Graph Contrastive Learning with Personalized Augmentation

Anonymous authors
Paper under double-blind review

Abstract

Graph contrastive learning (GCL) has emerged as an effective tool to learn representations for whole graphs in the absence of labels. The key idea is to maximize the agreement between two augmented views of each graph via data augmentation. Existing GCL models mainly focus on applying *identical augmentation strategies* for all graphs within a given scenario. However, real-world graphs are often not monomorphic but abstractions of diverse natures. Even within the same scenario (e.g., macromolecules and online communities), different graphs might need diverse augmentations to perform effective GCL. Thus, blindly augmenting all graphs without considering their individual characteristics may undermine the performance of GCL arts. However, it is non-trivial to achieve personalized allocation since the search space for all graphs is exponential to the number of graphs. To bridge the gap, we propose the first principled framework, termed as *G*raph contrastive learning with *P*ersonalized *A*ugmentation (GPA). It advances conventional GCL by allowing each graph to choose its own suitable augmentation operations. To cope with the huge search space, we design a tailored augmentation selector by converting the discrete space into continuous, which is a plug-and-play module and can be effectively trained with downstream GCL models end-to-end. Extensive experiments across 10 benchmark graphs from different types and domains demonstrate the superiority of GPA against state-of-the-art competitors. Moreover, by visualizing the learned augmentation distributions across different types of datasets, we show that GPA can effectively identify the most suitable augmentations for each graph based on its characteristics. The code is available at `https://anonymous.4open.science/r/GPA-2F2B/`.

1 Introduction

Graph contrastive learning (GCL) could learn effective representations of graphs (Hassani & Khasahmadi, 2020) in a self-supervised manner. It attracts considerable attention (Sun et al., 2020; Tan et al., 2022), given that labels are not available in many real-world networked systems. The core idea of GCL is to generate two augmented views for each graph by perturbing it, and then learn representations via maximizing the mutual information between the two views (Velickovic et al., 2019; Peng et al., 2020). Existing efforts can be roughly divided into two categories, i.e., node-level (Zhu et al., 2021; Wan et al., 2020) and graph-level (You et al., 2021). In this paper, we focus on the latter.

The performance of GCL is known to be heavily affected by the chosen augmentation types (Hassani & Khasahmadi, 2020; You et al., 2020; Jin et al., 2020), since different augmentations may impose different inductive biases about the data. Intensive recent works have been devoted to exploring effective augmentations for different graph scenarios (Veličković et al., 2019; Zhu et al., 2021; You et al., 2021; Chen et al., 2022). Typical augmentation strategies include node dropping, edge perturbation, subgraph sampling, and attribute masking. The best augmentation option is often data-driven and varies in graph scales or types (You et al., 2020; Jin et al., 2020). For example, (You et al., 2020) revealed that edge perturbation may benefit social networks but hurt biochemical molecules. Therefore, manually searching augmentation strategies for a given scenario would involve extensive trials and efforts, hindering the practical usages of GCL. Several studies have been proposed to address this issue by automating GCL.

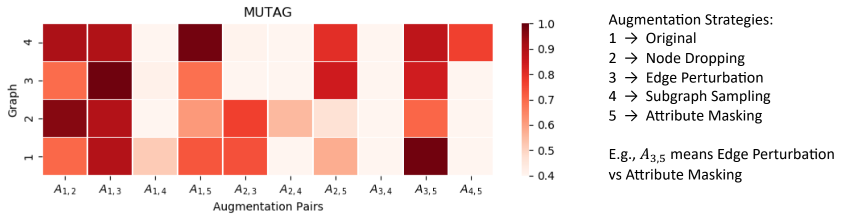


Figure 1: The effect of different augmentation strategies toward four randomly sampled graphs from MUTAG. X-axis denotes the id of augmentation pair. Y-axis is the graph id. The color represents the performance. The darker the color is, the better performance GCL achieves under the corresponding augmentation strategy.

Despite the recent advances, existing GCL might be suboptimal for augmentation configuration, since they apply a well-chosen but identical augmentation option to all graphs in a dataset. The rationale is: graphs in a scenario usually have different properties because the characteristics of real-world graphs are complex and diverse (West et al., 2001; Liu et al., 2019). For example, by slightly changing the structure of a molecular graph, its target function could be completely different (Morris et al., 2020). Similar observations have also been found in many other graph domains, such as social communities and protein-protein interaction networks (Thakoor et al., 2021). Motivated by these observations, we conduct a preliminary experiment on the MUTAG dataset to test how different augmentation types impact the GCL results. Results in Fig. 1 show that different graphs favor distinct perturbation operations to achieve their best performance. For example, graph 1 performs better when using *edge perturbation* vs *attribute masking*, while graph 2 prefers *original* vs *node dropping*. Such personalized phenomenon of graph instances, in terms of their desirable augmentation strategies, has never been explored in GCL. To bridge the gap, in this paper, we propose to develop an effective augmentation selector to identify the most informative perturbation operators for different graphs when performing GCL on a specific dataset.

However, it is a challenging task to perform personalized augmentations in GCL because of two major reasons. First, unlike the traditional GCL setting, the search space in our personalized scenario grows exponentially with the number of graphs N in a dataset. This search space is intolerable in practice since N could be thousands or even tens of thousands (Hu et al., 2020a). As a result, the conventional trial-and-error approach cannot be directly applied because each trail itself (i.e., testing an augmentation option for a dataset) is time-consuming. Second, the augmentation choices and GCL model naturally depend on and reinforce each other since the contrastive loss consists of augmented views and GCL encoder. That is, learning a better GCL requires well-chosen augmentation strategies (You et al., 2020), while selecting suitable augmentation operators needs the signals of GCL as feedback (You et al., 2021). Thus, how to perform effective personalized augmentation selector on a premise of such mutual effect is another challenge.

To tackle the aforementioned challenges, we propose a novel contrastive learning framework, dubbed GPA. Specifically, we aim to investigate two research questions. 1) What are the impacts of different augmentation strategies on a given graph (an instance in a graph dataset)? 2) Can we build a stronger automated GCL by allowing each graph instance to choose its favorable augmentation types? GPA works by iteratively updating a personalized augmentation selector and the GCL method, where the former aims to identify optimal augmentation types for each graph instance, and the latter is trained according to an instance-level contrastive loss defined on those assigned augmentation options.

Our **main contributions** are highlighted as follows. **First**, we focus on augmentation selection for graph contrastive learning, and propose an effective personalized augmentation framework (GPA). To the best of our knowledge, GPA is the first to assign personalized augmentation types to each graph based on its own characteristic. Moreover, we automate the personalized augmentation process, equipping GPA with broader applicability and practicability. **Second**, to cope with the huge search space, we develop a personalized augmentation selector to effectively infer optimal augmentation strategies by relaxing the discrete space to be continuous. To exploit the mutual reinforced effect between augmentation strategies and GCL, GPA is formulated as a bi-level optimization, which learns the augmentation selector and GCL method systematically. **Third**, we conduct extensive experiments to evaluate GPA on multiple graph benchmarks of diverse scales and types. Empirical results demonstrate the superiority of GPA against state-of-the-art GCL competitors.

