Spectrum Guided Topology Augmentation for Graph Contrastive Learning

Lu Lin[†§], Jinghui Chen[†], Hongning Wang[§]
[†]The Pennsylvania State University, [§]University of Virginia
{lulin,jzc5917}@psu.edu, hw5x@virginia.edu

Abstract

Graph contrastive learning (GCL) is a major self-supervised graph learning technique that aims to capture invariant properties of graphs via instance discrimination. Its performance heavily relies on the construction of multiple graph views yet it still remains unclear about *what makes effective topology augmentations*. Recent studies mainly perform topology augmentations in a uniformly random manner without considering graph properties. In this work, we aim to find principled ways for topology augmentations by exploring the invariance of graphs from the graph spectral perspective. Specifically, we propose a novel topology augmentation method guided by spectral change. Extensive experiments on both graph and node classification tasks demonstrate the effectiveness of our method in capturing the structural essence of graphs for self-supervised learning. The proposed method also brings promising performance in transfer learning and adversarial attack settings. We envision this work to provide a principled way for graph augmentation.

1 Introduction

Graph neural networks (GNNs) [26, 54, 60] have advanced graph representation learning in a (semi-) supervised manner, yet it requires supervised labels and may fail to generalize due to overfitting [43]. To obtain more generalizable, transferable, and robust representations, the self-supervised learning (SSL) paradigm has recently emerged which enables GNNs to learn from pretext tasks constructed on unlabeled graph-structured data [21, 20, 63, 23]. As the current state-of-the-art SSL technique, graph contrastive learning (GCL) has attracted the most attention due to its simplicity and remarkable empirical performance [55, 67, 17, 61, 48, 49].

A typical GCL method works by creating augmented views of the input graph and learning node (or graph) representations by contrasting related graph objects against unrelated ones. Different contrastive objects are studied, such as node-node [67, 68, 37], node-(sub)graph [56, 17, 47] and graph-graph [1, 49, 48] contrastive pairs. The goal of GCL is to maximize the congruence between the representations of graph objects in augmented views, following the mutual information maximization (InfoMax) principle [18]. This makes graph augmentation one of the most critical designs in GCL, as it determines the effectiveness of the contrastive objective. However, despite various GCL methods have been proposed, it remains a mystery about what makes the most effective graph augmentations.

Unlike images, which can be augmented by rotation or cropping to naturally highlight the main subject from the background, it is less intuitive and more challenging to augment graphs due to the complicated topology structure of diverse nature (e.g., citation networks [45], social networks [34], chemical and biomedical molecules [29, 20]). Most existing works perform topology augmentations in a uniformly random manner [62, 67, 49, 1]. Although such a strategy indeed achieves a certain level of empirical success, it is far from optimal: recent studies show that perturbations on different edges post *unequal* influence on the graph spectrum [8, 3] while the uniformly random edge perturbation adopted in many GCL methods treats all edges equally and ignores such differences. Given the

NeurIPS 2022 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2022).

importance of graph spectrum for spectral filters in GNNs [6], such a discrepancy between uniform edge perturbations and their non-uniform influence on the graph spectrum urges us to rethink a fundamental but not yet clearly answered question:

Will graph spectrum based topology augmentations be more effective for GCL?

In this paper, we answer this question affirmatively. By studying the influence of different edge perturbation strategies on graph spectrum, we observe a clear positive relationship between the overall spectral change on augmented graphs and the resulting GCL performance. This empirical finding further motivates us to propose a principled Graph Contrastive Learning scheme with Topology Augmentation guided by the Graph Spectrum, termed *GCL-TAGS*. Specifically, instead of perturbing edges uniformly at random, we search for graph augmentations that mostly change the graph spectrum of the input graph. By identifying sensitive edges where the graph spectrum is largely affected, GCL-TAGS allows the GNN encoder to focus on robust components (which can be hardly affected by small edge perturbations) in the spectral filters and to reduce its dependency on relatively vulnerable components (which can be easily affected). Therefore, the learned encoder captures the minimally sufficient information about the graph [51, 50] for the downstream tasks.

We extensively evaluate GCL-TAGS on various benchmark datasets consisting of social networks and molecules, which cover commonly seen graph learning tasks such as node classification, graph classification and regression. We also consider various settings to test the applicability of our proposed method including unsupervised learning, transfer learning and adversarial learning setting. In general, GCL-TAGS achieves remarkable performance gains compared to the state-of-the-art baselines. Meanwhile, GCL-TAGS is easy to plug and play with different GCL paradigms as it only requires a one-time pre-computation of the edge perturbation probability for topology augmentation.

2 Related Works

Existing graph SSL methods focus on *self-prediction* [14, 38, 13, 63, 23] and *contrastive learning*. We focus on contrastive learning, and will mainly discuss existing designs of topology augmentation.

Graph Contrastive Learning (GCL) leverages the InfoMax principle [18] to maximize the correspondence between related objects on the graph such that invariant property across objects is captured. Depending on how the positive objects are defined, one line of work treats different parts of a graph as positive pairs, while constructing negative examples from a corrupted graph [20, 22, 56, 37, 47]. In such works, contrastive pairs are defined as nodes [56] or substructures [47] v.s. the entire graph, and the input graph v.s. reconstructed graph [37]. The other line of works exploits *graph augmentation* to generate multiple views, which enable more flexible contrastive pairs [49, 1, 62, 17, 40, 48, 61, 9]. By generating augmented views, the GNN model is encouraged to encode crucial graph information that is invariant to different views. While both topology and feature augmentations are explored in prior GCL works, we focus on topology augmentation strategies. As a parallel effort in self-supervised learning, augmentation-free techniques [28, 58] recently arise, which avoid augmentation but requires special treatments (e.g., kNN search or clustering) to obtain positive and negative pairs.

Graph Topology Augmentation. The most widely adopted topology augmentation is the edge perturbation following *uniform distribution* [67, 49, 1, 62]. The underlying assumption is that each edge is equally important to the property of the input graph. However, a recent study shows that edge perturbations do not post equal influence to the graph spectrum [3] which summarizes a graph's structural property. To better preserve graph property that has been ignored by uniform perturbations, *domain knowledge* from network science is leveraged by considering the importance of edges measured via node centrality [68], the global diffusion matrix [17], and the random-walk based context graph [40]. While these works consider ad-hoc heuristics, our method targets the graph spectrum, which comprehensively summarizes global graph properties and plays a crucial role in the spectral filter of GNNs. To capture minimally sufficient information from the graph and remove redundancy that could compromise downstream performance, *adversarial training* strategy is paired with GCL for graph augmentation [48, 61, 9], following the information bottleneck (IB) [51] and InfoMin principle [50]. While the adversarial augmentation method requires frequent back-propagation during training, our method realizes a similar principle with a simpler but effective augmentation by maximizing the spectral change of graph with only one-time pre-computation. Recent analyses also find some data-centric properties which are essential for GCL when using generic graph augmentations, yielding insights about the necessity of inducing task-relevant variance

[52] and maintaining the low-frequency information [31]. The finding of maximizing the spectral difference between views contributes a new dimension of such properties to investigate GCL.

3 Preliminaries

Notations. We focus on connected undirected graphs $G = (\mathbf{X}, \mathbf{A})$ with n nodes and m edges, where $\mathbf{X} \in \mathbb{R}^{n \times d}$ describes node features, and $\mathbf{A} \in \mathbb{R}^{n \times n}$ denotes its adjacency matrix such that $A_{ij} = 1$ if an edge exists between node i and j, otherwise $A_{ij} = 0$. The unnormalized Laplacian matrix of the graph is defined as $\mathbf{L}_{\text{u}} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \text{diag}(\mathbf{A}\mathbf{1}_n)$ is the diagonal degree matrix with entry $D_{ii} = \sum_{i=1}^{n} A_{ij}$ and $\mathbf{1}_n$ being an all-one vector with dimension n. The normalized Laplacian matrix is further defined as $\mathbf{L}_{\text{norm}} = \text{Lap}(\mathbf{A}) = \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{I}_n$ is an $n \times n$ identity matrix.

Graph Spectrum. By treating node features as signals, one can apply graph signal processing (GSP) techniques to conduct graph filtering for representation learning. The most essential component of GSP is the graph shift operator (GSO), which commonly adopts the normalized Laplacian matrix $\mathbf{L}_{\text{norm}}$ and admits an eigendecomposition as $\mathbf{L}_{\text{norm}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\top}$. The diagonal matrix $\mathbf{\Lambda} = \text{eig}(\mathbf{L}_{\text{norm}}) = \text{diag}(\lambda_1, \ldots, \lambda_n)$ consists of the real eigenvalues which are known as *graph spectrum*, and the corresponding $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_n] \in \mathbb{R}^{n \times n}$ collecting the orthonormal eigenvectors are the *spectral bases*. Graph spectrum plays a significant role in analyzing and modeling graphs, as discussed in Appendix A. On one hand, it comprehensively summarizes important graph structural properties, including connectivity [5], clusterability [27] and diffusion distance [15]. On the other hand, at the essence of many graph models, including GNNs, is the *spectral filter* which is defined on the graph spectrum, and different filters can manipulate graph signals in various ways, such as smoothing and denoising [44], anomaly detection [33] and clustering [57].

Graph Representation Learning. Given a graph $G \in \mathcal{G}$, the goal of node representation learning is to train an encoder $f_\theta : \mathcal{G} \to \mathbb{R}^{n \times d'}$, such that $f_\theta(G)$ produces a low-dimensional vector for each node in G which can be served in downstream tasks, such as node classification. One can further obtain a graph representation by pooling the set of node representations via a readout function $g_\phi : \mathbb{R}^{n \times d'} \to \mathbb{R}^{d'}$, such that $g_\phi(f_\theta(G))$ outputs a low-dimensional vector for graph G which can be used in graph-level tasks such as graph classification or regression task.

Graph Contrastive Learning by Topology Augmentation. GCL methods generally apply graph augmentation to perturb the input graph and decrease the amount of information inherited from the original graph; then they leverage the InfoMax principle [18] over the perturbed graph views such that an encoder is trained to capture the remaining information [48]. Given a graph $G \in \mathcal{G}$ with adjacency matrix $\mathbf{A}$, we denote a *topology augmentation scheme* as $T(\mathbf{A})$ and a sampled augmented view as $t(\mathbf{A}) \sim T(\mathbf{A})$. GCL with two-branch augmentation can be formulated as the following problem:

$$\text{GCL} : \min_{\Theta} \mathcal{L}_{\text{GCL}}(t_1(\mathbf{A}), t_2(\mathbf{A}), \Theta), \text{ s.t. } t_i(\mathbf{A}) \sim T_i(\mathbf{A}), i \in \{1, 2\} \tag{1}$$

where the contrastive loss $\mathcal{L}_{\text{GCL}}$ measures the disagreement between representations from contrastive positive pairs, which can be defined among different levels of representations, such as node-node [67], graph-graph [48], and node-graph [56, 17] representations. The topology augmentation scheme determines a distribution from which perturbed graphs are sampled in augmented views, and its detailed formulation will be presented in Section 4.2.

4 Graph Contrastive Learning Guided by Graph Spectrum

In this section, we introduce our graph contrastive learning framework with topology augmentation guided by the change of graph spectrum (GCL-TAGS). We start with some empirical evidence on the relationship between the spectral change and GCL performance, then propose a new augmentation principle to maximize the spectral change when perturbing the graph.

