Boosting Graph Contrastive Learning via Graph Contrastive Saliency

Chunyu Wei [1 *] **Yu Wang** [1 *] **Bing Bai** [1 *] **Kai Ni** [2] **David J. Brady** [3] **Lu Fang** [1]

Abstract

Graph augmentation plays a crucial role in achieving good generalization for contrastive graph self-supervised learning. However, mainstream Graph Contrastive Learning (GCL) often favors random graph augmentations, by relying on random node dropout or edge perturbation on graphs. Random augmentations may inevitably lead to semantic information corruption during the training, and force the network to mistakenly focus on semantically irrelevant environmental background structures. To address these limitations and to improve generalization, we propose a novel self-supervised learning framework for GCL, which can adaptively screen the semantic-related substructure in graphs by capitalizing on the proposed gradient-based Graph Contrastive Saliency (GCS). The goal is to identify the most semantically discriminative structures of a graph via contrastive learning, such that we can generate semantically meaningful augmentations by leveraging on saliency. Empirical evidence on 16 benchmark datasets demonstrates the exclusive merits of the GCS-based framework. We also provide rigorous theoretical justification for GCS's robustness properties. Code is available at `https://github.com/weicy15/GCS`.

1. Introduction

Graph Neural Networks (GNNs) are powerful frameworks for modeling non-Euclidean structured data, with practical applications in the fields like biochemistry, physics, and social science (Senior et al., 2020; Hu et al., 2020a; Shlomi et al., 2021). However, conventional training methods for GNNs usually require human annotations, which requires domain expertise and is often time-consuming and expensive (Hu et al., 2020b; Sun et al., 2020). In contrast, self-supervised learning (SSL) pretraining offers a promising alternative, enabling effective representation learning of models without labeling efforts.

Contrastive learning is one of the most popular paradigms of SSL, demonstrating impressive progress in the field of computer vision (CV) (Chen et al., 2020; Grill et al., 2020). Recently, graph contrastive learning (GCL) techniques start to emerge by translating the central spirits of contrastive learning into graph-structured data (You et al., 2020). Among different algorithm variations, effective graph augmentation constructions turn out to be the bread and butter for the GCL methods to achieve success (Suresh et al., 2021). However, random augmentation is still a default for conventional graph SSL (You et al., 2020; Zhu et al., 2020; Rong et al., 2020), which lead to intrinsic drawbacks in achieving good representation generalization:

- **Semantic information corruption**. Random augmentation can easily corrupt semantic information and destroy the predictive structure of the graph. For example, specific topological structures in molecules are critical for chemistry functioning. Random augmentation may destroy such structures (Li et al., 2022b).
- **Environmental information overemphasis**. The predictive structure of a graph sometimes is only determined by a small subset of nodes/edges, with the majority of remaining nodes/edges being environmental background information irrelevant to semantics (Yu et al., 2022). Random augmentations might mistakenly encourage GNNs to focus on dominant background information during contrastive learning instead of semantics (Suresh et al., 2021).

To partially mitigate the above concerns, we probe an alternative solution and propose the unsupervised Graph Contrastive Saliency (GCS) learning method particularly tailored for graph SSL[1]. GCS takes inspiration from recent advancements in explanation methods such as Graph Rationalization (GR) (Wu et al., 2022; Sui et al., 2022; Liu et al.,

[*]Equal contribution [1]Department of Electronic Engineering, Tsinghua University, Beijing, China [2]Holo Technology (Beijing) Co., Ltd., Beijing, China [3]University of Arizona, Arizona, USA. Correspondence to: Lu Fang (www.luvision.net) <fanglu@tsinghua.edu.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[1]This paper focuses on training self-supervised GNNs for graph-level tasks, but the idea may generalize to node-level tasks.

