MA-GCL: Model Augmentation Tricks for Graph Contrastive Learning

Xumeng Gong[1], **Cheng Yang**[1*], **Chuan Shi**[1]

[1] Beijing University of Posts and Telecommunications
Xumeng1141@bupt.edu.cn,yangcheng@bupt.edu.cn, shichuan@bupt.edu.cn

Abstract

Contrastive learning (CL), which can extract the information shared between different contrastive views, has become a popular paradigm for vision representation learning. Inspired by the success in computer vision, recent work introduces CL into graph modeling, dubbed as graph contrastive learning (GCL). However, generating contrastive views in graphs is more challenging than that in images, since we have little prior knowledge on how to significantly augment a graph without changing its labels. We argue that typical data augmentation techniques (*e.g.*, edge dropping) in GCL cannot generate diverse enough contrastive views to filter out noises. Moreover, previous GCL methods employ two view encoders with exactly the same neural architecture and tied parameters, which further harms the diversity of augmented views. To address this limitation, we propose a novel paradigm named model augmented GCL (MA-GCL), which will focus on manipulating the architectures of view encoders instead of perturbing graph inputs. Specifically, we present three easy-to-implement model augmentation tricks for GCL, namely *asymmetric*, *random* and *shuffling*, which can respectively help alleviate high-frequency noises, enrich training instances and bring safer augmentations. All three tricks are compatible with typical data augmentations. Experimental results show that MA-GCL can achieve state-of-the-art performance on node classification benchmarks by applying the three tricks on a simple base model. Extensive studies also validate our motivation and the effectiveness of each trick. (Code, data and appendix are available at https://github.com/GXM1141/MA-GCL.)

Introduction

Contrastive learning (CL) has emerged as a promising paradigm for unsupervised vision representation learning (Chen et al. 2020; He et al. 2020; Grill et al. 2020; Van den Oord, Li, and Vinyals 2018; Hjelm et al. 2019). In short, typical CL methods will generate two views of the same sample by data augmentations, and then maximize the similarity between their encoded representations. In this way, CL can extract the information shared between different views (Tian et al. 2020), and thereby alleviate the task-irrelevant noises that only appear in a single view. Recently, CL is also introduced to graph domain for unsupervised graph representation learning, dubbed as graph contrastive learning (GCL) (You et al. 2020; Zhu et al. 2021; Hassani and Khasahmadi 2020; Zhu et al. 2020; Qiu et al. 2020; Suresh et al. 2021). GCL approaches have achieved competitive performance on various graph benchmarks compared to the counterpart trained with ground-truth labels.

However, generating data augmentations on graphs is more challenging than that on images. Theories on contrastive learning (Tian et al. 2020) show that good contrastive views should be diverse while keeping the task-relevant information untouched. In computer vision area, we have strong prior knowledge on how to significantly augment an image without changing its labels (*e.g.*, rotation or shift). While in graph learning area, dropping a single edge could possibly destroy a key connection (*e.g.,* an important chemical bond) related to downstream tasks. Nevertheless, most existing graph data augmentation (GDA) techniques (*e.g.,* node or edge dropping) choose to randomly perturb the graph topology (Velickovic et al. 2019; You et al. 2020; Hassani and Khasahmadi 2020; Zhu et al. 2020; Qiu et al. 2020) as contrastive views.

We argue that existing GDA techniques are in a dilemma: the perturbations that preserve sufficient task-relevant information cannot generate diverse enough augmentations to filter out noises. Moreover, previous GCL methods employ two view encoders with exactly the same neural architecture and tied parameters[1], which further harms the diversity of augmented views. Though some recent GCL methods generate augmentations in a heuristic (Zhu, Sun, and Koniusz 2021) or adversarial (You et al. 2021; Suresh et al. 2021) way, they still use two view encoders with identical architectures and fail to fully address this problem. Besides, a very recent work (Xia et al. 2022) proposes to perturb model parameters instead of graph inputs as augmentations. Note that the function of encoded semantics with respect to model parameters is rather complex. Hence the parameter perturbation drawn from a simple distribution (*e.g.*, Gaussian distribution) may harm the semantics in encoded representations. We will empirically validate our motivation in experiments.

In this work, we propose a novel paradigm for GCL to address the above limitations, named as MA-GCL (**M**odel

*Corresponding author.
Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1]Though some work (Thakoor et al. 2021; Xia et al. 2022) employs momentum update and hence makes two view encoders not entirely the same, we argue that such difference is not enough.

Augmented **G**raph **C**ontrastive **L**earning). We interpret graph neural network (GNN) encoders as a composition of propagation operators (*i.e.*, graph filters) and transformation operators (*i.e.*, weight matrix and non-linearity). Note that previous GCL methods adopt fixed and identical GNNs as view encoders, *i.e.*, the numbers of propagation and transformation operators, as well as their permutations, are fixed and identical for encoding different contrastive views. While in this work, we focus on perturbing the neural architectures of view encoders as contrastive views, and present three model augmentation tricks: (1) **Asymmetric strategy**: we will use two contrastive encoders with different numbers of propagation operators. We theoretically prove that this strategy can help alleviate high-frequency noises. (2) **Random strategy**: we will randomly vary the number of propagation operators in every epoch. The intuition behind is that varying the propagation depth can enrich the diversity of training instances, and hence help predict downstream task. (3) **Shuffling strategy**: we will shuffle the permutations of propagation and transformation operators in two view encoders. The intuition behind is that shuffling the order of operators will not change the semantics of an input graph, but will perturb the encoded representations as safer augmentations. All three tricks are simple and compatible with typical data augmentations.

We conduct experiments on six graph benchmarks to show the superiority of MA-GCL. For the implementation of MA-GCL, we combine the three tricks and apply them on a simple base model, which can be seen as a simplified version of GRACE (Zhu et al. 2020) and performs worse than GRACE and recent state-of-the-art (SOTA) methods. Experimental results show that MA-GCL achieves SOTA performance on 5 out of 6 benchmarks, and the relative improvement against best performed baseline can go up to 2.7%. Extensive experiments further show that all three tricks will contribute to the overall improvement, and the *Asymmetric strategy* is the most effective one among the three.

Our contributions are as follows:

(1) We highlight a key limitation in most GCL methods that the graph augmentations are not diverse enough to filter out noises. To overcome this limitation, we propose a novel paradigm named model augmentation for GCL, which will focus on perturbing the architectures of GNN encoders instead of graph inputs or model parameters.