4.1 Behavior of Edge Perturbation on Graph Spectrum

Given that graph spectrum is a comprehensive manifestation of graph structural properties [5, 27, 15], to further understand its role in GCL, we study how the behavior of edge perturbation on graph spectrum correlates with GCL performance.

Pre-analysis Setup. We take node representation learning on Cora dataset as an example, and adopt the same contrastive learning setup as in GRACE [67]. We consider two topology augmentation heuristics: 1) *uniform*: the original augmentation from GRACE [67] which removes edges with uniformly random probability; 2) *clustered*: a cluster-based strategy which removes edges between different clusters with a larger probability. The cluster-based strategy explicitly modifies the connectivity between clusters which may result in large change on the graph spectrum as suggested by recent studies [4, 30], thus it stands in sharp contrast to the uniform perturbations. For the two augmentation branches, one is fixed with the uniform version, and we compare the strategies when the other augmentation branch adopts the uniform or the clustered perturbation. Figure 1 shows their comparison with respect to *spectral change* (measured by the L_2 distance of graph spectrum between the original and the augmented graphs), and *F1 score* (measuring the downstream node classification performance). More experiment details are left in Appendix C.1.

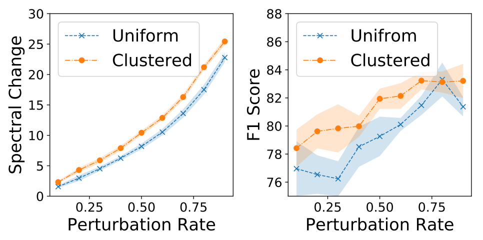


Figure 1: The spectral change (left) and downstream task performance (right) of two augmentation strategies.

Remarks. From Figure 1, we can clearly observe that under the same perturbation budget indicated by x-axis, the cluster-based strategy leads to a larger change on graph spectrum, while achieving better performance on the downstream task. This analysis suggests that an effective edge augmentation should pay more attention to sensitive edges that introduce large disturbance to graph spectrum. The performance gap between these two simple strategies suggests a distinct possibility to improve over the uniformly random augmentation. Unlike the proof-of-concept cluster-based heuristic, we aim on designing a principled topology augmentation by directly maximizing the spectral change.

4.2 Augmentation Scheme via Spectral Change Maximization

We now introduce our augmentation scheme guided by graph spectrum. We first define the edge perturbation based topology augmentation scheme determined by a Bernoulli *probability matrix*. Based on that, we formulate our augmentation principle as a spectral change maximization problem.

Edge Perturbation Based Augmentation Scheme. We focus on topology augmentation using edge perturbation. Following the GCL formulation in Eq. (1), we define topology augmentation $T(\mathbf{A})$ as a Bernoulli distribution $\mathcal{B}(\Delta_{ij})$ for each entry A_{ij}. All Bernoulli parameters for all entries constitute a *probability matrix* $\mathbf{\Delta} \in [0,1]^{n\times n}$. We can sample an edge perturbation matrix $\mathbf{E} \in \{0,1\}^{n\times n}$, where $E_{ij} \sim \mathcal{B}(\Delta_{ij})$ indicates whether to flip the edge between node i and j, and the edge is flipped if $E_{ij} = 1$ otherwise remaining unchanged. A sampled augmented graph is then obtained via:

$$t(\mathbf{A}) = \mathbf{A} + \mathbf{C} \circ \mathbf{E},\ \mathbf{C} = \bar{\mathbf{A}} - \mathbf{A} \tag{2}$$

where $\bar{\mathbf{A}}$ is the complement matrix of the adjacency matrix $\mathbf{A}$, calculated by $\bar{\mathbf{A}} = \mathbf{1}_n\mathbf{1}_n^\top - \mathbf{I}_n - \mathbf{A}$, with $(\mathbf{1}_n\mathbf{1}_n^\top - \mathbf{I}_n)$ denoting the fully-connected graph without self-loops. Therefore, $\mathbf{C} = \bar{\mathbf{A}} - \mathbf{A} \in \{-1,1\}^{n\times n}$ denotes legitimate edge adding or removing operations for each node pair: edge adding between node i and j is allowed if $C_{ij} = 1$, and edge removing is allowed if $C_{ij} = -1$. Taking the Hadamard product $\mathbf{C} \circ \mathbf{E}$ finally gives valid edge perturbations to the graph.

Since $\mathbf{E}$ is a matrix of random variables following Bernoulli distributions, we can easily obtain the *expectation* of sampled augmented graphs in Eq. (2) as $\mathbb{E}[t(\mathbf{A})] = \mathbf{A} + \mathbf{C} \circ \mathbf{\Delta}$. Therefore, the design of $\mathbf{\Delta}$ determines the topology augmentation scheme. Taking uniformly random edge removal as an example, the entry Δ_{ij} is set as a fixed dropout ratio if $C_{ij} = -1$; and 0 otherwise.

Spectral Change Maximization. The default uniform edge perturbation adopted in many GCL methods is far from satisfactory. Motivated by our observation in Section 4.1, instead of setting fixed values for $\mathbf{\Delta}$, we propose to optimize it guided by graph spectrum. Specifically, we aim to search for $\mathbf{\Delta}$ that in expectation maximizes the spectral difference between the original graph and the augmented graph. Recall that we denote the normalized Laplacian matrix of $\mathbf{A}$ as $\mathrm{Lap}(\mathbf{A})$, and the graph spectrum vector can be calculated by $\mathbf{\Lambda} = \mathrm{eig}(\mathrm{Lap}(\mathbf{A}))$. We formulate the following problem to search for the desired perturbation matrix $\mathbf{\Delta}$ in a single augmentation branch:

$$\text{Single-way scheme:}\ \max_{\mathbf{\Delta}\in\mathcal{S}} \|\mathrm{eig}(\mathrm{Lap}(\mathbf{A} + \mathbf{C}\circ\mathbf{\Delta})) - \mathrm{eig}(\mathrm{Lap}(\mathbf{A}))\|_2^2 \tag{3}$$

where $\mathcal{S} = \{\mathbf{s}|\mathbf{s} \in [0,1]^{n\times n}, \|\mathbf{s}\|_1 \leq \epsilon\}$ and ϵ controls the strength of graph perturbation. By solving Eq. (3), we obtain the optimal Bernoulli probability matrix $\mathbf{\Delta}^*$, from which we can sample augmented views that in expectation differ the most from the original graph in graph spectrum. Note that Eq. (3) only provides one augmented view, to further introduce flexibility for a two-branch augmentation framework and enlarge the spectral difference between the resulting two views, we extend Eq. (3) as follows:

$$\text{Two-way scheme:} \quad \max_{\mathbf{\Delta}_1, \mathbf{\Delta}_2 \in \mathcal{S}} \|\text{eig}(\text{Lap}(\mathbf{A} + \mathbf{C} \circ \mathbf{\Delta}_1)) - \text{eig}(\text{Lap}(\mathbf{A} + \mathbf{C} \circ \mathbf{\Delta}_2))\|_2^2 \tag{4}$$

where $\mathbf{\Delta}_i$ is the Bernoulli probability matrix for augmentation branch i's scheme $T_i(\mathbf{A})$ in Eq. (1). Note that Eq. (3) is a special case of Eq. (4) when setting $\mathbf{\Delta}_2 = \mathbf{0}$. Eq. (4) gives better flexibility yet also makes the nonconvex optimization problem harder to solve, thus we further simplify it by pushing two branches towards opposite directions: maximizing the spectral norm in one branch, while minimizing it in the other, which leads to the final objective for our augmentation scheme:

$$\text{Opposite-direction scheme:} \quad \max_{\mathbf{\Delta}_1 \in \mathcal{S}} \mathcal{L}_{\text{GS}}(\mathbf{\Delta}_1), \quad \text{and} \quad \min_{\mathbf{\Delta}_2 \in \mathcal{S}} \mathcal{L}_{\text{GS}}(\mathbf{\Delta}_2) \tag{5}$$

where $\mathcal{L}_{\text{GS}}(\mathbf{\Delta}) = \|\text{eig}(\text{Lap}(\mathbf{A} + \mathbf{C} \circ \mathbf{\Delta}))\|_2^2$ measures the Graph Spectral norm under augmentation scheme with $\mathbf{\Delta}$. For scheme $T_1(\mathbf{A})$, $\mathbf{\Delta}_1$ produces views that overall have larger spectral norm than the original graph, while for $T_2(\mathbf{A})$, $\mathbf{\Delta}_2$ produces views with smaller spectrum. We can understand them as setting a spectral boundary for the input graph such that the encoder is trained to capture information that is essential and robust regarding perturbations within this region.

Optimizing $\mathbf{\Delta}_1$ and $\mathbf{\Delta}_2$. Eq. (5) can be solved via projected gradient descent (for $\mathbf{\Delta}_2$) or ascent (for $\mathbf{\Delta}_1$). Taking $\mathbf{\Delta}_2$ as an example, its update works as follows:

$$\mathbf{\Delta}_2^{(t)} = \mathcal{P}_{\mathcal{S}}[\mathbf{\Delta}_2^{(t-1)} - \eta_t \nabla \mathcal{L}_{\text{GS}}(\mathbf{\Delta}_2^{(t-1)})] \tag{6}$$

where t is the iteration step, $\eta_t > 0$ is the learning rate for step t, and $\mathcal{P}_{\mathcal{S}}(\mathbf{a}) = \text{argmin}_{\mathbf{s}\in\mathcal{S}} \|\mathbf{s} - \mathbf{a}\|_2^2$ is the projection operator at $\mathbf{a}$ over the constraint set $\mathcal{S}$. The calculation of gradient $\nabla \mathcal{L}_{\text{GS}}(\mathbf{\Delta}_2^{(t-1)})$ is done via chain rule. We now explain how to obtain a closed-form gradient over eigenvalues as it looks less straightforward. For a real and symmetric matrix $\mathbf{L}$, one can obtain the derivatives of its k-th eigenvalue λ_k by: $\partial\lambda_k/\partial\mathbf{L} = \mathbf{u}_k\mathbf{u}_k^\top$ [42], where $\mathbf{u}_k$ is the corresponding eigenvector. Note that the derivative calculation requires distinct eigenvalues, which does not hold for graphs satisfying *automorphism* [10]. To avoid such cases, we add a small noise term to the adjacency matrix[1], e.g., $\mathbf{A} + \mathbf{C} \circ \mathbf{\Delta} + \varepsilon \times (\mathbf{N} + \mathbf{N}^\top)/2$, where each entry in $\mathbf{N}$ is sampled from a uniform distribution $\mathcal{U}(0,1)$ and ε is a very small constant. Such a noise addition will almost surely break the graph automorphism, thus enabling a valid gradient calculation of eigenvalues.