2022a) and gradient-based explanation technique like Class Activation Mapping (CAM) (Selvaraju et al., 2017). Unlike GR methods that always require ground-truth information of downstream tasks (Liu et al., 2022a), GCS jettisons the need for knowledge on downstream tasks. Nevertheless, GCS can still effectively help disentangle semantic and environmental background structures of the graph, by capitalizing only on the gradient-based saliency measurements. In contrast to CAM which was initially designed for euclidean-structured images, GCS extends beyond and makes gradient-based explanation an appealing tool for graph semantic explanation. Leveraging on the semantic and environmental structures decomposition of GCS, our proposed augmentation pipeline learns to (1) construct positive paired samples for GCL that preserve semantics in the graph and (2) suppress environmental background information via hard negative samples by intentionally destroying the identified semantics for the following GCL. We also incorporate an iterative refinement procedure to be associated with GCS. This procedure sequentially locates residual salient structures on the graph omitted in the last iteration and ensures that the extracted semantic substructure of the graph remains discriminative.

We extensively evaluate GCS-based framework on 16 benchmarks for molecule property-related and social network-related tasks. Empirical results demonstrate the superior performance of GCS compared to state-of-the-art graph contrastive learning methods, including AD-GCL (Suresh et al., 2021) and RGCL (Li et al., 2022b). We also demonstrate rigorous theoretical justification in terms of the robustness of the GCS-based framework.

2. Related Works

Graph Contrastive Learning Graph contrastive learning is one of the mainstream routes of graph self-supervised learning (Liu et al., 2022c), where the model is trained by maximizing the mutual information between two different views constructed from the original graph. Data augmentation plays a crucial role in contrastive learning. Mainstream GCL methods generate augmented views by randomly altering the anchor graph's topological structure, node properties, or edge attributes (Zhu et al., 2020; Qiu et al., 2020). GraphCL (You et al., 2020) systematically studies the impact of combining various random augmentations. However, randomness in augmentations can lead to undesired loss of semantic information. To address this issue, some works have employed domain knowledge to identify salient features in graph augmentations (Subramonian, 2021; Liu et al., 2022b; Zhu et al., 2021), which undermines the self-supervised learning's goal of reducing supervision. Additionally, AD-GCL (Suresh et al., 2021) adopts an adversarial edge dropping mechanism as augmentations to reduce the amount of redundant information taken by encoders.

Explanation Methods and Causal Learning In supervised learning, rationalization techniques refer to explanation methods that identify a small subset of input features by maximizing predictive performance based only on the subset, known as the rationale. To eliminate the spurious correlation between input features and the output, INVRAT (Chang et al., 2020) proposed the concept of invariant rationalization by modeling different environments as non-causal input to train predictors. Causal learning modeling-based methods also aim to extract causal semantics by identifying the spurious correlation between the class label and the semantic-irrelevant environmental variables. IRM (Rosenfeld et al., 2021) provided formal guarantees for improving invariant causal prediction on out-of-distribution generalization. By incorporating causal modeling into GNN optimization, StableGNN (Fan et al., 2021) presented a causal representation framework for GNN models to perform on out-of-distribution graphs. OOD-GNN (Li et al., 2022a) used a novel nonlinear graph representation decorrelation method that utilized random Fourier features to encourage GNNs to eliminate the statistical dependence between relevant and irrelevant graph representations. Recently, DIR (Wu et al., 2022) proposed a causal rationales approach for GNNs to improve interpretability and predictive performance on out-of-distribution data.

In this paper, we propose to use gradient-based saliency as an effective measurement for SSL augmentation constructions. A closely related work is RGCL (Li et al., 2022b) which extends the DIR (Wu et al., 2022) approach to GCL from the perspective of invariant rationale discovery, by utilizing a GNN-MLP combined rationale generator to generate rationales in a learnable manner. In comparison to RGCL which utilizes an additional network to generate the graph rationale, our proposed GCS conveniently captures graph saliency through the gradient computed as contrastive learning process evolves. GCS then generates efficient augmentations by respecting the extracted semantics/saliency and waives the need for an extra augmentation generation network. It is also worth noting that all the aforementioned works assume the availability of a downstream task whereas GCS is agnostic to downstream tasks' knowledge.

