# Community-Invariant Graph Contrastive Learning

**Shiyin Tan** [*1] **Dongyuan Li** [*1] **Renhe Jiang** [†2] **Ying Zhang** [3] **Manabu Okumura** [1]

## Abstract

Graph augmentation has received great attention in recent years for graph contrastive learning (GCL) to learn well-generalized node/graph representations. However, mainstream GCL methods often favor randomly disrupting graphs for augmentation, which shows limited generalization and inevitably leads to the corruption of high-level graph information, i.e., the graph community. Moreover, current knowledge-based graph augmentation methods can only focus on either topology or node features, causing the model to lack robustness against various types of noise. To address these limitations, this research investigated the role of the graph community in graph augmentation and figured out its crucial advantage for learnable graph augmentation. Based on our observations, we propose a community-invariant GCL framework to maintain graph community structure during learnable graph augmentation. By maximizing the spectral changes, this framework unifies the constraints of both topology and feature augmentation, enhancing the model's robustness. Empirical evidence on 21 benchmark datasets demonstrates the exclusive merits of our framework. Code is released on Github [https://github.com/CI-GCL.git].

## 1. Introduction

Graph representation learning on graph-structured data, such as molecules and social networks, has become one of the hottest topics in AI (Cao et al., 2023). Typical GNNs (Kipf & Welling, 2017) require large-scale task-specific labels, which are expensive and labor-intensive to collect. To alleviate this, graph contrastive learning (GCL) has been proposed as one of the most successful graph representation learning methods, drawing a lot of attention (Li et al., 2022b). The main goal of GCL is to maximize the agreement of node representations between two augmented views to capture graph invariance information (Tian et al., 2020). Among various GCL variations, effective graph augmentation turns out to be the bread and butter for achieving success (Wei et al., 2023). Early studies almost adopt random graph augmentation, such as randomly dropping edges or masking features (You et al., 2020). Researchers also attempt to incorporate expert knowledge to guide graph augmentation. For instance, GCA (Zhu et al., 2021) and MoCL (Sun et al., 2021) use network science or biomedical knowledge to constrain edge dropping probabilities. However, such random or knowledge-based graph augmentations are sensitive to different datasets (Shen et al., 2023) and may yield suboptimal performance (Yin et al., 2022).

To achieve better generalization and globally optimal performance, learnable graph augmentation is proposed to automatically disrupt redundant information as much as possible to share minimal yet sufficient core information between augmented views (Tong et al., 2021; Suresh et al., 2021). Although they have achieved great success, there still remain two open challenges worth exploring. (1) Community structure plays a crucial role in various downstream tasks, such as node classification and link prediction (Li et al., 2022a; Chen et al., 2023b). However, current GCL methods often randomly disrupt graphs during graph augmentation, which inevitably leads to the corruption of high-level graph information (i.e., community) and limits the generalization (Chiplunkar et al., 2018). (2) Current constraints employed in learnable graph augmentation methods primarily focus either on topology or node features (Li et al., 2022b). For instance, GAME (Wei et al., 2023) and GCL-SPAN (Lin et al., 2023) use spectrum-based constraints for topology augmentation. Due to the asymmetry of the feature matrix, their methods cannot be extended to feature augmentation. On the other hand, COSTA (Zhang et al., 2022) designs a covariance-preserving constraint for feature augmentation, which, however, lacks effectiveness in topology augmentation. By solely focusing on one type of graph augmentation (topology or feature), models may not fully exploit all available information and lack robustness against different types of noise (Liu et al., 2022b).

[*]Equal contribution with order determined by flipping a coin. [†] Corresponding Author. [1]Tokyo Institute of Technology [2]The University of Tokyo [3]RIKEN. Correspondence to: Renhe Jiang <jiangrh@csis.u-tokyo.ac.jp>. Ying Zhang conducted this research when she was worked at the Tokyo Institute of Technology.

*Proceedings of the 41st International Conference on Machine Learning*, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

To solve the aforementioned issues, we propose a general learnable Community-Invariant GCL framework (**CI-GCL**), which unifies constraints from both topology and feature augmentation to maintain CI for learnable graph augmentation. Specifically, when considering topology augmentation with a certain degree of disruption, we observe a nearly negative correlation between community and spectral changes (see Sec 4.1). Therefore, to maximize the topology perturbation while ensuring community invariance, we can simply maximize graph spectral changes during topology augmentation. To extend our CI constraint to feature augmentation, we convert the feature matrix into a symmetric bipartite feature matrix based on the bipartite graph co-clustering technique (Zhang et al., 2023a). This approach converts feature augmentation into bipartite feature augmentation, while elucidating the importance of features in maintaining community structure. For bipartite feature augmentation, we also observed a negative relationship between community and spectral changes, which is consistent with topology augmentation. This motivates us to apply our CI constraint to feature augmentation by maximizing graph spectral changes during bipartite feature augmentation. To summarize, the contributions of this research are:

- We propose a learnable CI-GCL framework to automatically maintain CI during graph augmentation by maximizing spectral change loss, improving the model's downstream performances.
- We theoretically show that the proposed CI constraint can be applied to both topology and feature augmentation, enhancing the model's robustness.
- Experiments on 21 widely used benchmarks demonstrate the effectiveness and robustness of CI-GCL.

## 2. Related Work

As an effective self-supervised learning paradigm, contrastive learning has achieved great success to learn text or image representations (Chen et al., 2020; Zhang et al., 2020). DGI (Velickovic et al., 2019) first adopted contrastive learning to learn robust graph representations that are invariant to various noise and operations. However, unlike Euclidean or sequential data, graphs are irregular non-Euclidean data and sensitive to minor structural augmentation (Shen et al., 2023), resulting in learned graph representations being ineffective. Given that among many GCL variations, graph augmentation shows its crucial advantage for graph representation learning, many studies attempted to investigate effective graph augmentation for GCL. Prior GCL almost adopts random graph augmentation. For example, GRACE (Zhu et al., 2020) firstly uses random edge dropping and feature masking as graph augmentations. After that, GraphCL (You et al., 2020) gives an extensive study on different combinations of graph augmentations including randomly node dropping, edge perturbation, subgraph sampling, and feature masking. To make GraphCL more flexible, JOAO (You et al., 2021) automatically selects the combination of different random graph augmentations. Due to the limited generalization of random augmentation, researchers start to incorporate expert knowledge as constraints for graph augmentation. For instance, Duan et al. (2022) and GraphAug (Luo et al., 2023) employ label-invariance between original and augmented views as constraints, which achieves great success in the graph-level classification task.

*Table 1.* Graph augmentation (Aug.) method comparison. An ideal method should support both topology and node feature augmentation, be adaptive to different datasets, be end-to-end differentiable and have efficient back-propagation (BP), and be CI and have unified constraints for any augmentation to against various noise.

| Property | Random or Constraint | | | Learnable Graph Augmentation | | | |
|---|---|---|---|---|---|---|---|
| | GraphCL | JOAO | GCA | AutoGCL | AD-GCL | GCL-SPAN | Ours |
| Topology Aug. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Feature Aug. | ✓ | ✓ | ✓ | ✓ | - | - | ✓ |
| Adaptive | - | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Differentiable | - | - | - | ✓ | ✓ | - | ✓ |
| Efficient BP | - | - | - | ✓ | ✓ | - | ✓ |
| Community | - | - | - | - | - | - | ✓ |
| Unified Constraint | - | - | - | - | - | - | ✓ |

$*$ Efficient BP means that the model adopts learnable graph augmentation in an end-to-end differentiable manner and is efficient enough for fast gradient computation via back-propagation (BP). Unified constraint refers to a constraint that can be generally applied to any type of graph augmentation, including both topology augmentation and feature augmentation.

Recent GCL focuses on fully parameterizing graph augmentation for utilizing learnable graph augmentation to automatically determine how to disrupt graphs (Chen et al., 2023a). For example, AutoGCL (Yin et al., 2022) and AD-GCL (Suresh et al., 2021) build a learnable graph generator that learns a probability distribution to help adaptively drop nodes and mask features. CGI (Wei et al., 2022) introduces the Information Bottleneck theory into GCL to remove unimportant nodes and edges between two augmented graphs by minimizing shared mutual information. GAME (Liu et al., 2022a) and GCL-SPAN (Lin et al., 2023) explore graph augmentation in spectral space, by maximizing spectral changes of high-frequency or all components to automatically drop edges. AdaGCL (Jiang et al., 2023) and GACN (Wu et al., 2023a) design graph generators and discriminators to automatically augment graphs in an adversarial style.

Compared with spectrum-based methods such as GCL-SPAN, GAME (Liu et al., 2022a), SGCL (Ghose et al., 2023) and $\text{SFA}_{\text{Info}}$ (Zhang et al., 2023b). GCL-SPAN explores high-level graph structural information by disturbing it, i.e., they design their augmented view by corrupting what they want to retain. In contrast, CI-GCL preserves the core information between two augmented graphs to help the encoder better understand what should be retrained during

graph augmentation. GAME focuses on the notion that spectrum changes of high-frequency should be larger than spectrum changes of low-frequency part, while CI-GCL considers all spectrum changes as a whole and endeavors to maintain community invariance during graph augmentation. SGCL aims to generate cropping and reordering augmented graphs, and $\text{SFA}_{\text{Info}}$ aims to balance and align data distribution shift between the original and augmented graph views. In contrast, CI-GCL aims to preserve community structural information by minimizing spectral changes. Another main difference between CI-GCL and these spectrum-based methods is that CI-GCL incorporates learnable graph augmentation, which makes it more flexible and generalizable for many different datasets. Overall, we are the first to point out the importance of community invariance for graph augmentation and propose a unified CI constraint for both topology and feature augmentation by simply maximizing spectral changes. Detailed comparisons are listed in Table 1.

## 3. Preliminary

Let $G = (\mathbf{X}, \mathbf{A})$ be a graph with $n$ nodes and $m$ edges, where $\mathbf{X} \in \mathbb{R}^{n \times d}$ describes node features and $\mathbf{A} \in \{0,1\}^{n \times n}$ denotes an adjacency matrix with $\mathbf{A}_{ij} = 1$ if an edge exists between node $i$ and $j$, otherwise $\mathbf{A}_{ij} = 0$. The normalized Laplacian matrix is defined as $\mathbf{L}_{\text{norm}} = \text{Lap}(\mathbf{A}) = \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ is an identity matrix, $\mathbf{D} = \text{diag}\,(\mathbf{A}\mathbf{1}_n)$ is the diagonal degree matrix with $\mathbf{1}_n \in \mathbb{R}^n$ being an all-one vector.

**Graph Spectrum.** The spectral decomposition of $\mathbf{L}_{\text{norm}}$ is defined as $\mathbf{L}_{\text{norm}} = \text{Lap}(\mathbf{A}) = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where the diagonal matrix $\mathbf{\Lambda} = \text{eig}\,(\text{Lap}\,(\mathbf{A})) = \text{diag}\,(\lambda_1, \ldots, \lambda_n)$ consists of real eigenvalues known as graph spectrum, and $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_n] \in \mathbb{R}^{n \times n}$ are the corresponding orthonormal eigenvectors known as the spectral bases (Gene et al., 2013).

**Graph Representation Learning.** Let $\mathcal{G}$ denote the whole graph space with $G \in \mathcal{G}$. Graph representation learning aims to train an encoder $f_\theta(\cdot) : \mathcal{G} \to \mathbb{R}^{n \times d'}$ to obtain node representations, where the learned node embedding $f_\theta(G)$ can be used in node-level downstream tasks. Then, it trains a readout function $r_\phi(\cdot) : \mathbb{R}^{n \times d'} \to \mathbb{R}^{d'}$ by pooling all node representations to obtain a low-dimensional vector for graph $G$, which can be used in graph-level tasks.

**Graph Contrastive Learning.** GCL trains the encoder $f_\theta(\cdot)$ to capture the maximum mutual information between the original graph and its perturbed view by graph augmentation. Formally, letting $T_1(G)$ and $T_2(G)$ denote two graph augmentation distributions of $G$, GCL is defined as follows:

$$\min_{\theta,\phi} \; \mathcal{L}_{\text{GCL}}(t_1(G),\, t_2(G),\, \theta, \phi), \tag{1}$$

where $t_m(G) \sim T_m(G)$ with $m \in \{1, 2\}$ and $\mathcal{L}_{\text{GCL}}$ measure the disagreement between two augmented graphs.

## 4. Methodology

We first show the importance of community invariance in GCL with preliminary analysis. Then, we introduce the details of our methodology CI-GCL as illustrated by Figure 2.

### 4.1. Preliminary Analysis

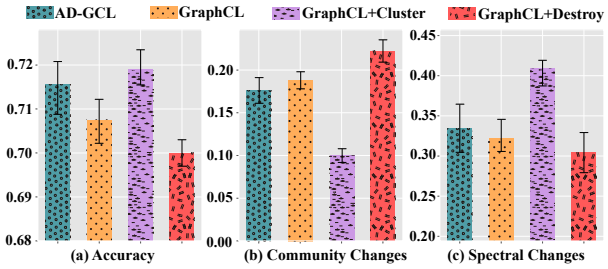


*Figure 1.* In unsupervised graph classification, we define community changes as the average ratio of the changed community labels over the number of nodes before and after graph augmentation by spectral clustering. Spectral changes are the eigenvalue changes between original and augmented graphs, using the $L_2$ distance.

Preserving community structure is crucial for learnable graph augmentation, i.e., perturbing a constrained number of edges or features that have least impact to community changes of the input graph. To show the benefits of preserving communities, we conduct a preliminary experiment by applying GraphCL (unlearnable graph augmentation) and AD-GCL (learnable graph augmentation) on the IMDB-B dataset. Specifically, we design the following four methods: (1) AD-GCL with uniformly edge dropping; (2) GraphCL with uniformly edge dropping; (3) GraphCL+Cluster augmentation that removes edges between different clusters with a higher probability; (4) GraphCL+Destroy augmentation that removes edges within the same cluster with a higher probability. Note that (3) preserves community structure, while (4) tends to disrupt community structure, as indicated by recent studies (Chiplunkar et al., 2018; Lin et al., 2022). We plot the accuracy for unsupervised graph classification as Figure 1(a) and the community changes as (b). From Figure 1(a-b), we can observe: (1) Methods' performance generally exhibits a nearly negative correlation with their community changes (i.e., less cluster changes yield higher accuracy); (2) AD-GCL outperforms GraphCL but underperforms GraphCL+Cluster. All these indicate preserving the community structure yields better results. Moreover, we also draw the spectral changes as Figure 1(c), as graph spectrum can reflect high-level graph structural information (Spielman, 2012; Lee et al., 2014). Through Figure 1(b-c), we can see that spectral changes are almost negatively correlated with community changes. That is to say, we can preserve community invariance during graph augmentation by maximizing spectral changes, based on which we expand on our methodology as follows.

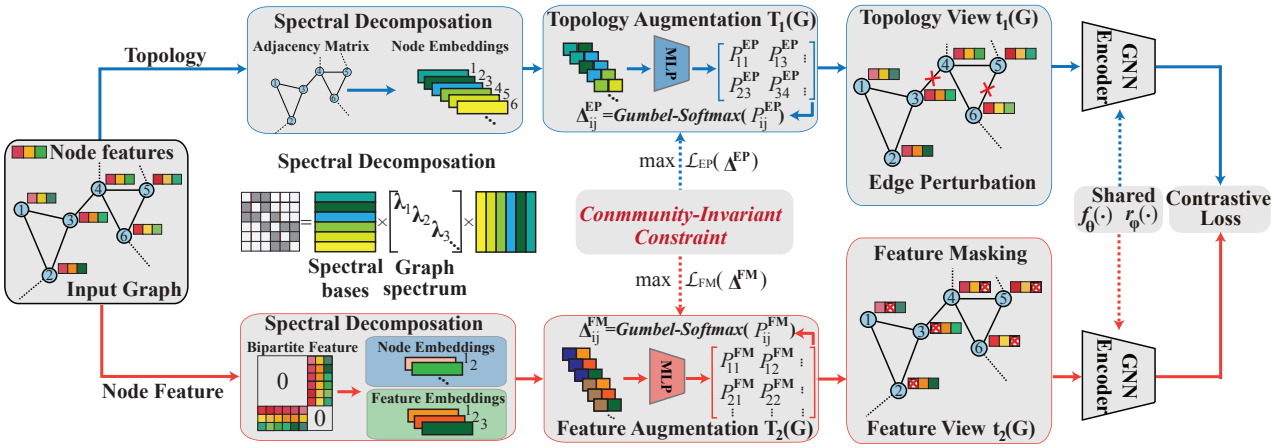


*Figure 2.* The proposed CI-GCL consists of two core components: (1) Learnable graph augmenter optimizes $T_m(G)$ to disrupt redundant information while ensuring community invariance from the original graph. (2) The GNN encoder $f_\theta(\cdot)$ and Readout $r_\phi(\cdot)$ maximize the mutual information between two augmented graphs by contrastive loss. We use edge dropping and feature masking as an instantiation.

### 4.2. Community-Invariant Graph Augmentation

**Topology Augmentation.** We conduct edge perturbation and node dropping operations as our topology augmentation. For edge perturbation, we define $T_1(G)$ as a Bernoulli distribution $Bern(P_{ij}^{\text{EP}})$ for each edge $A_{ij}$. Then, we can sample the edge perturbation matrices $\mathbf{\Delta}^{\text{EP}} \in \{0,1\}^{n\times n}$, where $\Delta_{ij} \sim Bern(P_{ij}^{\text{EP}})$ indicates whether to flip the edge between nodes $i$ and $j$. The edge is flipped if $\Delta_{ij}^{\text{EP}} = 1$; otherwise, it remains unchanged. A sampled augmented graph by edge augmentation can be formulated as:

$$t_1^{\text{EP}}(G) = \mathbf{A} + \mathbf{C} \circ \mathbf{\Delta}^{\text{EP}}, \ \mathbf{C} = \mathbf{A}^{\mathbf{c}} - \mathbf{A}, \tag{2}$$

where $\circ$ denotes an element-wise product and $\mathbf{A}^{\mathbf{c}}$ represents the complement matrix of $\mathbf{A}$, calculated as $\mathbf{A}^{\mathbf{c}} = \mathbf{J} - \mathbf{I}_n - \mathbf{A}$, where $\mathbf{J}$ denotes an all-one matrix. Thus, $\mathbf{C} \in \{-1,1\}^{n\times n}$ denotes all edge flipping operations, i.e., an edge is added between nodes $i$ and $j$ if $C_{ij} = 1$, and removed if $C_{ij} = -1$.