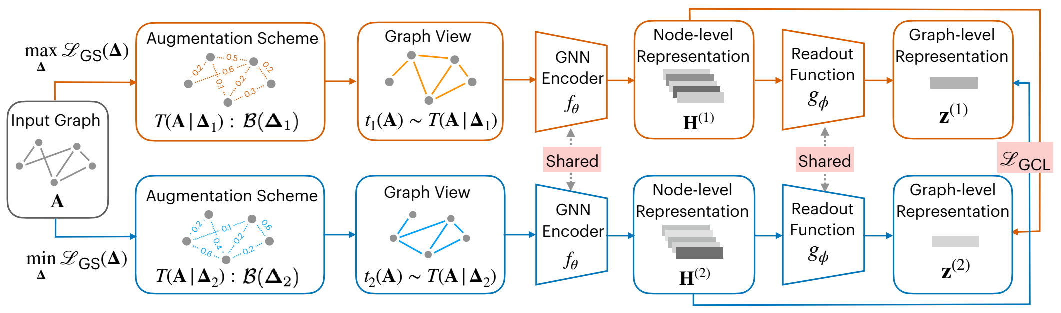
For T iterations, the time complexity of optimizing such a scheme is $\mathcal{O}(Tn^3)$ due to the eigen-decomposition eig$(\cdot)$ in $\mathcal{L}_{\text{GS}}$, which is prohibitively expensive for large graphs. To reduce the computational cost, instead of measuring the spectral change over all eigenvalues, we only maintain the K lowest- and highest-eigenvalues which are the most informative, as suggested by the spectral graph theory. The performance with different choices of K is studied in Appendix D.1. Using selective eigen-decomposition via the Lanczos Algorithm [36], the time complexity of augmentation scheme optimization is recuded to $\mathcal{O}(TKn^2)$ [2]. The scalability can be further improved by deploying the practical treatments in modeling large-scale graphs [40] (e.g., ego-nets sampling, batch training), which is left as our future work.

4.3 Formulation and Framework of GCL-TAGS

Figure 2 illustrates our GCL framework equipped with the spectrum-guided augmentation. A detailed algorithm can be found in Appendix B. We first pre-compute the probability matrices $\mathbf{\Delta}_1$ and $\mathbf{\Delta}_2$ via Eq. (5) to set up the augmentation scheme. For each iteration of contrastive learning, we sample two augmented graphs for the input graph $t_1(\mathbf{A}) \sim T(\mathbf{A}|\mathbf{\Delta}_1)$ and $t_2(\mathbf{A}) \sim T(\mathbf{A}|\mathbf{\Delta}_2)$. The augmented graphs are then fed into a GNN encoder f_θ, which outputs two sets of node representations $\mathbf{H}^{(1)}, \mathbf{H}^{(2)} \in \mathbb{R}^{n\times d'}$. A readout pooling function g_ϕ is further applied to aggregate and transform the node representations and obtain graph representations $\mathbf{z}^{(1)}, \mathbf{z}^{(2)} \in \mathbb{R}^{d'}$. Finally, the GNN encoder and

[1]The form of $(\mathbf{N} + \mathbf{N}^\top)/2$ is to keep the perturbed adjacency matrix symmetric for undirected graphs.

[2]Since we only require to precompute $\mathbf{\Delta}_1$ and $\mathbf{\Delta}_2$ once, the time complexity is totally acceptable.



$\max_{\Delta} \mathcal{L}_{\text{GS}}(\Delta)$
Input Graph
A
$\min_{\Delta} \mathcal{L}_{\text{GS}}(\Delta)$
Augmentation Scheme
$T(\mathbf{A}\,|\,\Delta_1) : \mathcal{B}(\Delta_1)$
Augmentation Scheme
$T(\mathbf{A}\,|\,\Delta_2) : \mathcal{B}(\Delta_2)$
Graph View
$t_1(\mathbf{A}) \sim T(\mathbf{A}\,|\,\Delta_1)$
Graph View
$t_2(\mathbf{A}) \sim T(\mathbf{A}\,|\,\Delta_2)$
GNN Encoder f_θ
Shared
GNN Encoder f_θ
Node-level Representation $\mathbf{H}^{(1)}$
Node-level Representation $\mathbf{H}^{(2)}$
Readout Function g_ϕ
Shared
Readout Function g_ϕ
Graph-level Representation $\mathbf{z}^{(1)}$
Graph-level Representation $\mathbf{z}^{(2)}$
$\mathcal{L}_{\text{GCL}}$

Figure 2: The framework of GCL-TAGS contains topology augmentation and contrastive objective. The opposite-direction augmentation scheme guided by graph spectrum is pre-computed following Eq. (5). The contrastive objective is to maximize the mutual information between node representations from one view and the graph representation from another view, and vice versa.

the readout function are trained by a contrastive objective $\mathcal{L}_{\text{GCL}}$ that maximizes the correspondence between local node representations of one view and the global graph representation of the other view. Given training graphs $\mathcal{G}$, we formulate GCL-TAGS as the following optimization problem:

$$\text{GCL-TAGS} : \min_{\theta,\phi} \mathcal{L}_{\text{GCL}}(t_1(\mathbf{A}), t_2(\mathbf{A}), \theta, \phi) = -\frac{1}{|\mathcal{G}|}\sum_{G\in\mathcal{G}}\left(\frac{1}{n}\sum_{i=1}^{n}\left(I(\mathbf{H}_i^{(1)}, \mathbf{z}^{(2)}) + I(\mathbf{H}_i^{(2)}, \mathbf{z}^{(1)})\right)\right)$$
$$\text{s.t. } t_i(\mathbf{A}) \sim T(\mathbf{A}|\boldsymbol{\Delta}_i), i \in \{1,2\}, \boldsymbol{\Delta}_1 = \text{argmax}_{\boldsymbol{\Delta}\in\mathcal{S}}\, \mathcal{L}_{\text{GS}}(\boldsymbol{\Delta}), \boldsymbol{\Delta}_2 = \text{argmin}_{\boldsymbol{\Delta}\in\mathcal{S}}\, \mathcal{L}_{\text{GS}}(\boldsymbol{\Delta}) \quad (7)$$

where $I(X_1, X_2)$ calculates the mutual information between variables X_1 and X_2, and we adopt InfoNCE as its estimator which is proven to be a lower bound of mutual information [53, 39]. Specifically, denoting cosine similarity as $\text{sim}(\cdot,\cdot)$, we estimate the mutual information as follows:

$$I(\mathbf{H}_i^{(a)}, \mathbf{z}^{(b)}) = \log \frac{\exp(\text{sim}(\mathbf{H}_i^{(a)}, \mathbf{z}^{(b)}))}{\sum_{j=1}^{n}\exp(\text{sim}(\widetilde{\mathbf{H}}_j, \mathbf{z}^{(b)}))} \quad (8)$$

where a and b index the augmented views, and $\widetilde{\mathbf{H}}$ is the node representations for a corrupted graph by randomly shuffling the features of the input graph [56, 17] to serve as negative examples. Note that the augmentation scheme is optimized prior to contrastive learning, which is a one-time computation thus does not introduce any extra complexity to the contrastive learning process.

5 Experiments

Experiment Setup: An extensive set of evaluations is performed for node classification, graph classification and regression tasks under unsupervised learning, transfer learning and adversarial attack settings. Our evaluations include various graph datasets ranging from citation networks, social networks to chemical molecules. We use GCN (for node prediction tasks) and GIN (for graph prediction tasks) as the base encoder for all methods to demonstrate the performance gain from contrastive learning. We adopt the following linear evaluation protocol for downstream tasks [48]: based on the representations given by the encoder, we train and evaluate a Logistic classifier or a Ridge regressor. We repeat our experiments for 10 times and report the mean and standard derivation of the evaluation metrics. We summarize the experimental details about datasets, baselines and configurations in Appendix C.

5.1 Unsupervised Learning

To evaluate the quality of learned representations, a linear model is trained for the downstream tasks using these learned representations as features and the resulting prediction performance is reported. The effectiveness of GCL-TAGS is evaluated on both node- and graph-level prediction tasks.

Node Classification Task. The datasets include Cora, Citeseer, PubMed citation networks [45], Wiki-CS hyperlink network [32], Amazon-Computer/Photo co-purchase network [46], and Coauthor-CS network [46]. We compare GCL-TAGS against GCL methods that augment topology with uniformly random edge perturbation (e.g., GRACE [67], BGRL [49], GBT [1]), centrality (GCA [68]), diffusion

Table 1: Node classification performance in *unsupervised* setting. The metric is *accuracy%*. **Bold** highlights that our method significantly outperforms baselines suggested by t-test with p-value$\leq$0.05.

	Dataset	Cora	Citeseer	PubMed	Wiki-CS	Amazon-Computer	Amazon-Photo	Coauthor-CS
	Raw-X	48.93±0.00	50.81±0.00	68.33±0.00	71.98±0.00	73.81±0.00	78.53±0.00	90.37±0.00
	S-GCN	81.34±0.35	70.42±0.45	79.82±0.41	77.19±0.12	86.51±0.54	92.42±0.16	93.03±0.31
	R-GCN	56.44±0.24	63.52±0.25	73.92±0.32	72.95±0.58	82.46±0.38	90.08±0.48	90.64±0.29
Baselines	GRACE [67]	83.33±0.43	72.10±0.54	78.72±0.13	80.14±0.48	89.53±0.35	92.78±0.30	91.12±0.20
	BGRL [49]	83.63±0.38	72.52±0.40	79.83±0.25	79.98±0.13	90.34±0.19	93.17±0.30	93.31±0.13
	GBT [1]	80.24±0.42	69.39±0.56	78.29±0.43	76.65±0.62	88.14±0.33	92.63±0.44	92.95±0.17
	MVGRL [17]	85.16±0.52	72.14±1.35	80.13±0.84	77.52±0.08	87.52±0.11	91.74±0.07	92.11±0.12
	GCA [68]	83.67±0.44	71.48±0.26	78.87±0.49	78.35±0.05	88.94±0.15	92.53±0.16	93.10±0.01
	GMI [37]	83.02±0.33	72.45±0.12	79.94±0.25	74.85±0.08	82.21±0.31	90.68±0.17	91.08±0.56
	DGI [56]	82.34±0.64	71.85±0.74	76.82±0.61	75.35±0.14	83.95±0.47	91.61±0.22	92.15±0.63
	GCL-TAGS	85.86±0.57	**72.76±0.63**	**81.54±0.24**	**82.13±0.15**	90.09±0.32	**93.52±0.26**	**93.91±0.24**

Table 2: Graph representation learning performance in *unsupervised* setting. TOP shows the biochemical and social network classification results on TU datasets (measured by *accuracy%*). BOTTOM shows the molecular regression (measured by *RMSE*) and classification (measured by *ROC-AUC%*) results on OGB datasets. **Bold** indicates that our method outperforms baselines with p-value$\leq$ 0.05.