(2) We present three effective model augmentation tricks for GCL, namely *asymmetric*, *random* and *shuffling*, which can respectively help alleviate high-frequency noises, enrich training instances and bring safer augmentations. All three tricks are simple, easy-to-implement and compatible with typical data augmentations.

(3) Experimental results show that, compared with recent SOTA GCL methods, MA-GCL can achieve SOTA performance on 5 out of 6 graph benchmarks by applying the three tricks on a simple base model. Extensive studies also validate our motivation and the effectiveness of each strategy.

Related Works

Graph Contrastive Learning Recently, many GCL methods were proposed (Sun et al. 2020; Wang et al. 2019; Zhu et al. 2020; Qiu et al. 2020) for unsupervised node/graph representation learning, and achieved the SOTA performance on this task. We roughly categorize previous GCL methods into two groups, depending on how they generate contrastive views: (1) The first group usually generates views based on different scales of substructures. InfoGraph (Sun et al. 2020) treats the graph structure of different scales (*e.g.,*, nodes, edges, triangles) as contrastive views to learn graph-level representations. GCC (Qiu et al. 2020) introduces InfoNCE to large-scale graph pre-training by adopting different sub-graphs as contrastive views. (2) The second group usually generates views by applying data augmentation on graph inputs. GRACE (Zhu et al. 2020) and GraphCL (You et al. 2020) randomly perturb the graph structures (*e.g.,*, node & edge dropping, graph sampling) to build contrastive views. MVGRL (Hassani and Khasahmadi 2020) and MV-CGC (Yuan et al. 2021) further employ graph diffusion to generate augmentation views of original graphs. BGRL (Thakoor et al. 2021) performs GCL without negative pairs inspired by BYOL (Grill et al. 2020). ARIEL (Feng et al. 2022a) further introduces an extra adversarial view to GCL. Recent methods propose to generate views in adaptive ways. GCA (Zhu et al. 2021) designs augmentation schemes based on different graph property (*e.g.,*, node degree, eigenvector, PageRank). DiGCL (Tong et al. 2021) uses Laplacian perturbation without changing directed graph structure. JOAO (You et al. 2021) and AD-GCL (Suresh et al. 2021) automatically select graph augmentations in adversarial ways. InfoGCL (Xu et al. 2021) uses task labels to select optimal views by information bottleneck principle. (Duan et al. 2022) proposes label-preserving augmentations (LPA) to improve GCL. RGCL (Li et al. 2022) creates rationale-aware views for CL. Besides the two groups above, there are some approaches generating views based on mathematical interpretations or parameter perturbations. COLES (Zhu, Sun, and Koniusz 2021) realizes better negative sampling strategy and extends Laplacian Eigenmaps for GCL. DSGC (Yang et al. 2022) conducts CL among views from hyperbolic and Euclidean spaces. GASSL (Yang, Zhang, and Yang 2021) generates views by adding perturbations to both input features and hidden layers. SimGRACE (Xia et al. 2022) adds Gaussian noises to model parameters as contrastive views without data augmentation. (Yu et al. 2022) adds uniform noises to the embedding space. Note that all these GCL methods employ two GNNs with the same fixed architecture as view encoders. In this work, we propose a novel paradigm for GCL, where the architectures of view encoders in MA-GCL are different and randomized.

Relevant Theories and Techniques In terms of the motivation of MA-GCL, the principle of keeping more task-relevant information and reducing more noise, has been studied through Information Bottleneck (TISHBY 1999) theory and minimal sufficient statistics (Soatto and Chiuso 2016) in computer vision (Tian et al. 2020). Some recent GCL methods (Suresh et al. 2021; Xu et al. 2021) are also developed with this principle. In terms of the randomness of MA-GCL, the idea of using randomized GNN architectures is quite different from Graph Random Neural Networks (GRAND) (Feng et al. 2020, 2022b). GRAND designs multiple channels of GNNs and ensembles them for prediction,

where the input features are randomized in each channel. Thus the neural architecture of GRAND is fixed, and the aim of GRAND is to address the over-smoothing and non-robustness issues in semi-supervised learning.

Notations and Preliminaries

Notations Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph, where $\mathcal{V} = \{1, ..., |\mathcal{V}|\}$ is the set of $|\mathcal{V}|$ vertices and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of $|\mathcal{E}|$ edges between vertices. $\boldsymbol{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ denotes the adjacency matrix of $\mathcal{G}$ with self-loops, where $\boldsymbol{A}_{i,j} = 1$ if $(v_i, v_j) \in \mathcal{E}$ or $i = j$. Each node v_i is associated with a feature vector $\boldsymbol{x}_i \in \boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$, where d is the feature dimension.

Graph Neural Networks (GNNs) GNNs (Kipf and Welling 2016; Velickovic et al. 2017; Zhou et al. 2020) generalize deep learning techniques into graphs for node encoding. Specifically, GNNs will stack multiple propagation layers and transformation layers, and then apply them to the raw features $\boldsymbol{X}$. Here we denote the operators of propagation layer and the transformation layer as g and h, respectively:

$$g(\boldsymbol{Z}; \boldsymbol{F}) = \boldsymbol{F}\boldsymbol{Z},\ h(\boldsymbol{Z}; \boldsymbol{W}) = \sigma(\boldsymbol{Z}\boldsymbol{W}), \tag{1}$$

where $\boldsymbol{Z} \in \mathbb{R}^{|\mathcal{V}|\times d_Z}$ is node embeddings, $\boldsymbol{F} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the graph filter matrix, $\boldsymbol{W} \in \mathbb{R}^{d_Z \times d_O}$ denotes the weight matrix of linear transformation, and σ is the nonlinear function (*e.g.,* ReLU). Graph filter $\boldsymbol{F}$ is a constant matrix based on the adjacency matrix $\boldsymbol{A}$, while $\boldsymbol{W}$ is trainable parameters. The output of g has the same shape with input matrix $\boldsymbol{Z}$, while the output of h is a $|\mathcal{V}| \times d_O$ dimensional representation matrix. With the help of the two operators, node representations can be propagated to their neighbors, and mapped to a new embedding space by linear and nonlinear transformations.