However, Eq.(2) cannot be directly applied to learnable graph augmentation, since Bernoulli sampling is non-differentiable. Inspired by Jang et al. (2017), we soften it from the discrete Bernoulli distribution space to the continuous space with a range $\Delta_{ij}^{\text{EP}} \in (0,1)^{n\times n}$ using *Gumbel-Softmax*, which can be formulated as:

$$\Delta_{ij}^{\text{EP}}(\epsilon) = \text{Softmax}((\log(P_{ij}^{\text{EP}}) + \epsilon)/\tau), \tag{3}$$

$$P_{ij}^{\text{EP}} = \text{Sigmoid}(\text{MLPs}(\text{Concat}(\mathbf{e}_i, \mathbf{e}_j))), \tag{4}$$

where $P_{ij}^{\text{EP}}$ controls whether to flip edge $A_{ij}$, MLPs are multilayer perceptions, $\mathbf{e}_i$ is the $i$-th node representation, $\text{Concat}(\cdot,\cdot)$ denotes the concatenation operation, $\epsilon \sim Gumbel(0,1)$,[1] and $\tau$ is the temperature factor that controls the approximation degree for the discrete categorical distributions. $\tau > 0$ results in a well-defined gradient $\partial\Delta_{ij}^{\text{EP}}/\partial P_{ij}^{\text{EP}}$, facilitating efficient optimization.

Node dropping can be considered as a type of edge dropping, i.e., removing all edges connected to this node. Thus, node dropping can be formulated similarly to Eq.(2) as:

$$t_1^{\text{ND}}(G) = \mathbf{A} + (-\mathbf{A}) \circ \mathbf{\Delta}^{\text{ND}}. \tag{5}$$

where $\mathbf{\Delta}^{\text{ND}}$ can be calculated by:

$$\mathbf{\Delta}^{\text{ND}} = (\mathbf{\Psi}^{\text{ND}} \cdot \mathbf{1}_n^\top + (\mathbf{\Psi}^{\text{ND}} \cdot \mathbf{1}_n^\top)^\top)/2, \tag{6}$$

$$\Psi_i^{\text{ND}}(\epsilon) = \text{Softmax}((\log(P_i^{\text{ND}}) + \epsilon)/\tau), \tag{7}$$

$$P_i^{\text{ND}} = \text{Sigmoid}(\text{MLPs}(\mathbf{e}_i)). \tag{8}$$

We combine $t_1^{(\text{EP},\text{ND})}(G)$ as topology augmentation $t_1(G)$.

**CI-based Topology Augmentation.** Inspired by the findings in Sec 4.1, we aim to optimize $\mathbf{\Delta}$ by simultaneously maximizing graph disruption while minimizing community changes for learnable topology augmentation in Eqs.(2,5). Based on the matrix perturbation theory (Bojchevski & Günnemann, 2019), we have the following definition.

**Definition 1.** Let $\lambda_k$ denote the $k$-th eigenvalue of the spectral decomposition of $\text{Lap}(\mathbf{A}) = \mathbf{U}\Lambda\mathbf{U}^\top$. For a single edge perturbation $A_{ij}$, it induces absolute changes in eigenvalues given by $\sum_{k=1}^n |\Delta\lambda_k| = \sum_{k=1}^n |(U_{ik} - U_{jk})^2 + (\lambda_k - 1)(U_{ik}^2 + U_{jk}^2)|$, and $\Delta\lambda_k$ denotes the $k$-th spectral change.

When optimizing $\sum_{k=1}^n |\Delta\lambda_k|$ in Definition 1, we argue that maintaining community invariance requires a categorized discussion on edge adding and edge dropping.

**Theorem 1.** The absolute spectral changes $\sum_{k=1}^n |\Delta\lambda_k|$ are upper bounded by $\|\mathbf{U}_{i\cdot} - \mathbf{U}_{j\cdot}\|_2^2 + \sum_{k=1}^n |\lambda_k - 1|$ and lower

[1] The $Gumbel(0,1)$ distribution can be sampled by calculating $\epsilon = -\log(-\log(u))$ with $u \sim \text{Uniform}(0,1)$ (Jang et al., 2017).

bounded by $\|\mathbf{U}_{i\cdot} - \mathbf{U}_{j\cdot}\|_2^2 - \sum_{k=1}^{n} |\lambda_k - 1|$, respectively. Here, $\mathbf{U}_{i\cdot}$ represents the $i$-th row vector of $\mathbf{U}$, denoting the $i$-th node embedding in the spectral space.

According to Theorem 1, maximizing spectral changes equates to maximizing their upper bound, i.e., flipping several edges between nodes with largest distances in spectral space. Shi & Malik (1997) states that node representations with larger distances always belong to different communities. Thus, we can maximize spectral changes during the edge dropping to preserve community invariance. However, we cannot conduct edge adding since adding edges between clusters always disrupts communities (Zhu et al., 2023). Conversely, minimizing spectral changes equates to minimizing their lower bound, i.e., flipping several edges between nodes with lowest distances, where nodes with lower distances always belong to the same cluster. Thus, we can minimize spectral changes during the edge adding, instead of edge dropping, since dropping edges within one cluster will disrupt communities (Zhu et al., 2023). We formulate the CI constraint for edge perturbation by jointly optimizing edge dropping and adding as follows:

$$\max_{\mathbf{\Delta}^{\text{ED}}, \mathbf{\Delta}^{\text{EA}} \in \mathcal{S}} \mathcal{L}_{\text{EP}}(\mathbf{\Delta}^{\text{EP}}) = \mathcal{L}_{\text{ED}}(\mathbf{\Delta}^{\text{ED}}) - \mathcal{L}_{\text{EA}}(\mathbf{\Delta}^{\text{EA}}), \quad (9)$$

$$\mathcal{L}_{\text{ED}}(\mathbf{\Delta}^{\text{ED}}) = \|\text{eig}(\text{Lap}(\mathbf{A} - \mathbf{A} \circ \mathbf{\Delta}^{\text{ED}})) - \text{eig}(\text{Lap}(\mathbf{A}))\|_2^2,$$

$$\mathcal{L}_{\text{EA}}(\mathbf{\Delta}^{\text{EA}}) = \|\text{eig}(\text{Lap}(\mathbf{A} + \mathbf{A}^{\mathbf{c}} \circ \mathbf{\Delta}^{\text{EA}})) - \text{eig}(\text{Lap}(\mathbf{A}))\|_2^2,$$

where $\mathcal{S} = \{\mathbf{S} | \mathbf{S} \in [0,1]^{n \times n}, \|\mathbf{S}\|_1 \leq \psi\}$, $\psi$ controls the perturbation strength, and $\mathcal{L}_(\mathbf{\Delta})$ represents graph spectral changes under different augmented operations.

Node dropping can be considered as one type of ED, which can be constrained by community invariance by:

$$\max_{\mathbf{\Delta}^{\text{ND}} \in \mathcal{S}} \|\text{eig}(\text{Lap}(\mathbf{A} - \mathbf{A} \circ \mathbf{\Delta}^{\text{ND}})) - \text{eig}(\text{Lap}(\mathbf{A}))\|_2^2. \quad (10)$$

By jointly optimizing Eqs.(9,10), we can maximize topology perturbation while maintaining community invariance.

**Feature Augmentation.** Similar to topology augmentation, we define $T_2(G)$ as a Bernoulli distribution $Bern(P_{ij}^{\text{FM}})$ for each feature $X_{ij}$. Then, we can sample feature masking matrix $\Delta_{ij}^{\text{FM}} \sim Bern(P_{ij}^{\text{FM}})$, indicating whether to mask the corresponding feature. A sampled augmented graph by feature masking can be formulated as:

$$t_2^{\text{FM}}(G) = \mathbf{X} + (-\mathbf{X}) \circ \mathbf{\Delta}^{\text{FM}}. \quad (11)$$

**CI-based Feature Augmentation.** Different from topology augmentation, $\mathbf{X} \in \mathbb{R}^{n \times d}$ is an asymmetric matrix lacking spectral decomposition. Theorem 1 is not applicable to feature augmentation. Moreover, discerning which feature has the least impact on community changes is challenging. Inspired by the co-clustering of bipartite graph (Nie et al., 2017), which can determine the importance of features for node clustering, we construct the feature bipartite graph as:

$$\widetilde{\mathbf{X}} = \begin{bmatrix} \mathbf{0} & \mathbf{X} \\ \mathbf{X}^\top & \mathbf{0} \end{bmatrix}_{(n+d)\times(n+d)}, \quad (12)$$

where the first $n$ rows of $\widetilde{\mathbf{X}}$ denote the original nodes, while the subsequent $d$ rows serve to represent features as feature nodes. Then, $\widetilde{X}_{ij}$, where $i \in \{1, \cdots n\}$ and $j \in \{(n+1), \cdots, (n+d)\}$, can be interpreted as the linking weight between $i$-th node with $j$-th feature.

**Theorem 2.** Let the singular value decomposition of the feature matrix $\mathbf{X}$ be denoted as $\text{svd}(\mathbf{D}_u^{-1/2}\mathbf{X}\mathbf{D}_v^{-1/2}) = \mathbf{U}\mathbf{\Lambda}_1\mathbf{V}^\top$ where $\mathbf{D}_u$ and $\mathbf{D}_v$ are the degree matrices of $\mathbf{X}$ and $\mathbf{X}^\top$, and $\mathbf{U}$ and $\mathbf{V}$ represent the left and right singular vectors, respectively. Then, $\text{eig}(\text{Lap}(\widetilde{\mathbf{X}})) = \mathbf{F}\mathbf{\Lambda}_2\mathbf{F}^\top$ where the $k$-th smallest eigenvector $\mathbf{F}_{\cdot k}$ is equal to the concatenation of $k$-th largest singular vectors: $\mathbf{F}_{\cdot k} = [\mathbf{U}_{\cdot k}; \mathbf{V}_{\cdot k}]$.

According to Theorem 2 and findings from Nie et al. (2017), if we can maintain community invariance of $\widetilde{\mathbf{X}}$, community structure will also be preserved in $\mathbf{X}$. Hence, we investigate the community-invariant constraint in $\widetilde{\mathbf{X}}$.

**Theorem 3.** Let $\lambda_k$ denote the $k$-th smallest eigenvalue of $\mathbf{\Lambda_2}$. When masking one feature $\widetilde{X}_{ij}$, the induced spectral changes are given by $\widetilde{\mathbf{X}}$ as $\sum_{k=1}^{n+d} |\Delta\lambda_k| = \sum_{k=1}^{n+d} |(F_{ik} - F_{jk})^2 + (\lambda_k - 1)(F_{ik}^2 + F_{jk}^2)|$, which are upper bounded by $\|\mathbf{F}_{i\cdot} - \mathbf{F}_{j\cdot}\|_2^2 + \sum_{k=1}^{n+d} |1 - \lambda_k|$ where $i \in \{1, .., n\}$ and $j \in \{(n+1), .., (n+d)\}$, $\mathbf{F}_{i\cdot}$ is the $i$-th row vector of $\mathbf{F}$.

Based on Theorem 3, maximizing spectral changes in $\widetilde{\mathbf{X}}$ under a constrained number of perturbation equals finding several largest embedding distances between nodes and feature nodes, i.e., these features have the least impact on community changes for these nodes (Zhang et al., 2023a). Thus, CI constraint for feature augmentation $\mathcal{L}_{\text{FM}}(\mathbf{\Delta}^{\text{FM}})$ can be formulated as follows:

$$\max_{\mathbf{\Delta}^{\text{FM}} \in \mathcal{S}} \|\text{eig}(\text{Lap}(\widetilde{\mathbf{X}} - \widetilde{\mathbf{X}} \circ \mathbf{\Delta}^{\text{FM}}) - \text{eig}(\text{Lap}(\widetilde{\mathbf{X}})))\|_2^2. \quad (13)$$

Finally, we parameterize $\mathbf{\Delta}^{\text{FM}}$ and ensure its differentiability in the feature augmentation, formulated as follows:

$$\Delta_{ij}^{\text{FM}}(\epsilon) = \text{Softmax}((\log(P_{ij}^{\text{FM}}) + \epsilon)/\tau), \quad (14)$$
$$P_{ij}^{\text{FM}} = \text{Sigmoid}(\text{MLPs}(\text{Concat}(\widetilde{\mathbf{U}}_{\cdot i}, \widetilde{\mathbf{V}}_{\cdot j}))).$$

### 4.3. Overall Framework of CI-GCL

As shown in Figure 2, we instantiate a graph contrastive learning framework with the proposed community-invariant constraint, namely CI-GCL. Specifically, we first conduct spectral decomposition on the adjacent matrix and feature bipartite matrix to obtain node and feature representations. Then, we consider these node and feature representations as input of MLPs for both topology and feature

augmentation, where we randomly initialize the parameters of MLPs. For each iteration of contrastive learning, we sample two augmented graphs by topology augmentation $t_1(G) \sim T_1(G|\mathbf{\Delta}^{\text{EP}}, \mathbf{\Delta}^{\text{ND}})$ and feature augmentation $t_2(G) \sim T_2(G|\mathbf{\Delta}^{\text{FM}})$. The augmented graphs are then fed into a GCN encoder $f_\theta(\cdot)$, which outputs two sets of node representations. A readout pooling function $r_\phi(\cdot)$ is applied to aggregate and transform the node representations and obtain graph representations $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}$. Following GraphCL (You et al., 2020), given training graphs $\mathcal{G}$, we use contrastive objective $\mathcal{L}_{\text{GCL}}$, which can be defined as:

$$\min_{\theta,\phi} \ \mathcal{L}_{\text{GCL}}\left(t_1\left(G\right), t_2\left(G\right), \theta, \phi\right) \tag{15}$$

$$= -\frac{1}{|\mathcal{G}|}\sum_{n=1}^{|\mathcal{G}|}\left(\log\frac{\exp(\text{sim}(\mathbf{z}_n^{(1)}, \mathbf{z}_n^{(2)}/\tau_2))}{\sum_{n'=1,n'\neq n}^{|\mathcal{G}|}\exp(\text{sim}(\mathbf{z}_n^{(1)}, \mathbf{z}_{n'}^{(2)})/\tau_2)}\right),$$

where $\tau_2$ is the temperature parameter, and we conduct minibatch optimization for Eq.(15) in our study.

### 4.4. Optimization and Scalability

**Optimization.** Eqs.(9,10,13) are jointly optimized via projected gradient descent. Taking $\mathbf{\Delta}^{\text{FM}}$ in Eq.(13) as an example, we can update the parameters $\mathbf{\Delta}^{\text{FM}}$ as:

$$\mathbf{\Delta}_t^{\text{FM}} = \mathcal{P}_{\mathcal{S}}\left(\mathbf{\Delta}_{(t-1)}^{\text{FM}} - \eta_t \nabla\mathcal{L}_{\text{FM}}\left(\mathbf{\Delta}_{(t-1)}^{\text{FM}}\right)\right), \tag{16}$$

where $\mathcal{P}_{\mathcal{S}}(\mathbf{\Delta}) = \arg\min_{\mathbf{S}\in\mathcal{S}} \|\mathbf{S} - \mathbf{\Delta}\|_F^2$ is defined as one projection operation at $\mathbf{\Delta}$ over the constraint set $\mathcal{S}$ and $\eta_t > 0$ is the learning rate for the $t$-th updating step. The gradient $\nabla\mathcal{L}_{\text{FM}}(\mathbf{\Delta}_{(t-1)}^{\text{FM}})$ can be calculated via a chain rule, with a closed-form gradient over eigenvalues, i.e., for $\mathbf{L}_{\text{norm}} = \text{Lap}(\mathbf{A} + \mathbf{C} \circ \mathbf{\Delta}^{\text{FM}})$, the derivatives of its $k$-th eigenvalue $\lambda_k$ is $\partial\lambda_k/\partial\mathbf{L}_{\text{norm}} = \mathbf{U}_{\cdot k}\mathbf{U}_{\cdot k}^\top$ (Rogers, 1970).

**Scalability.** Due to the eigendecomposition, the time complexity of optimizing Eqs.(9,10) is $\mathcal{O}(\text{Mn}^3)$ with M representing the number of iterations, due to the eigendecomposition, which is prohibitively expensive for large graphs. To reduce the computational cost, instead of conducting eigendecomposition on the full graph spectrum, we employ selective eigendecomposition on the $K$ lowest eigenvalues via the Lanczos Algorithm (Parlett & Scott, 1979), which can reduce time complexity to $\mathcal{O}(\text{Mn}^2\text{K})$. Similarly, we can use Truncated SVD (Halko et al., 2011) to obtain the $K$ highest eigenvalues of $\mathbf{X}$ and then concatenate them as approximate for eigendecomposition of $\widetilde{\mathbf{X}}$, thereby reducing time complexity from $\mathcal{O}(\text{M(n+d)}^3)$ to $\mathcal{O}(\text{Mn}\log\text{K})$. For more details, please refer to Appendix G.

