for protein-protein interaction networks and one fine-tuning/evaluation dataset are curated in [3]. We pre-train on the larger dataset then fine-tune/evaluate on smaller datasets of the same category using the given training/validation/test split. GIN is used with 5 layers and 300 hidden dimensions following [3].

Results. Table 3 and 4, reveal the following observations.

(iv) On datasets with rich/rigid knowledge, learned prior still beats contrastive learning with pre-fabricated augmentations but can underperform self-supervision with expert-designed pretext tasks. On the molecular datasets, the learned prior with proper principle(s) achieves significant improvement compared to GraphCL, boosting 7 of 8 downstream performances (Table 3). This again echoes our previous observation on relational graphs that the learnable prior results in better generalizability and scalability. Meanwhile, it can underperform the SOTA self-supervised methods with pretext tasks carefully designed for the molecule domain. Nevertheless, the best-performing learnable prior still manages to outperform other SOTA methods for 3 out of 8 datasets.

Unlike relational networks (e.g. citation networks earlier and PPI next) [6], molecules are required to follow rich and “rigid” chemistry rules to be semantically valid and their casual “augmentations” (e.g. modifying nodes/atoms or edges/bonds) can lead to invalidity, which is much more challenging for the generator to explore directly from data. Indeed, examples in [3, 4] show that improper domain-naive pretext tasks or augmentations lead to performance degradation for molecules; whereas some pretext tasks carefully designed to meet domain knowledge and human expertise boost downstream performance. Nevertheless, without assumption of the knowledge and the downstream labels, our learned prior has demonstrated robust competitiveness across datasets.

On the other hand, for the PPI dataset (and those in Section 4.1.1) where graphs are less embedded with the richness and/or rigidity of domain knowledge, we show the clear advantage of the learned prior that achieves the best performance (≥+3.28%) compared with all other SOTA competitors in Table 4. This again demonstrates the effectiveness of learned prior.

4.2 Further Analyses

With performance comparison with SOTA methods completed, we delve into the proposed models for even more insights.

4.2.1 Graph generation quality usually aligns with downstream performance. The first question we would like to answer is: what is the relationship between the quality of the learned graph generative model (specifically VGAE here) and the downstream performance? In the past research, it is widely adopted to evaluate VGAE using its reconstruction quality, which can be quantitatively measured by the performance of link prediction, a surrogate task [14, 27, 53]. We employ this commonly-used surrogate evaluation and find out that in most cases better reconstruction quality aligns with better downstream performance, as shown below.

Table 5 shows that under the guidance of the InfomIn and InfobN principles, the generator makes more precise link prediction (in both AUROC and AUPRC), leading to better downstream performance for 6 out of the 8 datasets. The same trend is observed among different checkpoints during training, as seen in Figure 4.

This observation further sheds light on the following insights. (i) The better reconstruction quality of the generator stands for more sufficient principled training to capture the prior knowledge, leading to better performance. Thus, the quality of the learned prior is inherently related to the generalizability of graph contrastive learning. This acts as the causal factor of the performance in most cases. (ii) Furthermore, the alignment offers a potential criterion for selecting the principle and early stopping while learning the prior (as stated in Section 4.1.1), without the need of human intervention or downstream validation.

4.2.2 Molecule-specific generator alone does not significantly benefit molecular datasets. Next we explore whether a domain-compatible molecule-specific generator, rather than the domain-agnostic graph generative model VGAE, can further boost the performance in Section 4.1.2 and potentially beat expert-designed pretext tasks in all 8 datasets. As mentioned earlier, it might be too challenging for VGAE without domain-adaptation to learn the rigid/rich domain knowledge of molecules purely from data. Therefore, we replace VGAE with GraphAF [54] which is not only molecule-specific but also sample-efficient compared with other molecule generators [55–57]. Efficiency is crucial in our case since sampling is performed in each feed-forward propagation of the framework (3). We show the results in Table 6 with the InfomIn principle incorporated.

The results above do not show a significant performance difference between the two graph generators. We believe that additional work is needed to bring out the potential benefit of domain-specific knowledge-infused graph generators.

4.2.3 Tuning principled-reward hyper-parameters could strengthen the competitive performance even more. Furthermore, we would like to examine the influence of two unique hyper-parameters in the reward function of the framework (3), δ values and the threshold determination, which we did not have the luxury to tune in earlier
Within the standard deviation of the best. The compared results are from the published papers.