3. Notations and Preliminaries

We begin with some preliminary concepts and notations for further exposition. In this work, we let $G = (V, E)$ denote an attributed graph where V is the node set and E is the edge set. Apart from the topological structure, graph G may have node attributes $\{X_v \in \mathbb{R}^F | v \in V\}$ and edge attributes $\{X_e \in \mathbb{R}^F | e \in E\}$ of dimension F. We use $\mathcal{N}_v$ to denote the neighbor set of node v.

Graph Representation Learning Let $\mathcal{G}$ denote the graph space and $G_i \in \mathcal{G}, i = 1, 2, \ldots, n$. Graph representation

learning aims to learn an encoder $f : \mathcal{G} \to \mathbb{R}^d$, where the learned graph embedding $f(G_i)$ can be further used in the downstream task. In the downstream task, each graph G_i is assigned with a label $y_i \in \mathcal{Y}$ with a fixed labeling rule $\mathcal{G} \to \mathcal{Y}$. We learn another model $q : \mathbb{R}^d \to \mathcal{Y}$ to predict y_i, which can be formulated as $q(f(G_i))$. We assume each G_i is independent and identically distributed (i.i.d.) sampled from an unknown distribution $\mathbb{P}_{\mathcal{G}}$ from space $\mathcal{G}$. And each (G_i, y_i) is also i.i.d sampled from a distribution $\mathbb{P}_{\mathcal{G}\times\mathcal{Y}} = \mathbb{P}_{\mathcal{G}}\mathbb{P}_{\mathcal{Y}|\mathcal{G}}$, where $\mathbb{P}_{\mathcal{Y}|\mathcal{G}}$ is the conditional distribution of the graph label in the downstream task.

We adopt GNNs as the encoder f. Formally, for a graph $G = (V, E)$, we denote the node representation for node v as $\boldsymbol{h}_v$ and it can be initialized as $\boldsymbol{h}_v^{(0)} = X_v$. By message passing mechanism (Suresh et al., 2021), the node representation $\boldsymbol{h}_v^{(k-1)}$ is updated using node v's neighborhood $\mathcal{N}_v$ during the k-th iteration, which can be expressed as:

$$\boldsymbol{h}_v^{(k)} = f_{\text{combine}}^{(k)}\left(\boldsymbol{h}_v^{(k-1)}, f_{\text{aggregate}}^{(k)}\left(\{\boldsymbol{h}_u^{(k-1)} | u \in \mathcal{N}_v\}\right)\right), \quad (1)$$

where $f_{\text{aggregate}}(\cdot)$ is a trainable function that maps the set of node representations to an aggregated vector, and f_{combine} is another trainable function that maps both v's current representation $\boldsymbol{h}_v^{(k-1)}$ and the aggregated vector to v's updated representation $\boldsymbol{h}_v^{(k)}$. After K iterations of the message passing as defined in Eq. (1), most GNN utilizes the pooling techniques on the final representation of V to obtain the graph representation as:

$$\boldsymbol{h}_G = f(G) = f_{\text{pool}}\left(\boldsymbol{h}_v^{(K)} | v \in V\right).$$

Existing works consider using different f_{combine}, $f_{\text{aggregate}}$ and f_{pool} (Hamilton et al., 2017; Velickovic et al., 2018).

Graph Contrastive Learning Graph Contrastive Learning (GCL) typically follows the InfoMax principle (Linsker, 1988), which aims to learn an encoder $f(\cdot)$ that maximizes the mutual information or the correspondence between the graph and its representation:

$$\text{InfoMax:} \quad \max_f I\left(f(G); G\right).$$

In order to optimize Eq. (3), GCL methods usually learn $f(\cdot)$ to attract semantically similar samples (i.e., positives) with G, and push semantically different samples (i.e., negatives) away from G. The positive pairs are usually generated through graph data augmentation (GDA) processes such as node perturbation on graphs.