Now we can formalize GNNs as the composition of multiple g and h operators. For example, a L-layer Graph Convolutional Network (GCN) (Kipf and Welling 2016) and a L-layer SGC (Wu et al. 2019) can be written as

$$\begin{aligned} &GCN(\boldsymbol{X}) = h_L \circ g \circ h_{L-1} \circ g \circ \cdots \circ h_1 \circ g(\boldsymbol{X}), \\ &SGC(\boldsymbol{X}) = h \circ g^{[L]}(\boldsymbol{X}), \end{aligned} \tag{2}$$

where $\circ$ denotes the composition of two operators, h_i denotes the i-th transformation layer of GCN, and $g^{[L]}$ denotes the composition of L operators of g. The graph filters in GCN and SGC are computed by normalizing the adjacency matrix $\boldsymbol{A}$ as: $\boldsymbol{F} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$, where $\boldsymbol{D}$ is the degree matrix. Following many previous GNNs (Chen et al. 2020; Feng et al. 2020; Cui et al. 2020), we define the graph filter as $\boldsymbol{F} = (1-\pi)\boldsymbol{I} + \pi\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$, where $\pi \in (0, 1)$ and $\boldsymbol{I}$ is the identity matrix. In our experiments, we fix $\pi = 0.5$.

Graph Contrastive Learning (GCL) Given a graph dataset $\mathcal{D}$ of observations, the purpose of GCL is to learn the representations of graphs or nodes in an unsupervised way (You et al. 2020; Zhu et al. 2021; Hassani and Khasahmadi 2020; Zhu et al. 2020). A typical GCL model consists of three key modules (You et al. 2020; Zhu et al. 2021; Suresh et al. 2021): graph data augmentation (GDA), view encoders, and contrastive loss. In particular, GDA (*e.g.,* node or edge dropping) will generate different augmentations of the same observation; view encoders (*e.g.,* GCN (Kipf and Welling 2016)) will transform the augmented graphs into view representations; contrastive loss (*e.g.,* InfoNCE) will maximize the consistency between different views, and can identify the invariant parts of view representations.

Formally, for each observation s (*e.g.,* a node or a graph), GDA will generate a pair of augmented views $(a(s), a'(s))$. Then GCL uses two view encoder functions (f, f') to map the augmentation pairs to corresponding representations $(\mathbf{z}, \mathbf{z}')$=$(f(a(s)), f'(a'(s))$. Typically, the two view encoders in GCL have exactly the same neural architecture (Zhu et al. 2021; Hassani and Khasahmadi 2020; Zhu et al. 2020), *i.e.,* $f = f'$. Finally, InfoNCE loss (Hjelm et al. 2019) will be minimized to enforce a similar representation across positive pairs:

$$\mathcal{L} = -\sum_{i=1}^{N} \log \frac{\exp\left(-|\mathbf{z_i} - \mathbf{z_i'}|^2/2\right)}{\exp\left(-|\mathbf{z_i} - \mathbf{z_i'}|^2/2\right) + \sum_{j\neq i}\exp\left(-|\mathbf{z_i} - \mathbf{z_j}|^2/2\right)}, \tag{3}$$

where $(\mathbf{z_i}, \mathbf{z_i'})$ and $(\mathbf{z_i}, \mathbf{z_j})$ are the positive and negative pairs of view representations, respectively. As shown in (Jing et al. 2021), the above square loss form is equivalent to the typical cosine similarity form of normalized embeddings.

Methodology

To address the limitations of existing GCL methods, we will focus on manipulating the neural architectures of view encoders as model augmentations. In this section, we will first present three strategies and their benefits separately. Then we will illustrate the overall algorithm by incorporating the three tricks to a simple base model.

Asymmetric Strategy

One-sentence Summary Using encoders with shared parameters but different numbers of propagation layers can alleviate high-frequency noises.

Main Idea Contrastive learning (CL) can extract the information shared between different views (Tian et al. 2020), and thereby filter out the task-irrelevant noises that only appear in a single view. As shown in Fig. 1 where the scale of each area denotes the amount of information, the learned representations of CL include both task-relevant information (area D) and task-irrelevant noises (area C). Intuitively, the two views should not be too far (limited information of D) or too close (too much noises of C). We argue that the two views in previous GCL methods are too close to each other: (1) Typical GDA techniques (*e.g.,* edge or node dropping) cannot generate diverse enough augmentations while keeping the task-relevant information untouched; (2) The two view encoders have exactly the same neural architecture and tied parameters, which strengthens the closeness between views.

To address this problem, we propose to use asymmetric view encoders with shared parameters but different numbers of propagation layers. As shown in Fig. 1(b), we can push the two views away from each other by different numbers of propagation layers, and shared parameters can make sure that the distance between them will not be too far. In this way, the noises in GCL (area C) can be alleviated. Note

(a) Symmetric (b) Asymmetric

Figure 1: An illustration of our motivation about the asymmetric strategy. Here V_1, V_2 and $\mathcal{Y}$ represent the information of two views and downstream tasks, respectively. The strategy can push the two views away from each other, and the task-irrelevant noises in GCL (area C) can be alleviated.

that the numbers of h and g operators are totally detached. For implementation, we employ two GNN encoders with the same number of h operators but different numbers of g operators. Now we will show that our implementation can help filter out high-frequency noises.

Benefits For simplicity, we ignore the non-linear transformations in GNN encoders to demonstrate our benefits. Without non-linear functions, the weight matrices in h operators collapse to a single weight matrix $\boldsymbol{W}$. Then we have the embeddings of two encoders as $\boldsymbol{Z} = \boldsymbol{F}^L\boldsymbol{X}\boldsymbol{W}$, $\boldsymbol{Z}' = \boldsymbol{F}^{L'}\boldsymbol{X}\boldsymbol{W}$, where L, L' are the numbers of g operators in two view encoders, and $L < L'$.

Assume that graph filter $\boldsymbol{F}$ can be factorized to $\boldsymbol{U}\Lambda\boldsymbol{U}^T$ by eigenvector decomposition, where $\boldsymbol{U}$ is the unitary matrix of eigenvectors and $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{|\mathcal{V}|})$ is the diagonal matrix of eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_{|\mathcal{V}|}$. A larger eigenvalue of $\boldsymbol{F}$ corresponds to a smaller eigenvalue of graph Laplacian matrix, which is often recognized as more important information for downstream tasks (Nt and Maehara 2019). Here we focus on analyzing graph signals and assume one-hot node features $\boldsymbol{X} = \boldsymbol{I}$. Then the embeddings of two views can be written as $\boldsymbol{Z} = \boldsymbol{U}\Lambda^L\boldsymbol{U}^T\boldsymbol{W}$, $\boldsymbol{Z}' = \boldsymbol{U}\Lambda^{L'}\boldsymbol{U}^T\boldsymbol{W}$, where $\boldsymbol{W} \in \mathbb{R}^{|\mathcal{V}|\times d_O}$.