2 Problem Statement

Notation. Let $\mathcal{G} = \{\mathcal{G}_n : 1 \leq n \leq N\}$ denote a graph set with N sample graphs, where $\mathcal{G}_n = (\mathcal{V}, \mathcal{E}) \in \mathcal{G}$ stands for an undirected graph with nodes $\mathcal{V}$ and edges $\mathcal{E}$. Each node $v \in \mathcal{V}$ in $\mathcal{G}_n$ is described by an F-dimensional feature vector $\mathbf{X}_v \in \mathbb{R}^F$. We use $\mathcal{A} = \{A_k : 1 \leq k \leq K\}$ to denote a set of data augmentation operators, where K is the maximum number of augmentation types of interest. Each augmentation operator $A_k : \mathcal{G}_n \to \tilde{\mathcal{G}}_n$ transforms a graph into its conceptually similar form with certain prior. In previous GCL studies, they focus on identifying two optimal augmentation types for the whole dataset $\mathcal{G}$, such as the augmentation pair $A_{i,j} = (A_i, A_j)$, where $A_i, A_j \in \mathcal{A}$. The optimal augmentation pair here is often manually picked via rules of thumb or trial-and-error. However, as shown in Fig. 1, different graphs within the dataset may favor different augmentation combinations. Therefore, we study personalized augmentation selection and formally define the research problem as below.

Definition 1 ***Personalized augmentation selection.*** *Given a set of graphs $\mathcal{G} = (\mathcal{G}_n : 1 \leq n \leq N)$, and the augmentation space $\mathcal{A} = \{A_k : 1 \leq k \leq K\}$ consisting of K different augmentation types, to perform GCL, personalized augmentation selection aims to find the optimal augmentation pair $A^n_{i,j} = (A^n_i, A^n_j)$ for each graph $\mathcal{G}_n \in \mathcal{G}$. The values of i and j are only determined by the characteristic of the n-th sample graph $\mathcal{G}_n$.*

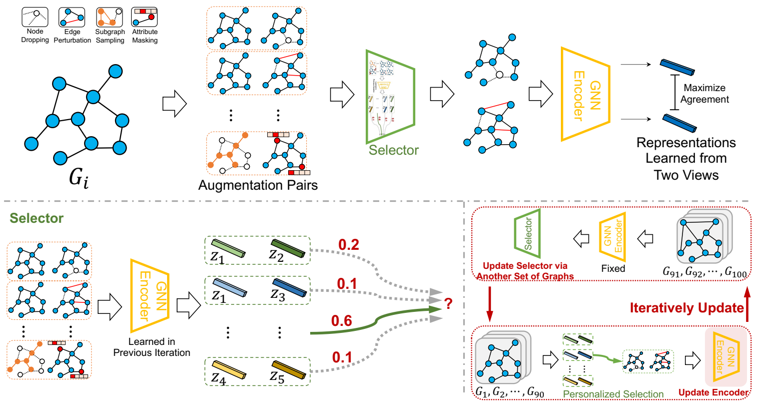


Figure 2: Illustration of our GPA framework. The *personalized augmentation selector* infers the two most informative augmentation operators, and the *GCL model* trains GNN encoder based on the sampled augmented views. Specifically, the *personalized augmentation selector* is learned to adjust its selection strategy to infer optimal augmentations on each graph, according to the characteristic of each graph and the *GCL model*'s performance, i.e., loss.

3 Methodology

In this section, we present the details of the proposed GPA shown in Fig. 2. In a nutshell, it contains two critical components: the *personalized augmentation selector* and the *GCL model*. The former module aims to infer augmentation choices for the downstream GCL methods when training them on the training set, while the later provides reward to update the augmentation selector based on the validation set. In the following, we first illustrate the exponential selection space of our personalized augmentation setting. Then, we elaborate the details of the augmentation selector and the GCL method. Finally, we show how to jointly optimize the two components in a unified perspective.

3.1 Personalized Augmentation Selection Space

Given the graph dataset $\mathcal{G} = \{\mathcal{G}_n : 1 \leq n \leq N\}$ and a pool of augmentation operators $\mathcal{A} = \{A_1, A_2, \cdots, A_K\}$, existing GCL efforts aim to select two informative operators (e.g., $(A_i, A_j \mid 1 \leq i, j \leq K)$) for N graphs to create their augmented views. Since the augmentation operators are shared for the whole dataset, the total selection space is $\binom{K+1}{2}$, i.e., sampling two operators with replacement. This collective selection strategy is widely adopted in existing GCL works. However, as discussed before, various sample graphs may favor different augmentation operators owing to the diversity of graph-structured data. Therefore, we propose to adaptively choose two augmentation strategies for different graphs. Following Definition 1, we define the selected augmentations for each graph $\mathcal{G}_n$ as $A^n_{i,j} = (A^n_i, A^n_j)$. Then, the potential augmentation selection size for each graph is $\binom{K+1}{2}$, and the total selection space for the whole dataset equals to $\binom{K+1}{2}^N$. Although K is empirically small (e.g., $K = 5$) in GCL domain, the total selection space in our personalized setting is still huge and intractable, since the complexity grows exponentially to the number of graphs. For instance, when $K = 5$ and $N = 100$, we already have 15^{100} selection configurations roughly. The situation is more serious in real-world scenarios where N is thousands or even tens of thousands. In this paper, we adopt five essential augmentation operators (i.e. $K = 5$), denoted by $\mathcal{A} = \{\textbf{Identical}, \textbf{NodeDrop}, \textbf{EdgePert}, \textbf{Subgraph}, \textbf{AttMask}\}$. These augmentations are initially proposed by the pioneering work (You et al., 2020), and have been demonstrated to be effective for contrastive learning (You et al., 2021). The details of these augmentations and the augmentation pairs are left in Appendix B

In summary, by considering personalized augmentation, the search space per dataset increases from $\binom{K+1}{2}$ to $\binom{K+1}{2}^N$. Therefore, common selection techniques such as rules of thumb or trial-and-errors adopted by prior GCL approaches (You et al., 2021; 2020) are no longer appropriate. Thus, a tailored augmentation selector is needed to effectively tackle the challenging personalized augmentation problem.

3.2 Personalized Augmentation Selector

In order to assign different augmentation operators to various sample graphs when performing GCL on a specific dataset, random selection is the intuitive solution. Its key idea is to randomly sample two augmentation types for each graph from the candidate set. Despite the simplicity, the random selection approach fails to control the quality of sampled augmentation operators. Therefore, directly coupling the existing GCL framework with random augmentation selection would lead to performance degradation (shown as the variant: GPA-random in Sec. 4.3).

To address this issue, we focus on data-driven search by making the augmentation selection process learnable. The principle idea is to parameterize our personalized augmentation selector with a deep neural network, which takes a query graph as input and outputs its optimal augmentation choices. There are two main hurdles to achieve this goal: (i) given the exponential augmentation space, how can we make our personalized augmentation selector scale to the real-world dataset with thousands of graphs; (ii) since the topology structure and node attributes are crucial to graph-structured data, how can our personalized augmentation selector exploit this information to produce a more precise augmentation choice? We illustrate our dedicated solutions below.

Given the augmentation pool $\mathcal{A} = \{A_i : 1 \leq i \leq K\}$ and a query graph $\mathcal{G}_n$, our augmentation selector is required to select the most two informative augmentations, e.g., (A^n_1, A^n_3), from the candidate set. This selection problem is well-known to be discrete and non-differentiable. Although enormous efforts based on evolution or reinforcement learning have been proposed to address the discrete selection problem, they are still not suitable for such a large selection space (illustrated in Sec. 3.1) and are far from utilizing the properties of graphs. Therefore, we propose to make the search space learnable by relaxing the discrete selection space to be continuous inspired by (Liu et al., 2018) and further make this relaxation consider the characteristic of graphs. Specifically, given the sampled augmentation pair $A^n_{i,j} = (A^n_i, A^n_j)$ of $\mathcal{G}_n$, its importance score $\hat{\alpha}^n_{i,j}$ is computed as

$$\hat{\alpha}^n_{i,j} = \frac{\exp(\alpha^n_{i,j})}{\sum_{i'j'} \exp(\alpha^n_{i',j'})}, \alpha^n_{i,j} = g_\theta(f_w(A^n_i(\mathcal{G}_n) \parallel A^n_j(\mathcal{G}_n))), \tag{1}$$

where g_θ denotes the score function that takes the representations of augmented views $A^n_i(\mathcal{G}_n)$ and $A^n_j(\mathcal{G}_n)$ as input. This design enforces the score estimation to take into account the topology and node attributes of $\mathcal{G}_n$. In practice, g_θ is parameterized as a two-layer MLP with a ReLU activation function. f_w is the GNN encoder for graph representation learning. $\parallel$ indicates the concatenation operation. Through Eq. 1, the discrete augmentation selection process reduces to learning a score function g_θ under the consideration of graph characteristic.