	Dataset	Biochemical Molecules				Social Networks				
		NCI1	PROTEINS	MUTAG	DD	COLLAB	RDT-B	RDT-M5K	IMDB-B	IMDB-M
	S-GIN	78.27±1.35	72.39±2.76	90.41±4.61	74.87±3.56	74.82±0.92	86.79±2.04	53.28±3.17	71.83±1.93	48.46±2.31
	R-GIN	62.98±0.10	69.03±0.33	87.61±0.39	74.22±0.30	63.08±0.10	58.97±0.13	27.52±0.61	51.86±0.33	32.81±0.57
Baselines	InfoGraph [47]	68.13±0.59	72.57±0.65	87.71±1.77	75.23±0.39	70.35±0.64	78.79±2.14	51.11±0.55	71.11±0.88	48.66±0.67
	GraphCL [62]	68.54±0.55	72.86±1.01	88.29±1.31	74.70±0.70	71.26±0.55	82.63±0.99	53.05±0.40	70.80±0.77	48.49±0.63
	MVGRL [17]	68.68±0.42	74.02±0.32	89.24±1.31	75.20±0.55	73.10±0.56	81.20±0.69	51.87±0.65	71.84±0.78	50.84±0.92
	AD-GCL [48]	69.67±0.51	73.59±0.65	89.25±1.45	74.49±0.52	73.32±0.61	85.52±0.79	53.00±0.82	71.57±1.01	49.04±0.53
	JOAO [61]	72.99±0.75	71.25±0.85	85.20±1.64	66.91±1.75	70.40±2.21	78.35±1.38	45.57±2.86	71.60±0.86	51.14±0.69
	GCL-TAGS	71.43±0.49	**75.78±0.41**	89.12±0.76	75.78±0.52	**75.01±0.45**	83.62±0.64	**54.10±0.49**	**73.65±0.69**	**52.16±0.72**

	Dataset	Regression (Metric: RMSE)			Classification (Metric: ROC-AUC%)				
		molesol	mollipo	molfreesolv	molbace	molbbbp	molclintox	moltox21	molsider
	S-GIN	1.173±0.057	0.757±0.018	2.755±0.349	72.97± 4.00	68.17±1.48	88.14±2.51	74.91±0.51	57.60±1.40
	R-GIN	1.706±0.180	1.075±0.022	7.526±2.119	75.07±2.23	64.48±2.46	72.29±4.15	71.53±0.74	62.29±1.12
Baselines	InfoGraph [47]	1.344±0.178	1.005±0.023	10.005±4.819	74.74±3.64	66.33±2.79	64.50±5.32	69.74±0.57	60.54±0.90
	GraphCL [62]	1.272±0.089	0.910±0.016	7.679±2.748	74.32±2.70	68.22±1.89	74.92±4.42	72.40±1.01	61.76±1.11
	MVGRL [17]	1.433±0.145	0.962±0.036	9.024±1.982	74.20±2.31	67.24±1.39	73.84±4.25	70.48±0.83	61.94±0.94
	AD-GCL [48]	1.217±0.087	0.842±0.028	5.150±0.624	76.37±2.03	68.24±1.47	80.77±3.92	71.42±0.73	63.19±0.95
	JOAO [61]	1.285±0.121	0.865±0.032	5.131±0.722	74.43±1.94	67.62±1.29	78.21±4.12	71.83±0.92	62.73±0.92
	GCL-TAGS	1.218±0.052	**0.802±0.019**	**4.531±0.463**	76.74±2.02	**69.59±1.34**	80.28±2.42	**72.83±0.62**	**64.87±0.88**

matrix (MVGRL [17]) and the original graph (e.g., GMI [37] and DGI [56]). We also consider a fully semi-supervised GCN (*S-GCN*), a randomly initialized untrained GCN (*R-GCN*) and using the raw node features as node representations (*Raw-X*). All the methods exploit a 2-layer GCN encoder and a downstream linear classifier with the same hyper-parameters for a fair comparison. We adopt random feature masking in GCL-TAGS, following the setup in SOTA works [68, 1].

Table 1 shows that on the node classification task, GCL-TAGS achieves state-of-the-art performance in 6 out of the 7 datasets, 5 of which are significantly better than others. Specifically, comparing GCL-TAGS with MVGRL and GCA which use domain knowledge of the graph (e.g., node centrality or graph diffusion), the performance gain suggests the advantage of the spectrum based augmentation over previous domain-knowledge based heuristics. Meanwhile, GCL-TAGS is shown to be more effective than GRACE, BGRL and GBT which adopt uniformly random augmentation[3] and use a node-level contrastive objective. It is noteworthy that the representations learned by GCL methods achieve better performance than the base encoder R-GCN with semi-supervision, which suggests the effectiveness of self-supervised learning when label information is limited.

[3]We also discuss the gain of spectrum augmentation in Appendix D.2 by directly plugging our proposed augmentation into these frameworks.

Graph Prediction Task. We test on TU biochemical and social networks [34], Open Graph Benchmark (OGB) [19] and ZINC [20, 11] chemical molecules, and Protein-Protein Interaction (PPI) biological networks [20, 69] for graph prediction. We compare GCL-TAGS with five GCL methods including InfoGraph [47], GraphCL [62], MVGRL [17], AD-GCL (with fixed regularization weight) [48] and JOAO (v2) [61]. We use a 5-layer GIN encoder for all methods, including a semi-supervised S-GIN and a randomly initialized R-GIN. A readout function with a graph pooling layer and a 2-layer MLP is applied to generate graph representations.

Table 2 summarizes the graph prediction performance. GCL-TAGS gives the best results on 13 out of 17 datasets, of which 10 are significantly better than others. Compared with GraphCL and JOAO which select the best combination of augmentations for each dataset from a pool of methods including edge perturbation, node dropping and subgraph sampling, GCL-TAGS using only edge perturbation based augmentation still outperforms them. This suggests the effectiveness of graph spectrum in guiding topology augmentation. Compared with MVGRL, our performance gain mainly comes from the augmentation scheme, as these two methods share similar contrastive objectives, and our augmentation guided by graph spectrum is clearly more effective than the widely adopted uniformly random augmentation. While AD-GCL and GCL-TAGS follow a similar principle to remove edges that carry non-important and redundant information, GCL-TAGS is more flexible since the augmentation scheme is optimized in an independent pre-computation step without interfering with the contrastive learning procedure.

5.2 Transfer Learning

This experiment evaluates the generalizability of the GNN models, which are pre-trained on some datasets and re-purposed on different but related datasets. Table 3 reports the performance on chemical and biological graph classification datasets from [20]. Appendix C.5 further studies an even more challenging setting [40] where the encoder is pre-trained on social networks and transferred to multiple out-of-domain tasks. A reference model without pre-training (*No-Pre-Train-GIN*) is compared to demonstrate the gain of pre-training. GCL-TAGS is shown to be more effective in learning generalizable encoders. This supports our augmentation principle: by perturbing edges that cause large spectral changes, the encoder is pre-trained to ignore unreliable structural information, such that the relationship between such information and downstream labels can be removed to mitigate the overfitting issue. The generalizability of the GNN encoder on molecule classification depends on the structural fingerprints such as *subgraphs* [7]. JOAO and GraphCL using *subgraph* sampling augmentation is outperformed by GCL-TAGS, which suggests that the graph spectrum could be another important fingerprint to study chemical and biological molecular properties.

5.3 Adversarial Attack Setting

This setting demonstrates the robustness property of the proposed augmentation when the input graph is adversarially poisoned. The representations are learned from graphs poisoned by different structural attack strategies, including *Random* (which randomly flips edges), *DICE* (which deletes edges internally and connects nodes externally across classes), *GF-Attack* (which maximizes a low-rank matrix approximation loss) and *Mettack* (which maximizes the training loss via meta-gradients). We test the perturbation ratios $\sigma \in \{0.05, 0.2\}$: $\sigma \times m$ edges are flipped for a graph with m edges.

Table 4 reports the node classification performance under adversarial attack. The encoders learned by GCL methods with graph augmentations are generally more robust to perturbed graph structure compared with S-GCN. GCL-TAGS outperforms baselines with a clear margin, even under the strong

Table 3: Graph classification performance in *transfer learning* setting on molecular classification task. The metric is *ROC-AUC%*. **Bold** indicates that our method outperforms baselines with p-value≤ 0.05.

Dataset	Pre-Train	ZINC-2M								PPI-306K
	Fine-Tune	BBBP	Tox21	SIDER	ClinTox	BACE	HIV	MUV	ToxCast	PPI
No-Pre-Train-GIN		65.8±4.5	74.0±0.8	57.3±1.6	58.0±4.4	70.1±5.4	75.3±1.9	71.8±2.5	63.4±0.6	64.8±1.0
Baselines	InfoGraph [47]	68.8±0.8	75.3±0.5	58.4±0.8	69.9±3.0	75.9±1.6	76.0±0.7	75.3±2.5	62.7±0.4	64.1±1.5
	GraphCL [62]	69.7±0.7	73.9±0.7	60.5±0.9	76.0±2.7	75.4±1.4	78.5±1.2	69.8±2.7	62.4±0.6	67.9±0.9
	MVGRL [17]	69.0±0.5	74.5±0.6	62.2±0.6	77.8±2.2	77.2±1.0	77.1±0.6	73.3±1.4	62.6±0.5	68.7±0.7
	AD-GCL [48]	70.0±1.1	76.5±0.8	63.3±0.8	79.8±3.5	78.5±0.8	78.3±1.0	72.3±1.6	63.1±0.7	68.8±1.3
	JOAO [61]	71.4±0.9	74.3±0.6	60.5±0.7	81.0±1.6	75.5±1.3	77.5±1.2	73.7±1.0	63.2±0.5	64.0±1.6
	GCL-TAGS	70.0±0.7	**78.0±0.5**	**64.7±0.5**	80.7±2.1	**79.9±0.7**	77.8±0.6	73.8±0.9	**64.2±0.4**	**70.0±0.8**

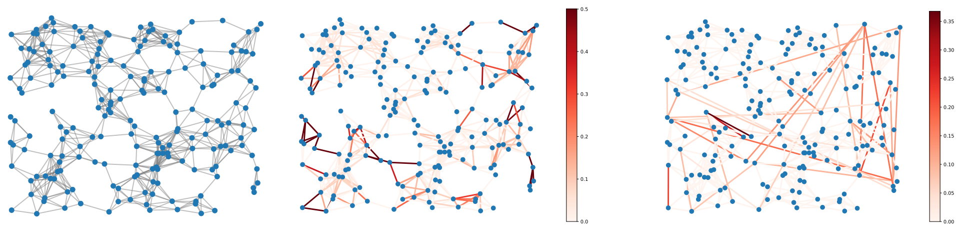
Table 4: Node classification performance on Cora in *adversarial attack* setting (measured by *accuracy%*). **Bold** indicates that our method outperforms baselines with p-value≤ 0.05.

	Attack	Clean	Random		DICE		GF-Attack		Mettack	
	Ratio σ		0.05	0.2	0.05	0.2	0.05	0.2	0.05	0.2
	S-GCN	81.34±0.35	81.11±0.32	80.02±0.36	79.42±0.37	78.37±0.42	80.12±0.33	79.43±0.32	50.29±0.41	31.04±0.48
Baselines	GRACE [67]	83.33±0.43	83.23±0.38	82.57±0.48	81.28±0.39	80.72±0.44	82.59±0.35	80.23±0.38	67.42±0.59	55.26±0.53
	BGRL [49]	83.63±0.38	83.12±0.34	83.02±0.39	82.83±0.48	81.92±0.39	82.10±0.37	80.98±0.42	70.23±0.48	60.42±0.54
	GBT [1]	80.24±0.42	80.53±0.39	80.20±0.35	80.32±0.32	80.20±0.34	79.89±0.41	78.25±0.49	63.26±0.69	53.89±0.55
	MVGRL [17]	85.16±0.52	85.28±0.49	84.21±0.42	83.78±0.35	83.02±0.40	83.79±0.39	82.46±0.52	73.43±0.53	61.49±0.56
	GCA [68]	83.67±0.44	83.33±0.46	82.49±0.37	82.20±0.32	81.82±0.45	81.83±0.36	79.89±0.47	58.25±0.68	49.25±0.62
	GMI [37]	83.02±0.33	83.14±0.38	82.12±0.44	82.42±0.44	81.13±0.49	82.13±0.39	80.26±0.48	60.59±0.54	53.67±0.68
	DGI [56]	82.34±0.64	82.10±0.58	81.03±0.52	80.48±0.38	79.89±0.43	81.30±0.54	79.88±0.58	71.42±0.63	63.93±0.58
	GCL-TAGS	85.86±0.57	**86.29±0.52**	**86.21±0.78**	**85.52±0.59**	**84.30±0.63**	**85.08±0.77**	**84.28±0.82**	**77.28±0.82**	**69.92±0.83**

Mettack which solicits the downstream label information. This demonstrates a good property of our proposed augmentation scheme: even though it is not explicitly designed for adversarial robustness, it achieves an effect that the encoder can stay invariant to the adversarially perturbed graph if its spectrum falls into the range captured by the opposite-direction augmentation scheme. Compared with the parallel efforts in designing robust GNNs [59, 8, 66, 24], the proposed augmentation provides a new insight using graph spectrum as a tool to study graph invariants and perturbations.