Contrastive loss aims to minimize the distance between embeddings of a positive pair. Following this idea, we focus on the numerator term in Eq. (3) and rewrite the loss as

$$\min_{\boldsymbol{W}} \sum_{i=1}^{|\mathcal{V}|} |\boldsymbol{z}_i - \boldsymbol{z}_i'|^2 = \min_{\boldsymbol{W}} \text{tr}((\boldsymbol{Z} - \boldsymbol{Z}')(\boldsymbol{Z} - \boldsymbol{Z}')^T), \quad \text{subject to} \quad \boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}, \tag{4}$$

where $\text{tr}(\cdot)$ denotes matrix trace, and the condition $\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}$ is to avoid trivial all-zero solutions.

Theorem 1. *The optimal $\boldsymbol{W}$ of the contrastive learning loss in Eq. (4) is*

$$\boldsymbol{W}^* = \boldsymbol{U}[k_1, k_2, \ldots k_{d_O}], \tag{5}$$

where $1 \leq k_1 < k_2 < \ldots k_{d_O} \leq |\mathcal{V}|$ are the best ks that minimize $(\lambda_k^L - \lambda_k^{L'})^2$, and $U[k_1, k_2, \ldots k_{d_O}]$ denotes the corresponding columns of $\boldsymbol{U}$.

The detailed proof of Theorem 1 is presented in Appendix.**A**. Now we will see which ks have the smallest $(\lambda_k^L - \lambda_k^{L'})^2$. The function $\min_\lambda(\lambda^L - \lambda^{L'})^2$ prefers $\lambda \to 0$ or $\lambda \to 1$. Taking the graph filter $\boldsymbol{F} = \frac{1}{2}\boldsymbol{I} + \frac{1}{2}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$ used in our experiments as an example, the eigenvalues of $\boldsymbol{F}$ fall in range $[0, 1]$. As shown in (Cui et al. 2020), for popular graph datasets such as Cora and Citeseer (Kipf and Welling 2016), the eigenvalues of $\boldsymbol{F}$ will fall in range $[\frac{1}{4}, 1]$ in practice. Besides, we also find that the filter $\boldsymbol{F}$ of these popular graph data has many eigenvalues close to 1, *e.g.,* the 100-th largest eigenvalue of $\boldsymbol{F}$ in Cora is still larger than 0.998. Therefore, $\boldsymbol{U}[k_1, k_2, \ldots k_{d_O}]$ is likely to be the first d_O columns of $\boldsymbol{U}$ in practice. Then the representations of view encoder are

$$\boldsymbol{Z} = \boldsymbol{U}\Lambda^L\boldsymbol{U}^T\boldsymbol{W} = \boldsymbol{U}\text{diag}(\lambda_1^L, \lambda_2^L, \ldots, \lambda_{d_O}^L, 0, 0, \ldots, 0), \tag{6}$$

which can perfectly filter out the high-frequency noises.

In contrast, if we adopt identical view encoders with $L = L'$, then $\boldsymbol{Z} - \boldsymbol{Z}' = \boldsymbol{U}(\Lambda^L - \Lambda^{L'})\boldsymbol{U}^T\boldsymbol{W} = 0$. Thus, the perturbations of GDA are no longer minor terms and have to be considered. Assume that the influence of GDA on eigenvalues is $\epsilon \in \mathbb{R}^{|\mathcal{V}|}$, we have $(\Lambda + \text{diag}(\epsilon))^L - \Lambda^L \approx L\text{diag}(\epsilon)\Lambda^{L-1}$. Thus the optimal $\boldsymbol{W}^*$ is determined by both filter $\boldsymbol{F}$ and the perturbations of GDA, which will probably incorporate high-frequency noises into learned representations.

Random Strategy

One-sentence Summary Randomly varying the number of propagation operators in view encoders at every epoch can help enrich training samples.

Main Idea Without loss of generality, we illustrate our idea based on SGC (Wu et al. 2019) encoder, *i.e.,* $f(\boldsymbol{X}) = h \circ g^{[L]}(\boldsymbol{X})$. During the training process of GCL, we propose to vary the number of operator g in the encoder f. Specifically, we randomly sample $\hat{L} \sim \text{Uniform}(lower, upper)$ in every epoch, instead of employing a fixed L. The intuition behind is that varying the propagation depth can enrich the diversity of training instances, and hence help predict downstream tasks. Now we will present the benefits more formally.

Benefits As a GNN encoder with L propagation operators of g, f will compute node representation $\boldsymbol{Z}_v \in \boldsymbol{Z}$ for every node v by aggregating L-hop neighbors of node v. In other words, f is equivalent to a function applied to the local computation tree (Nt and Maehara 2019) rooted at node v, with a depth of L.

Let $\mathcal{S} = \{v\}_{v=1}^{|\mathcal{V}|}$ be the training set of GCL. Then we can replace the observations in $\mathcal{S}$ from the individual node v to its L-hop computation tree t_v^L, and rewrite the training set as $\tilde{\mathcal{S}}^L = \{t_v^L\}_{v=1}^{|\mathcal{V}|}$. When the number of g operators is randomly chosen from $\hat{L} \sim \text{Uniform}(lower, upper)$ in every epoch, we can hypothesize that the set of samples changes to $\tilde{\mathcal{S}} = \bigcup_{\hat{L}\in[lower,upper]} \tilde{\mathcal{S}}^{\hat{L}}$. Therefore, the training set of GCL gets enlarged by several times. In fact, if we further take GDA (*e.g.,* edge dropping) into consideration, the training set will involve the computation trees and their sub-trees. There could be an even wider gap (*e.g.,* exponentially) between the set capacity with random $\hat{L}$ and that with fixed L.

Shuffling Strategy

One-sentence Summary Shuffling the permutation of propagation and transformation operators in every epoch brings safer augmentations.