## 5. Experiments

In our general experimental settings, we use GIN (Xu et al., 2019) as the base encoder for all baselines to ensure a fair comparison. Each experiment is repeated 10 times with different random seeds, and we report the mean and standard derivation of the corresponding evaluation metrics. We select several best-performing baselines for comparison, including classic GCL methods, such as MVGRL (Hassani & Ahmadi, 2020), InfoGraph (Sun et al., 2020), GraphCL, and JOAO, as well as GCL methods with learnable graph augmentation, such as SEGA (Wu et al., 2023b), GCS (Wei et al., 2023), GCL-SPAN, AD-GCL, and AutoGCL.

### 5.1. Quantitative Evaluation

#### 5.1.1. Comparison with State-of-the-Arts

To comprehensively demonstrate the effectiveness and generalizability of CI-GCL, following previous studies (Yin et al., 2022), we perform evaluations for graph classification and regression under three different experimental settings: unsupervised, semi-supervised, and transfer learning.

**Unsupervised Learning.** We first train graph encoders (i.e., GIN) separately for each of the GCL baselines using unlabeled data. Then, we fix parameters of these models and train an SVM classifier using labeled data. We use TU datasets (Morris et al., 2020) and OGB datasets (Hu et al., 2020a) to evaluate graph classification and regression, respectively. We adopt the provided data split for the OGB datasets and use 10-fold cross-validation for the TU datasets as it lacks such a split. Table 2 shows the performance on graph classification and Table 3 draws the performance on graph regression. In these tables, CI-GCL achieves the best results on 9 datasets and competitive results on the MUTAG and mollipo datasets. Specifically, CI-GCL achieves the highest averaged accuracy in graph classification (77.74%) and the lowest RMSE in graph regression (1.606), surpassing SOTA classification methods, such as SEGA (77.25%), GraphCL (75.71%), as well as SOTA regression methods, such as GCL-SPAN (2.184) and AD-GCL (2.403).

**Semi-Supervised Learning.** Following GraphCL, we employ 10-fold cross validation on each TU datasets using ResGCN (Pei et al., 2021) as the classifier. For each fold, different from the unsupervised learning setting, we only use 10% as labeled training data and 10% as labeled testing data for graph classification. As shown in Table 4, CI-GCL achieves highest averaged accuracy (74.0%) compared with SOTA baselines SEGA (72.9%) and AD-GCL (73.1%).

**Transfer Learning.** To show the generalization, we conduct self-supervised pre-training for baselines on the preprocessed ZINC-2M or PPI-306K dataset (Hu et al., 2020b) for 100 epochs and then fine-tune baselines on different downstream biochemical datasets. Table 5 shows that CI-GCL achieves best results on 6 datasets and comparable performance on the rest datasets with an averaged performance (75.1%), comparing with SOTA baseline SEGA (73.6%).

*Table 2.* Unsupervised representation learning classification accuracy (%) on TU Datasets. **Bold** denotes the best performance, and underline represents the second best performance. ❄ marks the reproduced results of the corresponding baselines by us.

| Method | NCI1 ↑ | PROTEINS ↑ | DD ↑ | MUTAG ↑ | COLLAB ↑ | RDT-B ↑ | RDT-M5K ↑ | IMDB-B ↑ | Avg. ↑ |
|---|---|---|---|---|---|---|---|---|---|
| InfoGraph | 76.20±1.0 | 74.44±0.3 | 72.85±1.7 | 89.01±1.1 | 70.65±1.1 | 82.50±1.4 | 53.46±1.0 | 73.03±0.8 | 74.02 |
| GraphCL | 77.87±0.4 | 74.39±0.4 | 78.62±0.4 | 86.80±1.3 | 71.36±1.1 | 89.53±0.8 | 55.99±0.3 | 71.14±0.4 | 75.71 |
| MVGRL | 76.64±0.3 | 74.02±0.3 | 75.20±0.4 | 75.40±7.8 | 73.10±0.6 | 82.00±1.1 | 51.87±0.6 | 63.60±4.2 | 71.48 |
| JOAO | 78.07±0.4 | 74.55±0.4 | 77.32±0.5 | 87.35±1.0 | 69.50±0.3 | 85.29±1.4 | 55.74±0.6 | 70.21±3.0 | 74.75 |
| SEGA | 79.00±0.7 | 76.01±0.4 | 78.76±0.6 | **90.21±0.7** | 74.12±0.5 | 90.21±0.7 | 56.13±0.3 | 73.58±0.4 | 77.25 |
| GCS❄ | 77.18±0.3 | 74.04±0.4 | 76.28±0.3 | 88.19±0.9 | 74.00±0.4 | 86.50±0.3 | 56.30±0.3 | 72.90±0.5 | 75.64 |
| GCL-SPAN❄ | 75.43±0.4 | 75.78±0.4 | 78.78±0.5 | 85.00±0.8 | 71.40±0.5 | 86.50±0.1 | 54.10±0.5 | 66.00±0.7 | 74.12 |
| AD-GCL❄ | 73.38±0.5 | 73.59±0.7 | 75.10±0.4 | 89.70±1.0 | 72.50±0.6 | 85.52±0.8 | 54.91±0.4 | 71.50±0.6 | 74.53 |
| AutoGCL❄ | 78.32±0.5 | 69.73±0.4 | 75.75±0.6 | 85.15±1.1 | 71.40±0.7 | 86.60±1.5 | 55.71±0.2 | 72.00±0.4 | 74.33 |
| CI+AD-GCL | 74.35±0.5 | 74.66±0.6 | 76.20±0.4 | 89.88±0.7 | 73.94±0.3 | 87.80±1.2 | 54.75±0.6 | 72.10±0.3 | 75.46 |
| CI+AutoGCL | 78.47±0.7 | 70.81±0.5 | 76.53±0.6 | 86.73±1.0 | 72.24±0.9 | 87.50±1.4 | 55.97±0.2 | 72.50±0.3 | 75.09 |
| **CI-GCL** | **80.50±0.5** | **76.50±0.1** | **79.63±0.3** | 89.67±0.9 | **74.40±0.6** | **90.80±0.5** | **56.57±0.3** | **73.85±0.8** | **77.74** |

*Table 3.* RMSE for unsupervised graph regression.

| Method | molesol ↓ | mollipo ↓ | molfreesolv ↓ | Avg. ↓ |
|---|---|---|---|---|
| InfoGraph | 1.344±0.18 | 1.005±0.02 | 10.00±4.82 | 4.118 |
| GraphCL | 1.272±0.09 | 0.910±0.02 | 7.679±2.75 | 3.287 |
| MVGRL | 1.433±0.15 | 0.962±0.04 | 9.024±1.98 | 3.806 |
| JOAO | 1.285±0.12 | 0.865±0.03 | 5.131±0.72 | 2.427 |
| GCL-SPAN | 1.218±0.05 | **0.802±0.02** | 4.531±0.46 | 2.184 |
| AD-GCL | 1.217±0.09 | 0.842±0.03 | 5.150±0.62 | 2.403 |
| **CI-GCL** | **1.130±0.13** | 0.816±0.03 | **2.873±0.32** | **1.606** |

**Summary.** From the above experimental results, we obtain the following three conclusions: (1) *Higher Effectiveness.* CI-GCL can achieve the best performance in three different experimental settings, attributed to its unified community-invariant constraint for graph augmentation. Compared to GraphCL and MVGRL, with similar contrastive objectives, the gain by CI-GCL mainly comes from the CI constraint and learnable graph augmentation procedure. While compared to AD-GCL and AutoGCL, with similar encoders, CI-GCL, guided by community invariance, is clearly more effective than the widely adopted uniformly random augmentation. (2) *Better Generalizability.* By maximizing spectral changes to minimize community changes, CI-GCL can obtain the encoder with better generalizability and transferability. Since the encoder is pre-trained to ignore the impact of community irrelevant information and mitigate the relationship between such information and downstream labels, solving the overfitting issue. Furthermore, previous studies, such as JOAO and GCL-SPAN, improve generalizability of the GNN encoder on molecule classification by exploring structural information like subgraph. We suggest that the community could be another way to study chemical and biological molecules. (3) *Wider Applicability.* By combing the CI constraint with AD-GCL and AutoGCL in Table 2 and Table 5, we also see significant improvements on almost all datasets, showing that the CI constraint could be a plug-and-play component for any learnable GCL frameworks.

#### 5.1.2. Ablation Study

We conduct an ablation study to evaluate the effectiveness of the proposed CI constraint on topology augmentation (TA) and feature augmentation (FA). We consider the following variants of CI-GCL: (1) **w/o TA**: remove TA on one branch. (2) **w/o FA**: remove FA on one branch. (3) **w/o CI on TA**: remove CI constraint on TA. (4) **w/o CI on FA**: remove CI constraint on FA. (5) **w/o CI on ALL**: remove CI constraint on both TA and FA. Experimental results in Table 6 demonstrate that the removal of either component of the method negatively impacts the ability of the graph representation learning to perform well. These results align with our hypothesis that random topology or feature augmentation without CI constraint corrupt community structure, thereby hindering model's performance in downstream tasks.

### 5.2. Qualitative Evaluation

#### 5.2.1. Robustness Against Various Noise

To showcase the robustness of CI-GCL, we conduct experiments in the adversarial setting. Following GraphCL, we conduct *Random* noise attack, with perturbation ratios $\sigma \in \{0.05, 0.30\}$, on topology $\mathbf{A}$ and feature $\mathbf{X}$ of the input graph, respectively. Specifically, for the topology attack, we randomly flip $\sigma \times m$ edges with $m$ as the total number of edges. For the feature attack, we randomly select $\sigma \times d$ features to add Gaussian noise with $d$ as the total number of features. Baselines without designed feature augmentation are set with random feature masking. Figure 3 reports the graph classification performance. From Figure 3, we have the following three findings. (1) CI-GCL outperforms four best-performing GCL methods in both topology and feature attack, demonstrating its strong robustness. (2) CI-GCL and GCL-SPAN are more robustness than other baselines in the topology attack, showing that preserving high-level graph structure can improve robustness than random graph augmentation. While CI-GCL can better focus on community invariance to outperform GCL-SPAN. (3) CI-GCL is more

*Table 4.* Accuracy (%) for 10% labeled semi-supervised graph classification.

| Method | NCI1 ↑ | PROTEINS ↑ | DD ↑ | COLLAB ↑ | RDT-B ↑ | RDT-M5K ↑ | GITHUB ↑ | Avg. ↑ |
|---|---|---|---|---|---|---|---|---|
| No Pre-train | 73.72±0.2 | 70.40±1.5 | 73.56±0.4 | 73.71±0.3 | 86.63±0.3 | 51.33±0.4 | 60.87±0.2 | 70.0 |
| GraphCL | 74.63±0.3 | 74.17±0.3 | 76.17±1.4 | 74.23±0.2 | 89.11±0.2 | 52.55±0.5 | 65.81±0.8 | 72.3 |
| JOAO | 74.48±0.3 | 72.13±0.9 | 75.69±0.7 | 75.30±0.3 | 88.14±0.3 | 52.83±0.5 | 65.00±0.3 | 71.9 |
| SEGA | 75.09±0.2 | 74.65±0.5 | 76.33±0.4 | 75.18±0.2 | 89.40±0.2 | 53.73±0.3 | 66.01±0.7 | 72.9 |
| AD-GCL | 75.18±0.4 | 73.96±0.5 | 77.91±0.7 | 75.82±0.3 | 90.10±0.2 | 53.49±0.3 | 65.89±0.6 | 73.1 |
| AutoGCL | 67.81±1.6 | 75.03±3.5 | 77.50±4.4 | **77.16±1.5** | 79.20±3.5 | 49.91±2.7 | 58.91±1.5 | 69.3 |
| **CI-GCL** | **75.86±0.8** | **76.28±0.3** | **78.01±0.9** | 77.04±1.5 | **90.29±1.2** | **54.47±0.7** | **66.36±0.8** | **74.0** |

*Table 5.* ROC-AUC (%) for graph classification under transfer Learning settings.

| Pre-Train | ZINC 2M | | | | | | | | PPI-306K | |
|---|---|---|---|---|---|---|---|---|---|---|
| Fine-Tune | BBBP ↑ | Tox21 ↑ | ToxCast ↑ | SIDER ↑ | ClinTox ↑ | MUV ↑ | HIV ↑ | BACE ↑ | PPI ↑ | Avg. ↑ |
| No Pre-train | 65.8±4.5 | 74.0±0.8 | 63.4±0.6 | 57.3±1.6 | 58.0±4.4 | 71.8±2.5 | 75.3±1.9 | 70.1±5.4 | 64.8±1.0 | 66.7 |
| MVGRL | 69.0±0.5 | 74.5±0.6 | 62.6±0.5 | 62.2±0.6 | 77.8±2.2 | 73.3±1.4 | 77.1±0.6 | 77.2±1.0 | 68.7±0.7 | 71.4 |
| SEGA | 71.9±1.1 | 76.7±0.4 | 65.2±0.9 | 63.7±0.3 | **85.0±0.9** | 76.6±2.5 | 77.6±1.4 | 77.1±0.5 | 68.7±0.5 | 73.6 |
| GCS❄ | 72.5±0.5 | 74.4±0.4 | 64.4±0.2 | 61.9±0.4 | 66.7±1.9 | **77.3±1.7** | 78.7±1.4 | 82.3±0.3 | 70.3±0.5 | 72.1 |
| GCL-SPAN | 70.0±0.7 | **78.0±0.5** | 64.2±0.4 | 64.7±0.5 | 80.7±2.1 | 73.8±0.9 | 77.8±0.6 | 79.9±0.7 | 70.0±0.8 | 73.2 |
| AD-GCL❄ | 67.4±1.0 | 74.3±0.7 | 63.5±0.7 | 60.8±0.9 | 58.6±3.4 | 75.4±1.5 | 75.9±0.9 | 79.0±0.8 | 64.2±1.2 | 68.7 |
| AutoGCL❄ | 72.0±0.6 | 75.5±0.3 | 63.4±0.4 | 62.5±0.6 | 79.9±3.3 | 75.8±1.3 | 77.4±0.6 | 76.7±1.1 | 70.1±0.8 | 72.5 |
| CI+AD-GCL | 68.4±1.1 | 74.5±0.9 | 64.0±0.8 | 61.4±0.9 | 59.8±3.2 | 76.5±1.7 | 77.0±0.9 | 80.0±0.8 | 65.3±1.1 | 69.6 |
| CI+AutoGCL | 73.9±0.7 | 76.4±0.3 | 63.8±0.3 | 63.9±0.6 | 80.9±3.1 | 76.3±1.3 | 78.8±0.7 | 78.8±1.1 | 70.9±0.7 | 73.7 |
| **CI-GCL** | **74.4±1.9** | 77.3±0.9 | **65.4±1.5** | **64.7±0.3** | 80.5±1.3 | 76.5±0.9 | **80.5±1.3** | **84.4±0.9** | **72.3±1.2** | **75.1** |

*Table 6.* Ablation study on unsupervised graph classification.

| Method | NCI1 ↑ | PROTEIN ↑ | DD ↑ | MUTAG ↑ | Avg. ↑ |
|---|---|---|---|---|---|
| w/o TA | 78.8±0.6 | 74.6±0.6 | 77.8±0.8 | 86.1±0.9 | 79.3 |
| w/o FA | 79.2±0.4 | 75.1±0.2 | 78.1±0.3 | 86.3±0.9 | 79.6 |
| w/o CI on TA | 79.9±0.6 | 75.7±0.9 | 78.8±0.3 | 87.6±0.5 | 80.5 |
| w/o CI on FA | 80.0±0.3 | 75.9±1.4 | 78.7±0.8 | 87.5±1.4 | 80.5 |
| w/o CI on ALL | 78.6±0.9 | 74.8±0.9 | 78.3±0.3 | 86.5±1.8 | 79.5 |
| CI-GCL | **80.5±0.5** | **76.5±0.1** | **79.6±0.3** | **89.6±0.9** | **81.5** |

robust than other baselines in the feature attack since we also apply the uniformed CI constraint to feature augmentation.

### 5.2.2. Effectiveness in Community Preservation

To explore the ability of community preservation, we draw community changes in Table 7, where community changes are defined as the averaged number of changes of community labels of nodes before and after graph augmentation by spectral clustering. We observe that CI-GCL can effectively preserve community structure due to the proposed CI constraint. Furthermore, refer to Table 2, we can find that methods with larger community disruption, such as GraphCL and AutoGCL, underperform others with smaller community disruption. We also provide a visualization of CI-GCL on widely used synthetic graphs with 1,000 samples (Kim & Oh, 2021), that is suited for analysis since it possesses clear community structure (Kim & Oh, 2021). We train our models in an unsupervised learning manner on this dataset and randomly select one example for visualization. In Figure 4(A-C), CI-GCL effectively preserve

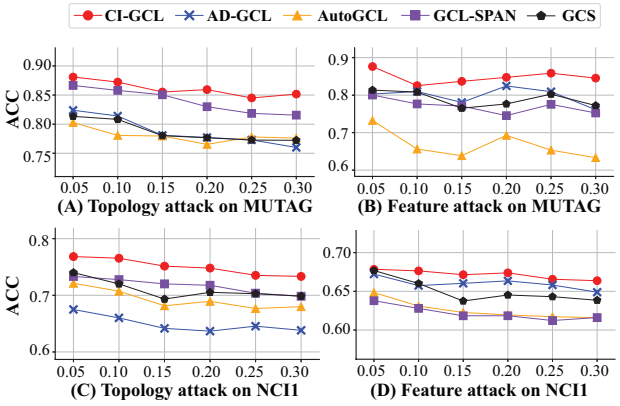


*Figure 3.* Accuracy (%) under noise attack on two datasets.

community structure by removing edges between clusters (Green lines) and add edges within each cluster (Red lines), while GraphCL destroys communities by randomly add and remove edges. In Figure 4(D), with x- and y-axis represent nodes and features, respectively, CI-GCL can effectively maintain important features for community invariance.