5.4 Behavior of Spectrum Guided Augmentation

To intuitively show the spatial change caused by perturbing the graph spectrum, we visualize a case study on a random geometric graph in Figure 3 where3a draws the original graph, 3b and 3c show the perturbation probability obtained by maximizing and minimizing $\mathcal{L}_{\mathrm{GS}}(\mathbf{\Delta})$, respectively. We can observe that maximizing the spectral norm assigns larger probability to remove edges bridging clusters such that the clustering effect becomes more obvious, while minimizing the spectral norm tends to add edges connecting clusters such that the clustering effect is blurred. Intuitively, such an augmentation perturbs these spurious edges that can easily affects the structural property (e.g. clustering) to preserve structural invariant, and the information about these edges are disentangled and minimized in the learned representations by contrastive learning. Such an interplay of spectral change and spatial change is also justified theoretically in Appendix D.3.



(a) Original graph (b) $\mathbf{\Delta}_1 = \mathrm{argmax}_{\mathbf{\Delta}} \mathbf{L}_{\mathrm{GS}}(\mathbf{\Delta})$ (c) $\mathbf{\Delta}_2 = \mathrm{argmin}_{\mathbf{\Delta}} \mathbf{L}_{\mathrm{GS}}(\mathbf{\Delta})$

Figure 3: A case study of the spectrum-guided augmentation scheme on a random geometric graph.

6 Conclusion

In this work, we proposed a principled topology augmentation scheme guided by graph spectrum. Our principle is that a well-behaving GNN encoder should stay invariant to sensitive structures that cause large changes on graph spectrum. To achieve this goal, we searched for the augmentation scheme that would mostly change the graph spectrum of the input graph, which leads to an opposite-direction augmentation scheme that changes the graph spectrum towards opposite directions. Based on the augmentation scheme, we developed a practical GCL framework GCL-TAGS, which has shown remarkable advantage in a variety of settings. Currently we only focus on topology augmentation, and ignore its interplay with node features, which could be another important dimension in GCL.

References

[1] Piotr Bielak, Tomasz Kajdanowicz, and Nitesh V Chawla. Graph barlow twins: A self-supervised representation learning framework for graphs. *arXiv preprint arXiv:2106.02466*, 2021.

[2] Aleksandar Bojchevski and Stephan Günnemann. Adversarial attacks on node embeddings via graph poisoning. In *International Conference on Machine Learning*, pages 695–704. PMLR, 2019.

[3] Heng Chang, Yu Rong, Tingyang Xu, Yatao Bian, Shiji Zhou, Xin Wang, Junzhou Huang, and Wenwu Zhu. Not all low-pass filters are robust in graph convolutional networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[4] Ashish Chiplunkar, Michael Kapralov, Sanjeev Khanna, Aida Mousavifar, and Yuval Peres. Testing graph clusterability: Algorithms and lower bounds. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 497–508. IEEE, 2018.

[5] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.

[7] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

[8] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 169–177, 2020.

[9] Shengyu Feng, Baoyu Jing, Yada Zhu, and Hanghang Tong. Adversarial graph contrastive learning with information regularization. *arXiv preprint arXiv:2202.06491*, 2022.

[10] Chris D Godsil. On the full automorphism group of a graph. *Combinatorica*, 1(3):243–256, 1981.

[11] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[12] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020.

[13] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[14] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[15] David K. Hammond, Yaniv Gur, and Chris R. Johnson. Graph diffusion distance: A difference measure for weighted graphs based on the graph laplacian exponential kernel. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 419–422, 2013.

[16] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[17] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pages 4116–4126. PMLR, 2020.

[18] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[19] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[20] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020.

[21] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020.

[22] Yizhu Jiao, Yun Xiong, Jiawei Zhang, Yao Zhang, Tianqi Zhang, and Yangyong Zhu. Sub-graph contrast for scalable self-supervised graph representation learning. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 222–231. IEEE, 2020.

[23] Wei Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zitao Liu, and Jiliang Tang. Self-supervised learning on graphs: Deep insights and new direction, 2020.

[24] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 66–74, 2020.

[25] Nabil Kahale. Eigenvalues and expansion of regular graphs. *J. ACM*, 42(5):1091–1106, sep 1995.

[26] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[27] James R Lee, Shayan Oveis Gharan, and Luca Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM (JACM)*, 61(6):1–30, 2014.

[28] Namkyeong Lee, Junseok Lee, and Chanyoung Park. Augmentation-free self-supervised learning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7372–7380, 2022.

[29] Michelle M. Li, Kexin Huang, and Marinka Zitnik. Representation learning for networks in biology and medicine: Advancements, challenges, and opportunities. *CoRR*, abs/2104.04883, 2021.

[30] Lu Lin, Ethan Blaser, and Hongning Wang. Graph structural attack by spectral distance, 2021.

[31] Nian Liu, Xiao Wang, Deyu Bo, Chuan Shi, and Jian Pei. Revisiting graph contrastive learning from the perspective of graph spectrum. *arXiv preprint arXiv:2210.02330*, 2022.

[32] Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

[33] Benjamin A Miller, Michelle S Beard, and Nadya T Bliss. Matched filtering for subgraph detection in dynamic networks. In *2011 IEEE Statistical Signal Processing Workshop (SSP)*, pages 509–512. IEEE, 2011.

[34] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020.

[35] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001.

[36] Beresford N Parlett and David S Scott. The lanczos algorithm with selective orthogonalization. *Mathematics of computation*, 33(145):217–238, 1979.

[37] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. Graph representation learning via graphical mutual information maximization. In *Proceedings of The Web Conference 2020*, pages 259–270, 2020.

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

[39] Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR, 2019.

[40] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1150–1160, 2020.

[41] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 385–394, 2017.

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

[43] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

[44] Michael T Schaub and Santiago Segarra. Flow smoothing and denoising: Graph signal processing in the edge-space. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 735–739. IEEE, 2018.

[45] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[46] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[47] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2019.

[48] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. Adversarial graph augmentation to improve graph contrastive learning. *Advances in Neural Information Processing Systems*, 34, 2021.

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

[50] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *Advances in Neural Information Processing Systems*, 33:6827–6839, 2020.

[51] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[52] Puja Trivedi, Ekdeep Singh Lubana, Mark Heimann, Danai Koutra, and Jayaraman J Thiagarajan. Analyzing data-centric properties for contrastive learning on graphs. *arXiv preprint arXiv:2208.02810*, 2022.

[53] Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv e-prints*, pages arXiv–1807, 2018.

[54] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. accepted as poster.

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

[56] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep Graph Infomax. In *International Conference on Learning Representations*, 2019.

[57] Hoi-To Wai, Santiago Segarra, Asuman E Ozdaglar, Anna Scaglione, and Ali Jadbabaie. Community detection from low-rank excitations of a graph filter. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4044–4048. IEEE, 2018.

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

[59] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples on graph data: Deep insights into attack and defense. *IJCAI*, 2019.

[60] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[61] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. In *International Conference on Machine Learning*, pages 12121–12132. PMLR, 2021.

[62] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823, 2020.

[63] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. When does self-supervision help graph convolutional networks? In *international conference on machine learning*, pages 10871–10880. PMLR, 2020.

[64] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR, 2021.

[65] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. GraphSAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020.

[66] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1399–1407, 2019.

[67] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep Graph Contrastive Representation Learning. In *ICML Workshop on Graph Representation Learning and Beyond*, 2020.

[68] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, WWW '21, page 2069–2080, New York, NY, USA, 2021. Association for Computing Machinery.

[69] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.

A A Review of Graph Spectrum

Graph spectrum plays a significant role in analyzing graph property (e.g., connectivity, cluster structure, diameter etc.) and is the foundation of spectral filters in GNNs. This motivates us to guide our proposed topology augmentation method using graph spectrum.

Graph Spectrum and Graph Property. The graph spectrum summarizes important properties related to a graph's global structure, which has been studied in graph spectral theory. We list some widely discussed properties revealed by graph spectrum to support our design: graph spectrum can be used as a comprehensive proxy for capturing graph properties in GCL.

- *Algebraic connectivity* [5], also known as Fiedler eigenvalue, of a graph is the second-smallest eigenvalue (counting multiple eigenvalues separately) of the Laplacian matrix. This eigenvalue is greater than 0 if and only if the graph is a connected graph. A corollary is that the number of times 0 appears as an eigenvalue in the Laplacian is the number of connected components in the graph. The magnitude of this value reflects how well connected the overall graph is.
- *Diameter* of a graph can be upper and lower bounded from its spectrum [5]. If the graph has r distinct eigenvalues, its diameter d is at most $r-1$. Meanwhile, if the graph has m edges and n nodes, we can bound the diameter by the first and second smallest non-zero eigenvalues as $1/2m\lambda_1 \geq d \geq 4/n\lambda_2$. For all $k \geq 2$, we also have $d \leq k \log n/\lambda_k$.
- *Clusterability* of a graph reveals how easy it is to partition the graph, which can be captured by the differences between the smallest successive eigenvalues of connected graphs. The difference between the first two eigenvalues, often referred to as the spectral gap, upper and lower bounds the graph expansion and conductance by the Cheeger inequality [25]. Nevertheless, analogous results also hold for higher-order eigenvalues [27].
- *Diffusion distance* [15] between two nodes v_i and v_j can be defined as $\mathcal{D}(v_i, v_j) = \|[\phi(\mathbf{L})]_{i,:} - [\phi(\mathbf{L})]_{j,:}\|_2^2 = \sum_{l=1}^{n} \phi(\lambda_l)^2 (\mathbf{u}_l[i] - \mathbf{u}_l[j])^2$, where $\phi(\mathbf{L}) = \mathbf{U}\phi(\mathbf{\Lambda})\mathbf{U}^\top$ and $\phi(x)$ is a decreasing kernel function such as $\phi(x) = e^{-tx}$. Therefore, a map that separates nodes with a specific diffusion distance is obtained when embedding graph nodes using eigenvectors.