Table 3: Transfer learning on molecular datasets from [3]. Red numbers indicate the best performances (AUROC, %) and those within the standard deviation of the best. We compare with the published results.

| Methods          | BBBP   | Tox21  | ToxCast | SIDER  | ClinTox | MUV    | HIV    | BACE   |
|------------------|--------|--------|---------|--------|---------|--------|--------|--------|
| 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 |
| Infomax          | 68.8±0.8 | 73.5±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 |
| 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 | 79.2±0.9 |
| 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 |
| 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 |
| GraphCL          | 69.6±0.67 | 73.8±0.66 | 62.4±0.57 | 60.5±0.88 | 75.9±2.65 | 69.8±0.266 | 78.4±1.22 | 75.38±1.44 |
| LP-InfoMin       | 71.47±0.66 | 74.60±0.70 | 63.13±0.30 | 60.52±0.75 | 72.39±1.50 | 70.51±2.25 | 76.43±0.85 | 78.86±1.66 |
| LP-InfoBN        | 71.68±0.99 | 74.45±0.51 | 62.39±0.60 | 60.80±0.62 | 76.73±1.53 | 72.03±1.17 | 77.03±0.75 | 81.15±1.33 |
| LP-Info(Min&BN)  | 71.40±0.55 | 74.54±0.45 | 63.04±0.30 | 59.70±0.43 | 74.81±2.73 | 72.99±2.28 | 76.96±1.10 | 80.21±1.36 |

Figure 4: Link prediction performance (AUROC and AUPRC, %) of the generator checkpoints on two representative large-scale datasets ogbg-ppa and ogbg-code.

Table 4: Transfer learning on PPI dataset from [3]. Red numbers indicate the best performances (AUROC, %) and those within the standard deviation of the best. The compared results are from the published papers.

| Methods          | PPI    |
|------------------|--------|
| No pre-train.    | 64.8±1.0 |
| Infomax          | 64.1±1.5 |
| EdgePred         | 65.7±1.3 |
| AttrMasking      | 65.2±1.6 |
| ContextPred      | 64.4±1.3 |
| GraphCL          | 67.8±0.85 |
| LP-InfoMin       | 68.24±0.87 |
| LP-InfoBN        | 71.16±0.28 |
| LP-Info(Min&BN)  | 70.10±0.76 |

Results. Previously we fix $\delta = 0.01$ and set the threshold at the mean value of the (corresponding estimated mutual information in the batch) due to the limited computational resources and the large amount of experiments. Here on four small-scale benchmarks, COLLAB, RDT-B, RDT-M and GITHUB from the TUDataset, with the InfoMin principle, we tune $(\delta, \gamma)$ threshold $\in \{0.1, 0.01, 0.001\} \times \{\text{mean-sd, mean, mean+sd}\}$ (sd: standard deviation) while fixing the optimal $\gamma_*$ with results shown in Figure 5.

Results echo the similar observation as in Section 4.1.1, that, the choice of the hyper-parameter $(\delta, \gamma)$ is dataset-relevant owing to the distinctively heterogeneous nature of graph data. We notice that earlier chosen values $(\delta, \gamma) = (0.01, \text{mean})$ located in the middle patch of heatmaps in Figure 5, are not optimal (even not within top-3 performance on 3 out of 4 datasets), whereas our models performances further improve their competitiveness against SOTA methods.

4.2.4 Generative graphs connections are sparse to capture patterns. We lastly visualize the generated connections probability under the guidance of Info(Min&BN) in Figure 6. Compared with input graphs, we observe the sparsity in the generated connections, that tries to preserve links among cliques and some key connection across cliques. This sparsification is useful to capture patterns of original graphs and generate contrastive views. We leave the quantitative assessment of the “usefulness” in future work.

5 CONCLUSIONS

In this paper, we target more adaptive, automatic and generalizable graph self-supervised learning, by introducing a learnable prior and a framework to learn it. Leveraging the SOTA GraphCL framework as the base model, we extend the prefabricated discrete prior into a learnable continuous one parameterized by a graph generative model. In addition, principles such as InfoMin and InfoBN are incorporated to avoid collapsing into trivial solutions. The resulting framework is formulated as bi-level optimization. Empirically, this first attempt to incorporate the adaptive and dynamic learned prior with GNN, performs on par with the SOTA competitors on small benchmarks, and generalizes better on large-scale datasets, without resorting to human expertise of domain knowledge or tedious trial-and-error relying on downstream validation.
Table 5: Link prediction performance (AUROC and AUPRC, %) of VGAE generators on eight pre-training datasets. Better link prediction results are marked in red if accompanied with better downstream performances, as shown in Table 2, 3 and 4.