Definition 1 (GDA). *For a graph $G \in \mathcal{G}$, $T(G)$ denote a graph augmentation action of G, which is a distribution defined over $\mathcal{G}$ conditioned on G. $t(G) \in \mathcal{G}$ denote a sample from $T(G)$.*

Correspondingly, given two augmentation distributions T_1 and T_2, the objective of GCL can be defined as:

$$\max_f I\left(f\left(t_1(G)\right), f\left(t_2(G)\right)\right),$$

where $t_m(G) \sim T_m(G), m \in \{1, 2\}$. In practice, $I(\cdot, \cdot)$ is usually hard to estimate (Belghazi et al., 2018) and can be instead approximated by contrastive loss function, such as NCE (Misra & van der Maaten, 2020), InfoNCE (van den Oord et al., 2018), and NT-Xent (Chen et al., 2020).

4. Methodology

We present our unsupervised Graph Contrastive Saliency (GCS) framework. Our motivation is to respectively identify which components/nodes of the training data best preserve *s*emantic content of the graph, that represents the intrinsic true latent classes of the graph; and to also identify the complementary nodes causing semantic-irrelevant variations due the change of "environment". Through such decomposition into "semantic" and "environment" components, GCS can generate diverse data augmentations with improved invariance among semantically similar graphs, and with suppressed spurious correlations out of environmental variations. The augmentations are then used to construct positive and negative samples for the following graph contrastive learning with less bias, leading to improved generalization.

4.1. Unsupervised Semantic Screening for Graph Contrastive Learning

Semantic screening aims at finding the semantically discriminative substructures in a graph. These could be certain functional groups of molecules, or certain groups of people in social networks. In order to capture these semantically meaningful nodes, we take inspiration from the gradient-based class activation map (CAM) (Selvaraju et al., 2017). CAM is conventionally used to locate salient regions in images. In contrast, we propose a novel non-euclidean gradient-based explanation technique called the graph contrastive saliency (GCS) method. GCS extends beyond CAM and is exclusively suitable for training graph representation learning in an unsupervised manner. We elaborate detailed GCS procedure in the following section.

Overview The overall framework of GCS is illustrated in Figure 1. Blue nodes represent the semantics to be preserved, while red nodes represent the detected environmental information. In each refine iteration, the upper left is the original graph, the lower left is the residual graph out of the previous iteration (by masking newly obtained saliency nodes from original graph). The gradually evolving subgraph in the lower right represents the saliency scores learned by the model where darker color indicates higher saliency score. The upper right graph is the resultant graph with semantic nodes and environmental nodes determined by GCS via thresholding the saliency scores in the lower right graph. GCS refines the saliency results through multiple iterations and obtains the final augmentation in the third iteration, i.e., the upper right graph.

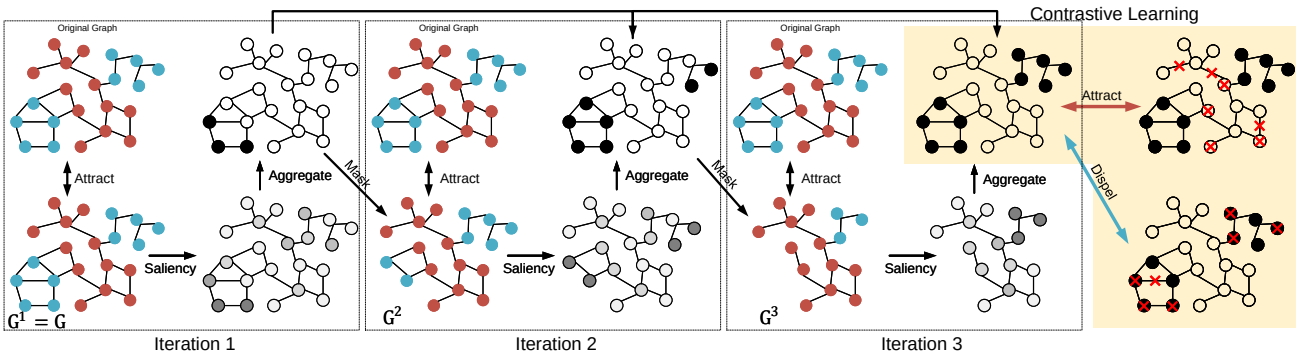

Figure 1. Illustration of the Iterative Graph Contrastive Saliency (T = 3) procedure and the generated augmentations based on GCS. Blue: semantic content; Red: enviromental content; Grey (darker) : saliency score (higher); Black: final semantic by thresholding the saliency score; White: final enviromental content by thresholding saliency score.