Graph Spectrum and Spectral Filter. By viewing graph embedding models from a signal processing perspective, the normalized Laplacian $\mathbf{L}_{\text{norm}}$ serves as a graph shift operator and defines the frequency domain of a graph. As a result, its eigenvectors $\mathbf{U}$ can be considered as the graph Fourier bases, and the eigenvalues $\mathbf{\Lambda}$ (a.k.a., graph spectrum) correspond to the frequency components. Take one column of node features $\mathbf{X}$ as an example of graph signal, which can be compactly represented as $\mathbf{x} \in \mathbb{R}^n$. The graph Fourier transform of graph signal $\mathbf{x}$ is given by $\hat{\mathbf{x}} = \mathbf{U}^\top \mathbf{x}$ and the inverse graph Fourier transform is then $\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}$. The graph signals in the Fourier domain are filtered by amplifying or attenuating the frequency components $\mathbf{\Lambda}$.

At the essence of GNNs is the *spectral convolution*, which can be defined as the multiplication of a signal vector $\mathbf{x}$ with a spectral filter g_ϕ in the Fourier domain [6]:

$$g_\phi(\mathbf{L}) \star \mathbf{x} = g(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top)\mathbf{x} = \mathbf{U} g_\phi(\mathbf{\Lambda})\mathbf{U}^\top \mathbf{x} \tag{9}$$

The filter g_ϕ defines a smooth transformation function of the graph spectrum. One can apply a spectral filter and graph Fourier transformation to manipulate graph signals in various way, such as smoothing and denoising [44], abnormally detection [33] and clustering [57]. Here we show how the spectral convlution is defined in two popular GNNS used in our experiments: GCN [26] and GIN [60].

The vanilla GCN [26] is a first-order approximation of Chebyshev polynomial filter [16] with $g_\phi(\mathbf{\Lambda}) = (2 - \mathbf{\Lambda})\phi$, and the corresponding convolution for d-dimensional signal $\mathbf{X}$ is:

$$g_\phi^{\text{GCN}}(\mathbf{L}) \star \mathbf{X} = \mathbf{U}(2-\mathbf{\Lambda})\mathbf{U}^\top \mathbf{X}\mathbf{\Phi} = (\mathbf{I}_n + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{X}\mathbf{\Phi} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{\Phi} \tag{10}$$

where $\mathbf{\Phi} \in \mathbf{R}^{d \times d'}$ is the matrix of spectral filter parameters, and a renormalization trick $\mathbf{I}_n + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \rightarrow \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ is applied by adding self-loop $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$. GIN [60] with equal discriminative power as WL test designs spectral convolution as:

$$g_\phi^{\text{GIN}}(\mathbf{L}) \star \mathbf{X} = \mathbf{U}(2 + \epsilon - \mathbf{\Lambda})\mathbf{U}^\top \mathbf{X}\mathbf{\Phi} = (\mathbf{I}_n(1+\epsilon) + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{X}\mathbf{\Phi} \tag{11}$$

where ϵ can be a learnable parameter or a fixed scalar. Since the spectral filters $g_\phi(\mathbf{\Lambda})$ are the key in graph convolutions to encode graph signals that are transformed in the Fourier domain. The output of the spectral filters is then transformed back to the spatial domain to generate node representations. Therefore, we aim to augment graphs to influence the graph spectrum and the filtered graph signals, such that the encoder with altered spectral filters is encouraged to stay invariant to such perturbations through GCL.

B Algorithm of GCL-TAGS

Algorithm 1 illustrates the detailed steps of GCL-TAGS. Note that only one training graph is included (e.g., $L = 1$) for node representations learning, and multiple graphs are used (e.g., $L \geq 2$) for graph representation learning.

Algorithm 1: GCL-TAGS

```
Input   : Data $\{G_l = (\mathbf{X}_l, \mathbf{A}_l) \sim \mathcal{G} | l = 1 \cdots, L\}$; GNN encoder $f_\theta$; Readout function $g_\phi$.
Params  : Augmentation optimization step $M$ and learning rate $\eta$;
          Contrastive learning step $N$ and learning rate $\beta$.
Output  : Trained encoder $f_{\theta^*}$ and readout function $f_{\phi^*}$.
1  /* Lower-level optimization for augmentation scheme in Eq. (7) */
2  for each graph $l \leftarrow 1$ to $L$ do
3      Initialize Bernoulli parameters for each graph: $\mathbf{\Delta}_{l,1}^{(0)} \in [0,1]^{n \times n}, \mathbf{\Delta}_{l,2}^{(0)} \in [0,1]^{n \times n}$;
4      for $t \leftarrow 1$ to $M$ do
5          /* Optimize one direction of scheme: $\max_{\mathbf{\Delta} \in \mathcal{S}} \mathcal{L}_{\texttt{GS}}(\mathbf{\Delta})$ */
6          $\mathcal{L}_{\text{GS}}(\mathbf{\Delta}_{l,1}^{(t-1)}) = \|\text{eig}(\text{Lap}(\mathbf{A} + \mathbf{C} \circ \mathbf{\Delta}_{l,1}^{(t-1)}))\|_2^2$;
7          $\mathbf{\Delta}_{l,1}^{(t)} \leftarrow \mathcal{P}_{\mathcal{S}}[\mathbf{\Delta}_{l,1}^{(t-1)} + \eta \nabla \mathcal{L}_{\text{GS}}(\mathbf{\Delta}_{l,1}^{(t-1)})]$;
8          /* Optimize the other direction of scheme: $\min_{\mathbf{\Delta} \in \mathcal{S}} \mathcal{L}_{\texttt{GS}}(\mathbf{\Delta})$ */
9          $\mathcal{L}_{\text{GS}}(\mathbf{\Delta}_{l,2}^{(t-1)}) = \|\text{eig}(\text{Lap}(\mathbf{A} + \mathbf{C} \circ \mathbf{\Delta}_{l,2}^{(t-1)}))\|_2^2$;
10         $\mathbf{\Delta}_{l,2}^{(t)} \leftarrow \mathcal{P}_{\mathcal{S}}[\mathbf{\Delta}_{l,2}^{(t-1)} - \eta \nabla \mathcal{L}_{\text{GS}}(\mathbf{\Delta}_{l,2}^{(t-1)})]$ based on Eq. (6);
11     end
12     $\mathbf{\Delta}_{l,1} \leftarrow \mathbf{\Delta}_{l,1}^{(M)}, \mathbf{\Delta}_{l,2} \leftarrow \mathbf{\Delta}_{l,2}^{(M)}$;
13 end
14 /* Upper-level optimization for contrastive learning in Eq. (7) */
15 Initialize encoder and readout function: $\theta^{(0)}, \phi^{(0)}$;
16 for $t \leftarrow 1$ to $N$ do
17     Sample a batch of graphs $\{G_1, \cdots, G_Q\}$;
18     /* Sample augmented views for this graph based on Eq. (2) */
19     for $l \leftarrow 1$ to $Q$ do
20         Sample perturbations from Bernoulli distributions: $\mathbf{E}_{l,1} \sim \mathcal{B}(\mathbf{\Delta}_{l,1}), \mathbf{E}_{l,2} \sim \mathcal{B}(\mathbf{\Delta}_{l,2})$;
21         Calculate topology augmentations: $\mathbf{A}_{l,1} = \mathbf{A} + \mathbf{C} \circ \mathbf{E}_{l,1}, \mathbf{A}_{l,2} = \mathbf{A} + \mathbf{C} \circ \mathbf{E}_{l,2}$;
22         Randomly mask node features to obtain $\mathbf{X}_{l,1}, \mathbf{X}_{l,2}$ following [68, 1] if applicable;
23         Two graph views are generated as $G_{l,1} = (\mathbf{X}_{l,1}, \mathbf{A}_{l,1}), G_{l,2} = (\mathbf{X}_{l,2}, \mathbf{A}_{l,2})$
24     end
25     /* Optimize contrastive objective $\min_{\theta,\phi} \mathcal{L}_{\texttt{GCL}}$ */
26     Define $\mathcal{L}(G_{l,1}, G_{l,2}, \theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} \Big( I(\mathbf{H}_i^{(1)}, \mathbf{z}^{(2)}) + I(\mathbf{H}_i^{(2)}, \mathbf{z}^{(1)}) \Big)$ for $G_l$;
27     Calculate objective: $\mathcal{L}_{\text{GCL}}(\theta^{(t-1)}, \phi^{(t-1)}) = -\frac{1}{Q} \sum_{l=1}^{Q} \mathcal{L}(G_{l,1}, G_{l,2}, \theta^{(t-1)}, \phi^{(t-1)})$;
28     Update the encoder: $\theta^{(t)} \leftarrow \theta^{(t-1)} - \beta \nabla_\theta \mathcal{L}_{\text{GCL}}(\theta^{(t-1)}, \phi^{(t-1)})$;
29     Update the readout function: $\phi^{(t)} \leftarrow \phi^{(t-1)} - \beta \nabla_\phi \mathcal{L}_{\text{GCL}}(\theta^{(t-1)}, \phi^{(t-1)})$;
30 end
Output  : Encoder $f_{\theta^{(N)}}$ and readout function $h_{\phi^{(N)}}$
```

C Experiment Setup Details

This section includes the detailed setup for all experiments, including the procedure of conducting pre-analysis, datasets, baselines, and hyper-parameter settings. The experiments were performed on Nvidia GeForce RTX 2080Ti (12GB) GPU cards for most datasets, and RTX A6000 (48GB) cards for PubMed and Coauthor-CS datasets. Optimizing memory use for large graphs will be our future work.

C.1 Pre-analysis Experiment of Figure 1

We now introduce the detailed information to reproduce Figure 1 on Cora. This experiment is to show that uniformly random edge perturbation adopted in many GCL methods is not effective enough to reveal essential graph properties, described by graph spectrum. Since graph spectrum is closely related to graph properties such as clusterability (as discussed in Appendix A), in contrast to the uniform edge perturbation, we created a node cluster based strategy as follows: We first grouped the edges among nodes by whether the end nodes belong to the same cluster, treating nodes' class labels as their clusters. For inter-cluster edges, we assign a larger removing probability, while for intra-cluster edges we assign a smaller removing probability. Note that in expectation, we remove the same amount of edges as the uniformly *random* strategy, but allocate different probabilities to these two groups of edges. We should note the purpose of this experiment is only to demonstrate the impact of graph spectrum for GCL, and we used the class label of nodes as a proxy about the graph spectrum (due to the relation between clusterability and graph spectrum). Our proposed solution GCL-TAGS is fully unsupervised, i.e., it does *not* depend on the node labels at all.

Specifically, an edge removing ratio σ indicated by the x-axis of Figure 1 represents the augmentation strength: for an input graph with m edges, we remove $\sigma \cdot m$ edges to generate an augmented graph. For the uniformly random augmentation method (*Uniform*), each edge is assigned an equal removing probability as σ; for the cluster-based augmentation heuristic (*Clustered*), given m_{inter} inter-cluster edges and $m_{\text{intra}} = m - m_{\text{inter}}$ intra-cluster edges, we increase the removing probability of each inter-cluster edge as $\sigma_{\text{inter}} = \min\{1.2\sigma, \sigma \cdot m/m_{\text{inter}}\}$, and the removing probability of each intra-cluster edge is decreased to $\sigma_{\text{intra}} = (\sigma \cdot m - \sigma_{\text{inter}} \cdot m_{\text{inter}})/m_{\text{intra}}$ to make sure that in expectation $\sigma \cdot m$ edges are removed as in the uniform strategy.