After the personalized selector is well-trained, let $\alpha^n = [\hat{\alpha}^n_{1,1}, \cdots, \hat{\alpha}^n_{i,j}, \cdots, \hat{\alpha}^n_{K,K}]$ denote the vector of importance scores associated with all different augmentation pairs of the graph $\mathcal{G}_n$. The optimal augmentation choice of $\mathcal{G}_n$ can be obtained by selecting the augmentation pair with the maximum score in α^n. For example, $A^n = (A^n_i, A^n_j)$ if $\hat{\alpha}^n_{i,j} = \arg\max_{i',j'} \hat{\alpha}^n_{i',j'}$.

To summarize, the above equation provides a principled solution to our personalized augmentation setting. On one hand, it allows augmentation selection in such a large search space via the simple forward propagation of a shallow neural network, i.e., g_θ. On the other hand, it can also infer the most informative augmentations for each graph based on its own characteristic for downstream GCL model training(discussed in Sec. 3.3).

3.3 GCL Model Learning

After the personalized augmentation selector is plugged in, we can adopt it to train the GCL model. Notice that our model is applicable to arbitrary GCL methods that rely on two augmented views as input. In this section, we mainly focus on the most popular and generic GCL architecture - GraphCL (You et al., 2020) as the backbone and leave other specific architectures for future work.

Assume (A^n_i, A^n_j) is the optimal augmentation pair of graph $\mathcal{G}_n$ identified by our personalized augmentation selector, and $f_w(\cdot)$ is the GNN encoder. GraphCL proposes to learn $f_w(\cdot)$ by maximizing the agreement between the two augmented views, i.e., $A^n_i(\mathcal{G}_n), A^n_j(\mathcal{G}_n)$. The GNN encoder can encode the whole graph into a hidden space $\mathbb{R}^D$. In practice, a shared projection head function $\mathbb{R}^D \to \mathbb{R}^D$ is often applied upon the output of GNN encoder to improve the model capacity. In the following sections, we abuse the notation f_w to denote both the GNN encoding function and the projection function. Based on this notation, we can formally calculate the instance-level contrastive loss as follows:

$$\mathcal{L}(\mathcal{G}_n) = -\log \frac{\exp(\text{sim}(f_w(A^n_i(\mathcal{G}_n)), f_w(A^n_j(\mathcal{G}_n)))/\tau)}{\sum_{n'=1,n'\neq n}^{N} \exp(\text{sim}(f_w(A^n_i(\mathcal{G}_n)), f_w(A^{n'}_j(\mathcal{G}_{n'})))/\tau)}, \tag{2}$$

where $\text{sim}(\cdot,\cdot)$ denotes the cosine similarity function and τ is the temperature parameter. By minimizing Eq. 2, it encourages the two augmented views of the same sample graph to have similar representations, while enforces the augmented representations of disparate graphs to be highly distinct. As the sum operation over all graphs $\mathcal{G}$ in the denominator of Eq. 2 is computationally prohibitive, GCL is often trained under minibatch sampling (You et al., 2020), where the negative views are generated from the augmented graphs within the same minibatch.

Although Eq. 2 looks similar to the traditional contrastive loss, the key difference between them is that the augmentation operators A^n_i, A^n_j are closely related to the sample graph $\mathcal{G}_n$ in our formulation. That is, the augmentation operators learned by our model vary from one graph to another according to their own characteristics, which are previously enforced to be the same regardless of the graph's diverse nature. Benefiting from considering the graph's personality, the GCL method can learn the basic but essential features of graphs and thus achieve more expressive representations for downstream tasks, as empirically verified in Sec. 4.

3.4 Model Optimization

Until now, we have illustrated the detailed personalized augmentation selector as well as the downstream GCL framework, the remaining question is how to effectively train the two modules. The naive solution is to first train the personalized augmentation selector separately, and then optimize the GCL method using the identified personalized augmentations as input. However, such a task-agnostic approach is sub-optimal, since it does not take the mutual reinforced effect between augmentation and the GCL method into consideration. This is because learning a better GCL method requires optimal augmentation strategies since they can augment more personalized features to distinguish it from other objects. Meanwhile, obtaining suitable augmentation strategies also needs the signals of a better GCL method as guidance for optimization. As a result, without linking the two modules in a principled way, it is almost infeasible to enforce the *personalized augmentation selector* to accurately infer optimal augmentation strategy for improving the performance of the *GCL method*. To this end, we propose to tackle this problem by jointly training the two modules under a bi-level optimization, expressed as:

$$\arg\min_{\theta}\ \mathcal{L}_{valid}(w^*(\theta), \theta) s.t. \quad w^*(\theta) = \arg\min_{w} \mathcal{L}_{train}(w, \theta), \tag{3}$$

where θ denotes the trainable parameters of the personalized augmentation selector, and w is the parameters for the GCL method. The upper-level objective $\mathcal{L}_{valid}(w^*(\theta), \theta)$ aims to find θ that minimizes the validation rewards on the validation set given the optimal w^*, and the lower-level objective $\mathcal{L}_{train}(w, \theta^*)$ targets to optimize w by minimizing the contrastive loss based on the training set with θ fixed. We want to remark that GPA only exploits the signals from the self-supervisory task itself without accessing labels. Thus, compared with conventional supervised methods, the validation set here only contains a set of graphs without label information, which is much easier to construct, e.g., randomly sampling 10% of the training set.

By optimizing Eq. 3, the personalized augmentation selector and the target GCL model will be jointly trained to reinforce their reciprocal effects. Since deriving exact solutions for this bi-level problem is indeed analytically intractable, we adopt the alternating gradient descent algorithm to solve it as follows.

3.4.1 Lower-level optimization.

With θ fixed, we can update w with the standard gradient descent procedure as below.

$$w' = w - \xi \nabla_w \mathcal{L}_{train}(w, \theta), \tag{4}$$

where $\mathcal{L}_{train} = \mathbb{E}_{\mathcal{G}_{train}} \mathcal{L}(\mathcal{G}_n)$ with $\mathcal{L}(\mathcal{G}_n)$ is instance-level contrastive loss defined in Eq. 2, $\mathcal{G}_{train}$ denotes the training set, and ξ is the learning rate.