*Table 7.* Community changes (%) in unsupervised learning.

| Method | NCI1 ↓ | PROTEINS ↓ | MUTAG ↓ | IMDB-B ↓ | Avg. ↓ |
|---|---|---|---|---|---|
| GraphCL | 30.6 | 25.2 | 29.9 | 29.2 | 28.7 |
| GCS | 27.6 | 30.9 | 26.5 | 32.5 | 29.3 |
| GCL-SPAN | 15.1 | 11.2 | 14.5 | 19.7 | 15.1 |
| AD-GCL | 7.0 | 4.8 | 5.6 | 18.0 | 8.8 |
| AutoGCL | 34.2 | 31.0 | 30.3 | 33.4 | 32.2 |
| **CI-GCL** | **5.9** | **4.3** | **3.3** | **13.8** | **6.8** |

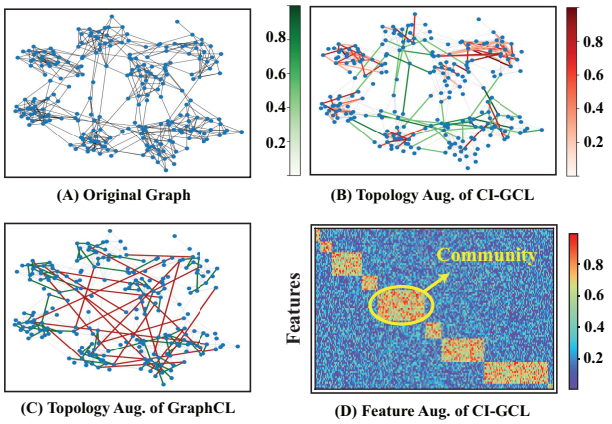


*Figure 4.* A case study of TA and FA of GraphCL and CI-GCL. (B-C) share the same color map and Green lines are edge dropping and Red lines are edge adding.

### 5.3. Node Classification

We conduct experiments on the node classification task and compare it with some recent and well-performing baselines: GRACE (Zhu et al., 2020), MVGCL (Hassani & Ahmadi, 2020), GCA (Zhu et al., 2021), ABGML (Chen et al., 2023a), GraphACL (Xiao et al., 2023), SFA (Zhang et al., 2023b), MA-GCL (Gong et al., 2023). Specifically, we utilize a random split setting by dividing the datasets into 10% for training, 10% for validation, and 80% for testing, respectively. The results are shown in Table 8. As depicted in Table 8, CI-GCL achieves the second-best performance on the Cora, CiteSeer, PubMed, and Amazon-Photo datasets, and comparable performance on the Amazon-Computers dataset, demonstrating the effectiveness of CI-GCL framework for the node classification task.

*Table 8.* Results of node classification with random split. **Bold** denotes the best performance, and underline represents the second best performance.

| Method | Cora ↑ | CiteSeer ↑ | PubMed ↑ | Amazon-C ↑ | Amazon-P ↑ |
|---|---|---|---|---|---|
| GRACE | 83.40±1.0 | 69.50±1.4 | 83.10±0.6 | 87.46±0.2 | 92.15±0.2 |
| MVGCL | 83.10±0.1 | 73.30±0.1 | 84.20±0.1 | 87.52±0.1 | 91.74±0.0 |
| GCA | 82.60±0.9 | 72.20±0.7 | 83.40±0.4 | 87.85±0.3 | 92.53±0.2 |
| ABGML | 78.90±0.3 | 65.49±0.4 | 83.05±0.5 | 88.38±0.3 | 92.82±0.4 |
| GraphACL | 84.72±0.3 | 44.47±0.2 | 83.14±0.2 | 88.93±0.3 | 92.89±0.2 |
| SFA | **85.89±0.1** | **75.31±0.1** | **86.29±0.1** | **89.24±0.3** | 93.53±0.1 |
| MA-GCL | 84.79±0.4 | 73.26±0.1 | 85.74±0.4 | 86.09±0.3 | **93.57±0.1** |
| CI-GCL | 85.50±0.3 | 73.58±0.2 | 85.90±0.4 | 86.57±0.3 | 93.54±0.4 |

### 5.4. Hyperparameter sensitives

We provide detailed hyperparameter sensitive experiments in the following. Specifically, CI-GCL includes three parameters. $\alpha$ and $\beta$ control the strength of community-invariant constraints on topology and feature augmentation, respectively. And $K$ denotes the number of eigenvalues and eigenvectors of spectral decomposition.

We evaluate parameter sensitivity by varying one target hyperparameter while keeping the others fixed. We employ 10-fold cross-validation, with 80% of the data used for training, 10% for validation, and 10% for testing. We conduct parameter sensitive experiments on the validation datasets. We show the experiment result of the MUTAG and PROTEIN datasets in Figure 5. From Figure 5, we observe that hyperparameters are insensitive within their respective optimal ranges. For more detailed discussion about hyperparameter settings, please refer to Appendix C.

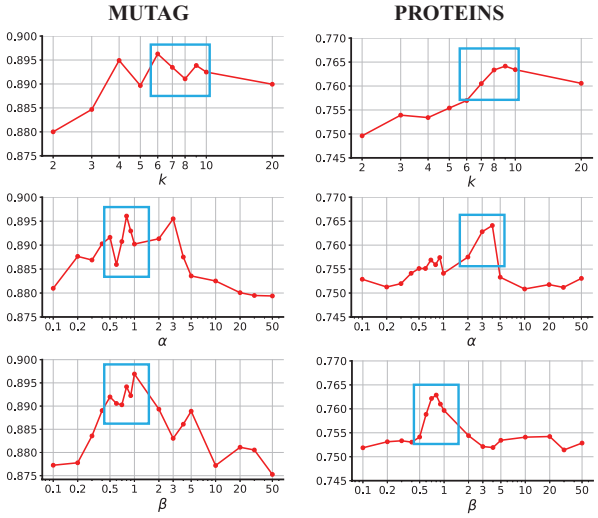


*Figure 5.* The parameter sensitivity involves the selected number of spectrum $K$, the balance weight of the topological constraint $\boldsymbol{\alpha}$, and the balance weight of the feature constraint $\boldsymbol{\beta}$.

## 6. Conclusion

In this work, we aimed to propose an unified constraint that can be applied to both topology and feature augmentation, to ensure community invariance and benefit for downstream tasks. To achieve this goal, we searched for the augmentation scheme that would maximize spectral changes of the input graph's topology and features, which can also minimize community changes. Our proposed community-invariant constraint can be paired with various GCL frameworks. We plan to explore more high-level graph information as constraints for learnable graph augmentation and apply our framework to many real-world applications in the future.

## Limitations

First, while the complexity of our algorithm is not high, the spectral decomposition of each augmented graph must be calculated one by one. Consequently, this process is time-consuming if no parallel computing is used. Second, for graph-level tasks, our approach is capable of handling inductive problems. However, for node-level tasks, our approach remains transductive due to the use of factorization.

## Impact Statement

In this section, we elaborate on the broader impacts of our work from the following two aspects. (1) Learnable Graph Augmentations. With the rapid development of GCL, learnable graph augmentation has become a significant research topic in the machine-learning community. Compared to current learnable graph augmentation methods, our work introduces control over the augmentation scheme in joint learning settings. Considering that CI constraint as a kind of expert knowledge, we can perceive our work as establishing a connection between expert knowledge-guided augmentations and learnable augmentations through the design of specific constraints. (2) Community and Spectrum: Despite significant advances in GCL, theoretical foundations regarding the relations between community preservation and spectrum design remain lacking. Our work highlights the significant potential of graph spectrum and community preservation in GCL, which may assist others in comprehending the graph spectrum. Moreover, we do not anticipate any direct negative impacts on society from our findings.

## References


Adhikari, B., Zhang, Y., Ramakrishnan, N., and Prakash, B. A. Sub2vec: Feature learning for subgraphs. In *Advances in Knowledge Discovery and Data Mining*, volume 10938, pp. 170–182, 2018.

Blakely, D., Lanchantin, J., and Qi, Y. Time and space complexity of graph convolutional networks. *Accessed on: Dec*, 31:2021, 2021.

Bojchevski, A. and Günnemann, S. Adversarial attacks on node embeddings via graph poisoning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 695–704, 2019.

Cao, Y., Chai, M., Li, M., and Jiang, C. Efficient learning of mesh-based physical simulation with bi-stride multi-scale graph neural network. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 3541–3558, 2023.

Chen, D., Zhao, X., Wang, W., Tan, Z., and Xiao, W. Graph self-supervised learning with augmentation-aware contrastive learning. In *Proceedings of the ACM Web Conference*, pp. 154–164, 2023a.

Chen, H., Zhao, Z., Li, Y., Zou, Y., Li, R., and Zhang, R. CS-GCL: community-strength-enhanced graph contrastive learning. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence*, pp. 2059–2067, 2023b.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. E. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 1597–1607, 2020.

Chiplunkar, A., Kapralov, M., Khanna, S., Mousavifar, A., and Peres, Y. Testing graph clusterability: Algorithms and lower bounds. In *IEEE Annual Symposium on Foundations of Computer Science*, pp. 497–508, 2018.

Dhillon, I. S. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining*, pp. 269–274, 2001.

Duan, H., Vaezipoor, P., Paulus, M. B., Ruan, Y., and Maddison, C. J. Augment with care: Contrastive learning for combinatorial problems. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 5627–5642, 2022.

Francis, J. Theq r transformation. a unitary analogue to thel r transformation. *Comput. J*, 4:265–271, 1961.

Gene, G., Van Loan, C. F., and et al. *Matrix computations*. JHU press, 2013.

Ghose, A., Zhang, Y., Hao, J., and Coates, M. Spectral augmentations for graph contrastive learning. In Ruiz, F. J. R., Dy, J. G., and van de Meent, J. (eds.), *International Conference on Artificial Intelligence and Statistics, 25-27 April 2023, Palau de Congressos, Valencia, Spain*, volume 206 of *Proceedings of Machine Learning Research*, pp. 11213–11266. PMLR, 2023.

Gong, X., Yang, C., and Shi, C. MA-GCL: model augmentation tricks for graph contrastive learning. In *Thirty-Seventh AAAI Conference on Artificial Intelligence*, pp. 4284–4292, 2023.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*, pp. 855–864, 2016.

Halko, N., Martinsson, P.-G., and Tropp, J. A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

Hassani, K. and Ahmadi, A. H. K. Contrastive multi-view representation learning on graphs. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 4116–4126, 2020.

Hogben, L. *Handbook of linear algebra*. CRC press, 2013.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark:

Datasets for machine learning on graphs. In *Proceedings of the 33th Advances in Neural Information Processing Systems*, 2020a. URL https://arxiv.org/abs/2005.00687.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V. S., and Leskovec, J. Strategies for pre-training graph neural networks. In *Proceedings of the 8th International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=HJlWWJSFDH.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Jiang, Y., Huang, C., and Huang, L. Adaptive graph contrastive learning for recommendation. In *Proceedings of the 29th International Conference on Knowledge Discovery and Data Mining*, pp. 4252–4261. ACM, 2023.

Kim, D. and Oh, A. How to find your friendly neighborhood: Graph attention design with self-supervision. In *9th International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Wi5KUNlqWty.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Lee, J. R., Gharan, S. O., and Trevisan, L. Multiway spectral partitioning and higher-order cheeger inequalities. *J. ACM*, 61(6):37:1–37:30, 2014.

Li, B., Jing, B., and Tong, H. Graph communal contrastive learning. In *Proceedings of the ACM Web Conference*, pp. 1203–1213, 2022a.

Li, S., Wang, X., Zhang, A., Wu, Y., He, X., and Chua, T. Let invariant rationale discovery inspire graph contrastive learning. In *Proceedings of 39th International Conference on Machine Learning*, volume 162, pp. 13052–13065, 2022b.

Li, Y., Hu, Y., Fu, L., Chen, C., Yang, L., and Zheng, Z. Community-aware efficient graph contrastive learning via personalized self-training. *CoRR*, abs/2311.11073, 2023. URL https://arxiv.org/abs/2311.11073.

Lin, L., Blaser, E., and Wang, H. Graph structural attack by perturbing spectral distance. In *Proceedings of the 28th International Conference on Knowledge Discovery and Data Mining*, pp. 989–998, 2022.

Lin, L., Chen, J., and Wang, H. Spectral augmentation for self-supervised learning on graphs. In *The 11th International Conference on Learning Representations*, 2023. URL https://openreview.net/pdf?id=DjzBCrMBJ_p.

Liu, N., Wang, X., Bo, D., Shi, C., and Pei, J. Revisiting graph contrastive learning from the perspective of graph spectrum. In *Proceedings of the 35th Advances in Neural Information Processing Systems*, 2022a. URL https://openreview.net/forum?id=L0U7TUWRt_X.

Liu, S., Ying, R., Dong, H., Li, L., Xu, T., Rong, Y., Zhao, P., Huang, J., and Wu, D. Local augmentation for graph neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 14054–14072, 2022b.

Luo, Y., McThrow, M., Au, W. Y., Komikado, T., Uchino, K., Maruhashi, K., and Ji, S. Automated data augmentations for graph classification. In *The Proceedings of the 11th International Conference on Learning Representations*, 2023. URL https://openreview.net/pdf?id=vTb1JI0Gps_.

Mayr, A., Klambauer, G., Unterthiner, T., Steijaert, M., Wegner, J. K., Ceulemans, H., Clevert, D.-A., and Hochreiter, S. Large-scale comparison of machine learning methods for drug target prediction on chembl. *Chemical science*, 9(24):5441–5451, 2018.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*, 2020. URL https://chrsmrrs.github.io/datasets/.

Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., and Jaiswal, S. graph2vec: Learning distributed representations of graphs. *CoRR*, 2017. URL http://arxiv.org/abs/1707.05005.

Ng, A. Y., Jordan, M. I., and Weiss, Y. On spectral clustering: Analysis and an algorithm. In Dietterich, T. G., Becker, S., and Ghahramani, Z. (eds.), *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pp. 849–856. MIT Press, 2001.

Nie, F., Wang, X., Deng, C., and Huang, H. Learning A structured optimal bipartite graph for co-clustering. In *Proceedings of the 30th Advances in Neural Information Processing Systems*, pp. 4129–4138, 2017.

Parlett, B. N. and Scott, D. S. The lanczos algorithm with selective orthogonalization. *Mathematics of computation*, 33(145):217–238, 1979.

Pei, Y., Huang, T., van Ipenburg, W., and Pechenizkiy, M. Resgcn: Attention-based deep residual modeling for anomaly detection on attributed networks. In *International Conference on Data Science and Advanced Analytics*, pp. 1–2. IEEE, 2021.

Pothen, A., Simon, H. D., and Liou, K.-P. Partitioning sparse matrices with eigenvectors of graphs. *SIAM journal on matrix analysis and applications*, 11(3):430–452, 1990.

Rogers, L. C. Derivatives of eigenvalues and eigenvectors. *AIAA journal*, 8(5):943–944, 1970.

Shen, X., Sun, D., Pan, S., Zhou, X., and Yang, L. T. Neighbor contrastive learning on learnable graph augmentation. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, pp. 9782–9791, 2023.

Shervashidze, N., Vishwanathan, S. V. N., Petri, T., Mehlhorn, K., and Borgwardt, K. M. Efficient graphlet kernels for large graph comparison. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, volume 5, pp. 488–495, 2009.

Shervashidze, N., Schweitzer, P., van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-lehman graph kernels. *J. Mach. Learn. Res.*, 12:2539–2561, 2011.

Shi, J. and Malik, J. Normalized cuts and image segmentation. In *Conference on Computer Vision and Pattern Recognition*, pp. 731–737. IEEE Computer Society, 1997.

Spielman, D. Spectral graph theory. *Combinatorial scientific computing*, 18:18, 2012.

Sterling, T. and Irwin, J. J. ZINC 15 - ligand discovery for everyone. *J. Chem. Inf. Model.*, 55(11):2324–2337, 2015.

Sun, F., Hoffmann, J., Verma, V., and Tang, J. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *Proceedings of the 8th International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1908.01000.

Sun, M., Xing, J., Wang, H., Chen, B., and Zhou, J. Mocl: Data-driven molecular fingerprint via knowledge-aware contrastive learning from molecular graph. In *Proceedings of the 27th International Conference on Knowledge Discovery and Data Mining*, pp. 3585–3594, 2021.

Suresh, S., Li, P., Hao, C., and Neville, J. Adversarial graph augmentation to improve graph contrastive learning. *Proceedings of the 34th Advances in Neural Information Processing Systems*, 34:15920–15933, 2021.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. What makes for good views for contrastive learning? In *Proceedings of the 33th Advances in Neural Information Processing Systems*, 2020. URL https://openreview.net/forum?id=FRnVp1JFaj.

Tong, Z., Liang, Y., Ding, H., Dai, Y., Li, X., and Wang, C. Directed graph contrastive learning. In *Proceedings of the 34th Advances in Neural Information Processing Systems*, pp. 19580–19593, 2021.

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. In *Proceedings of the 7th International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rklz9iAcKQ.

von Luxburg, U. A tutorial on spectral clustering. *Stat. Comput.*, 17(4):395–416, 2007. doi: 10.1007/S11222-007-9033-Z. URL https://doi.org/10.1007/s11222-007-9033-z.

Wang, Y., Zhang, J., Li, H., Dong, Y., Yin, H., Li, C., and Chen, H. Clusterscl: Cluster-aware supervised contrastive learning on graphs. In *Proceedings of the ACM Web Conference*, pp. 1611–1621, 2022.

Wei, C., Liang, J., Liu, D., and Wang, F. Contrastive graph structure learning via information bottleneck for recommendation. In *Proceedings of the 35th Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=lhl_rYNdiH6.

Wei, C., Wang, Y., Bai, B., Ni, K., Brady, D., and Fang, L. Boosting graph contrastive learning via graph contrastive saliency. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 36839–36855, 2023.