Main Idea Existing GDA techniques usually randomly perturb the graph topology and will take the risk of destroying key connections (*e.g.,* dropping an important chemical bond) related to downstream tasks. To address this problem, we propose to use different permutations of operators g and h in two view encoders as safer augmentations. Formally, if view encoder f has L operators of g and N operators of h, then f can be written as

$$f(\boldsymbol{X}) = h_N \circ g^{[K_N]} \circ h_{N-1} \circ \dots g^{[K_2]} \circ h_1 \circ g^{[K_1]}(\boldsymbol{X}), \quad (7)$$

where $K_1 \dots K_N \geq 0$ and $\sum_{i=1}^{N} K_i = L$. Then we will use a different set of $K'_1, K'_2 \dots K'_N \geq 0$ with $\sum_{i=1}^{N} K'_i = L$ for another view encoder f'. The intuition behind is that shuffling the order of propagation and transformation operators will not change the semantics of an input graph, but will perturb the encoded representations as safer augmentations. Now we will discuss more about the relationship between our strategy and previous GDA techniques.

Benefits As shown in (Wang and Isola 2020), contrastive learning includes an important step of alignment, whose metric is as follows:

$$\mathcal{L}_{align} \triangleq \mathbb{E}_{x \sim \mathcal{D}, x' \sim p_{pos}(x)}[||f_\theta(x) - f'_{\theta'}(x')||_2], \quad (8)$$

where $p_{pos}(x)$ is the distribution of data augmentation, and θ, θ' denote the learnable parameters in encoders f, f'.

Previous GCL methods based on data augmentation (Zhu et al. 2021; You et al. 2020) can be interpreted as $f' = f$ and $\theta' = \theta$ in Eq. (8). Those based on perturbing model parameters (Yang, Zhang, and Yang 2021; Xia et al. 2022) can be interpreted as $f' = f$, $x' = x$ and $\theta' = \theta + \mathcal{N}$ where $\mathcal{N}$ is random perturbation noise (*e.g.*, drawn from a Gaussian distribution). Note that the function of encoded semantics with respect to model parameters is rather complex. Hence the parameter perturbation drawn from a simple distribution may also harm the semantics in encoded representations. While in this strategy, we have $\theta' = \theta$ and $x' = x$. Note that our f and f' will be the same if we ignore all the non-linear transformations. Therefore, f' can provide safer augmentations by utilizing the perturbations brought by non-linear transformations.

Implementation of MA-GCL

Now we will present our MA-GCL by applying the three strategies on a simple base model. The base model adopts random edge/feature dropping, graph filter $\boldsymbol{F} = \frac{1}{2}\boldsymbol{I} + \frac{1}{2}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$, 2-layer embedding projector, and InfoNCE loss for learning. Note that the base model can be seen as a simplified version of GRACE (Zhu et al. 2020) with only intra-view modeling. The pseudo code is shown in Alg. 1. After the training phase, we will use a fixed encoder architecture and drop the embedding projector for evaluation: similar to the architecture of GCN, we set $K_1 = K_2 = \dots K_N = K$, where $K \in \{1, 2\}$ is a hyper-parameter.

Algorithm 1: Implementation of MA-GCL

Input: Adjacency matrix $\boldsymbol{A}$; feature matrix $\boldsymbol{X}$; GDA distribution $\mathcal{T}$; Random range $[low, high]$; N transformation operators $h_1, h_2 \dots h_N$; Embedding projector proj$(\cdot)$;

Output: N learned operators $h_1, h_2 \dots h_N$.

for *every epoch* **do**

- Draw two graph augmentations: $a, a' \sim \mathcal{T}$, $(\boldsymbol{A}_1, \boldsymbol{X}_1) = a(\boldsymbol{A}, \boldsymbol{X}), (\boldsymbol{A}_2, \boldsymbol{X}_2) = a'(\boldsymbol{A}, \boldsymbol{X})$;
- Draw K_i, K'_i from Uniform$(low, high)$ for $i = 1 \dots N$;
- Assure $L = \sum_{i=1}^{N} K_i \neq \sum_{i=1}^{N} K'_i = L'$, and $\forall\, i$ $K_i \neq K'_i$;
- Calculate graph filters in g_1, g_2: $\boldsymbol{F}_1 = \frac{1}{2}\boldsymbol{I} + \frac{1}{2}\boldsymbol{D_1}^{-\frac{1}{2}}\boldsymbol{A_1}\boldsymbol{D_1}^{-\frac{1}{2}}$, $\boldsymbol{F}_2 = \frac{1}{2}\boldsymbol{I} + \frac{1}{2}\boldsymbol{D_2}^{-\frac{1}{2}}\boldsymbol{A_2}\boldsymbol{D_2}^{-\frac{1}{2}}$;
- Augmented view encoders: $f_1 = h_N \circ g_1^{[K_N]} \dots h_1 \circ g_1^{[K_1]}$, $f_2 = h_N \circ g_2^{[K'_N]} \dots h_1 \circ g_2^{[K'_1]}$;
- View encoding: $\boldsymbol{Z}_1 = f_1(\boldsymbol{X_1}), \boldsymbol{Z}_2 = f_2(\boldsymbol{X_2})$
- Update parameters by contrastive loss $\mathcal{L}(\text{proj}(\boldsymbol{Z}_1), \text{proj}(\boldsymbol{Z}_2))$;

end

Experiments

In this section, we conduct experiments on node classification task[2] to show the effectiveness of MA-GCL. We start by introducing the experimental setup. Then we present the performance of MA-GCL against SOTA GCL methods on graph benchmarks. We also demonstrate the advantages of each proposed strategy and verify our motivation by extensive experiments.

Experimental Setup

Datasets We evaluate our approach on six benchmark datasets of node classification, which have been widely used in previous GCL methods. Specifically, citation datasets include Cora, CiteSeer and PubMed (Kipf and Welling 2016), co-purchase and co-author datasets include Amazon-Photo, Amazon-Computers and Coauthor-CS (Shchur et al. 2018).

Evaluation protocol Following the protocol and implementation of GCA (Zhu et al. 2021), we will learn node representations by different GCL methods in an unsupervised manner, and then train the same linear classifier as post-processing for evaluation. We will report classification accuracy as evaluation metric. For three citation datasets, we evaluate the models on the public splits (Kipf and Welling 2016). For co-purchase and co-author datasets, we randomly split the datasets, where 10%, 10%, and the rest 80% of nodes

[2]We will explore graph classification and node clustering tasks in Appendix.**B**. The appendix also includes details of datasets, environment and hyper-parameter settings. The results on hyper-parameter sensitivity and model efficiency are provided as well.