3.4.2 Upper-level optimization.

Since it is not intuitive to directly calculate the gradient *w.r.t.* θ over all augmentation options, we first define the upper-level objective based on Eq. 2 as below:

$$\mathcal{L}_{valid}(w^*(\theta), \theta) = \sum_{A_i, A_j \in \mathcal{A}} \sum_{\mathcal{G}_n \in \mathcal{G}_{valid}} \hat{\alpha}^n_{i,j} \mathcal{L}(\mathcal{G}_n), \tag{5}$$

where $\hat{\alpha}^n_{i,j}$ is the selecting score computed by the score function g_θ with parameter θ defined in Eq. 1. $\mathcal{G}_{valid}$ denotes the validation set. Based on the above loss function, we can update θ by fixing w, expressed as:

$$\theta' = \theta - \xi \nabla_\theta \mathcal{L}_{valid}(w^*(\theta), \theta). \tag{6}$$

However, evaluating the gradient *w.r.t.* θ exactly is intractable and computationally expensive, since it requires solving for the optimal $w^*(\theta)$ whenever θ gets updated. To approximate the optimal solution $w^*(\theta)$, we propose to take one step of gradient descent update for w, without solving the lower-level optimization completely by training until convergence. To further compute the gradient of θ, we apply chain rule to differentiate $\mathcal{L}_{train}(w'(\theta), \theta)$ with respect to θ via w', where w' is defined in Eq. 4. The full derivation is left in Appendix C. Here, we directly present the final result:

$$\nabla_\theta \mathcal{L}_{valid}(w^*(\theta), \theta) \approx \nabla_\theta \mathcal{L}_{valid}(w', \theta) - \xi \frac{\nabla_\theta \mathcal{L}_{train}(w^+, \theta) - \nabla_\theta \mathcal{L}_{train}(w^-, \theta)}{2\epsilon}. \tag{7}$$

By alternating the update rules in Eq. 4 and Eq. 6, we are able to progressively learn the two modules. Although an optimizer with the theoretical guarantee of convergence for the bi-level problem in Eq. 3 remains an open challenge, alternating gradient descent algorithm has been widely adopted to solve similar objectives in Bayesian optimization (Snoek et al., 2012), automatic differentiation (Shaban et al., 2019), and adversarial training (Wang et al., 2019a). The complete optimization procedure of our model and some level of empirical convergence are shown in Appendix D.

3.5 More Discussions on GPA

In addition to augmentation selection, another line of research focuses on generating views via a trainable generator, such as AutoGCL (Yin et al., 2022) and AD-GCL (Suresh et al., 2021). Specifically, the augmented view is obtained by sampling the original one based on the edge or node probabilities predicted by the generator. As such, these methods could limit their applications to simple augmentation strategies (e.g., edge dropping and node masking) and cannot be adopted for applications that require complex augmentation strategies, such as subgraph sampling and motif-based augmentations (e.g., graphon (Ruiz et al., 2020)). In contrast, as an augmentation selection

Table 1: An overview of graph contrastive learning methods. Our GPA model is the first automated GCL effort that supports personalized augmentation in terms of various augmentation types.

	Auto	Simple Augmentation	Complex Augmentation	Applicability	Personalized
GraphCL	-	✓	✓	✓	-
JOAO	✓	✓	✓	✓	-
AD-GCL	✓	✓	-	-	✓
AutoGCL	✓	✓	-	-	✓
Ours	✓	✓	✓	✓	✓

method, GPA has broad applicability, i.e., it is flexible to incorporate new augmentation strategies created by domain experts that might be useful by modifying the augmentation pool without extra effort. We summarize the difference in Table 1

4 Experiments

We evaluate the performance of GPA on multiple graph datasets with various scales and types. We focus on exploring the following research questions. **Q1**: How effective is GPA in performing graph representation learning against state-of-the-art GCL methods in unsupervised and semi-supervised evaluation tasks? **Q2**: How effective is the proposed personalized augmentation selector in identifying augmentations across various datasets? **Q3**: Compared with random selection, how effective is our proposed personalized augmentation selector? Besides, what are the impacts of hyperparameters on GPA, such as the embedding dimension d of the score function? We leave the introduction of datasets, baselines, and experiment setting in Appendix E.

Table 2: Unsupervised learning performance for graph classification in TUdatasets (Averaged accuracy $\pm$ std. over 10 runs). The **bold** numbers denote the best performance and the numbers in blue represent the second best performance

Dataset	NCI1	PROTEINS	DD	MUTAG	COLLAB	RDT-B	RDT-M5K	IMDB-B	Avg.Rank
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	4.75
GraphCL	77.87 ± 0.41	74.39 ± 0.45	78.62 ± 0.40	86.80 ± 1.34	71.36 ± 1.15	$\mathbf{89.53 \pm 0.84}$	55.99 ± 0.28	71.14 ± 0.44	3.88
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	5.00
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	4.50
AD-GCL	69.67 ± 0.51	73.59 ± 0.65	74.49 ± 0.52	88.62 ± 1.27	73.32 ± 0.61	85.52 ± 0.79	53.00 ± 0.82	71.57 ± 1.01	5.13
AutoGCL	$\mathbf{82.00 \pm 0.29}$	75.80 ± 0.36	77.57 ± 0.60	88.64 ± 1.08	70.12 ± 0.68	88.58 ± 1.49	$\mathbf{56.75 \pm 0.18}$	73.30 ± 0.40	2.38
GPA	80.42 ± 0.41	$\mathbf{75.94 \pm 0.25}$	$\mathbf{79.90 \pm 0.35}$	$\mathbf{89.68 \pm 0.80}$	$\mathbf{76.17 \pm 0.10}$	89.32 ± 0.38	53.70 ± 0.19	$\mathbf{74.64 \pm 0.35}$	2.38

Table 3: Semi-supervised learning performance for graph classification

Dataset	NCI1	PROTEINS	DD	RDT-B	RDT-M5K	GITHUB	ogbg-molhiv	Avg.Rank
GAE	74.36 ± 0.24	70.51 ± 0.17	74.54 ± 0.68	87.69 ± 0.40	53.58 ± 0.13	63.89 ± 0.52	-	6.67
InfoGraph	74.86 ± 0.26	72.27 ± 0.40	75.78 ± 0.34	88.66 ± 0.95	53.61 ± 0.31	65.21 ± 0.88	-	4.50
GraphCL	74.63 ± 0.25	74.17 ± 0.34	76.17 ± 1.37	89.11 ± 0.19	52.55 ± 0.45	65.81 ± 0.79	55.48 ± 1.32	4.29
JOAO	74.48 ± 0.25	72.13 ± 0.92	75.69 ± 0.67	88.14 ± 0.25	52.83 ± 0.54	65.00 ± 0.30	56.83 ± 1.39	5.71
JOAOv2	74.86 ± 0.39	73.31 ± 0.48	75.81 ± 0.73	88.79 ± 0.65	52.71 ± 0.28	66.60 ± 0.60	57.39 ± 1.39	4.00
AD-GCL	75.18 ± 0.31	73.96 ± 0.47	$\mathbf{77.91 \pm 0.73}$	$\mathbf{90.10 \pm 0.15}$	53.49 ± 0.28	67.13 ± 0.52	-	2.33
AutoGCL	73.75 ± 2.25	$\mathbf{75.65 \pm 2.40}$	77.50 ± 4.41	79.80 ± 3.47	49.91 ± 2.70	62.46 ± 1.51	-	5.83
GPA	$\mathbf{75.50 \pm 0.14}$	74.27 ± 1.11	76.68 ± 0.81	$\mathbf{89.99 \pm 0.32}$	$\mathbf{54.92 \pm 0.35}$	$\mathbf{68.31 \pm 0.13}$	$\mathbf{60.76 \pm 1.01}$	1.57

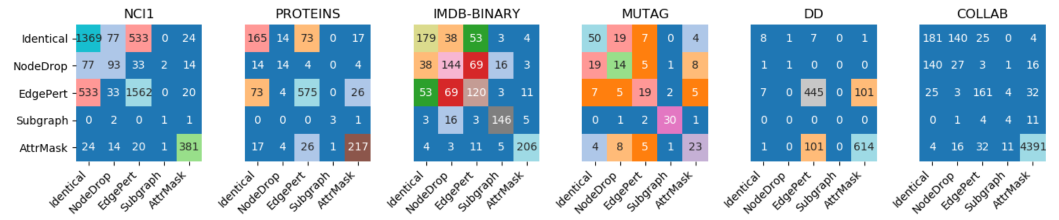
4.1 Comparison with Baselines

We start by comparing the performance of GPA with the state-of-the-art baseline methods under two settings (**Q1**). Table 2 & Table 3 report the results of all methods on diverse datasets under unsupervised and semi-supervised settings, respectively. We have the following **Obs**ervations.