Wu, C., Wang, C., Xu, J., Liu, Z., Zheng, K., Wang, X., Song, Y., and Gai, K. Graph contrastive learning with generative adversarial network. In *Proceedings of the 29th International Conference on Knowledge Discovery and Data Mining*, pp. 2721–2730, 2023a.

Wu, J., Chen, X., Shi, B., Li, S., and Xu, K. SEGA: structural entropy guided anchor view for graph contrastive learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 202, pp. 37293–37312, 2023b.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Xiao, T., Zhu, H., Chen, Z., and Wang, S. Simple and asymmetric graph contrastive learning without augmentations. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *Proceedings of the 7th International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Yanardag, P. and Vishwanathan, S. V. N. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1365–1374, 2015.

Yin, Y., Wang, Q., Huang, S., Xiong, H., and Zhang, X. Autogcl: Automated graph contrastive learning via learnable view generators. In *36th AAAI Conference on Artificial Intelligence*, pp. 8892–8900, 2022.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5812–5823, 2020.

You, Y., Chen, T., Shen, Y., and Wang, Z. Graph contrastive learning automated. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 12121–12132, 2021.

Zhang, H., Nie, F., and Li, X. Large-scale clustering with structured optimal bipartite graph. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(8):9950–9963, 2023a.

Zhang, Y., Zhu, H., Song, Z., Koniusz, P., and King, I. COSTA: covariance-preserving feature augmentation for graph contrastive learning. In *The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2524–2534, 2022.

Zhang, Y., Zhu, H., Song, Z., Koniusz, P., and King, I. Spectral feature augmentation for graph contrastive learning and beyond. In Williams, B., Chen, Y., and Neville, J. (eds.), *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pp. 11289–11297. AAAI Press, 2023b.

Zhang, Z., Zhao, Z., Lin, Z., Zhu, J., and He, X. Counterfactual contrastive learning for weakly-supervised vision-language grounding. In *Advances in Neural Information Processing Systems*, 2020. URL https://openreview.net/forum?id=p7l6QRLKwCX.

Zhu, J., Zhao, W., Yang, H., and Nie, F. Joint learning of anchor graph-based fuzzy spectral embedding and fuzzy k-means. *IEEE Trans. Fuzzy Syst.*, 31(11):4097–4108, 2023.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. Deep graph contrastive representation learning. *CoRR*, abs/2006.04131, 2020. URL https://arxiv.org/abs/2006.04131.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. Graph contrastive learning with adaptive augmentation. In *The Web Conference*, pp. 2069–2080, 2021.

Zitnik, M., Sosič, R., Feldman, M. W., and Leskovec, J. Evolution of resilience in protein interactomes across the tree of life. *Proceedings of the National Academy of Sciences*, 116(10):4426–4433, 2019.

# A. Experiment Setup Details

## A.1. Hardware Specification and Environment

We conduct our experiments using a single machine equipped with an Intel i9-10850K processor, Nvidia GeForce RTX 3090Ti (24GB) GPUs for the majority datasets. For the COLLAB, RDT-B, and RDT-M5K datasets, we utilized RTX A6000 GPUs (48GB) with batch sizes exceeding 512. The code is written in Python 3.10 and we use PyTorch 2.1.0 on CUDA 11.8 to train the model on the GPU. Implementation details can be accessed at the provided link

https://github.com/CI-GCL.git

## A.2. Details on Datasets

### A.2.1. TU DATASETS

TU Datasets (Morris et al., 2020) [2] provides a collection of benchmark datasets. We use several biochemical molecules and social networks for graph classification, as summarized in Table A1. The data collection is also accessible in the PyG [3] library, which employs a 10-fold evaluation data split. We used these datasets for the evaluation of the graph classification task in unsupervised and semi-supervised learning settings.

*Table A1.* Statistics of TU Datasets for the unsupervised or semi-supervised graph classification task. # means Number of.

| Dataset | Category | # Graphs | Avg. # Nodes | Avg. # Edges |
|---|---|---|---|---|
| NCI1 | Biochemical Molecule | 4,110 | 29.87 | 32.30 |
| PROTEINS | Biochemical Molecule | 1,113 | 39.06 | 72.82 |
| DD | Biochemical Molecule | 1,178 | 284.32 | 715.66 |
| MUTAG | Biochemical Molecule | 188 | 17.93 | 19.79 |
| COLLAB | Social Network | 5,000 | 74.49 | 2,457.78 |
| RDT-B | Social Network | 2,000 | 429.63 | 497.75 |
| RDT-M5K | Social Network | 4,999 | 508.52 | 594.87 |
| IMDB-B | Social Network | 1,000 | 19.77 | 96.53 |
| Github | Social Network | 12,725 | 113.79 | 234.64 |

*Table A2.* Statistics of MoleculeNet dataset for the downstream transfer learning. # means Number of.

| Dataset | Category | Utilization | # Graphs | Avg. # Nodes | Avg. # Edges |
|---|---|---|---|---|---|
| ZINC-2M | Biochemical Molecule | Pre-Traning | 2,000,000 | 26.62 | 57.72 |
| PPI-306K | Biology Networks | Pre-Traning | 306,925 | 39.82 | 729.62 |
| BBBP | Biochemical Molecule | FineTuning | 2,039 | 24.06 | 51.90 |
| TOX21 | Biochemical Molecule | FineTuning | 7,831 | 18.57 | 38.58 |
| TOXCAST | Biochemical Molecule | FineTuning | 8,576 | 18.78 | 38.52 |
| SIDER | Biochemical Molecule | FineTuning | 1,427 | 33.64 | 70.71 |
| CLINTOX | Biochemical Molecule | FineTuning | 1,477 | 26.15 | 55.76 |
| MUV | Biochemical Molecule | FineTuning | 93,087 | 24.23 | 52.55 |
| HIV | Biochemical Molecule | FineTuning | 41,127 | 25.51 | 54.93 |
| BACE | Biochemical Molecule | FineTuning | 1,513 | 34.08 | 73.71 |
| PPI | Biology Networks | FineTuning | 88,000 | 49.35 | 890.77 |

### A.2.2. MOLECULENET AND PPI DATASETS

We follow the same transfer learning setting of Hu et al. (2020b). For pre-training, we use 2 million unlabeled molecules sampled from the ZINC15 database (Sterling & Irwin, 2015) for the chemistry domain, and 395K unlabeled protein ego-networks derived from PPI networks (Mayr et al., 2018) representing 50 species for the biology domain. During the

[2] https://chrsmrrs.github.io/datasets/

[3] https://pytorch-geometric.io

fine-tuning stage, we use 8 larger binary classification datasets available in MoleculeNet (Wu et al., 2018) for the chemistry domain and PPI networks (Zitnik et al., 2019) for the biology domain. All these datasets are obtained from SNAP [4]. Summaries of these datasets are presented in Table A2.

#### A.2.3. OGB CHEMICAL MOLECULAR DATASETS

Open Graph Benchmark (OGB) chemical molecular datasets (Hu et al., 2020a) are employed for both graph classification and regression tasks in an unsupervised learning setting. OGB [5] hosts datasets designed for chemical molecular property classification and regression, as summarized in Table A3. These datasets can be accessed through the OGB platform, and are also available within the PyG library [6].

*Table A3.* Statistics of OGB chemical molecular dataset for both graph classification and regression tasks.

| Dataset | Task Type | # Graphs | Avg. # Nodes | Avg. # Edges | # Task |
|---|---|---|---|---|---|
| ogbg-molesol | Regression | 1,128 | 13.3 | 13.7 | 1 |
| ogbg-molipo | Regression | 4,200 | 27.0 | 29.5 | 1 |
| ogbg-molfreesolv | Regression | 642 | 8.7 | 8.4 | 1 |
| ogbg-molbace | Classification | 1,513 | 34.1 | 36.9 | 1 |
| ogbg-molbbbp | Classification | 2,039 | 24.1 | 26.0 | 1 |
| ogbg-molclintox | Classification | 1,477 | 26.2 | 27.9 | 2 |
| ogbg-moltox21 | Classification | 7,831 | 18.6 | 19.3 | 12 |
| ogbg-molsider | Classification | 1,427 | 33.6 | 35.4 | 27 |

### A.3. Detailed Model Configurations, Training and Evaluation Process

#### A.3.1. PRE-ANALYSIS EXPERIMENTAL SETTINGS

We outline detailed settings of the pre-analysis experiment for replication purposes. Specifically, our experiments are conducted on the IMDB-B dataset. Both the GraphCL and AD-GCL methods utilized a GIN encoder with identical architecture, including a graph-graph level contrastive loss, mean pooling readout, as well as hyperparameters, such as 2 convolutional layers with an embedding dimension of 32). Both strategies underwent 100 training iterations to acquire graph representations, which were subsequently evaluated by using them as features for a downstream SVM classifier. We employ spectral clustering (Ng et al., 2001) to obtain node cluster labels. We first apply spectral clustering on the original graph to obtain node cluster labels, using the Eigengap heuristic (von Luxburg, 2007) to determine the optimal number of clusters. Subsequently, maintaining the fixed number of clusters, we apply spectral clustering separately on the two perspectives of the augmented graphs, GraphCL + Cluster and GraphCL + Destroy, to obtain node cluster labels in the augmented views after graph augmentation. By comparing these labels with those obtained from the original graph, we approximate the trend of community average changes for nodes.

#### A.3.2. UNSUPERVISED LEARNING SETTINGS

For unsupervised learning, we employ a 2-layer GIN (Xu et al., 2019) encoder with a 2-layer MLP for projection, followed by a mean polling readout function. The embedding size was set to 256 for both TU and OGB datasets. We conduct training for 100 epochs with a batch size of 256, utilizing the Adam optimizer with a learning rate of $0.01$. We adopt the provided data split for the OGB datasets and employ 10-fold cross-validation for the TU datasets, as the split is not provided. Initially, we pre-train the encoder using unlabeled data with a contrastive loss. Then, we fix the encoder and train with an SVM classifier for unsupervised graph classification or a linear prediction layer for unsupervised graph regression to evaluate the performance. Experiments are performed 10 times with mean and standard deviation of Accuracy(%) scores for TUDataset classification, RMSE scores for OGB datasets regression, and ROC-AUC (%) for OGB datasets classification.

[4] https://snap.stanford.edu/gnn-pretrain/
[5] https://ogb.stanford.edu/
[6] https://pytorch-geometric.io

#### A.3.3. SEMI-SUPERVISED LEARNING SETTINGS

Following GraphCL (You et al., 2020), we employ a 10-fold cross-validation on each of the TU datasets using a ResGCN (Pei et al., 2021) classifier. For each fold, 10% of each data is designed as labeled training data and 10% as labeled testing data. These splits are randomly selected using the StratifiedKFold method [7]. During pre-training, we tune the learning rate in the range {0.01, 0.001, 0.0001} (using Adam optimizer) and the number of epochs in {20, 40, 60, 80, 100} through grid search. For fine-tuning, an additional linear graph prediction layer is added on top of the encoder to map the representations to the task labels. Experiments are performed 10 times with mean and standard deviation of Accuracy(%) scores.

#### A.3.4. TRANSFER LEARNING SETTINGS

In transfer learning, we first conduct self-supervised pre-training on the pre-processed ZINC-2M or PPI-306K dataset for 100 epochs. Subsequently, we fine-tune the backbone model (GIN encoder used in (Hu et al., 2020b)) on binary classification biochemical datasets. During the fine-tuning process, the encoder is equipped with an additional linear graph prediction layer on top, facilitating the mapping of representations to the task labels. Experiments are conducted 10 times, and the mean and standard deviation of ROC-AUC(%) scores are reported.

#### A.3.5. ADVERSARIAL LEARNING SETTINGS

In robustness experiments, we conduct random noise attacks by introducing edge perturbations to the graph topology, and introducing Gaussian noise to the attributes of the input graph. Each experiment is performed within the context of unsupervised graph classification across various TU Datasets. Specifically, we first train the model using contrastive loss, following the same approach as in the unsupervised setting. During evaluation, we poison the topology and features of the input graph, and then asses its performance using an SVM classifier. We vary the ratio of added noise, denoted as $\sigma$, ranging from 0.05 to 0.3 with a step of 0.05. In the topological robustness experiment, we randomly add and remove $\sigma * m$ edges, where $m$ is the number of edges in a single graph. In the feature robustness experiment, we randomly select $\sigma * n * d$ positions to introduce noises, where $n$ is the number of nodes, and $d$ is the dimension of features.

#### A.3.6. SYNTHETIC DATASET SETTINGS

For the experiment depicted in Figure 4, we randomly generate 1,000 synthetic graphs using Random Partition Graph method (Kim & Oh, 2021) [8] and train models in an unsupervised learning manner. The parameters for graphs generation are set as follows: $num_class$=8, $num_nodes_per_class$=30, $node_homophily_ratio$=0.96, $average_degree$=5. After training, we select one graph to show the augmented edges, where green lines represent edge dropping and red lines represent edge adding, as shown in Figure 4(B). We compare it with the random edge perturbation in Figure 4(C) GraphCL. Furthermore, Figure 4(D) displays the inverted probability $(1 - p)$ for feature masking. The feature is obtained through the eigendecomposition of the corresponding adjacency matrix.

## B. More Discussion on Existing GCL Methods

### B.1. Introduction to Selected Best-performing Baselines

Here, we briefly introduce some important baselines for graph self-supervised learning.

- **MVGRL** (Hassani & Ahmadi, 2020) MVGRL maximizes the mutual information between the local Laplacian matrix and a global diffusion matrix.
- **InfoGraph** (Sun et al., 2020) InfoGraph maximizes the mutual information between the graph-level representation and the representations of substructures of different scales.
- **Attribute Masking** (Hu et al., 2020b) Attribute Masking learns the regularities of the node/edge attributes distributed over graph structure to capture inherent domain knowledge.
- **Context Prediction** (Hu et al., 2020b) Context Prediction predicts the surrounding graph structures of subgraphs to pre-train a backbone GNN, so that it maps nodes appearing in similar structural contexts to nearby representations.

[7] https://scikit-learn.org
[8] https://pytorch-geometric.io

- **GraphCL** (You et al., 2020) GraphCL learns unsupervised representations of graph data through contrastive learning with random graph augmentations.
- **JOAO** (You et al., 2021) JOAO leverages GraphCL as the baseline model and automates the selection of augmentations when performing contrastive learning.
- **SEGA** (Wu et al., 2023b) SEGA explores an anchor view that maintains the essential information of input graphs for graph contrastive learning, based on the theory of graph information bottleneck. Moreover, based on the structural information theory, we present a practical instantiation to implement this anchor view for graph contrastive learning.
- **AD-GCL** (Suresh et al., 2021) AD-GCL aims to avoid capturing redundant information during the training by optimizing adversarial graph augmentation strategies in GCL and designing a trainable edge-dropping graph augmentation.
- **AutoGCL** (Yin et al., 2022) AutoGCL employs a set of learnable graph view generators with node dropping and attribute masking and adopts a joint training strategy to train the learnable view generators, the graph encoder, and the classifier in an end-to-end manner. We reproduce this algorithm by removing the feature expander part for the unsupervised setting.
- **GCS** (Wei et al., 2023) GCS utilizes a gradient-based approach that employs contrastively trained models to preserve the semantic content of the graph. Subsequently, augmented views are generated by assigning different drop probabilities to nodes and edges.
- **GCL-SPAN** (Lin et al., 2023) GCL-SPAN develops spectral augmentation to guide topology augmentations. Two views are generated by maximizing and minimizing spectral changes, respectively. The probability matrices are pre-computed and serve as another input for the GCL framework. We reproduce it by adding 10-fold cross-validation and report the mean accuracy.

### B.2. Summary of the Most Similar Studies

- Compared with spectrum-based methods such as GCL-SPAN, GAME (Liu et al., 2022a), SGCL (Ghose et al., 2023) and $\text{SFA}_{\text{Info}}$ (Zhang et al., 2023b). GCL-SPAN explores high-level graph structural information by disturbing it, i.e., they design their augmented view by corrupting what they want to retain. In contrast, CI-GCL preserves the core information between two augmented graphs to help the encoder better understand what should be retrained during graph augmentation. GAME focuses on the notion that spectrum changes of high-frequency should be larger than spectrum changes of low-frequency part, while CI-GCL considers all spectrum changes as a whole and endeavors to maintain community invariance during graph augmentation. SGCL aims to generate cropping and reordering augmented graphs, and $\text{SFA}_{\text{Info}}$ aims to balance and align data distribution shift between the original and augmented graph views. In contrast, CI-GCL aims to preserve community structural information by minimizing spectral changes. Another main difference between CI-GCL and these spectrum-based methods is that CI-GCL incorporates learnable graph augmentation, which makes it more flexible and generalizable for many different datasets.
- Compared with community-based GCL methods such as gCooL (Li et al., 2022a), ClusterSCL (Wang et al., 2022), and CSGCL (Li et al., 2023). These methods first attempt to obtain community labels for each node using clustering methods such as K-means. Then they consider these community labels as the supervised signal to minimize the changes of community labels before and after the GCL framework. However, since they still employ random graph augmentation, they disrupt community structure during graph augmentation, which causes them to underperform some basic GCL baselines such as GraphCL on graph classification and node classification. Different from these methods, the proposed CI constraint can be applied for graph augmentation to improve effectiveness.
- Compared with other GCL methods with learnable graph augmentation, such as AutoGCL and AD-GCL. To the best of our knowledge, we propose the first unified community-invariant constraint for learnable graph augmentation. With the proposed CI constraint, we can better maintain high-level graph information (i.e., community), which can enhance effectiveness, generalizability, and robustness for downstream tasks.