Graph Contrastive Saliency The basic thread of thinking is to find the critical semantic substructure on G' that maximizes mutual information $I(G', G)$, where $G' = t(G)$ is augmentation of graph G with similar semantics. In the meanwhile, these substructure should maintain discriminativeness to distinguish G from other graphs. Formally speaking, we define graph representation $h_{G'}$ as:

$$h_{G'} = f(G') = f_{\text{pool}}\left(\boldsymbol{h}_v | v \in V'\right) ,$$

where $\boldsymbol{h}_v \in \mathbb{R}^d$ is node v's final-layer representation $\boldsymbol{h}_v^{(K)}$ and for simplicity, we omit its subscript (K). Perturbation to some node representation of graph G, i.e., $\boldsymbol{h}_v \pm \Delta\boldsymbol{h}, v \in V$, will cause the change ΔI of the mutual information $I(G', G) = I(h_{G'}, h_G)$. If ΔI turns out large, the disturbed node on G through the mapping of f is considered critical in distinguishing the semantic changes between G and G'. Inspired from Mo et al. (2021), we perform "trail perturbation" by computing the closed form gradient of estimated mutual information $I(h_{G'}, h_G)$, with respect to the p-th dimension of $h_v, v \in V'$, to locate these semantically important substructure of the G:

$$\alpha_{v,p} = \frac{\partial I\left(h_{G'}, h_G\right)}{\partial h_{v,p}} , \tag{2}$$

where we later elaborate the used estimator on $I(h_{G'}, h_G)$ in Section 4.2. These gradients $\alpha_{v,p}$ flowing back are global-average-pooled over all the nodes v to obtain the "importance weight" of dimension p:

$$\alpha_p = \text{ReLU}\left(\frac{1}{|V'|}\sum_{V'} \alpha_{v,p}\right) , \tag{3}$$

where $\text{ReLU}(\cdot)$ is used to discard the negative signals. Inspired by CAM, we define the saliency score ω_v of each node v as the weighted sum of all the dimensions of the node's representation, yielding:

$$\omega_v = \text{Normalize}\left(\text{ReLU}\left(\sum_{p=1}^{d} \alpha_p h_{v,p}\right)\right) , \tag{4}$$

where $\text{Normalize}(x) = \frac{x - \min x}{\max x - \min x}$ is a normalization function that maps the elements to $[0, 1]$. For edge saliency score, it is intuitively legitimate to average the corresponding node saliency scores: $\omega_{uv} = (\omega_u + \omega_v)/2$. After calculating both the node and edge saliency scores, we *binarize* the scores by thresholding respectively at hyperparameter ϕ_{node} and ϕ_{edge} in order to generate corresponding mask generate the node mask vector $\mathbf{M}_V \in \mathbb{R}^{|V|}$ and edge mask vector $\mathbf{M}_E \in \mathbb{R}^{|E|}$. This procedure can be formally presented as:

$$\mathbf{M}_V[u] = \begin{cases} 1 , & u \in V \ \ \& \ \ \omega_u > \phi_{\text{node}} \\ 0 , & \text{Otherwise} \end{cases} ,$$

$$\mathbf{M}_E[u, v] = \begin{cases} 1 , & e_{u,v} \in E \ \ \& \ \ \omega_{uv} > \phi_{\text{edge}} \\ 0 , & \text{Otherwise} \end{cases} .$$