When conducting the contrastive learning procedure following GRACE [67], one augmentation branch used the uniformly random edge removing strategy, and the other branch adopted either the uniform or the clustered strategy. Both branches included a random feature masking with ratio 0.3. For these two GCL methods based on different augmentation strategies, the experiment setup is as follows: both methods used a GCN encoder with the same architecture and hyper-parameters (e.g., 2 convolutional layers with the embedding dimension of 32). Both performed 1000 training iterations to obtain node representations, whose quality was evaluated by using them as features for a downstream node classification task.

We compared the clustered augmentation based GCL with the uniform augmentation based GCL from two aspects: their performance in the downstream task and the spectral change on the augmented graphs. The right-side of Figure 1 reports the mean and standard derivation of *F1 score* for 10 runs with different random seeds, which measures the downstream task performance. Meanwhile, we calculated the eigenvalues of the normalized Laplacian matrix of the input graph ($\mathbf{\Lambda}$), the augmented graphs with uniform strategy ($\mathbf{\Lambda}'_{\text{uniform}}$) and the augmented graphs with clustered strategy ($\mathbf{\Lambda}'_{\text{clustered}}$). The left-side of Figure 1 reports the L_2 distance of eigenvalues between the input and augmented graphs (e.g., $\|\mathbf{\Lambda} - \mathbf{\Lambda}'_{\text{uniform}}\|_2$ and $\|\mathbf{\Lambda} - \mathbf{\Lambda}'_{\text{clustered}}\|_2$) to measure the spectral change.

C.2 Summary of Datasets

The proposed GCL-TAGS is evaluated on 25 graph datasets. Specifically, for the *node classification task*, we included Cora, Citeseer, PubMed citation networks [45], Wiki-CS hyperlink network [32], Amazon-Computer and Amazon-Photo co-purchase network [46], and Coauthor-CS network [46]. For the *graph classification and regression tasks*, we employed TU biochemical and social networks [34], Open Graph Benchmark (OGB) [19] and ZINC [20, 11] chemical molecules, and Protein-Protein Interaction (PPI) biological networks [20, 69]. We summarize the statistics of these datasets and briefly introduce the experiment settings on them.

Table 5: Node classification dataset. The metric is *accuracy*.

Data Name	#Nodes	#Edges	#Features	#Classes	Cluster Coefficient
Cora	2,708	5,278	1,433	7	0.2407
Citeseer	3,327	4,552	3,703	6	0.1415
PubMed	19,717	44,324	500	3	0.0602
Wiki-CS	11,701	215,863	300	10	0.4527
Amazon-Computers	13,752	245,861	767	10	0.3441
Amazon-Photos	7,650	119,081	745	8	0.4040
Coauthor-CS	18,333	81,894	6,805	15	0.3425

Table 6: TU Benchmark Datasets [34] for graph classifcation task in unsupervised learning setting. The metric used for classification task is *accuracy*.

Data Type	Name	#Graphs	Avg. #Nodes	Avg. #Edges	#Classes
Biochemical Molecules	NCI1	4,110	29.87	32.30	2
	PROTEINS	1,113	39.06	72.82	2
	MUTAG	188	17.93	19.79	2
	DD	1,178	284.32	715.66	2
Social Networks	COLLAB	5,000	74.5	2457.78	3
	REDDIT-BINARY	2,000	429.6	497.75	2
	REDDIT-MULTI-5K	4,999	508.8	594.87	5
	IMDB-BINARY	1,000	19.8	96.53	2
	IMDB-MULTI	1,500	13.0	65.94	3

- A collection of datasets were used to evaluate *node classification* performance in both *unsupervised learning* and *adversarial attack* settings, and Table 5 summarizes the statistics of these datasets. **Cora, Citeseer, PubMed** citation networks [45] contain nodes representing documents and edges denoting citation links. The task is to predict the research topic of a document given its bag-of-word representation. **Wiki-CS** hyperlink network [32] consists of nodes corresponding to Computer Science articles, with edges based on hyperlinks. The task is to predict the branch of the field about the article using its 300-dimension pretrained GloVe word embeddings. **Amazon-Computer, Amazon-Photo** co-purchase networks [46] have nodes being items and edges representing that two items are frequently bought together. Given item reviews as bag-of-word node features, the task is to map items to their respective item category. **Coauthor-CS** network [46] contains node to be authors and edges to be co-author relationship. Given keywords of each author's papers, the task is to map authors to their respective field of study. All of these datasets are included in the PyG (PyTorch Geometric) library[4].

- Two sets of datasets were used to evaluate *graph prediction* tasks under the *unsupervised learning* setting. **TU Datasets** [34] provides a collection of benchmark datasets, and we used several biochemical molecules and social networks for graph classification as summarized in Table 6. The data collection is also included in the PyG library following a 10-fold evaluation data split. We used these datasets for evaluation of the graph classification task in unsupervised learning setting. **Open Graph Benchmark (OGB)** [19] contains datasets for chemical molecular property classification and regression tasks, which are summarized in Table 7. This data collection can be load via the OGB platform [5], and we used its processed format available in PyG library.

- A set of biological and chemical datasets were used to evaluate *graph classification* task under the *transfer learning* setting, summarized in Table 8. Following the transfer learning pipeline in [20], an encoder was first pre-trained on a large biological Protein-Protein Interaction (**PPI**) network or **ZINC** chemical molecule dataset, and then was evaluated on small datasets from the same domains.

[4]https://pytorch-geometric.readthedocs.io/en/latest/index.html

[5]https://ogb.stanford.edu/docs/graphprop/

Table 7: OGB chemical molecular datasets [19] for both graph classification and regression tasks in unsupervised learning setting. The evaluation metric used for regression task is *RMSE*, and for classification is *ROC-AUC*.

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

Table 8: Biological interaction and chemical molecular datasets [20] for graph classification task in transfer learning setting. The evaluation metric is *ROC-AUC*.

Data Type	Stage	Name	#Graph	Avg. #Nodes	Avg. #Degree
Protein-Protein Interaction Networks	Pre-training	PPI-306K	306,925	39.82	729.62
	Fine-tuning	PPI	88,000	49.35	890.77
Chemical Molecules	Pre-training	ZINC-2M	2,000,000	26.62	57.72
	Fine-tuning	BBBP	2,039	24.06	51.90
		Tox21	7,831	18.57	38.58
		SIDER	1,427	33.64	70.71
		ClinTox	1,477	26.15	55.76
		BACE	1,513	34.08	73.71
		HIV	41,127	25.52	54.93
		MUV	93,087	24.23	52.55
		ToxCast	8,576	18.78	38.52

C.3 Summary of GCL Baselines

We compared GCL-TAGS against seven self-supervised learning baselines for *node representation learning*, including GRACE [67], its extension GCA [68], BGRL [49], GBT [1], MVGRL [17], GMI [37] and DGI [56]. Meanwhile, five baselines designed for *graph representation learning* were compared, including InfoGraph [47], GraphCL [62], MVGRL [17], AD-GCL (with fixed regularization weight) [48] and JOAO (v2) [61]. In the contrastive objective design of these methods, different contrastive modes are adopted to compare node-level or graph-level representations. We summarize them based on their contrastive modes as follows:

- *Node v.s. node* mode specifies the contrastive examples as node pairs in a local perspective, which focuses on node-level representation learning to serve node prediction tasks. In particular, **GRACE** [67] employs uniformly random edge removing to generate two views, and treats the same node from two views as positive pairs, and all the other nodes as negatives. **GCA** [68] extends GRACE [67] with an adaptive augmentation considering the node centrality. **BGRL** [49] adopts uniformly random edge removing augmentation and applies a bootstrapping [12] framework to avoid collapse without negative sampling. **GBT** [1] uses uniformly random edge removing as graph augmentation and a Barlow-twins [64] objective to avoid collapse without requiring negative sampling. **GMI** [37] maximizes a general form of graphical mutual information defined on both features and edges between nodes in input graph and reconstructed output graph.
- *Graph v.s. node* mode takes graph and node pairs as contrastive examples to decide whether they are from the same graph, which obtains both node- and graph-level representations. In particular, **MVGRL** [17] maximizes the mutual information between the local Laplacian matrix and a global diffusion matrix, which obtains both node-level and graph-level representations that can serve for both node and graph prediction tasks. **DGI** [56] proposes to maximize the mutual information between representations of local nodes and the entire graph, in contrast with a corrupted graph by node shuffling. **InfoGraph** [47] aims to maximize the mutual information between the

representations of entire graphs and substructures (e.g., nodes, edges and triangles) with different granularity, and it is evaluated on graph-level prediction tasks.

- *Graph v.s. graph* mode treats contrastive examples as graph pairs from a global perspective, which mainly targets on graph-level representation learning for graph prediction tasks. Specifically, **AD-GCL** [48] aims to avoid capturing redundant information during the training by optimizing adversarial graph augmentation strategies in GCL, and designs a trainable non-i.i.d. edge-dropping graph augmentation. **JOAO** [61] adopts a bi-level optimization framework to search the optimal strategy among multiple types of augmentations such as uniform edge or node dropping, subgraph sampling. **GraphCL** [62] extensively studies graph structure augmentations including random edge removing, node dropping and subgraph sampling.

C.4 Hyper-parameter Setting

Training Configuration. We summarize the configuration of our GCL framework, including the GNN encoder and training parameters. For node representation learning, we used GCN [26] encoder, and set the number of GCN layers to 2, the size of hidden dimension for each layer to 512. The training epoch is 1000. For graph representation learning, we adopted GIN [60] encoder with 5 layers, which was concatenated by a readout function that adds node representations for pooling. The embedding size was set to 32 for TU dataset and 300 for OBG dataset. We used 100 training epochs with batch size 32. In all the experiments, we used the Adam optimizer with learning rate 0.001 and weight decay 10^{-5}. For data augmentation, we adopted both edge perturbation and feature masking, whose perturbation ratio σ_e and σ_f were tuned by grid search among $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ based on the validation set. Note that in our formulation Eq. (7), the augmentation strength $\epsilon = \sigma_e \cdot m$ where m is the number of edges in the input graph.

Evaluation Protocol. When evaluating the quality of learned representations on downstream tasks in an unsupervised setting, we adopted the evaluation protocol proposed in [48]. Specifically, based on the representations given by the encoder, we trained and evaluated a Logistic classifier or a Ridge regressor with L_2 regularization, whose weight was tuned among $\{0.001, 0.01, 0.1, 1.0, 10.0, 100.0\}$ on the validation set for each dataset.

C.5 Out-of-Domain Transfer Learning

A more challenging transfer learning setting is proposed in [40], which supports model training on multiple graphs from academic and social networks and transferring to different downstream tasks for other datasets. We further conducted a transfer learning experiment under such setting to demonstrate the out-of-domain transferability of our proposed solution GCL-TAGS. Specifically, we pre-train the encoder on the Yelp dataset [65] which contains 716,847 nodes and 13,954,819 edges; and then the encoder is fine-tuned and evaluated on the US-airport dataset [41] for node classification task and three TU social networks [34] for graph classification task via 10-fold CV. We sample 80,000 2-hop ego-nets from the Yelp dataset for pre-training, and use node degree as the node feature for all datasets.

Table 9 summarizes the results of downstream classification accuracy. Overall, we can observe satisfactory performance gain using contrastive learning to pre-train the GIN encoder on Yelp dataset under such a cross-domain setting. Meanwhile, our proposed method GCL-TAGS outperforms other contrastive learning methods on three out of four downstream datasets.