### B.3. The Difference between CI-GCL, CI+AD-GCL and CI+AutoGCL

- Different Graph Augmentation Operations: CI+AD-GCL solely considers the edge dropping augmentation, and CI+AutoGCL solely considers the node dropping operation, while CI-GCL encompasses nearly all augmentation operations, including edge perturbation, node perturbation, and feature masking, significantly improving the our model's classification accuracy. Furthermore, our proposed community-invariant constraints are applied to both topology and feature augmentation, thereby enhancing the model's robustness against different types of noise.
- Different Graph Augmentors: Both CI+AD-GCL and CI+AutoGCL adopts GNN+MLP as graph augmentors. Specifically, they utilize the original graph as input for the GNN encoder to obtain node representations. In contrast, CI-GCL employs spectral decomposition+MLP as the graph augmentor. Node representations are obtained through spectral decomposition, and the decomposed eigenvalues are further utilized for the community-invariant constraint. This approach is chosen because Community-Invariant constraint and spectrum information are strongly related.
- Different Optimization Strategy: CI+AD-GCL, unlike CI-GCL and CI+AutoGCL, does not employ joint learning for updating. The parameters of the Encoder and Augmentor are updated separately. In contrast, both CI-GCL and CI+AutoGCL utilize joint learning to simultaneously update the Augmentor and Encoder, offering greater potential to achieve optimal performance.

## C. Parameter Sensitive Experiments

### C.1. Hyperparameters Sensitivity in Overall Objective Function

Before delving into the sensitivity of hyperparameters, we first introduce all the hyperparameters used in this study. The overall objective function of CI-GCL in this study can be formulated as follows:

$$\min\ \mathcal{L}_{\text{overall}} = \mathcal{L}_{\text{GCL}} - \boldsymbol{\alpha}\mathcal{L}_{\text{TA}}(\boldsymbol{\Delta}^{\text{TA}}) - \boldsymbol{\beta}\mathcal{L}_{\text{FA}}(\boldsymbol{\Delta}^{\text{FA}}), \tag{C.1}$$

$$\text{where}\quad \mathcal{L}_{\text{TA}}(\boldsymbol{\Delta}^{\text{TA}}) = \mathcal{L}_{\text{ED}}(\boldsymbol{\Delta}^{\text{ED}}) + \mathcal{L}_{\text{ND}}(\boldsymbol{\Delta}^{\text{ND}}) - \mathcal{L}_{\text{EA}}(\boldsymbol{\Delta}^{\text{EA}}), \tag{C.2}$$

$$\text{and}\quad \mathcal{L}_{\text{FA}}(\boldsymbol{\Delta}^{\text{FA}}) = \mathcal{L}_{\text{FM}}(\boldsymbol{\Delta}^{\text{FM}}) \tag{C.3}$$

where TA and FA represent topology augmentation and feature augmentation, respectively. $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are hyperparameters controlling the strength of community-invariant constraints on topology and feature augmentation, respectively. Another important hyperparameter $K$, as mentioned in Section 4.4, denotes the number of eigenvalues and eigenvectors.

We evaluate parameter sensitivity by varying the target hyperparameter while keeping other hyperparameters fixed. We select three datasets MUTAG, IMDB-B, and PROTEINS, from the TU datasets to evaluate parameter sensitivity in the unsupervised setting using 10-fold cross-validation. In our experimental settings, we adopt a 2-layer GIN encoder with a 2-layer MLP for projection, followed by a mean pooling readout function. The embedding size is fixed at 256. We conduct training for 100 epochs with a batch size of 256. Throughout all experiments, we employ the Adam optimizer with a learning rate of $0.01$.

**Effect of $K$** Fixing $\boldsymbol{\alpha} = 0.8$ and $\boldsymbol{\beta} = 1.0$ unchanged, we vary $K$ from 2 to 20 with a step of 1 to evaluate its performance. As shown in Figure C1 (the first column), CI-GCL demonstrates significant improvement as the selected number increases from 2 to 6. We attribute this improvement to the small number of spectrum that only contain a rough community structure of the graph. Conducting topology or feature augmentations constrained by such an incomplete community structure may further compromise the distinguishability of the generated positive views. Therefore, it is important to use a sufficient number of spectrum to obtain a confident augmentation view. Figure C1 demonstrates that increasing the number of spectrum substantially enhances performance, with the best results achieved at 6. While MUTAG, IMDB-B, and PROTEINS are relatively small datasets, we recommend using a larger number of spectrum for larger graphs.

**Effect of $\boldsymbol{\alpha}$** Fixing $\boldsymbol{\beta} = 1.0$ and $K = 6$ unchanged, We vary $\boldsymbol{\alpha}$ from $0.1$ to $50$ with a step of $0.1$ (from $0.1$ to $1$), and $1$ (from $1$ to $5$), and $10$ (from $10$ to $30$) to evaluate its performance. As shown in Figure C1 (the second column), CI-GCL demonstrates significant improvement when the weight of the topological constraint increases from $0.1$ to $3$. We attribute this improvement to the weight of $\mathcal{L}_{\text{TA}}$ controlling the rate of community invariance. However, if we further increase the value, we observe a decreasing trend in performance. This may occur because the topological community constraint may dominate the overall loss, leading to overfitting of the topological augmentation view. We recommend setting the best parameter for $\alpha$ within the range of $(0.7 \sim 1.0)$. For datasets with an average number of nodes larger than 40, we suggest setting $\alpha$ within the range of $(2 \sim 5)$.

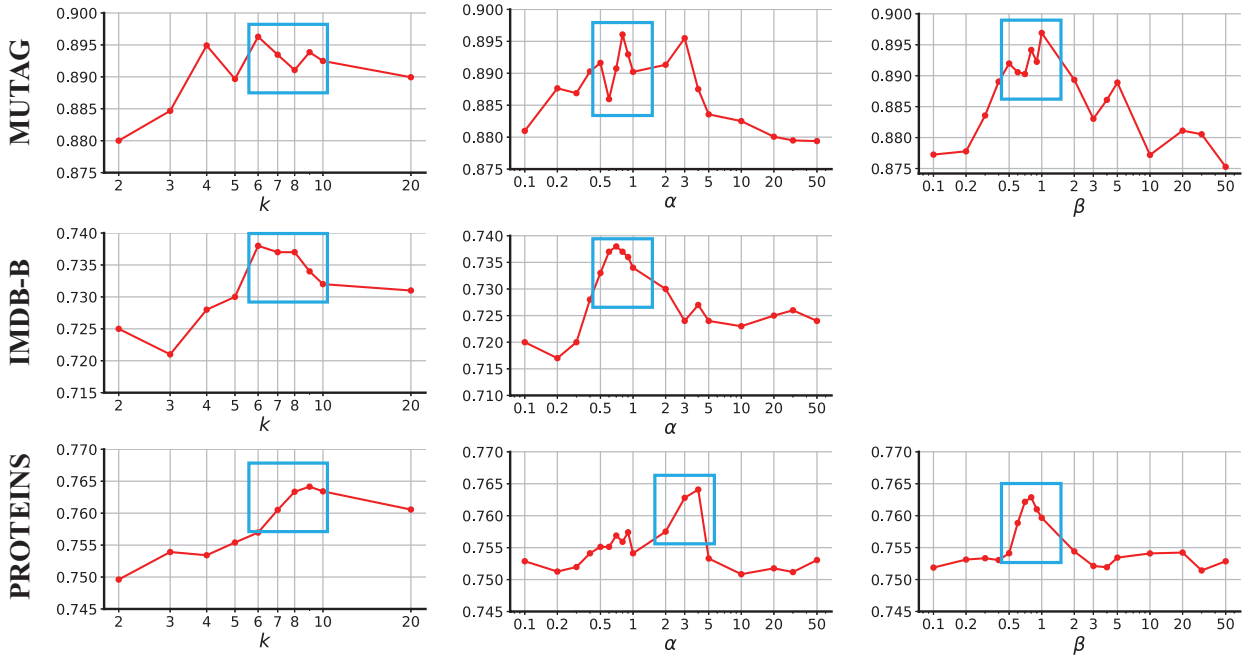


*Figure C1.* The parameter sensitivity involves the selected number of spectrum $K$, the balance weight of the topological constraint $\boldsymbol{\alpha}$, and the balance weight of the feature constraint $\boldsymbol{\beta}$.

**Effect of $\boldsymbol{\beta}$** Fixing $\boldsymbol{\alpha} = 0.8$ and $K = 6$ unchanged, we vary $\beta$ from $0.1$ to $50$ using the same step range as $\boldsymbol{\alpha}$. As shown in Figure C1 (the third column), CI-GCL demonstrates significant improvement when the weight of feature-wise constraint increases from $0.1$ to $1.0$. We attribute this improvement to the weight of $\mathcal{L}_{FM}$, which controls the selection of important features. However, if we further increase the value, we observe a decreasing trend in performance. This may also occur because the feature community constraint may dominate the overall loss, leading to overfitting of the feature augmentation view. We recommend setting the best parameter for $\beta$ within the range of $(0.5 \sim 1.0)$.

### C.2. Analysis of Perturbation Strength

The value of $\psi$ controls the strength of perturbation when generating augmented graphs. A larger $\psi$ value indicates that more edges will be either dropped or added. The number of perturbed edges in each graph is restricted to $\psi = \sigma_e \times m$, where $m$ represents the number of edges in the input graph and $\sigma_e$ denotes the perturbation rate. To analyze the impact of perturbation strength, we examine the influence of the perturbation ratio $\sigma_e$ on augmentations. We vary $\sigma_e$ from 0.05 to 0.5 with a step size of 0.05 across datasets NCI1, PROTEINS, DD, and MUTAG. The performance comparison is illustrated in Figure C2, conducted within the context of an unsupervised graph classification task. We recommend selecting the optimal perturbation strength within the range of $0.1$ to $0.3$.

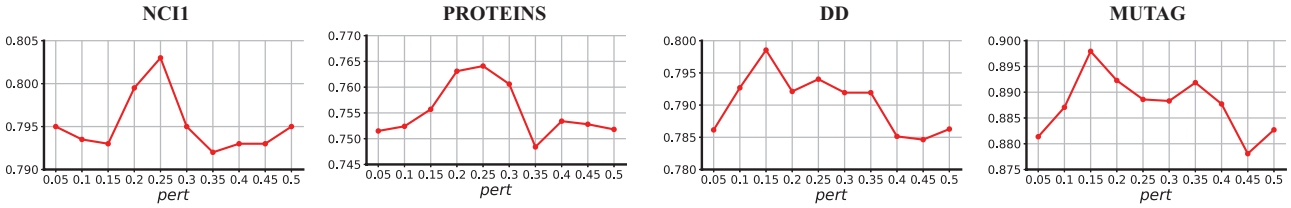


*Figure C2.* Graph classification performance when tuning perturbation ratio $\sigma_e$.

*Table D1.* Unsupervised graph classification. We pre-train using the whole dataset to learn graph embeddings and feed them into a downstream SVM classifier with 10-fold cross-validation.

| Method | NCI1 | PROTEINS | DD | MUTAG | COLLAB | RDT-B | RDT-M5K | IMDB-B | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| GL | - | - | - | 81.66±2.1 | - | 77.34±0.1 | 41.01±0.1 | 65.87±0.9 | - |
| WL | 80.01±0.5 | 72.92±0.5 | - | 80.72±3.0 | - | 68.82±0.4 | 46.06±0.2 | 72.30±3.4 | - |
| DGK | 80.31±0.4 | 73.30±0.8 | - | 87.44±2.7 | - | 78.04±0.4 | 41.27±0.2 | 66.96±0.5 | - |
| node2vec | 54.89±1.6 | 57.49±3.5 | - | 72.63±10.2 | - | - | - | - | - |
| sub2vec | 52.84±1.5 | 53.03±5.5 | - | 61.05±15.8 | - | 71.48±0.4 | 36.68±0.42 | 55.26±1.5 | - |
| graph2vec | 73.22±1.8 | 73.30±2.0 | - | 83.15±9.2 | - | 75.78±1.0 | 47.86±0.26 | 71.10±0.5 | - |
| InfoGraph | 76.20±1.0 | 74.44±0.3 | 72.85±1.7 | 89.01±1.1 | 70.65±1.1 | 82.50±1.4 | 53.46±1.0 | 73.03±0.8 | 74.02 |
| GraphCL | 77.87±0.4 | 74.39±0.4 | 78.62±0.4 | 86.80±1.3 | 71.36±1.1 | 89.53±0.8 | 55.99±0.3 | 71.14±0.4 | 75.71 |
| MVGRL | 76.64±0.3 | 74.02±0.3 | 75.20±0.4 | 75.40±7.8 | 73.10±0.6 | 82.00±1.1 | 51.87±0.6 | 63.60±4.2 | 71.48 |
| JOAO | 78.07±0.4 | 74.55±0.4 | 77.32±0.5 | 87.35±1.0 | 69.50±0.3 | 85.29±1.4 | 55.74±0.6 | 70.21±3.0 | 74.75 |
| SEGA | 79.00±0.7 | 76.01±0.4 | 78.76±0.6 | **90.21±0.7** | 74.12±0.5 | 90.21±0.7 | 56.13±0.3 | 73.58±0.4 | 77.25 |
| GCS❄ | 77.18±0.3 | 74.04±0.4 | 76.28±0.3 | 88.19±0.9 | 74.00±0.4 | 86.50±0.3 | 56.30±0.3 | 72.90±0.5 | 75.64 |
| GCL-SPAN❄ | 75.43±0.4 | 75.78±0.4 | 78.78±0.5 | 85.00±0.8 | 71.40±0.5 | 86.50±0.1 | 54.10±0.5 | 66.00±0.7 | 74.12 |
| AD-GCL❄ | 73.38±0.5 | 73.59±0.7 | 75.10±0.4 | 89.70±1.0 | 72.50±0.6 | 85.52±0.8 | 54.91±0.4 | 71.50±0.6 | 74.53 |
| AutoGCL❄ | 78.32±0.5 | 69.73±0.4 | 75.75±0.6 | 85.15±1.1 | 71.40±0.7 | 86.60±1.5 | 55.71±0.2 | 72.00±0.4 | 74.33 |
| W/O CI ON ALL | 78.67±0.9 | 74.89±0.9 | 78.33±0.3 | 86.53±1.8 | - | - | - | - | - |
| W/O CI ON TA | 79.98±0.6 | 75.74±0.9 | 78.84±0.3 | 87.65±0.5 | 73.49±0.9 | 88.93±0.5 | 55.36±0.3 | 70.34±0.8 | 76.29 |
| W/O CI ON FA | 80.02±0.3 | 75.99±1.4 | 78.75±0.8 | 87.56±1.4 | - | - | - | - | - |
| CI-GCL | **80.50±0.5** | **76.50±0.1** | **79.63±0.3** | 89.67±0.9 | **74.40±0.6** | **90.80±0.5** | **56.57±0.3** | **73.85±0.8** | **77.74** |

## D. More Experimental Results

### D.1. Complete Results for Unsupervised Learning

**Baselines.** We compare CI-GCL with kernel-based methods, such as Graphlet Kernel (GL) (Shervashidze et al., 2009), Weisfeiler-Lehman sub-tree kernel (WL) (Shervashidze et al., 2011), and Deep Graph Kernel (DGK) (Yanardag & Vishwanathan, 2015). Unsupervised graph learning methods like node2vec (Grover & Leskovec, 2016), sub2vec (Adhikari et al., 2018) and graph2vec (Narayanan et al., 2017). Classic GCL methods, such as MVGRL (Hassani & Ahmadi, 2020), InfoGraph (Sun et al., 2020), GraphCL (You et al., 2020), and JOAO (You et al., 2021), SEGA (Wu et al., 2023b). And learnable GCL methods, such as GCS (Wei et al., 2023), GCL-SPAN (Lin et al., 2023), AD-GCL (Suresh et al., 2021), AutoGCL (Yin et al., 2022). The complete results for unsupervised graph classification results are shown in Table D1. We also report the complete results on OGB dataset in Table D2.

*Table D2.* Unsupervised graph regression (measured by RMSE) and classification (measured by ROC-AUC) results on OGB datasets. We pre-train using the whole dataset to learn graph embeddings and feed them into a downstream classifier with scaffold split. Each experiment is performed 10 times and take the average accuracy as a result.

| Task | Regression (RMSE) | | | Classification (AUC) | | | | |
|---|---|---|---|---|---|---|---|---|
| Method | molesol | mollipo | molfreesolv | molbace | molbbbp | molclintox | moltox21 | molsider |
| InfoGraph | 1.344± 0.18 | 1.005±0.02 | 10.005±4.82 | 74.74±3.60 | 66.33±2.79 | 64.50±5.32 | 69.74±0.57 | 60.54±0.90 |
| GraphCL | 1.272±0.09 | 0.910±0.02 | 7.679±2.75 | 74.32±2.70 | 68.22±1.89 | 74.92±4.42 | 72.40±1.01 | 61.76±1.11 |
| MVGRL | 1.433±0.15 | 0.962±0.04 | 9.024±1.98 | 74.20±2.31 | 67.24±1.39 | 73.84±4.25 | 70.48±0.83 | 61.94±0.94 |
| JOAO | 1.285±0.12 | 0.865±0.03 | 5.131±0.72 | 74.43±1.94 | 67.62±1.29 | 78.21±4.12 | 71.83±0.92 | 62.73±0.92 |
| GCL-SPAN | 1.218±0.05 | **0.802±0.02** | 4.531±0.46 | 76.74±2.02 | 69.59±1.34 | 80.28±2.42 | 72.83±0.62 | 64.87±0.88 |
| AD-GCL | 1.217±0.09 | 0.842±0.03 | 5.150±0.62 | 76.37±2.03 | 68.24±1.47 | **80.77±3.92** | 71.42±0.73 | 63.19±0.95 |
| CI-GCL | **1.130±0.13** | 0.816±0.03 | **2.873±0.32** | **77.26±1.51** | **70.70±0.67** | 78.90±2.59 | **73.59±0.66** | **65.91±0.82** |

### D.2. Transfer learning setting

The full results for transfer learning are shown in Table D3.