Datasets	Cora	CiteSeer	PubMed	Coauthor-CS	Amazon-C	Amazon-P	Avg. Acc.	Avg. Rank
GCN	82.5 ± 0.4	71.2 ± 0.3	79.2 ± 0.3	93.03 ± 0.3	86.51 ± 0.5	92.42 ± 0.2		
GAT	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3	92.31 ± 0.2	86.93 ± 0.3	92.56 ± 0.4	-	-
InfoGCL	83.5 ± 0.3	73.5 ± 0.4	79.1 ± 0.2	-	-	-		
DGI	82.3 ± 0.6	71.8 ± 0.7	76.8 ± 0.3	92.15 ± 0.6	83.95 ± 0.5	91.61 ± 0.2	83.10	8.5
GRACE	81.7 ± 0.4	71.5 ± 0.5	80.7 ± 0.4	92.93 ± 0.0	87.46 ± 0.2	92.15 ± 0.2	84.44	6.5
MVGRL	83.4 ± 0.3	73.0 ± 0.3	80.1 ± 0.6	92.11 ± 0.1	87.52 ± 0.1	91.74 ± 0.0	84.63	6.5
BGRL	81.7 ± 0.5	72.1 ± 0.5	80.2 ± 0.4	93.01 ± 0.2	88.23 ± 0.3	92.57 ± 0.3	84.63	6.5
GCA	83.4 ± 0.3	72.3 ± 0.1	80.2 ± 0.4	93.10 ± 0.0	87.85 ± 0.3	92.53 ± 0.2	84.89	4.0
SimGRACE	77.3 ± 0.1	71.4 ± 0.1	78.3 ± 0.3	93.45 ± 0.4	86.04 ± 0.2	91.39 ± 0.4	82.98	8.5
COLES	81.2 ± 0.4	71.5 ± 0.2	80.4 ± 0.7	92.65 ± 0.1	79.64 ± 0.0	89.00 ± 0.5	82.40	8.8
ARIEL	82.5 ± 0.1	72.2 ± 0.2	80.5 ± 0.3	93.35 ± 0.0	88.27 ± 0.2	91.43 ± 0.2	84.71	4.8
CCA-SSG	**83.9 ± 0.4**	73.1 ± 0.3	81.3 ± 0.4	93.37 ± 0.2	88.42 ± 0.3	92.44 ± 0.1	85.42	2.3
Base Model	81.1 ± 0.4	71.4 ± 0.1	79.1 ± 0.4	92.86 ± 0.3	87.65 ± 0.2	91.19 ± 0.3	83.88	9.0
MA-GCL	83.3 ± 0.4	**73.6 ± 0.1**	**83.5 ± 0.4**	**94.19 ± 0.1**	**88.83 ± 0.3**	**93.80 ± 0.1**	**86.20**	**1.2**

Table 1: Performance of node classification. Here we use public splits on Cora, Citeseer and PubMed.

Datasets	Cora	CiteSeer	PubMed
GRACE	83.4 ± 1.0	69.5 ± 1.4	83.1 ± 0.6
GCA	82.6 ± 0.9	72.2 ± 0.7	83.4 ± 0.4
SimGRACE	77.2 ± 0.6	71.4 ± 0.6	81.8 ± 0.7
COLES	82.5 ± 0.9	72.3 ± 0.6	84.7 ± 0.2
ARIEL	84.3 ± 1.0	72.7 ± 1.1	81.6 ± 0.5
CCA-SSG	83.9 ± 0.7	72.7 ± 0.8	84.2 ± 0.2
Base Model	82.2 ± 0.7	70.7 ± 0.8	82.1 ± 0.5
MA-GCL	**84.8 ± 0.4**	**73.3 ± 0.3**	**85.7 ± 0.3**

Table 2: Performance on Cora, Citeseer and PubMed under random splits.

are selected for the training, validation and test set, respectively (Zhu et al. 2021; Zhang et al. 2021). For each dataset, we report the average accuracy and standard deviation over 5 runs in different random seeds. Note that random seeds will also change the splits in co-purchase and co-author datasets.

For MA-GCL, we apply our strategies to the base model described in previous section. We use two h operators and multiple g operators in view encoder f for all datasets. We evaluate our model with $K_1 = K_2 = 2$ for Cora and CiteSeer, and $K_1 = K_2 = 1$ for other datasets. More details about hyper-parameter settings are provided in Appendix.**C**.

Baselines We consider a number of node representation learning baselines including very recent SOTA GCL methods. Baselines trained without labels: DGI (Velickovic et al. 2019), GRACE (Zhu et al. 2020), MVGRL (Hassani and Khasahmadi 2020), BGRL (Thakoor et al. 2021), GCA (Zhu et al. 2021), COLES (Zhu, Sun, and Koniusz 2021), CCA-SSG (Zhang et al. 2021), Ariel (Feng et al. 2022a) and SimGRACE (Xia et al. 2022). Baselines trained with labels: GCN (Kipf and Welling 2016), GAT (Velickovic et al. 2017) and InfoGCL (Xu et al. 2021). Note that SimGRACE is a graph classification method, we use its variant for node classification tasks by changing GIN (Xu et al. 2018) encoders to GCN. All the baselines run with the same evaluation protocol.

Comparison with Baseline Methods

We report the performance of node classification in Table 1. We **bold** the best method trained without labels in each column, and underline the best performed baseline. We can see that MA-GCL can achieve SOTA performance on 5 out of 6 graph benchmarks, and the relative improvement can go up to 2.7%. Considering that the public splits on Cora, Citeseer and PubMed might not be representative, we also investigate another benchmark setting (Feng et al. 2022a) with random splits on these three datasets, and compare with the most competitive baselines. As shown in Table 2, MA-GCL consistently outperforms baseline methods, which demonstrates the effectiveness of MA-GCL.