Obs.1. With personalized augmentations for each graph, GPA outperforms vanilla GCL methods with fixed augmentation per dataset. By identifying different augmentations for different sample graphs, GPA performs generally better than vanilla GCL methods on two evaluation scenarios (Table 2 and Table 3). Specifically, in the semi-supervised evaluation task, GPA consistently outperforms GAE, InfoGraph, GraphCL, JOAO, and JOAOv2 on all datasets. In the unsupervised setting, GPA outperforms the vanilla GCL methods on 6 out of 8 datasets. In particular, GPA improves 7.8%, 9.9%, 9.6%, and 6.7% over InfoGraph, JOAOv2, JOAO, and GraphCL on COLLAB in Table 2, respectively. This observation validates the effectiveness of performing personalized augmentation in GCL training.

Obs.2. Across diverse datasets, GPA performs better than (or on par with) the view-generated GCL methods on two evaluation settings. On all datasets originating from diverse domains, GPA generally performs better or sometimes on par with the state-of-the-art AD-GCL and AutoGCL methods, as shown in the average rank. In unsupervised setting, GPA outperforms AD-GCL on all datasets while GPA beats AutoGCL on 6 out of 8 datasets. In semi-supervised setting, GPA outperforms both AD-GCL and AutoGCL on NCI1, RDT-B, RDT-M5K, and GITHUB datasets and achieves the second best on PROTEINS.

Obs.3. GPA scales well on large datasets. To study the scalability of our model, we further conduct experiment on the large-scale OGB dataset: ogbg-molhiv. View-generated GCL methods are excluded for this dataset since they are not officially tested on OGB datasets. From the Table 3, GPA consistently performs better than GraphCL and JOAO. To be specific, GPA improves 9.5% and 5.9% over GraphCL and JOAOv2 on ogbg-molhiv, respectively.



NCI1	Identical	NodeDrop	EdgePert	Subgraph	AttrMask
Identical	1369	77	533	0	24
NodeDrop	77	93	33	2	14
EdgePert	533	33	1562	0	20
Subgraph	0	2	0	1	1
AttrMask	24	14	20	1	381

PROTEINS	Identical	NodeDrop	EdgePert	Subgraph	AttrMask
Identical	165	14	73	0	17
NodeDrop	14	14	4	0	4
EdgePert	73	4	575	0	26
Subgraph	0	0	0	3	1
AttrMask	17	4	26	1	217

IMDB-BINARY	Identical	NodeDrop	EdgePert	Subgraph	AttrMask
Identical	179	38	53	3	4
NodeDrop	38	144	69	16	3
EdgePert	53	69	120	3	11
Subgraph	3	16	3	146	5
AttrMask	4	3	11	5	206

MUTAG	Identical	NodeDrop	EdgePert	Subgraph	AttrMask
Identical	50	19	7	0	4
NodeDrop	19	14	5	1	8
EdgePert	7	5	19	2	5
Subgraph	0	1	2	30	1
AttrMask	4	8	5	1	23

DD	Identical	NodeDrop	EdgePert	Subgraph	AttrMask
Identical	8	1	7	0	1
NodeDrop	1	1	0	0	0
EdgePert	7	0	445	0	101
Subgraph	0	0	0	0	0
AttrMask	1	0	101	0	614

COLLAB	Identical	NodeDrop	EdgePert	Subgraph	AttrMask
Identical	181	140	25	0	4
NodeDrop	140	27	3	1	16
EdgePert	25	3	161	4	32
Subgraph	0	1	4	4	11
AttrMask	4	16	32	11	4391

Figure 3: Augmentation distribution learned by GPA over molecules, bioinformatics, and social networks, in terms of the unsupervised setting.

4.2 Personalized Augmentation Analysis

To study the effectiveness of our model in identifying informative augmentation types for various graphs (**Q2**), we visualize the learned augmentation distribution on Fig. 3. By comparing across different types of datasets, we observe the following.

Obs.4. By learning from the data, GPA can effectively assign different augmentations for various datasets. Our model GPA can identify different augmentations for different sample graphs, and allow different datasets to have their own augmentation distributions (see Fig. 3). Specifically, on MUTAG, 19 graphs prefer (*Identical, NodeDrop*) augmentations, while 30 graphs favor (*Subgraph, Subgraph*) augmentation combinations. Notice that (*Subgraph, Subgraph*) will generate two different subgraph-perturbation-induced augmented views, owning to sample randomness. Besides, COLLAB more likes the (*AttMask, AttMask*) augmentation pair, while DD prefers (*EdgePert, EdgePert*) operations. These observations empirically echo the necessity of performing personalized augmentation for GCL methods.

Another promising observation is that our model can assign (*Identical, Identical*) choice (i.e., two identical views) to some portion of graphs over all datasets. Given that the mutual information between two identical views (i.e., representations) is always maximized, such pure identical augmentations can be regarded as a skip operation. That is, these graphs abandon themselves during the GCL model training. The possible reason is that the existing augmentation strategies are not suitable to capture their characteristics or damage their semantic meanings. In this case, blindly selecting any combination of other augmentation types may incur huge performance degradation or noise. This observation sheds light on designing more advanced augmentation strategies beyond the current basic augmentations. On the other hand, it verifies the effectiveness of the proposed augmentation selector in skipping noisy graphs during model training by providing identical augmentations.

4.3 Ablation Study

To further investigate the effectiveness of the proposed personalized augmentation selector (**Q3**), we compare it with a random-search based variant, i.e., GPA-random. GPA-random replaces the personalized augmentation selector with a random mechanism. Specifically, it assigns one random

Table 4: Ablation study of GPA under unsupervised setting in terms of mean classification accuracy

	NCI1	PROTEINS	DD	MUTAG	IMDB-B
GPA-random	77.71 ± 0.60	74.21 ± 0.39	77.25 ± 0.69	86.08 ± 2.93	71.80 ± 1.13
GPA	$\mathbf{80.42 \pm 0.41}$	$\mathbf{75.94 \pm 0.25}$	$\mathbf{79.90 \pm 0.35}$	$\mathbf{89.68 \pm 0.80}$	$\mathbf{74.64 \pm 0.35}$

augmentation pair to each graph. Noticed that GPA-random still assigns different augmentations to each graph while GraphCL assigns one pre-defined pair of augmentation strategies to the whole dataset. Table 4 shows the results in terms of unsupervised setting. From the table, we can observe that GPA consistently performs better than GPA-random in all cases. In particular, GPA improves 3.5%, 2.3%, 3.4%, 4.2%, and 4.0% over GPA-random on NCI1, PROTEINS, DD, MUTAG, and IMDB-B, respectively. This comparison validates our motivation to develop a tailored and learnable personalized augmentation selector for GCL methods. We leave the analysis for the impacts of hyperparameter d on GPA in Appendix F

5 Related Work

In this section, we briefly review some related works in the automated graph contrastive learning and leave graph representation learning and conventional GCL in Appendix A. For comprehensive review, please refer to (Liu et al., 2021) and (Wu et al., 2020). Recently, some automated GCL methods have been proposed to seek optimal augmentation without repetitive trials. These methods can be roughly categorized into two types.

One is **GCL with view generation**, which targets to train a view generator to predict the edge or node probabilities of the original graph and then generate augmented views by sampling edges or nodes based on the possibilities (You et al., 2022). Two SOTA models in this type are AD-GCL (Suresh et al., 2021) and AutoGCL (Yin et al., 2022). Despite its effectiveness, this type of work is limited to simple augmentation strategies, i.e., edge dropping or node masking, and further loses its applicability to complex graphs and scenarios. Notably, in our experiments, we only include AD-GCL and AutoGCL for comparison because their empirical results are generally better than the work in (You et al., 2022).