*Table D3.* Transfer Learning is conducted with various manually designed pre-training schemes, with different manually designed pre-training schemes, following the pre-training setting and dataset splitting as outlined in (Hu et al., 2020b). Each experiment is repeated 10 times, with mean and standard deviation calculated for the ROC-AUC scores.

| **Pre-Train** | **ZINC 2M** | | | | | | | | **PPI-306K** | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Fine-Tune** | **BBBP** | **Tox21** | **ToxCast** | **SIDER** | **ClinTox** | **MUV** | **HIV** | **BACE** | **PPI** | **Avg.** |
| No Pre-train | 65.8±4.5 | 74.0±0.8 | 63.4±0.6 | 57.3±1.6 | 58.0±4.4 | 71.8±2.5 | 75.3±1.9 | 70.1±5.4 | 64.8±1.0 | 66.72 |
| Infomax | 68.8±0.8 | 75.3±0.5 | 62.7±0.4 | 58.4±0.8 | 69.9±3.0 | 75.3±2.5 | 76.0±0.7 | 75.9±1.6 | 64.1±1.5 | 69.60 |
| EdgePred | 67.3±2.4 | 76.0±0.6 | 64.1±0.6 | 60.4±0.7 | 64.1±3.7 | 74.1±2.1 | 76.3±1.0 | 79.9±0.9 | 65.7±1.3 | 69.76 |
| AttrMasking | 64.3±2.8 | 76.7±0.4 | 64.2±0.5 | 61.0±0.7 | 71.8±4.1 | 74.7±1.4 | 77.2±1.1 | 79.3±1.6 | 65.2±1.6 | 70.48 |
| ContextPred | 68.0±2.0 | 75.7±0.7 | 63.9±0.6 | 60.9±0.6 | 65.9±3.8 | 75.8±1.7 | 77.3±1.0 | 79.6±1.2 | 64.4±1.3 | 70.16 |
| GraphCL | 69.7±0.7 | 73.9±0.7 | 62.4±0.6 | 60.5±0.9 | 76.0±2.7 | 69.8±2.7 | 78.5±1.2 | 75.4±1.4 | 67.9±0.9 | 70.45 |
| MVGRL | 69.0±0.5 | 74.5±0.6 | 62.6±0.5 | 62.2±0.6 | 77.8±2.2 | 73.3±1.4 | 77.1±0.6 | 77.2±1.0 | 68.7±0.7 | 71.37 |
| JOAO | 70.2±1.0 | 75.0±0.3 | 62.9±0.5 | 60.0±0.8 | 81.3±2.5 | 71.7±1.4 | 76.7±1.2 | 77.3±0.5 | 64.4±1.4 | 71.05 |
| SEGA | 71.9±1.1 | 76.7±0.4 | 65.2±0.9 | 63.7±0.3 | **85.0±0.9** | 76.6±2.5 | 77.6±1.4 | 77.1±0.5 | 68.7±0.5 | 73.61 |
| GCS❄ | 72.5±0.5 | 74.4±0.4 | 64.4±0.2 | 61.9±0.4 | 66.7±1.9 | **77.3±1.7** | 78.7±1.4 | 82.3±0.3 | 70.3±0.5 | 72.05 |
| GCL-SPAN | 70.0±0.7 | **78.0±0.5** | 64.2±0.4 | 64.7±0.5 | 80.7±2.1 | 73.8±0.9 | 77.8±0.6 | 79.9±0.7 | 70.0±0.8 | 73.23 |
| AD-GCL❄ | 67.4±1.0 | 74.3±0.7 | 63.5±0.7 | 60.8±0.9 | 58.6±3.4 | 75.4±1.5 | 75.9±0.9 | 79.0±0.8 | 64.2±1.2 | 68.79 |
| AutoGCL❄ | 72.0±0.6 | 75.5±0.3 | 63.4±0.4 | 62.5±0.6 | 79.9±3.3 | 75.8±1.3 | 77.4±0.6 | 76.7±1.1 | 70.1±0.8 | 72.59 |
| **CI-GCL** | **74.4±1.9** | 77.3±0.9 | **65.4±1.5** | **64.7±0.3** | 80.5±1.3 | 76.5±0.9 | **80.5±1.3** | **84.4±0.9** | **72.3±1.2** | **75.11** |

### D.3. Adversarial Attacks Experiment

The full results are shown in Figure D1 and Figure D2. As the results show, CI-GCL exhibits greater robustness than other baselines against both topology-wise and feature-wise attacks.

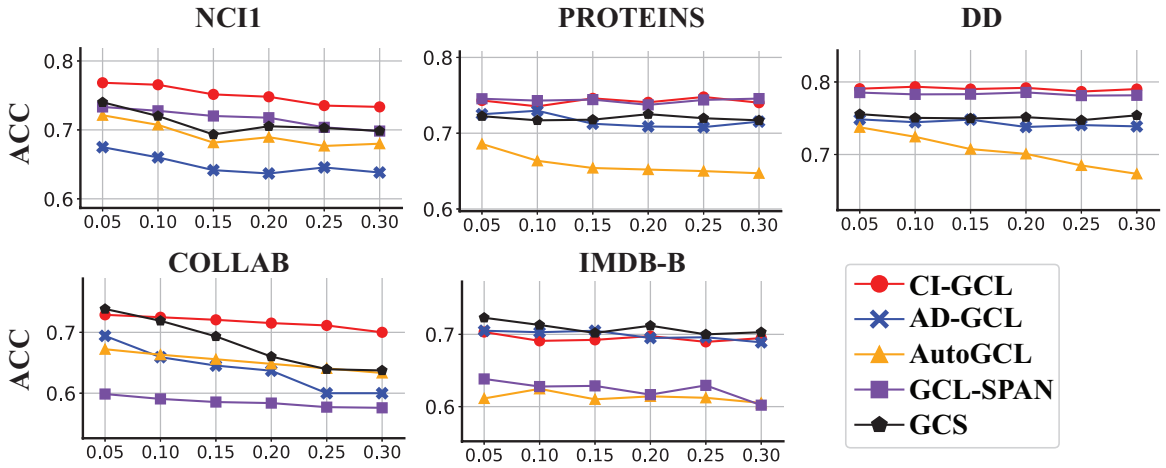


*Figure D1.* Accuracy under noise attack on graph's topology.

### D.4. SPATIAL BEHAVIOR OF SPECTRAL AUGMENTATION SPAN

The Case Study on Random Partition Graph. To intuitively demonstrate the spatial change caused by spectral augmentation, we present a case study on a random geometric graph in Figure D3. Specifically, Figure D3(A) depicts the original graph, while Figure D3(B) illustrates the perturbation probability with the community-invariant constraint. Figure D3(C) demonstrates the perturbation edges of uniform augmentations. We observe that with our community-invariant constraint, the cluster structure is preserved after augmentation, which assigns a higher probability to removing edges between clusters and adding edges within clusters. On the other hand, random topological augmentation uniformly removes and adds edges, causing the cluster effect to become blurred.

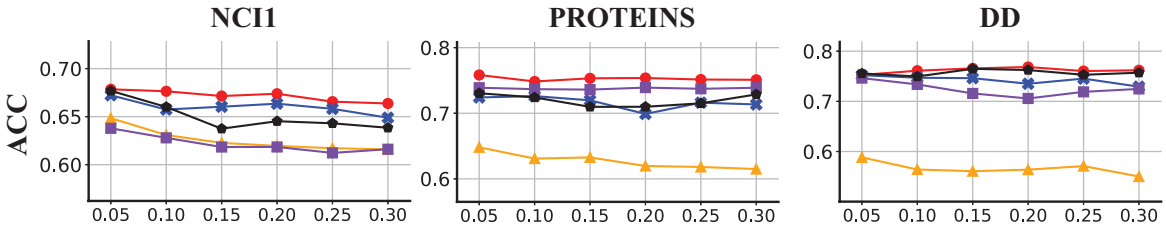


*Figure D2.* Accuracy under noise attack on graph's attributes.

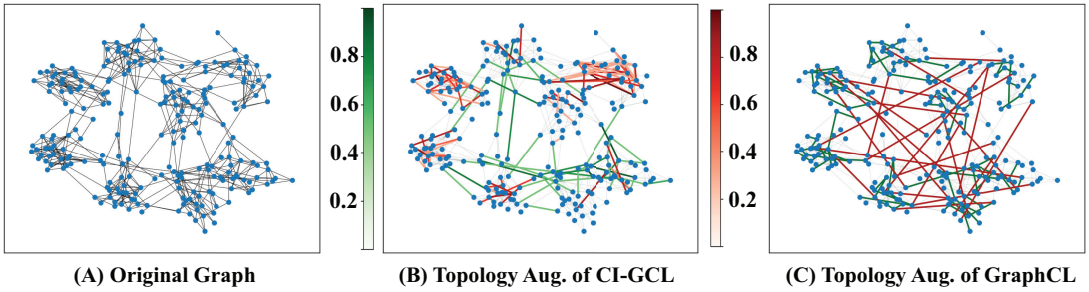


*Figure D3.* An case study of the community-invariant augmentation on a Random Partition Graph. In (B-C), green lines represent edge dropping operation and red lines represent edge adding operation.

## E. Algorithms

Algorithm 1 illustrates the detailed steps of developing CI-ACL.

**Algorithm 1:** Learnable Graph Contrastive Learning with Community-Invariant Constraint

**Input** : Datasets $\{G : G \in \mathcal{G}\}$, initial encoder $f_\theta(\cdot)$, readout function $r_\phi(\cdot)$, iteration number T.
**Hyperparameters** : Topological constraint coefficient $\boldsymbol{\alpha}$, feature constraint coefficient $\boldsymbol{\beta}$, number of eigenvector $K$.
**Output** : Optimized encoder $f_\theta(\cdot)$.

**for** $t \leftarrow$ *1 to T* **do**
  **for** *Sampled a minibatch of graphs* $\{G_n = (\mathbf{A}_n, \mathbf{X}_n) : n = 1, \ldots, N\}$ **do**
    **for** $n \leftarrow 1$ *to* $N$ **do**
      // Spectral Decomposition.
      $\mathbf{U} \leftarrow$ spectral decomposition($\mathbf{A}_n$)
      $\mathbf{F} \leftarrow$ SVD($\mathbf{X}_n$)
      // Graph Augmentation.
      **for** *any one edge* $A_{ij}$*, the* $i$*-th node, and feature* $X_{il}$ *in* $G_n$ **do**
        $\boldsymbol{\Delta}^{\text{EP}}_{ij}$ = Gumble-Softmax(MLPs(Concat $(\mathbf{U}_{i\cdot}, \mathbf{U}_{j\cdot})$))
        $\boldsymbol{\Delta}^{\text{ND}}_{ij}$ = Gumble-Softmax(MLPs($\mathbf{U}_{i\cdot}$))
        $\boldsymbol{\Delta}^{\text{FM}}_{il}$ = Gumble-Softmax(MLPs($[\mathbf{F}_{i\cdot}, \mathbf{F}_{l\cdot}]$))
      **end**
      // Sampling Two Augmented Views.
      $t_1^{\text{TA}}(G_n) = \mathbf{A} + \mathbf{C} \circ \boldsymbol{\Delta}$, // Topology Augmentation.
      $t_2^{\text{FA}}(G_n) = \mathbf{X} + (-\mathbf{X}) \circ \boldsymbol{\Delta}^{\text{FM}}$ // Feature Augmentation.
      // GNNs Encoding Layers.
      Encode Topology View: $\mathbf{z}_n^{(1)} = r_\phi(f_\theta(t_1^{\text{TA}}(G_n))$
      Encode Feature View: $\mathbf{z}_n^{(2)} = r_\phi(f_\theta(t_2^{\text{FA}}(G_n)))$
      // Compute Community-Invariant Constraint.
      Compute $\mathcal{L}_{\text{TA}}(\boldsymbol{\Delta}^{\text{TA}})$ to measure the spectral change as Eqs. (9,10)
      Compute $\mathcal{L}_{\text{FA}}(\boldsymbol{\Delta}^{\text{FA}})$ to measure the spectral change as Eq. (13)
    **end**
    // Graph Contrastive loss for the minibatch
    Compute $\mathcal{L}_{\text{GCL}} = -\frac{1}{N}\sum_{n=1}^{N}\left(\log \frac{\exp(\text{sim}(\mathbf{z}_n^{(1)}, \mathbf{z}_n^{(2)}/\tau_2))}{\sum_{n'=1, n'\neq n}^{N} \exp(\text{sim}(\mathbf{z}_n^{(1)}, \mathbf{z}_{n'}^{(2)})/\tau_2)}\right)$
    // Jointly Optimize the Overall Objective Function
    Jointly optimize by minimizing $\mathcal{L}_{\text{overall}} = \mathcal{L}_{\text{GCL}} - \boldsymbol{\alpha}\mathcal{L}_{\text{TA}}(\boldsymbol{\Delta}^{\text{TA}}) - \boldsymbol{\beta}\mathcal{L}_{\text{FA}}(\boldsymbol{\Delta}^{\text{FA}})$.
  **end**
**end**

# F. Proofs

## F.1. The Proof of Theorem 1 in the Draft

**Definition F1.** *(Eigenvalue Perturbation)* Assume matrix $\mathbf{A}'$, the altered portion is represented by $\Delta\mathbf{A} = \mathbf{A}' - \mathbf{A}$, and the changed degree is denoted as $\Delta\mathbf{D}$. According to matrix perturbation theory (Hogben, 2013), the change in amplitude for the $y$-th eigenvalue can be represented as:

$$\Delta\lambda_y = \lambda'_y - \lambda_y = \mathbf{u}_y^\top \Delta\mathbf{A}\mathbf{u}_y - \lambda_y \mathbf{u}_y^\top \Delta\mathbf{D}\mathbf{u}_y + \mathcal{O}(\|\Delta\mathbf{A}\|). \tag{F.1}$$

**Lemma F1.** If we only flip one edge $(i, j)$ on adjacency matrix $\mathbf{A}$, the change of $y$-th eigenvalue can be write as

$$\Delta\lambda_y = \Delta w_{ij} \left(2u_{yi} \cdot u_{yj} - \lambda_y \left(u_{yi}^2 + u_{yj}^2\right)\right), \tag{F.2}$$

where $u_{yi}$ is the $i$-th entry of $y$-th eigenvector $\mathbf{u}_y$, and $\Delta w_{ij} = (1 - 2A_{ij})$ indicates the edge flip, i.e $\pm 1$.

*Proof.* Let $\Delta\mathbf{A}$ be a matrix with only 2 non-zero elements, namely $\Delta A_{ij} = \Delta A_{ji} = 1 - 2A_{ij}$ corresponding to a single edge flip $(i, j)$, and $\Delta\mathbf{D}$ the respective change in the degree matrix, i.e. $\mathbf{A}' = \mathbf{A} + \Delta\mathbf{A}$ and $\mathbf{D}' = \mathbf{D} + \Delta\mathbf{D}$.

Denote with $\mathbf{e}_i$ the vector of all zeros and a single one at position $i$. Then, we have $\Delta\mathbf{A} = \Delta w_{ij}(\mathbf{e}_i\mathbf{e}_j^\top + \mathbf{e}_j\mathbf{e}_i^\top)$ and $\Delta\mathbf{D} = \Delta w_{ij}(\mathbf{e}_i\mathbf{e}_i^\top + \mathbf{e}_j\mathbf{e}_j^\top)$.

Based on eigenvalue perturbation formula (F.1) by removing the high-order term $\mathcal{O}(\|\Delta\mathbf{A}\|)$, we have:

$$\Delta\lambda_y \approx \mathbf{u}_y^\top(\Delta\mathbf{A} - \lambda_y\Delta\mathbf{D})\mathbf{u}_y \tag{F.3}$$

Substituting $\Delta\mathbf{A}$ and $\Delta\mathbf{D}$, we conclude Eq. F.2. □

**Theorem F1.** The constraint on the lowest $k$ eigenvalues of the normalized Laplacian matrix $\mathbf{L}_{\text{norm}}$ ensures the preservation of the community structure of nodes.

*Proof.* Firstly, we separate $\Delta\mathbf{A} = \Delta_{\mathbf{A}+} - \Delta_{\mathbf{A}-}$ , where $\Delta_{\mathbf{A}+}$ and $\Delta_{\mathbf{A}-}$ indicate which edge is added and deleted, respectively. To analyze the change of eigenvalues in spectral space corresponding to the perturbation of edges in spatial space, we first consider the situation that only augments one edge for both edge-dropping ($\Delta w_{ij} = -1$) and edge-adding ($\Delta w_{ij} = 1$):

① In the case of edge dropping ($\Delta w_{ij} = -1$ in Lemma F1), we have

$$\Delta\lambda_y = -2u_{yi} \cdot u_{yj} + \lambda_y \left(u_{yi}^2 + u_{yj}^2\right) \tag{F.4}$$

$$= \left(u_{yi} - u_{yj}\right)^2 + (\lambda_y - 1)\left(u_{yi}^2 + u_{yj}^2\right) \tag{F.5}$$

If we only drop the edge $(i, j)$ that makes a large change in the eigenvalues. We have the objective function as

$$\underset{\{i,j|\Delta w_{ij}=-1\}}{\arg\max} \sum_{y=1}^{n} |\Delta\lambda_y| = |\left(u_{yi} - u_{yj}\right)^2 + (\lambda_y - 1)\left(u_{yi}^2 + u_{yj}^2\right)| \tag{F.6}$$

$$\leq \sum_{y=1}^{n} |\left(u_{yi} - u_{yj}\right)^2| + \sum_{y=1}^{n} |\lambda_y - 1|\left(u_{yi}^2 + u_{yj}^2\right) \tag{F.7}$$

$$= \|\mathbf{U}_{i\cdot} - \mathbf{U}_{j\cdot}\|^2 + \sum_{y=1}^{n} |\lambda_y - 1|\left(u_{yi}^2 + u_{yj}^2\right) \tag{F.8}$$

$$\leq \|\mathbf{U}_{i\cdot} - \mathbf{U}_{j\cdot}\|^2 + \sum_{y=1}^{n} |\lambda_y - 1|\left(u_{y1}^2 + u_{y2}^2 + \cdots + u_{yn}^2\right) \tag{F.9}$$

$$= \|\mathbf{U}_{i\cdot} - \mathbf{U}_{j\cdot}\|^2 + \sum_{y=1}^{n} |\lambda_y - 1| \tag{F.10}$$

Notice that $\mathbf{u}_y$ is the $y$-th column of $\mathbf{U}$, so $u_{yi} = U_{iy}$ and $\mathbf{U}_{i,\cdot} = [u_{0i}, u_{1i}, \ldots, u_{ni}]$. From the first item in Eq. F.10, we prefer to select the nodes with larger distances in the eigenvector spaces, i.e. two nodes belonging to different communities (The relationship between eigenvector and community structure is proved in Theorem F4).