| Principles | COLLAB | RDT-B | RDT-M5K | GITHUB | ogbg-ppa | ogbg-code | Trans-Mol | Trans-PPI |
|------------|--------|-------|---------|--------|----------|-----------|-----------|-----------|
| AUROC (%)  | InfoMin| 71.28 | 97.32   | 99.08  | 78.68    | 96.53     | 92.39     | 64.54     | 71.20     |
|            | InfoBN | 69.44 | 97.29   | 99.31  | 81.20    | 95.24     | 94.06     | 83.55     | 71.32     |
| AUPRC (%)  | InfoMin| 80.84 | 96.62   | 98.67  | 78.49    | 95.94     | 90.08     | 64.14     | 69.34     |
|            | InfoBN | 79.13 | 96.59   | 98.97  | 80.47    | 95.30     | 91.66     | 82.51     | 70.65     |

Table 6: Learned prior performance with different generators under the guidance of InfoMin, in the transfer learning setting on molecular datasets. Red numbers indicate the best performances (AUROC, %).

| Methods | BBBP | Tox21 | ToxCast | SIDER | ClinTox | MUV | HIV | BACE |
|---------|------|-------|---------|-------|---------|-----|-----|------|
| VGAE    | 71.47±0.66 | 74.60±0.70 | 63.13±0.30 | 60.52±0.75 | 72.39±1.50 | 70.51±2.25 | 76.43±0.85 | 78.86±1.66 |
| GraphAF | 70.55±0.63 | 73.51±0.43 | 62.03±0.33 | 61.32±1.32 | 77.47±1.91 | 72.25±1.18 | 76.30±1.34 | 78.43±2.36 |

Figure 5: Impact of hyper-parameters of the reward function for InfoMin, δ and threshold, on COLLAB, RDT-B, RDT-M and GITHUB datasets. Warmer colors represent the relatively better performances.

Figure 6: Visualizing the connection probability of generative models guided by InfoMin&BN for 4 samples from COLLAB.

Our proposed learnable prior further exploits the power of deep learning and big data in the graph domain, and therefore is of broad interests and wide applications as in recommendation systems, drug discovery and combinatorial optimization.

ACKNOWLEDGMENT
The study is in part funded by NIH (R35GM124952 to YS).

REFERENCES
[1] Yaochen Xie, Zhao Xu, Zhengyang Wang, and Shuiwang Ji. Self-supervised learning of graph neural networks: A unified review. arXiv preprint arXiv:2102.10757, 2021.
[2] Yixin Liu, Shuiru Pan, Ming Jin, Chuan Zhou, Feng Xia, and Philip S Yu. Graph self-supervised learning: A survey. arXiv preprint arXiv:2103.00111, 2021.
[3] Weihua Hu, Bowen Liu, Joseph Gomes, Marina Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. arXiv preprint arXiv:1905.12265, 2019.
[4] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. Advances in Neural Information Processing Systems, 33, 2020.

[5] Xiao Liu, Fanjun Zhang, Zhenyu Hou, Zhaoyu Wang, Li Mian, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. arXiv preprint arXiv:2006.08228, 1(2), 2020.

[6] Monassar N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.

[7] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. Layer-dependent importance sampling for training deep and large graph convolutional networks. arXiv preprint arXiv:1911.07325, 2019.

[8] Yuning You and Yang Shen. Cross-modality protein embedding for compound-protein affinity and contact prediction. arXiv preprint arXiv:2012.00651, 2020.

[9] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. L2-Gen: Layer-wise and learned efficient training of graph convolutional networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2127–2135, 2020.

[10] Daosil Hwang, Junyoung Park, Sunyoung Kwon, Kyung-Min Kim, Jung-Woo Ha, and Hyeonwoo J Kim. Self-supervised auxiliary learning with meta-paths for heterogeneous graphs. arXiv preprint arXiv:2007.08294, 2020.

[11] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. When does self-supervision help graph convolutional networks? In International Conference on Machine Learning, pages 10871–10880. PMLR, 2020.

[12] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 701–710, 2014.

[13] Jian Tang, Meng Qu, Minghe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In Proceedings of the 24th international conference on world wide web, pages 1076–1077, 2015.

[14] Thomas N Kipf and Max Welling. Variational graph auto-encoders. arXiv preprint arXiv:1611.07802, 2016.

[15] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. arXiv preprint arXiv:1908.01000, 2019.

[16] Kaveh Hassani and Amir Hossein Khazaehmadi. Contrastive multi-view representation learning on graphs. arXiv preprint arXiv:2006.05582, 2020.