The other type, **GCL with augmentation selection**, avoids the limitation of applicability by adopting predefined augmentation strategies. Specifically, the predefined augmentation strategies include simple ones such as node dropping and edge masking and complicated ones like sampling subgraph and motif-based augmentation. Thus, instead of training view generators, this category of work focuses on selecting optimal strategies. A typical model in this category is JOAO (You et al., 2021). It automatically selects the ideal augmentation strategies for each dataset via learning the strategies' importance.

6 Conclusions and Future Work

In this paper, we study the augmentation selection problem for graph contrastive learning. Existing GCL efforts mainly focus on employing two shared augmentation strategies for all graphs in the dataset based on the assumption that they contain similar nature. Here, we argue that such collective augmentation selection is suboptimal in practice due to the heterogeneity of graph structure data. Different graphs should have different augmentation preferences. To bridge the gap, we propose a novel graph contrastive learning framework with personalized augmentation termed as GPA. GPA not only allows each graph to select its optimal augmentation types, but also automates the selection via a personalized augmentation selector, which can be jointly trained with downstream GCL models under a bi-level optimization. Empirical results on the graph classification task demonstrate the superiority of GPA against state-of-the-art GCL methods in terms of unsupervised and semi-supervised settings, across multiple benchmark datasets with various types such as molecules, bioinformatics, and social networks.

REFERENCES

Deli Chen, Yanyai Lin, Lei Li, Xuancheng Ren Li, Jie Zhou, Xu Sun, et al. Distance-wise graph contrastive learning. *IJCAI*, 2022.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *ICLR*, 2018.

Ming Ding, Jie Tang, and Jie Zhang. Semi-supervised learning on graphs with generative adversarial nets. In *CIKM*, pp. 913–922, 2018.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Alberto Garcia Duran and Mathias Niepert. Learning graph representations with embedding propagation. *NeurIPS*, 30, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *NeurIPS*, 30, 2017.

Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *ICML*, pp. 4116–4126. PMLR, 2020.

Zhenyu Hou, Xiao Liu, Yuxiao Dong, Chunjie Wang, Jie Tang, et al. Graphmae: Self-supervised masked graph autoencoders. *arXiv preprint arXiv:2205.10803*, 2022.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *NeurIPS*, 33:22118–22133, 2020a.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *KDD*, pp. 1857–1867, 2020b.

Roshni G Iyer, Wei Wang, and Yizhou Sun. Bi-level attention graph neural networks. In *ICDM*, pp. 1126–1131. IEEE, 2021.

Yizhu Jiao, Yun Xiong, Jiawei Zhang, Yao Zhang, Tianqi Zhang, and Yangyong Zhu. Sub-graph contrast for scalable self-supervised graph representation learning. In *ICDM*, pp. 222–231. IEEE, 2020.

Wei Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zitao Liu, and Jiliang Tang. Self-supervised learning on graphs: Deep insights and new direction. *arXiv preprint arXiv:2006.10141*, 2020.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NeurIPS*, 2016.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *ICLR*, 2018.

Ninghao Liu, Qiaoyu Tan, Yuening Li, Hongxia Yang, Jingren Zhou, and Xia Hu. Is a single vector enough? exploring node polysemy for network embedding. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 932–940, 2019.

Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *TKDE*, 2021.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, 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)*, 2020. URL `www.graphlearning.io`.

Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. Graph representation learning via graphical mutual information maximization. In *WWW*, pp. 259–270, 2020.

Jiezhong Qiu, 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 *SIGKDD*, pp. 1150–1160, 2020.

Luana Ruiz, Luiz Chamon, and Alejandro Ribeiro. Graphon neural networks and the transferability of graph neural networks. *Advances in Neural Information Processing Systems*, 33:1702–1712, 2020.

Amirreza Shaban, Ching-An Cheng, Nathan Hatch, and Byron Boots. Truncated back-propagation for bilevel optimization. In *AISTATS*, pp. 1723–1732. PMLR, 2019.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *NeurIPS*, 25, 2012.

Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *ICLR*, 2020.

Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. Adversarial graph augmentation to improve graph contrastive learning. *NeurIPS*, 34, 2021.

Qiaoyu Tan, Ninghao Liu, and Xia Hu. Deep representation learning for social network analysis. *Frontiers in big Data*, 2:2, 2019.

Qiaoyu Tan, Ninghao Liu, Xiao Huang, Rui Chen, Soo-Hyun Choi, and Xia Hu. Mgae: Masked autoencoders for self-supervised learning on graphs. *arXiv preprint arXiv:2201.02534*, 2022.

Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Veličković, and Michal Valko. Bootstrapped representation learning on graphs. In *ICLR 2021 Workshop on Geometrical and Topological Representation Learning*, 2021.

Yizhou Sun Ting Chen, Song Bian. Are powerful graph neural nets necessary? a dissection on graph classification. *CoRR*, abs/1905.04579, 2019.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *ICLR (Poster)*, 2(3):4, 2019.

Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *ICLR*, 2019.

Sheng Wan, Shirui Pan, Jian Yang, and Chen Gong. Contrastive and generative graph convolutional networks for graph-based semi-supervised learning. *AAAI*, 2020.

Haonan Wang, Jieyu Zhang, Qi Zhu, and Wei Huang. Augmentation-free graph contrastive learning. *arXiv preprint arXiv:2204.04874*, 2022.

Jingkang Wang, Tianyun Zhang, Sijia Liu, Pin-Yu Chen, Jiacen Xu, Makan Fardad, and Bo Li. Towards a unified min-max framework for adversarial exploration and robustness. *arXiv preprint arXiv:1906.03563*, 2019a.

Lu Wang, Wenchao Yu, Wei Wang, Wei Cheng, Wei Zhang, Hongyuan Zha, Xiaofeng He, and Haifeng Chen. Learning robust representations with graph denoising policy network. In *ICDM*, pp. 1378–1383. IEEE, 2019b.

Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

Lirong Wu, Haitao Lin, Cheng Tan, Zhangyang Gao, and Stan Z Li. Self-supervised learning on graphs: Contrastive, generative, or predictive. *IEEE TKDE*, 2021.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *TNNLS*, 32(1):4–24, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2018.

Yihang Yin, Qingzhong Wang, Siyu Huang, Haoyi Xiong, and Xiang Zhang. Autogcl: Automated graph contrastive learning via learnable view generators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8892–8900, 2022.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *NeurIPS*, 33:5812–5823, 2020.

Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. *ICML*, 2021.

Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. Bringing your own view: Graph contrastive learning without prefabricated data augmentations. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pp. 1300–1309, 2022.

Jiaqi Zeng and Pengtao Xie. Contrastive self-supervised learning for graph classification. In *AAAI*, volume 35, pp. 10824–10832, 2021.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *WWW*, pp. 2069–2080, 2021.

Table 5: Distinct graph augmentation pairs.

$A_{i,j}$	A_i, A_j	Augmentation	Augmentation
$A_{1,1}$	A_1,A_1	Identical	Identical
$A_{1,2}$	A_1, A_2	Identical	Node Dropping
$A_{1,3}$	A_1, A_3	Identical	Edge Perturbation
$A_{1,4}$	A_1, A_4	Identical	Subgraph
$A_{1,5}$	A_1, A_5	Identical	Attribute Masking
$A_{2,2}$	A_2, A_2	Node Dropping	Node Dropping
$A_{2,3}$	A_2, A_3	Node Dropping	Edge Perturbation
$A_{2,4}$	A_2, A_4	Node Dropping	Subgraph
$A_{2,5}$	A_2, A_5	Node Dropping	Attribute Masking
$A_{3,3}$	A_3, A_3	Edge Perturbation	Edge Perturbation
$A_{3,4}$	A_3, A_4	Edge Perturbation	Subgraph
$A_{3,5}$	A_3, A_5	Edge Perturbation	Attribute Masking
$A_{4,4}$	A_4, A_4	Subgraph	Subgraph
$A_{4,5}$	A_4, A_5	Subgraph	Attribute Masking
$A_{5,5}$	A_5, A_5	Attribute Masking	Attribute Masking

A RELATED WORK

A.1 GRAPH REPRESENTATION LEARNING

With the rapid development of graph neural networks (GNNs), a large number of GNN-based graph representation learning frameworks have been proposed (Wu et al., 2020; Wang et al., 2019b; Iyer et al., 2021; Wang et al., 2022; Tan et al., 2019), which exhibit promising performance. Typically, these methods can be divided into supervised and unsupervised categories. While supervised methods (Chen et al., 2018; Ding et al., 2018; Kipf & Welling, 2017; Veličković et al., 2018) achieve empirical success with the help of labels, reliable labels are often scarce in real-world scenarios. Thus, unsupervised graph learning approaches (Kipf & Welling, 2016; Garcia Duran & Niepert, 2017; Hamilton et al., 2017; Hou et al., 2022) have broader application potential. For example, one of the well-known methods is GAE (Kipf & Welling, 2016) which learns graph representations by reconstructing the network structure under the autoencoder approach. Another popular approach GraphSAGE (Hamilton et al., 2017) aims to train GNNs by a random-walk based objective.