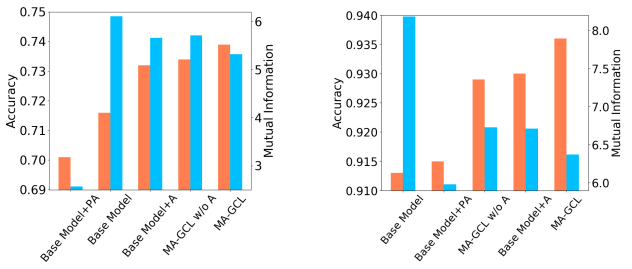
Ablation Studies

We conduct ablation experiments to prove the effectiveness of each model augmentation strategy: **A**symmetric, **R**andom and **S**huffling. We compare the full model (3 strategies) with base model (0 strategy) and six ablated models (1 or 2 strategies). A model with or without asymmetric strategy determines whether two view encoders have the same number of g operators; a model with or without random strategy determines whether the encoder architectures are fixed or randomized in every epoch; a model with or without shuffling strategy determines whether two view encoders have different operator permutations. Note that the asymmetric strategy indicates that the permutations of two view encoders cannot be exactly the same. Thus for models with asymmetric strategy but without shuffling strategy, we will force $K_i = K'_i$ for $i < N$. The results are reported in Table 3.

In summary, if an ablated model is further combined with asymmetric/random/shuffling strategy, its accuracy can be respectively improved by 0.86/0.65/0.54% on average. Hence all three strategies have positive effects on the overall performance of MA-GCL, and the asymmetric strategy is the

Datasets	Cora	CiteSeer	PubMed	Coauthor-CS	Amazon-C	Amazon-P	Avg. Acc.	Avg. Rank
Base Model+A	82.5 ± 0.5	73.1 ± 0.4	82.3 ± 0.3	93.41 ± 0.0	88.13 ± 0.2	93.10 ± 0.1	85.42	4.2
Base Model+R	82.4 ± 0.3	72.5 ± 0.1	82.8 ± 0.4	93.12 ± 0.1	88.06 ± 0.1	92.26 ± 0.3	85.19	6.3
Base Model+S	81.9 ± 0.2	72.7 ± 0.2	81.8 ± 0.1	93.13 ± 0.0	87.94 ± 0.2	92.66 ± 0.2	85.02	6.5
MA-GCL w/o A	82.5 ± 0.6	73.3 ± 0.4	82.9 ± 0.4	93.31 ± 0.1	88.09 ± 0.2	92.86 ± 0.3	85.49	4.3
MA-GCL w/o R	83.3 ± 0.6	73.0 ± 0.0	83.1 ± 0.4	93.47 ± 0.0	88.67 ± 0.2	93.41 ± 0.2	85.83	2.8
MA-GCL w/o S	82.7 ± 0.3	73.1 ± 0.4	83.3 ± 0.5	93.92 ± 0.1	88.34 ± 0.4	93.01 ± 0.0	85.73	2.8
Base Model	81.1 ± 0.4	71.4 ± 0.1	79.1 ± 0.4	92.86 ± 0.3	87.65 ± 0.2	91.19 ± 0.3	83.88	8.0
MA-GCL	83.3 ± 0.4	73.6 ± 0.1	83.5 ± 0.4	94.19 ± 0.1	88.83 ± 0.3	93.80 ± 0.1	86.20	1.0

Table 3: Ablation Studies of MA-GCL. Base Model (0 strategy), Base Model+X (1 strategy), MA-GCL w/o X (2 strategies), and MA-GCL (3 strategies).



(a) CiteSeer (b) Amazon-Photo

Figure 2: Experimental results for motivation verification. Orange columns (estimation of area D in Fig. 1) denote the classification accuracies of different methods; Blue columns (estimation of area C+D in Fig. 1) denote the mutual information between two views. A method with higher orange column and lower blue column indicates that more task-relevant knowledge can be encoded via CL with less irrelevant noises. The other four datasets are in Appendix.**B**.

most effective one among the three. Also note that the performance of our base model is weaker than that of SOTA GCL baselines, which validates the superiority of our paradigm of model augmentation.

Motivation Verification

Now we will conduct experiments to verify our motivation that (1) typical GDA techniques alone (*e.g.*, edge dropping) cannot generate diverse enough augmentations, and thus will bring much noise (area C in Fig. 1) into learned representations; (2) directly perturbing model parameters may harm the semantics (area D in Fig. 1) in encoded representations.

Setup We propose to estimate the information in area C and D for different GCL methods. According to the InfoMin principle (Tian et al. 2020), a method is better if it has a larger area D (*i.e.,* task-relevant information) and a smaller area C (*i.e.,* task-irrelevant noises). However, it is hard to directly estimate the information in area C or D alone. Thus we employ MINE (Belghazi et al. 2018) to compute the mutual information between two views, as an estimation of **area C+D**. Given the representation trained by CL, we will use its accuracy on downstream tasks as an estimation of **area D**. For a fair comparison, we investigate five methods based on our backbone model: **Base Model** represents typical GCL methods with random data augmentations; **Base Model+PA** adds parameter perturbations to the base model, which represents a recent paradigm (Xia et al. 2022) of parameter augmentation (PA); **Base Model+A**, **MA-GCL w/o A** and **MA-GCL** are employed to demonstrate our effectiveness.

Results As shown in Fig. 2, we have the following observations: (1) Base Model always has the highest mutual information but the worst two accuracies, which supports our statement that views in typical GCL methods are too close to each other and bring much noises; (2) Base Model+PA has low accuracies and mutual information, which shows that perturbing model parameters will significantly harm the semantics in encoded representations; (3) By comparing Base Model against Base Model+A, and MA-GCL w/o A against MA-GCL, we can see that the asymmetric strategy can effectively push two views away from each other, result in a significant decrease of blue columns, and thereby reduce task-irrelevant noises. The above observations validate our motivation and strategy design.

Conclusion

In this paper, we highlight a key limitation of previous GCL methods that their contrastive views are too close to effectively filter out noises. Then we propose a novel paradigm named model augmentation, which focuses on manipulating the neural architectures of GNN view encoders instead of perturbing graph inputs or model parameters. Specifically, we present three model augmentation tricks for GCL: *asymmetric*, *random* and *shuffling*, which can respectively help alleviate high-frequency noises, enrich training instances and bring safer augmentations. With the three tricks, two contrastive views can keep a proper distance from each other, *i.e.*, neither too far (losing task-relevant semantics) nor too close (introducing unnecessary noises). Experiments show that MA-GCL can achieve SOTA performance by applying the three tricks on a simple base model as a plug-and-play component. We hope this work can provide a new direction of GCL, and future work may consider generalize the idea of model augmentation to graphs with heterophily.

Acknowledgments.

This work is supported in part by the National Natural Science Foundation of China (No. U20B2045, 62192784, 62172052, 62002029, 62006129, 62172052, U1936014).