[17] Jiezhih Qu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 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.

[18] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. arXiv preprint arXiv:1809.10541, 2018.

[19] Zhen Peng, Yuxiang Dong, Minnan Luo, Xiao-Ming Wu, and Qinghua Zheng. Self-supervised graph representation learning via global context prediction. arXiv preprint arXiv:2003.01604, 2020.

[20] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. arXiv preprint arXiv:2010.14945, 2020.

[21] Tong Zhao, Yozhen Liu, Leonardo Naves, Oliver Woodford, Meng Jiang, and Neil Shah. Data augmentation for graph neural networks. arXiv preprint arXiv:2006.00830, 2020.

[22] Kezhi Kong, Guohao Li, Mucong Ding, Zuxuan Wu, Chen Zhu, Bernard Ghanem, Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020).

[23] Weihuia Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasa, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. arXiv preprint arXiv:2005.06087, 2020.

[24] Eric W Weisstein. Stochastic function. MathWorld–A Wolfram Web Resource. https://mathworld.wolfram.com/StochasticFunction.html.

[25] Aditya Grover and Stefano Ermon. Amortized variational compressive sensing, 2018.

[26] Vincent Fortuin. Prions in bayesian deep learning: A review, 2021.

[27] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. In International Conference on Machine Learning, pages 610–619. PMLR, 2018.

[28] Domenico Giannone, Michele Lenza, and Giorgio E Primiceri. Prior selection for vector autoregressions. Review of Economics and Statistics, 97(2):436–451, 2015.

[29] Robert E Kass and Larry Wasserman. The selection of prior distributions by formal rules. Journal of the American statistical Association, 91(435):1343–1370, 1996.

[30] Jingkang Wang, Tianyuan Zhang, Siia Ju, Pin-Yu Chen, Jiaxin Xu, Makan Fardad, and Bo Li. Towards a unified min-max framework for adversarial exploration and robustness. arXiv preprint arXiv:1906.03563, 2019.

[31] Stephen Boyd, Stephen P. Boyd, and Lawn Vandenberghe. Convex optimization. Cambridge university press, 2004.

[32] Mohamed Ismaiel Belghazi, Aris Thek Korat, Sathish Kumar, Sharjeel Otair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In International Conference on Machine Learning, pages 531–540. PMLR, 2018.

[33] Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. Graph information bottleneck. arXiv preprint arXiv:2010.12811, 2020.

[34] Junchi Yu, Tingyang Xu, Yu Rong, Yatian Bin, Junzhou Huang, and Ran He. Graph information bottleneck for subgraph recognition. arXiv preprint arXiv:2010.05653, 2020.

[35] Shantanu Thakoor, Corentin Tallec, Mohammad Gheislaghi Aazar, Rémi Monus, Petar Velickovic, and Michal Valko. Boosted representation learning on graphs. arXiv preprint arXiv:2105.06514, 2021.

[36] Yinglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Philip Isola. What makes for good views for contrastive learning. arXiv preprint arXiv:2005.10343, 2020.

[37] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck. Neural computation, 97(2):436–451, 2015.

[38] Teague Sterling and John J Irwin. Zinc 15-ligand discovery for everyone. Journal of chemical information and modeling, 55(11):2324–2337, 2015.

[39] Zheqian Zhu, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In International Conference on Machine Learning, pages 2323–2332. PMLR, 2018.

[40] Meng Liu, Keqiang Yan, Bora Oztekin, and Shuiwang Ji. Graphfem: Molecular graph generation with energy-based models. arXiv preprint arXiv:2102.05083, 2021.

[41] Shunai Lin, Pan Zhou, Zi-Yuan Hu, Shoujia Wang, Ruizhi Zhao, Yefeng Zheng, Liang Lin, Eric Xing, and Xiaodan Liang. Prototype graph contrastive learning. arXiv preprint arXiv:2105.04906, 2021.

[42] Annalakshmi Narayanan, Mahinath Chandramohan, Rajasekar Venkatesan, Li Hui, Chen Yang, and Shantanu Jaswal. Graph2vec: Learning distributed representations of graphs. arXiv preprint arXiv:1707.05065, 2017.
APPENDIX

A EXPERIMENTS ON CLASSICAL SMALL BENCHMARKS

We assay our methods on classical small benchmarks of MUTAG and PTC-MR datasets with the standard setting of unsupervised learning [58]. We adopt the GraphCL backbone architecture and hyper-parameters in [4] for the learned priors, and compare with state-of-the-art (SOTA) approaches of Graph2Vec [59], InfoGraph [15], MVGRL [16], GCC [17] and GraphCL [4].