A.2 GRAPH CONTRASTIVE LEARNING

Graph contrastive learning (GCL) has attracted significant attention in the past two years for self-supervised graph learning (Wu et al., 2021). It learns GNN encoder by maximizing the agreement between representations of a graph in its different augmented views, so that similar graphs are close to each other, while dissimilar ones are spaced apart. Many GCL efforts have been devoted to node level (Wan et al., 2020; Peng et al., 2020; Hu et al., 2020b), subgraph level (Qiu et al., 2020; Jiao et al., 2020), and graph level (You et al., 2020; Zeng & Xie, 2021) scenarios, where the key challenge lies in designing effective graph augmentations. Recently, GraphCL (You et al., 2020) introduced four types of graph augmentations, including node dropping, edge perturbation, subgraph sampling, and node attribute masking, and showed that different graph applications may favor different augmentation combinations. However, the optimal augmentation configuration for a given dataset is mainly determined by either domain experts or extensive trial-and-errors, thus limiting the boarder applications of GCL in practice.

B DETAILS OF AUGMENTATION STRATEGIES

The graph augmentation pairs are summarized in Table 5. The details of augmentation strategies are listed below.

- **NodeDrop.** Given the graph $\mathcal{G}_n$, NodeDrop randomly discards a fraction of the vertices and their connections. The dropping probability of each node follows the i.i.d. uniform distribution. The underlying assumption is that missing part of vertices does not damage the semantic information of $\mathcal{G}_n$.

- **EdgePert.** The connectivity in $\mathcal{G}_n$ is perturbed through randomly adding or dropping a certain portion of edges. We also follow the i.i.d. uniform distribution to add/drop each edge. The underlying prior is that the semantic meaning of $\mathcal{G}_n$ is robust to the variance of edges.
- **AttMask.** AttMask masks the attributes of a certain proportion of vertices. Similarly, each node's masking possibility follows the i.i.d. uniform distribution. Attribute masking implies that the absence of some vertex attributes does not affect the semantics of $\mathcal{G}_n$.
- **Subgraph.** This augmentation method samples a subgraph from the given graph $\mathcal{G}_n$ based on random walk. It believes that most of the semantic meaning of $\mathcal{G}_n$ can be preserved in its local structure.

C Approximate Gradient

We show the full derivation of calculating the gradient *w.r.t.* θ as follows. When optimizing θ, we fix the optimal $w^*(\theta)$ and update θ as follows:

$$\theta' = \theta - \xi \nabla_\theta \mathcal{L}_{valid}(w^*(\theta), \theta), \tag{8}$$

where ξ is the learning rate for one step. As illustrated in the main body, we adopt w' as $w^*(\theta)$. Therefore, the gradient of θ can be approximated as:

$$\begin{aligned} \nabla_\theta \mathcal{L}_{valid}(w^*(\theta), \theta) &\approx \nabla_\theta \mathcal{L}_{valid}(w', \theta) \\ &\approx \nabla_\theta \mathcal{L}_{valid}(w - \xi \nabla_w \mathcal{L}_{train}(w, \theta), \theta). \end{aligned} \tag{9}$$

Applying chain rule to Eq. 9, we further obtain the gradient of θ as:

$$\begin{aligned} \nabla_\theta \mathcal{L}_{valid}(w^*(\theta), \theta) \approx \nabla_\theta \mathcal{L}_{valid}(w', \theta) \\ - \xi \nabla^2_{\theta,w} \mathcal{L}_{train}(w, \theta) \nabla_{w'} \mathcal{L}_{valid}(w', \theta) \end{aligned} \tag{10}$$

Since the computation cost of the second term in Eq. 10 is still high, it can be further approximated by the finite difference method:

$$\begin{aligned} &\xi \nabla^2_{\theta,w} \mathcal{L}_{train}(w, \theta) \nabla_{w'} \mathcal{L}_{valid}(w', \theta) \\ &\approx \frac{\nabla_\theta \mathcal{L}_{train}(w^+, \theta) - \nabla_\theta \mathcal{L}_{train}(w^-, \theta)}{2\epsilon} \end{aligned} \tag{11}$$

where $w^\pm = w \pm \epsilon \nabla_{w'} \mathcal{L}_{valid}(w', \theta)$ and ϵ denotes a small scalar.

D Optimization

The complete optimization procedure of GPA is shown in the Algorithm 1. We also show some level of empirical convergence in Figure 4.

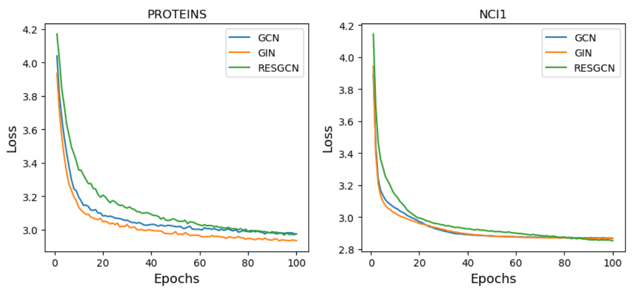


Figure 4: Empirical training curves of approximate gradient scheme in GPA on datasets PROTEINS and NCI1 with different GNN encoders.

Algorithm 1: The framework of GPA

```
Input: A graph dataset $\mathcal{G}$, the personalized augmentation selector $g_\theta(\cdot)$, and a GCL model
       $f_w(\cdot)$;
Output: The well-trained GCL model;
1  Split the input graph dataset $\mathcal{G}$ into train $\mathcal{G}_{train}$ and validation $\mathcal{G}_{valid}$ set;
2  Initialize the selector parameter $\theta$ and the GCL model parameter $w$;
3  while not converge do
4      Randomly sample a minibatch of graphs from the training set;
5      Infer the optimal augmentation pairs for sampled graphs using the personalized
         augmentation selector $g_\theta(\cdot)$;
6      Update parameters $w$ of the GCL learner based on the sampled graphs and the identified
         augmentation types according to Eq. 4;
7      Randomly sample a batch of graphs from validation set;
8      Compute the rewards based on the sampled validation graphs using the updated $w'$
         according to Eq. 5;
9      Update parameters $\theta$ of the personalized augmentation selector according to Eq. 6 and
         Eq. 7;
10 Return The well-trained GCL model.
```

Table 6: Statistics of the datasets.