② In the case of edge adding ($\Delta w_{ij} = 1$ in Lemma F1), we have

$$\Delta\lambda_y = 2u_{yi} \cdot u_{yj} - \lambda_y \left(u_{yi}^2 + u_{yj}^2\right) \tag{F.11}$$

$$= -\left(u_{yi} - u_{yj}\right)^2 - \left(\lambda_y - 1\right)\left(u_{yi}^2 + u_{yj}^2\right) \tag{F.12}$$

If we only add the edge $(i, j)$ that makes a small change in the eigenvalues. We have the objective function as

$$\underset{\{i,j|\Delta w_{ij}=+1\}}{\arg\min} \sum_{y=1}^{n} |\Delta\lambda_y| = |\left(u_{yi} - u_{yj}\right)^2 + \left(\lambda_y - 1\right)\left(u_{yi}^2 + u_{yj}^2\right)| \tag{F.13}$$

$$\geq \sum_{y=1}^{n} |\left(u_{yi} - u_{yj}\right)^2| - \sum_{y=1}^{n} |1 - (\lambda_y)| \left(u_{yi}^2 + u_{yj}^2\right) \tag{F.14}$$

$$\geq \|\mathbf{U}_{i\cdot} - \mathbf{U}_{j\cdot}\|^2 - \sum_{y=1}^{n} |(1 - \lambda_y)| \left(u_{y1}^2 + u_{y2}^2 + \cdots + u_{yn}^2\right) \tag{F.15}$$

$$= \|\mathbf{U}_{i\cdot} - \mathbf{U}_{j\cdot}\|^2 - \sum_{y=1}^{n} |(1 - \lambda_y)| \tag{F.16}$$

From the first item in Eq. F.16, we prefer to select nodes with smaller distances in the eigenvector spaces, i.e. nodes belonging to one community (Theorem F4).

Previously, we have proven that the constraint on the lowest $k$ eigenvalues of $\mathbf{L}$ ensures the preservation of community structure when we only augment one edge. Next, we will demonstrate that the perturbation of more than one edge still aligns with this theory.

Suppose we augment $m$ edges, similar to Lemma F.2, we replace $\mathbf{\Delta A} = \sum_{(i,j)\in\{m \text{ edges}\}} \Delta w_{ij}(\mathbf{e}_i\mathbf{e}_j^\top + \mathbf{e}_j\mathbf{e}_i^\top)$, and $\mathbf{\Delta D} = \sum_{(i,j)\in\{m \text{ edges}\}} \Delta w_{ij}(\mathbf{e}_i\mathbf{e}_i^\top + \mathbf{e}_j\mathbf{e}_j^\top)$, we Substituting $\Delta\mathbf{A}$ and $\Delta\mathbf{D}$ of Eq. (F.3), we get:

$$\Delta\lambda_y = \sum_{(i,j)\in\{m \text{ edges}\}} \Delta w_{ij} \left(2u_{yi} \cdot u_{yj} - \lambda_y \left(u_{yi}^2 + u_{yj}^2\right)\right), \tag{F.17}$$

By replacing $\Delta w_{ij} \left(2u_{yi} \cdot u_{yj} - \lambda_y \left(u_{yi}^2 + u_{yj}^2\right)\right)$ with Eqs. (F.10, F.16), we could easily see that the community preserving theory is satisfied. □

### F.2. Proof of Theorem 2

**Theorem F2.** Given a bipartite matrix $\mathbf{B}$, we have the adjacency matrix of the bipartite graph as:

$$\mathbf{M} = \begin{bmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{0} \end{bmatrix}, \tag{F.18}$$

The eigenvalue of matrix $\mathbf{M}$ can be represented as $\lambda = 1 - \sigma$, where $\sigma$ is the singular value of $\mathbf{B}$. And the eigenvectors of matrix $\mathbf{M}$ can be represented as the concatenation of singular vectors of $\mathbf{B}$ and $\mathbf{B}^\top$ (Dhillon, 2001).

*Proof.* The Laplacian matrix and Degree matrix of $\mathbf{M}$ can be represented as $\mathbf{L}$ and $\mathbf{D}$. We can split these into:

$$\mathbf{L} = \begin{bmatrix} \mathbf{D}_1 & -\mathbf{B} \\ -\mathbf{B}^T & \mathbf{D}_2 \end{bmatrix}, \text{ and } \mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2 \end{bmatrix}, \tag{F.19}$$

where $\mathbf{D}_1$, $\mathbf{D}_2$ are diagonal matrices such that $D_1(i, i) = \sum_j B_{i,j}$ and $D_2(j, j) = \sum_i B_{i,j}$. Thus the generalized eigen problem $\mathbf{Lf} = \lambda\mathbf{Df}$ may be written as:

$$\begin{bmatrix} \mathbf{D}_1 & -\mathbf{B} \\ -\mathbf{B}^T & \mathbf{D}_2 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{D}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \tag{F.20}$$

Assuming that both $\mathbf{D}_1$ and $\mathbf{D}_2$ are nonsingular, we can rewrite the above equations as

$$\begin{aligned} \mathbf{D}_1^{1/2}\mathbf{x} - \mathbf{D}_1^{-1/2}\mathbf{B}\mathbf{y} &= \lambda \mathbf{D}_1^{1/2}\mathbf{x} \\ -\mathbf{D}_2^{-1/2}\mathbf{B}^\top\mathbf{x} + \mathbf{D}_2^{1/2}\mathbf{y} &= \lambda \mathbf{D}_2^{1/2}\mathbf{y} \end{aligned} \tag{F.21}$$

Letting $\mathbf{u} = \mathbf{D}_1^{1/2}\mathbf{x}$ and $\mathbf{v} = \mathbf{D}_2^{1/2}\mathbf{y}$, we can get

$$\begin{aligned} \mathbf{D}_1^{-1/2}\mathbf{B}\mathbf{D}_2^{-1/2}\mathbf{v} &= (1-\lambda)\mathbf{u} \\ \mathbf{D}_2^{-1/2}\mathbf{B}^\top\mathbf{D}_1^{-1/2}\mathbf{u} &= (1-\lambda)\mathbf{v} \end{aligned} \tag{F.22}$$

These are precisely the equations that define the singular value decomposition (SVD) of the normalized matrix $\mathbf{D}_1^{-1/2}\mathbf{B}\mathbf{D}_2^{-1/2}$. In particular, $\mathbf{u}$ and $\mathbf{v}$ are the left and right singular vectors of $\mathbf{D}_1^{-1/2}\mathbf{B}\mathbf{D}_2^{-1/2}$ respectively, while $(1-\lambda)$ is the corresponding singular value. Thus the $k$-th smallest eigenvalue of $\mathbf{L}_{\text{norm}}$ equals to $k$-th largest singular value of $\mathbf{D}_1^{-1/2}\mathbf{B}\mathbf{D}_2^{-1/2}$, $\lambda_k = 1 - \sigma_k$. Furthermore, the corresponding eigenvector of $\mathbf{L}_{\text{norm}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ is given by

$$\mathbf{f}_k = \begin{bmatrix} \mathbf{u}_k \\ \mathbf{v}_k \end{bmatrix} \tag{F.23}$$

There is a computational benefit on $\mathbf{B}$ that $\mathbf{B}$ is of size $n \times d$, and $\mathbf{L}$ is of size $(n+d) \times (n+d)$.

□

### F.3. Proof of Theorem 3

**Theorem F3.** The constraint on the highest $k$ singular values of the feature matrix $\mathbf{X}$ ensures the preservation of the bipartite community structure of both nodes and features.

*Proof.* Consider we have the feature matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, intuitively, the common features, containing the community structure, should be preserved as core components and the features that are irrelevant to community structure should be randomly masking. As (Nie et al., 2017) studied, we could apply spectral clustering on matrix $\widetilde{\mathbf{X}}$ to obtain the co-clustering of both nodes and features.

$$\widetilde{\mathbf{X}} = \begin{bmatrix} 0 & \mathbf{X} \\ \mathbf{X}^\top & 0 \end{bmatrix} \tag{F.24}$$

where $\widetilde{\mathbf{X}} \in \mathbb{R}^{(n+d)\times(n+d)}$. When we perform the eigen decomposition on $\widetilde{\mathbf{X}}$, the first $n$ rows of obtained eigenvectors represent the node embedding, and the rest $d$ rows are representations of features. Similar to the edge-dropping situation in Theorem F1, we could preserve the community structure by maximizing the spectral change of $\widetilde{\mathbf{X}}$. The objective function could be written as

$$\begin{aligned} \underset{\{i\in[1,\cdots,n], j\in[n+1,\cdots,n+d] \mid \Delta P_{ij}=0\}}{\arg\max} \sum_{y=1}^{n+d} |\Delta\lambda_y| &= |\left(f_{yi} - f_{yj)}\right)^2 + (\lambda_y - 1)\left(f_{yi}^2 + f_{yj}^2\right)|, && \text{(F.25)} \\ &\geq \sum_{y=1}^{n+d} |\left(f_{yi} - f_{yj}\right)^2| - \sum_{y=1}^{n+d} |1-\lambda_y|\left(f_{yi}^2 + f_{yj}^2\right) && \text{(F.26)} \\ &\geq \|\mathbf{F}_{i\cdot} - \mathbf{F}_{j\cdot}\|^2 - \sum_{y=1}^{n+d} |1-\lambda_y|\left(f_{y1}^2 + f_{y2}^2 + \cdots + f_{yn}^2\right) && \text{(F.27)} \\ &= \|\mathbf{F}_{i\cdot} - \mathbf{F}_{j\cdot}\|^2 - \sum_{y=1}^{n} |1-\lambda_y| && \text{(F.28)} \end{aligned}$$

where $\mathbf{f}_y$ and $\lambda_y$ are the $y$-th eigenvector and eigenvalue of $\widetilde{\mathbf{X}}$'s normalized Laplacian matrix. Similar to edge-dropping, we prefer to mask the feature $j$ for node $i$ when they do not belong to the same community, which means that the feature is irrelevant to the sample.

According to the relationship between eigenvectors and community structures (proved in Theorem F4), we only need to constrain the change of several lowest eigenvalues to preserve community structures. And then we prove that $\lambda_y = 1 - \sigma_y$ on Theorem F2, where $\sigma_y$ is the singular value of matrix $\mathbf{D}_1^{-1/2}\mathbf{X}\mathbf{D}_2^{-1/2}$. Therefore, the constraint on the highest $k$ singular values of $\mathbf{D}_1^{-1/2}\mathbf{X}\mathbf{D}_2^{-1/2}$ ensures the preservation of the bipartite community structure of both nodes and features.

□

### F.4. The relation between community and spectrum

**Theorem F4.** The spectral decomposition can indicate a relaxed solution of graph vertex partition. Given a graph $G$ with a Laplacian matrix $\mathbf{L}$, and the spectral decomposition is indicated as $\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where $\mathbf{D}$ is the degree matrix and $\mathbf{U}$, $\mathbf{\Lambda}$ are eigenvectors, eigenvalues correspondingly. The second smallest eigenvalue and its corresponding eigenvector indicate a bipartition of the graph.

*Proof.* Given a partition of nodes of a graph (split $V$ into two disjoint sets $S_A$ and $S_B$), let $\mathbf{x}$ be an indicator vector for the partition, $x_i = 1$ if node $i$ is in $S_A$, and $-1$, otherwise. Let $d_i$ be the degree of node $i$, and $w_{ij}$ is the weight of edge $(i, j)$. Based on (Shi & Malik, 1997), a normalized cut could be write as

$$Ncut(S_A, S_B) = \frac{\sum_{x_i>0, x_j<0} -w_{ij}x_ix_j}{\sum_{x_i>0} d_i} + \frac{\sum_{x_i<0, x_j>0} -w_{ij}x_ix_j}{\sum_{x_i>0} d_i}. \tag{F.29}$$

The optimal partition is computed by minimizing the normalized cut: $\min_{\mathbf{x}} Ncut(\mathbf{x})$. By setting $\mathbf{y} = (1 + \mathbf{x}) - b(1 - \mathbf{x})$, $k = \frac{\sum_{x_i>0} d_i}{\sum_i d_i}$ and $b = \frac{k}{1-k}$, we could rewrite it as

$$\min_{\mathbf{x}} Ncut(\mathbf{x}) = \min_{\mathbf{y}} \frac{\mathbf{y}^\top\mathbf{L}\mathbf{y}}{\mathbf{y}^\top\mathbf{D}\mathbf{y}}. \tag{F.30}$$

with the condition $y_i \in 1, -b$, and $\mathbf{y}^\top\mathbf{D}\mathbf{1} = 0$.

According to the Rayleigh quotient (Gene et al., 2013), we can minimize Eq. F.30 by solving the generalized eigenvalue system, if $\mathbf{y}$ is relaxed to take on real values.

$$\mathbf{L}\mathbf{y} = \lambda\mathbf{D}\mathbf{y} \tag{F.31}$$

By writing $\mathbf{z} = \mathbf{D}^{\frac{1}{2}}\mathbf{y}$, we could transform the Eq. F.31 to a standard eigensystem:

$$\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\mathbf{z} = \lambda\mathbf{z} \tag{F.32}$$

Because the Laplacian matrix is positive semidefinite (Pothen et al., 1990), we can easily verify that $\mathbf{z}_0 = \mathbf{D}^{\frac{1}{2}}\mathbf{1}$ is the smallest eigenvector of Eq. F.32 with eigenvalue 0. And correspondingly, $\mathbf{y}_0 = \mathbf{1}$ is the smallest eigenvector with an eigenvalue of 0 in the general eigensystem F.31, but do not satisfy the condition $\mathbf{y}^\top\mathbf{D}\mathbf{1} = 0$. According to the Lemma F2, we could know that the second smallest eigenvector $\mathbf{z}_1$ is the solution of Eq. F.30, because $\mathbf{z}_1^\top\mathbf{z}_0 = \mathbf{y}_1^\top\mathbf{D}\mathbf{1} = 0$ satisfy the condition in Eq. F.30.

Therefore, the second smallest eigenvalue and its corresponding eigenvector indicate a bipartition of the graph. While the next cut must be perpendicular to others, which is the third smallest eigenvalue corresponding eigenvector $\mathbf{z}_2^\top\mathbf{z}_1 = \mathbf{z}_2^\top\mathbf{z}_0$. Recursively, the $k$-dimensional eigenvectors can represent the community structure of a graph to some extent. □

**Lemma F2.** A simple fact about the Rayleigh quotient (Gene et al., 2013): Let $\mathbf{A}$ be a real symmetric matrix. Under the constraint that $\mathbf{x}$ is orthogonal to the $j-1$ smallest eigenvectors $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{j-1}$, the quotient $\frac{\mathbf{x}^\top\mathbf{A}\mathbf{x}}{\mathbf{x}^\top\mathbf{x}}$ is minimized by the next smallest eigenvector $\mathbf{x}_j$ and its minimum value is the corresponding eigenvalue $\lambda_j$.

## G. More Detailed Time Complexity Analysis

Assuming $M$ as the number of training epochs, $B$ as the batch size, $n$ as the average number of nodes in the graph, $d$ as the dimension of node features, $K$ as the number of selected minimum eigenvalues, $L$ as the number of layers in the GNN, $d_h$ as the dimension of the encoder hidden layer, and $|G|$ as the number of networks. The time complexity of each component of our pipeline ars shown as follows:

- For spectral decomposition: we apply the Lanczos algorithm (Parlett & Scott, 1979) to perform spectral decomposition once to obtain eigenvalues and eigenvectors, reducing the time complexity from $\mathcal{O}(|G|n^3)$ to $\mathcal{O}(|G|n^2K)$.
- For the MLP layer computes the probability of $m$ edges and $nd$ features: the topological part has a time complexity of $\mathcal{O}(MBmKd_h)$, and the feature part has a time complexity of $\mathcal{O}(MBndKd_h)$.
- For topology augmentation view construction: we employ the QR algorithm (Francis, 1961) to compute the eigenvalues of the topology augmented graph, reducing the time complexity to $\mathcal{O}(MBn^2)$ for the Community-Invariant loss.
- For feature augmentation view construction, we directly apply Truncated SVD (Halko et al., 2011) to decompose $X \in \mathbb{R}^{n\times d}$, reducing the time complexity from $\mathcal{O}(MB(n+d)^3)$ to $\mathcal{O}(MBnd\log K)$.
- For the GNN encoder part: time complexity is $\mathcal{O}(MBL(md_h + nd_h^2)$ according to (Blakely et al., 2021).
- For contrastive loss calculation: the time complexity is $\mathcal{O}(MB^2d_h)$.

Almost all graph contrastive learning frameworks, such as AD-GCL, and AutoGCL, share the same time complexity of step-2, step-5, and step-6. The additional time complexity of CI-GCL for view construction is $\mathcal{O}(MBn\max\{n, d\log K\})$. CI-GCL has an approximate similar time complexity to the most recent GCL methods with learnable graph augmentation, such as GCL-SPAN and GCS.