References

Belghazi, M. I.; Baratin, A.; Rajeshwar, S.; Ozair, S.; Bengio, Y.; Courville, A.; and Hjelm, D. 2018. Mutual information neural estimation. In *ICML*.

Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; and Li, Y. 2020. Simple and deep graph convolutional networks. In *ICML*.

Cui, G.; Zhou, J.; Yang, C.; and Liu, Z. 2020. Adaptive graph encoder for attributed graph embedding. In *KDD*.

Duan, H.; Vaezipoor, P.; Paulus, M. B.; Ruan, Y.; and Maddison, C. 2022. Augment with Care: Contrastive Learning for Combinatorial Problems. In *ICML*.

Feng, S.; Jing, B.; Zhu, Y.; and Tong, H. 2022a. Adversarial graph contrastive learning with information regularization. In *WWW*.

Feng, W.; Dong, Y.; Huang, T.; Yin, Z.; Cheng, X.; Kharlamov, E.; and Tang, J. 2022b. GRAND+: Scalable Graph Random Neural Networks. In *WWW*.

Feng, W.; Zhang, J.; Dong, Y.; Han, Y.; Luan, H.; Xu, Q.; Yang, Q.; Kharlamov, E.; and Tang, J. 2020. Graph random neural networks for semi-supervised learning on graphs. *NeurIPS*.

Grill, J.-B.; Strub, F.; Altché, F.; Tallec, C.; Richemond, P.; Buchatskaya, E.; Doersch, C.; Avila Pires, B.; Guo, Z.; Gheshlaghi Azar, M.; et al. 2020. Bootstrap your own latent-a new approach to self-supervised learning. *NeurIPS*.

Hassani, K.; and Khasahmadi, A. H. 2020. Contrastive multi-view representation learning on graphs. In *ICML*.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 9729–9738.

Hjelm, R. D.; Fedorov, A.; Lavoie-Marchildon, S.; Grewal, K.; Bachman, P.; Trischler, A.; and Bengio, Y. 2019. Learning deep representations by mutual information estimation and maximization. *ICLR*.

Jing, L.; Vincent, P.; LeCun, Y.; and Tian, Y. 2021. Understanding dimensional collapse in contrastive self-supervised learning. *ICLR*.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *ICLR*.

Li, S.; Wang, X.; Zhang, A.; Wu, Y.; He, X.; and Chua, T.-S. 2022. Let Invariant Rationale Discovery Inspire Graph Contrastive Learning. In *ICML*.

Nt, H.; and Maehara, T. 2019. Revisiting graph neural networks: All we have is low-pass filters. *NeurIPS*.

Qiu, J.; Chen, Q.; Dong, Y.; Zhang, J.; Yang, H.; Ding, M.; Wang, K.; and Tang, J. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *KDD*.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

Soatto, S.; and Chiuso, A. 2016. Modeling Visual representations: Defining properties and deep approximations. *ICLR*.

Sun, F.-Y.; Hoffman, J.; Verma, V.; and Tang, J. 2020. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *ICLR*.

Suresh, S.; Li, P.; Hao, C.; and Neville, J. 2021. Adversarial graph augmentation to improve graph contrastive learning. *NeurIPS*.

Thakoor, S.; Tallec, C.; Azar, M. G.; Munos, R.; Veličković, P.; and Valko, M. 2021. Bootstrapped representation learning on graphs. In *ICLR 2021 Workshop on Geometrical and Topological Representation Learning*.

Tian, Y.; Sun, C.; Poole, B.; Krishnan, D.; Schmid, C.; and Isola, P. 2020. What makes for good views for contrastive learning? *NeurIPS*.

TISHBY, N. 1999. The information bottleneck method. In *Proc. 37th Annual Allerton Conference on Communications, Control and Computing*.

Tong, Z.; Liang, Y.; Ding, H.; Dai, Y.; Li, X.; and Wang, C. 2021. Directed Graph Contrastive Learning. *NeurIPS*.

Van den Oord, A.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv e-prints*.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *ICLR*.

Velickovic, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep Graph Infomax. *ICLR*.

Wang, M.; Yu, L.; Da Zheng, Q. G.; Gai, Y.; Ye, Z.; Li, M.; Zhou, J.; Huang, Q.; Ma, C.; et al. 2019. Deep Graph Library: Towards efficient and scalable deep learning on graphs.(2019). *arXiv preprint arXiv:1909.01315*.

Wang, T.; and Isola, P. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*.

Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. 2019. Simplifying graph convolutional networks. In *ICML*.

Xia, J.; Wu, L.; Chen, J.; Hu, B.; and Li, S. Z. 2022. SimGRACE: A Simple Framework for Graph Contrastive Learning without Data Augmentation. In *WWW*.

Xu, D.; Cheng, W.; Luo, D.; Chen, H.; and Zhang, X. 2021. Infogcl: Information-aware graph contrastive learning. *NeurIPS*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How Powerful are Graph Neural Networks? In *ICLR*.

Yang, H.; Chen, H.; Pan, S.; Li, L.; Yu, P. S.; and Xu, G. 2022. Dual Space Graph Contrastive Learning. *WWW*.

Yang, L.; Zhang, L.; and Yang, W. 2021. Graph Adversarial Self-Supervised Learning. *NeurIPS*.

You, Y.; Chen, T.; Shen, Y.; and Wang, Z. 2021. Graph contrastive learning automated. In *ICML*.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph contrastive learning with augmentations. *NeurIPS*.

Yu, J.; Yin, H.; Xia, X.; Chen, T.; Cui, L.; and Nguyen, Q. V. H. 2022. Are graph augmentations necessary? Simple graph contrastive learning for recommendation. In *SIGIR*.

Yuan, J.; Yu, H.; Cao, M.; Xu, M.; Xie, J.; and Wang, C. 2021. Semi-Supervised and Self-Supervised Classification with Multi-View Graph Neural Networks. In *CIKM*.

Zhang, H.; Wu, Q.; Yan, J.; Wipf, D.; and Yu, P. S. 2021. From canonical correlation analysis to self-supervised graph neural networks. *NeurIPS*.

Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2020. Graph neural networks: A review of methods and applications. *AI Open*.

Zhu, H.; Sun, K.; and Koniusz, P. 2021. Contrastive laplacian eigenmaps. *NeurIPS*.

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

Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; and Wang, L. 2021. Graph contrastive learning with adaptive augmentation. In *WWW*.