	$\mid \mathcal{G} \mid$	Avg.Nodes	Avg.Edges	$\#Label$
NCI1	$4,110$	29.87	32.30	2
PROTEINS	$1,113$	39.06	72.82	2
DD	$1,178$	284.32	715.66	2
MUTAG	188	17.93	19.79	2
COLLAB	$5,000$	74.49	$2,457.78$	3
IMDB-BINARY	$1,000$	19.77	96.53	2
REDDIT-BINARY	$2,000$	429.63	497.75	2
REDDIT-MULTI-5K	$4,999$	508.52	594.87	5
GITHUB	$12,725$	113.79	234.64	2
ogbg-molhiv	$41,127$	25.5	27.5	2

E Experiment settings

E.1 datasets

For a comprehensive comparison, we evaluate the performance of GPA on eleven widely used benchmark datasets. Specifically, we include two small molecules networks (**NCI1** and **MUTAG**), two bioinformatics networks (**DD** and **PROTEINS**), and five social networks (**COLLAB**, **REDDIT-BINARY**, **REDDIT-MULTI-5K**, **IMDB-BINARY**, and **GITHUB**) from TUDatasets (Morris et al., 2020). To evaluate the scalability of our model, we also use one large-scale OGB (Hu et al., 2020a) dataset **ogbg-molhiv**. The data statistics are summarized in Table 6.

E.2 Baselines

To validate the effectiveness of GPA, we compare against three categories of state-of-the-art competitors. First, to evaluate the effectiveness of contrastive learning, we include one traditional network embedding method **GAE** (Kipf & Welling, 2016). Second, to study why we need personalized augmentation, we include classic GCL methods that assign identical augmentation strategies for all graphs **InfoGraph** (Sun et al., 2020), **GraphCL** (You et al., 2020), **JOAO** (You et al., 2021) and its variant **JOAOv2** (You et al., 2021). Note that JOAO and JOAOv2 are two sample-based automated GCL methods that focus on selecting the most suitable predefined augmentation strategies for each dataset. They are the most relevant baselines to our proposed GPA.

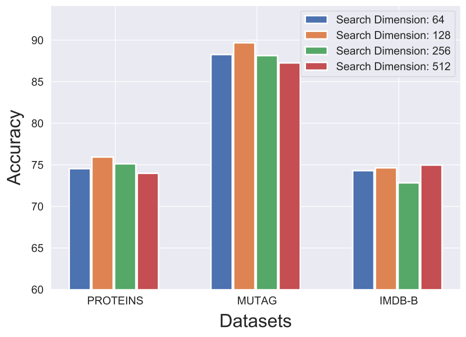


Figure 5: Personalized augmentation selector dimension analysis of GPA

E.3 Learning protocols

Following common protocols (Sun et al., 2020; You et al., 2020), we aim to evaluate the performance of GPA in unsupervised and semi-supervised settings. In our model training phase, we randomly split 10% of graphs in each dataset into the validation set and use the remaining for training. After the model is trained, in unsupervised setting, we train an SVM classifier on the graph representations generated by the trained GCL model, and apply 10-fold cross-validation to evaluate the performance. For semi-supervised setting, we finetune the GCL model (its GNN encoder) with a logistic regression layer for semi-supervised learning, where the labeled sample ratio is 0.1. To avoid randomness, we repeat the process for ten times and report the averaged results.

E.4 Implementation details

Our model is built upon Pytorch and PyG (PyTorch Geometric) library (Fey & Lenssen, 2019). We train our model with Adam optimizer using a fixed batch size of 128. Similar to GraphCL (You et al., 2020), the default augmentation ratio is set to 0.2 for all augmentation types. In unsupervised setting, we adopt a three-layer GIN (Xu et al., 2018) encoder with hidden dimension 128 for all datasets. In semi-supervised scenarios, we employ a five-layer ResGCN (Ting Chen, 2019) encoder with dimension 128 for TUDatasets (Morris et al., 2020), while a five-layer GIN (Xu et al., 2018) encoder with dimension 300 for OGB datasets as suggested in (Hu et al., 2020a). There is one hyper-parameter in our model, i.e., the hidden dimension d of score function g_θ. We search d within the set $\{128, 256, 512\}$. The impact of the hidden dimension is analyzed in Appendix F.

F Parameter Sensitivity Analysis

We now study the impact of the parameter, i.e., the hidden dimension d of the score function. Specifically, we search d from the set $\{64, 128, 256, 512\}$ and plot the results of GPA on three representative datasets in Fig. 5. Similar observations are obtained by other datasets. From the figure, we can see that GPA generally performs stably across various dimension choices. In experiments, we fix $d = 128$ for all datasets.

G Complexity analysis of GPA

We analyze the complexity of GPA and illustrate its impact on the GCL backbone –GraphCL (You et al., 2020). Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and the GNN encoder f_w. The time complexity for the GNN backbones used in common graph learning tasks is $\mathcal{O}(|\mathcal{V}| + |\mathcal{E}|)$. Since the proposed GPA method is built upon GraphCL (You et al., 2020) and the personalized augmentation selector g_θ, the additional cost is mainly caused by the selector. For GraphCL, it performs two encoder computations per iteration plus a prediction head. If we assume the backward cost is similar to that of the forward pass, the complexity of GraphCL is $4C_{\text{encoder}}(|\mathcal{V}| + |\mathcal{E}|) + 2C_{\text{head}}(N_{\text{batch}}) + C_{\text{loss}} + 2C_{\text{aug}}$, where $C_{\cdot}$ are constants rely on the neural architectures, and N_{batch} is the batch size. For the selector g_θ, it takes K encoder computations per iteration to generate graph representations and simple multilayer perceptrons (MLPs) to estimate the sampling probability. Thus its complexity is $KC_{\text{encoder}}(|\mathcal{V}| + |\mathcal{E}|) + C_{\text{selector}}(N_{\text{batch}}) + KC_{\text{aug}}$. Therefore, the total complexity for GPA and GraphCL are $(2K+4)C_{\text{encoder}}(|\mathcal{V}|+|\mathcal{E}|)+(2K+2)C_{\text{aug}}$ and $4C_{\text{encoder}}(|\mathcal{V}|+|\mathcal{E}|)+2C_{\text{aug}}$, respectively, since the costs of GNN encoder and data augmentation are significantly higher than others. Assume the time cost for data augmentation is 5 times of GNN encoder, i.e., $C_{\text{aug}} = 5C_{\text{encoder}}$, then the time complexity of GPA is $\frac{2K+4+(2K+2)*5}{4+2*5} = \frac{12K+14}{14}$ times that of GraphCL. Given that $K = 5$ in our experiments, the complexity of GPA is around 6 times of GraphCL in theory. However, due to the inefficient implementation of our current code (e.g., for loop in the loss function and perform K augmentations in series), the empirical complexity of GPA is less than 7 times of GraphCL. It is noteworthy that parallel techniques such as multiprocessing could be used to accelerate the training.

H Comparison of GPA and JOAO

While JOAO and GPA all focus on training GraphCL in an automated fashion, they differ in two crucial ways: 1) The research problem is different. JOAO aims to learn a shared sampling distribution for a given dataset, but GPA targets to learn a sampling vector for each instance (i.e., graph) in the dataset. Notably, due to randomness, JOAO can sample different augmentations to different graphs to some extent. However, since the sampling distribution is fixed, it is “fake” personalization because it does not consider the characteristics of different graphs. In contrast, GPA could explicitly generate personalized sampling distribution to each graph, which is tailored and more challenging as the augmentation space grows exponentially to the dataset size. 2) The sampler optimization process is different. JOAO adopts a sampling-based approach to directly train a K-dimensional sampling distribution parameters. However, such a solution cannot be applied to our personalized scenarios because i) the total trainable parameters will be linear to the size of the dataset, i.e., $K \times |\mathcal{V}|$, where $\mathcal{V}$ is the dataset; and ii) it is restricted to transductive settings and cannot generate distribution vectors for new graphs, i.e., unseen graphs during training. To this end, we adopt a new approach to achieve personalized augmentation allocation by parameterizing the sampler as a neural network taking the topological and node attributes of a graph as input. It reduces the trainable parameters of the sampler to be independent with the size of the dataset.