D Model Analysis

D.1 Influence of Choosing Eigenvalues

To reduce the time complexity of eigen-decomposition when calculating the spectrum norm $\mathcal{L}_{\text{GS}}(\mathbf{\Delta})$, we can approximate the norm by only using the K lowest- and highest-eigenvalues. The time complexity of optimizing the augmentation scheme in Eq. (5) with T iterations is $\mathcal{O}(TKn^2)$. This experiment shows the influence of K to the resulting GCL performance. Since the graphs encountered in the node prediction tasks are much larger than those in graph prediction tasks, we used the node classification datasets in Table 5 to conduct this experiment. Specifically, we test influence of K on four large graphs representing different types of networks: PubMed citation network, Wiki-CS

Table 9: Node and graph classification performance under *out-of-domain transfer learning* setting. The metric is *accuracy%*. **Bold** indicates that our method outperforms baselines with p-value≤ 0.05.

Dataset	Pre-Train	Yelp			
	Fine-Tune	Node Classification	Graph Classification		
		US-Airport	COLLAB	RDT-B	IMDB-B
No-Pre-Train-GIN		62.42±1.27	74.82±0.92	86.79±2.04	71.83±1.93
Baseline	MVGRL	63.83±0.97	74.78±0.84	86.24±1.26	73.21±1.54
	AD-GCL	–	75.11±0.70	88.72±1.53	74.34±1.23
	JOAO	–	75.35±0.93	87.65±1.72	75.15±1.67
GCL-TAGS		**65.21±0.86**	**76.37±0.73**	88.41±1.12	**75.89±1.20**

hyperlink network, Amazon-Computers co-purchase network and Coauthor-CS network. We tuned K among $\{50, 100, 200, 500, 1000, 5000\}$ for each of the datasets containing $n \geq 10,000$ nodes. The other components of GCL maintained the same, except the resulting augmentation scheme using spectrum norm with different K.

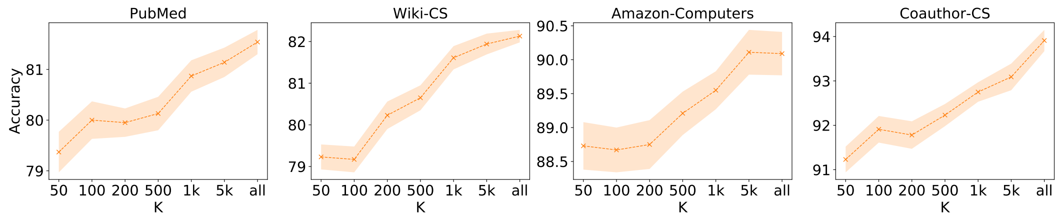


Figure 4: Node classification performance when choosing K lowest- and highest-eigenvalues

Figure 4 demonstrates the performance of GCL-TAGS on the node classification task when different K was used to generate the augmentation scheme. The x-axis denotes the value of K with "all" indicating that all the eigenvalues were used. The performance decreases marginally when we used a smaller K, and generally when $K = 1000$ we can still achieve a comparable performance. This suggests that low and high eigenvalues are already quite informative in capturing graph structural properties. Similar phenomenon is also discussed in previous works [30]: small eigenvalues carry smoothly varying signals (e.g., similar neighbor nodes within the same cluster), while high eigenvalues carry sharply varying signals (e.g., dissimilar nodes from different clusters).

D.2 Gain of Spectrum Guided Augmentation on Other GCL Frameworks

In this experiment, we use an ablation study to evaluate the effectiveness of the graph spectrum guided topology augmentation scheme when applied to different contrastive learning frameworks. We focus on GCL for node-level representation learning, as this line of work adopts distinct contrastive objectives (e.g., bootstrapping in BGRL, and Barlow twins in GBT) and contrastive modes (e.g., node v.s. node in GRACE, and node v.s. graph in MVGRL), such that we can comprehensively demonstrate the effectiveness of our proposed augmentation in covering a variety of GCL frameworks.

Specifically, we replace the original uniformly random edge removing augmentation in GRACE, BGRL, GBT, and the diffusion matrix based augmentation in MVGRL with the proposed spectrum based augmentation scheme, and use *-TAGS* as suffix to denote them. Note that MVGRL-TAGS is basically GCL-TAGS since it uses the same contrastive objective as in MVGRL such that both node- and graph-level representations are obtained to serve a broader range of downstream tasks.

Table 10 shows the results of plugging our augmentation scheme on four types of GCL frameworks. We can observe that our augmentation scheme does not depend on a particular contrastive objective, but brings a clear performance gain across different GCL frameworks. Intuitively, our augmentation captures the essential structural properties by perturbing edges that cause large spectral change. Therefore, no matter what contrastive objective or mode is used, maximizing the correspondence of two views encourages the encoder to ignore the information carried by such sensitive edges. This demonstrates the importance of studying graph spectral properties for graph augmentation.

Table 10: Node classification performance under *unsupervised* setting. We plug the spectrum based augmentation to different GCL frameworks, highlighted with suffix *-TAGS*. The metric is *accuracy%*. **Bold** highlights that the GCL framework with plugging our augmentation method outperforms its original version with p-value≤ 0.05.

Dataset	Cora	Citeseer	PubMed	Wiki-CS	Amazon-Computer	Amazon-Photo	Coauthor-CS
GRACE [67]	83.33±0.43	72.10±0.54	78.72±0.13	80.14±0.48	89.53±0.35	92.78±0.30	91.12±0.20
GRACE-TAGS	**84.21±0.51**	**72.87±0.58**	**79.94±0.22**	80.63±0.47	89.95±0.41	92.56±0.34	**91.98±0.20**
BGRL [49]	83.63±0.38	72.52±0.40	79.83±0.25	79.98±0.13	90.34±0.19	93.17±0.30	93.31±0.13
BGRL-TAGS	**84.34±0.42**	72.73±0.44	**80.78±0.32**	**81.04±0.22**	90.12±0.21	**93.58±0.39**	93.77±0.21
GBT [1]	80.24±0.42	69.39±0.56	78.29±0.43	77.30±0.62	88.02±0.32	92.23±0.35	92.85±0.31
GBT-TAGS	**82.43±0.51**	**71.12±0.48**	**80.05±0.49**	**78.89±0.54**	**89.04±0.43**	92.78±0.43	92.95±0.37
MVGRL [17]	85.16±0.52	72.14±1.35	80.13±0.84	77.52±0.08	87.52±0.11	91.74±0.07	92.11±0.12
MVGRL-TAGS	85.86±0.57	**72.76±0.63**	**81.54±0.24**	**82.13±0.15**	90.09±0.32	**93.52±0.26**	**93.91±0.24**

D.3 Spatial Behavior of Spectrum Guided Augmentation

We provide a theoretical justification which reveals the interplay of spectral change and spatial change. As derived in Theorem 2 of [2], given an edge flip $\Delta w_{ij} = (1 - 2A_{ij}) \in \{-1, 1\}$ between node i and j (e.g. if $\Delta w_{ij} = 1$, adding edge (i, j); otherwise removing edge (i, j)), the k-th eigenvalue is changed as $\lambda'_k = \lambda_k + \Delta\lambda_k$. $\Delta\lambda_k$ can be approximated by:

$$\Delta\lambda_k = \Delta w_{ij}(2u_{ki} \cdot u_{kj} - \lambda_k(u_{ki}^2 + u_{kj}^2)) \tag{12}$$

where u_k is the k-th eigenvector corresponding to the eigenvalue λ_k. If we only focus on the magnitude of eigenvalue change, we can obtain:

$$|\Delta\lambda_k| = |(u_{ki} - u_{kj})^2 + (\lambda_k - 1)(u_{ki}^2 + u_{kj}^2)| \tag{13}$$

Remarks. Since the eigenvectors are normalized, we can treat $(u_{ki}^2 + u_{kj}^2)$ as a constant as it is a relatively stable value. Based on the theory in spectral clustering [35], if the eigenvectors of node i and node j have a larger difference (i.e., $\|u_{\cdot,i} - u_{\cdot,j}\|_2$ is large), these two nodes should belong to different clusters. The first term in Eq. (13) suggests a larger eigenvalue change, if u_{ki} and u_{kj} have a larger difference. Therefore, flipping the edge between nodes from different clusters (thus with a larger $(u_{ki} - u_{kj})^2$) results in a larger spectral change. The second term suggests that such an effect becomes more obvious for eigenvalues close to 0 or 2.

D.4 The Convergence of Optimizing the Spectrum Guided Augmentations

In this section, we show how the graph spectrum norm changes as the augmentation optimization proceeds following Eq. (5). To better show the relative change of graph spectrum compared with the original graph, we calculate $\mathcal{L}_{\text{GS}}(\mathbf{\Delta})$ normalized by the spectrum norm of the original graph, that is, $\mathcal{L}_{\text{GS}}(\mathbf{\Delta})/\mathcal{L}_{\text{GS}}(\mathbf{0}) = \|\text{eig}(\text{Lap}(\mathbf{A} + \mathbf{C} \circ \mathbf{\Delta}))\|_2^2/\|\text{eig}(\text{Lap}(\mathbf{A}))\|_2^2$. The value of normalized $\mathcal{L}_{\text{GS}}(\mathbf{\Delta})/\mathcal{L}_{\text{GS}}(\mathbf{0})$ is reported in Figure 5.

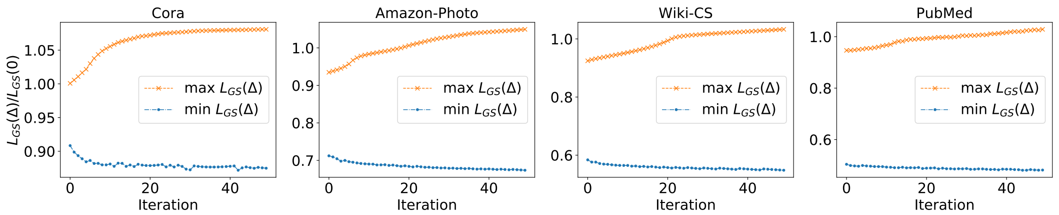


Figure 5: The value of relative spectral change $\mathcal{L}_{\text{GS}}(\mathbf{\Delta})/\mathcal{L}_{\text{GS}}(\mathbf{0})$ when maximizing the spectrum norm (orange) or minimizing it (blue) via Eq. (5)

Based on the graph spectral theory, the eigenvalues are bounded within $[0, 2]$, thus the L2-norm of the graph spectrum is also bounded by the total number of nodes, For example, the values of the original spectrum norm $\mathcal{L}_{\text{GS}}(\mathbf{0})$ for Cora, Amazon-Photo, Wiki-CS and PubMed are $24.88, 22.13, 31.79$, and 65.26 respectively. From Figure 5, we can observe that maximizing $\mathcal{L}_{\text{GS}}(\mathbf{\Delta})$ indeed results in an a larger spectrum norm compared with the original graph (i.e. $\mathcal{L}_{\text{GS}}(\mathbf{\Delta})/\mathcal{L}_{\text{GS}}(\mathbf{0}) > 1$), while minimizing it achieves a smaller spectrum norm (i.e. $\mathcal{L}_{\text{GS}}(\mathbf{\Delta})/\mathcal{L}_{\text{GS}}(\mathbf{0}) < 1$).