Results in Table 7 reach the consistent observation as in Section 4.1.1 result (i), that on small benchmarks (MUTAG and PTC-MR are even smaller than those in Table 2), learned prior performs on par with GraphCL using tedious trial-and-error on prefabricated augmentations, as well as other SOTA methods.

Table 7: Unsupervised learning on classical small benchmarks. Reported numbers are classification accuracies (%).

| Methods       | MUTAG     | PTC-MR    |
|---------------|-----------|-----------|
| Graph2Vec     | 83.2±9.3  | 60.2±6.9  |
| InfoGraph     | 89.0±1.1  | 61.7±1.7  |
| MVGRL         | 89.7±1.1  | 62.5±1.7  |
| GCC           | 86.4±0.5  | 58.4±1.2  |
| GraphCL       | 86.8±1.3  | 58.4±1.7  |
| LP-InfoMin    | 88.69±0.84| 62.34±0.83|
| LP-InfoBN     | 88.68±0.41| 61.84±0.59|
| LP-Info(Min&BN)| 88.94±0.69| 61.67±0.49|

B ABLATION ON BASE MODEL

We validate the efficacy of learned priors on different base models. Except for GraphCL, we choose another simple and effective variant, BRGL [48] for the experiment. We examine on the semi-supervised learning setting as in Section 4.1.1 with the COLLAB dataset.

Results in Table 8 show a similar phenomenon as in Section 4, that compared with prefabricated augmentations, the learned prior leads to the better generalization especially guided by a proper principle. We demonstrate the above observation holds with different base models.

Table 8: Semi-supervised learning on COLLAB of learned priors with different base models.

| Methods       | GraphCL | BRGL    |
|---------------|---------|---------|
| None          | 74.23±0.21 | 74.41±0.19 |
| LP-InfoMin    | 74.66±0.14 | 74.69±0.30 |
| LP-InfoBN     | 74.61±0.28 | 74.44±0.12 |
| LP-Info(Min&BN)| 74.84±0.31| 74.87±0.12 |

C ABLATIONS ON HYPER-PARAMETER OF LEARNED PRIOR

We perform ablation studies on three key hyper-parameters of learned priors: γ (in the last paragraph of Section 3.2) controlling the trade-off between InfoMin and InfoBN principles; δ (in equation (3)) depicting the value of attenuated rewards; and threshold (in Section 3.2) determining when to trigger the attenuated rewards. Experiments are conducted on the semi-supervised learning setting as in Section 4.1.1 with the COLLAB dataset.

Results in Table 9 state the performance of the incorporated principle is stable insensitive to the trade-off factor γ within a certain wide range (in COLLAB around 0.3 to 0.7). Since it is dataset-dependent, we determine its value via validation per dataset. Results in Table 10 show the attenuated reward value should not be over small or large (where the positive reward is 1), and we by default set it as 0.1 which is safe. Results in Table 11 illuminate the threshold greater than or equal to the (principled) loss mean is necessary to reach a good performance, and we by default configure it as mean.

Table 9: Ablation study on γ of learned priors in semi-supervised learning on COLLAB.

| γ     | 0   | 0.1  | 0.3  | 0.5  | 0.7  | 0.9  | 1    |
|-------|-----|------|------|------|------|------|------|
| Acc(%)| 74.66| 74.81| 74.86| 74.84| 74.87| 74.59| 74.61|

Table 10: Ablation study on δ of learned priors in semi-supervised learning on COLLAB.

| δ     | 0.01 | 0.1  | 0.5  |
|-------|------|------|------|
| Acc(%)| 74.44| 74.84| 74.64|

Table 11: Ablation study on the threshold value of learned priors in semi-supervised learning on COLLAB.

| Thres. | mean-2std | mean-std | mean  | mean+std | mean+2std |
|--------|-----------|----------|-------|----------|-----------|
| Acc(%) | 74.43     | 74.42    | 74.84 | 74.87    | 74.78     |

D EFFICIENCY ANALYSIS

Since the learned prior is composed of graph contrastive learning and graph generative model, its pre-training complexity is equal to the addition of the complexities of two components. We empirically show the running time in Table 12. Although we observe that the generator is more expensive to train, its scalability on large datasets is promising as reported in [14, 27, 39].

Table 12: Efficiency comparison of pre-training 100 epochs in semi-supervised learning on COLLAB.

| Methods | GraphCL | Generator | LP |
|---------|---------|-----------|----|
| Time(min)| 23      | 51        | 68 |