Iterative Refinement An additional advantage of our GCS method is that it can be further improved through an iterative refinement procedure in computing Eq. (4) similarly to Mo et al. (2021). The intuition is to remove the old identified semantic contents to construct new G', estimating the new mutual information $I(h_{G'}, h_G)$ based on these removals on G', then identify the residual new salient structure in G' that maximizes new $I(h_{G'}, h_G)$, and finally update the Eq. (4). This procedure can iterate several times (a hyperparameter) to best screen the semantic representative structures in G and obtain the final ω_v. Mathematically, at the t iteration, we perform the GCS on the original G with G^t to compute the saliency score:

$$\omega^t = \text{GCS}(G^t, G) ,$$

where ω^t is the saliency score (of both nodes and edges) in the t-th iteration. We initialize G^1 with the original graph, i.e., $G^1 = G$. In order to continuously capture the saliency part on the graph, We always aggregate the new GCS scores

with the old ones as follows:

$$\overline{\omega^t} = \max_{l \le t} \omega^l \,. \tag{5}$$

Finally, we binarize the saliency score $\overline{\omega^t}$ to obtain the actual masks:

$$\mathbf{M}_V^t, \mathbf{M}_E^t = \text{Binarize}\left(\overline{\omega^t}\right) \,,$$

and we mask the salient regions of the graph with the (reverse of) current mask to obtain the updated G^{t+1}:

$$G^{t+1} = \left(V \odot \left(\mathbf{1} - \mathbf{M}_V^t\right), E \odot \left(\mathbf{1} - \mathbf{M}_E^t\right)\right) \,. \tag{6}$$

Here the saliency region in the t-th iteration will be masked in G^{t+1}, which enables our GCS process to explore the unmasked region. After iterating T times according to Eq. (6), we use the final aggregated mask $\mathbf{M}_V^T$ and $\mathbf{M}_E^T$ at time T as the output node mask to indicate whether the node or edge is considered as the semantics of graph G. The complete refinement procedure is illustrated in Figure 1.

4.2. Graph Contrastive Saliency-based Augmentations

After obtaining the semantic mask $\mathbf{M}_V^T$ and $\mathbf{M}_E^T$ as in Section 4.1, it is now legitimate to decompose the input graph G into the semantic subgraph G^s and the environment subgraph G^e such that we can generate new augmentations based the decomposition:

$$G^s = \left(V \odot \mathbf{M}_V^T, E \odot \mathbf{M}_E^t\right) \,,$$
$$G^e = \left(V^T \odot \left(\mathbf{1} - \mathbf{M}_V^t\right), E^T \odot \left(\mathbf{1} - \mathbf{M}_E^t\right)\right) .$$

In order to force the GNN to learn the semantic invariance between G and its augmentations, it is critical to preserve the semantic of augmentation G' during the generative process of augmentation. We then keep the learned G^s invariant and perturb G^e by node dropout, edge dropout, or other graph augmentation approaches to create the "positive" pair of views of G:

$$G^{m,+} = t_m(G^e) + G^s \,, \tag{7}$$

where $t_m(G) \sim T_m(G), m \in \{1, 2\}$. We then optimize the following objective according to Definition 1 to maximize the mutual information between positive samples:

$$\mathcal{L}_{\text{sc}}: \ \max_f I\left(f\left(G^{1,+}\right), f\left(G^{2,+}\right)\right) \,. \tag{8}$$

Note that the graph model may focus wrongly and rely too much on the environmental information to take shortcuts in reaching an agreement for positive samples. This will cause environmental augmentation overemphasis and provides helpless (and harmful) information to learn useful semantic invariance. To remedy this issue, we can generate hard "negative" graph samples to be composed of nodes dominated by background information. The training then explicitly pushes away the representations of G and its hard negative sample G^- to learn even the smallest and "delicate" change in semantics, even if the environmental background information is dominating. Given this motivation, and by inspiration from Li et al. (2022b) and Mo et al. (2021), we also create hard negative views for each graph by corrupting its semantics, in addition to using the negative views derived from other graphs. Specifically, for each G, we keep its environment subgraph G^e intact, and perform a certain ratio of node dropout and edge dropout on the G^s to contaminate its semantics:

$$G^- = G^e + t'(G^s) \,, \tag{9}$$

where G^- is the constructed hard negative and $t'(G) \sim T'(G)$ represents the graph data augmentation with a certain ratio of node dropout and edge dropout. The objective is to also dispel these hard negatives through penalizing:

$$\mathcal{L}_{\text{env}}: \ \min_f I(f(G), f(G^-)) \,. \tag{10}$$

Our overall objective function can be equivalently viewed as a Lagrangian of the objectives Eq. (8) and Eq. (10):

$$\min_f \mathcal{L} = -I\left(f\left(G^{1,+}\right), f\left(G^{2,+}\right)\right) + \lambda I\left(f\left(G\right), f\left(G^-\right)\right) \,, \tag{11}$$

where λ is the hyperparameter that controls the strength of the hard negatives. Note that our GCS-based SSL is a backbone-agnostic graph SSL framework, which is applicable to different backbones.

Without loss of generality, we adopt InfoNCE as the estimator for mutual information $I(\cdot, \cdot)$ in Eq. (11) and in Eq. (2). InfoNCE loss is known as a lower bound of the mutual information (Poole et al., 2019) in the context of contrastive SSL (van den Oord et al., 2018; Chen et al., 2020). During the training, given a mini-batch of N graphs $\{G_i\}_{i=1}^N$, let $z'_i = f(G'_i)$ and $z''_i = f(G''_i)$ for any arbitrary G'_i and G''_i, the mutual information between variables $f(G'), f(G'')$ is approximately estimated as:

$$\hat{I}\left(f(G'), f(G'')\right) = \frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp\left({z'_i}^T z''_i/\tau\right)}{\exp\left({z'_i}^T z''_i/\tau\right) + \sum_{j=1, j\neq i}^{N} \exp\left({z'_i}^T z''_j/\tau\right)} \,. \tag{12}$$

Here, G'_i and G''_i are viewed as sampled realizations of variables G' and G''. We plug $\hat{I}\left(f(G'), f(G'')\right)$ estimated via Eq. (12) to replace $I\left(f(G'), f(G'')\right)$ in Eq. (11), where the definitions of G' and G'' become clear depending on the contexts in Eq. (11).

5. Analysis

The proposed augmentation method leverages on the explanation of Eq. (4). Since interpretability methods are often prone to robustness issues (Ghorbani et al., 2019; Adebayo et al., 2018; Slack et al., 2020; Bai et al., 2021), it is

critical to provide a theoretical understanding of robustness properties of Eq. (4). This section serves this purpose and establishes the provable robustness guarantee of the proposed method. In detail, we demonstrate that the term inside the ReLU of Eq. (4) satisfies C-Lipschitz continuity with some Lipschitz constant C, and Eq. (4) is therefore guaranteed to be robust to small changes when the number of nodes in the graph approximately approaches infinity. We firstly demonstrate the used definitions and assumptions required to support our theorem.

Definition 2 (***C*-Lipschitz).** *A function f is C-Lipschitz if there exists a constant C that satisfies $\|f(x) - f(\hat{x})\|_2 \leq C\|x - \hat{x}\|_2$.*

Assumption 1.
i) *Admissible loss functions $\mathcal{L}(h_{v,p})$ are first-order differentiable.*

ii) *The first order derivative of loss function $\mathcal{L}(h_{v,p})$, i.e., $\tilde{\mathcal{L}}(h_{v,p}) = \frac{\partial \mathcal{L}(h_{v,p})}{\partial h_{v,p}}$ is in bound in $[-\tilde{\mathcal{L}}_{max}, \tilde{\mathcal{L}}_{max}]$, i.e., $|\tilde{\mathcal{L}}(h_{v,p})| \leq \tilde{\mathcal{L}}_{max}$, and $|h_{v,p}| \leq h_{max}$.*

iii) *The p^{th} dimension of feature $h_{v,p}$ across $v \in V$ nodes follows Gaussian distribution, i.e., $h_{v,p} \sim \mathcal{N}(\mu_p, \sigma_p^2), \forall v$, where μ_p is the mean value, and σ_p^2 is the variance. Define $\boldsymbol{\mu}_v$ has value μ_p at the p^{th} dimension in $\boldsymbol{\mu}_v$.*

Given the above definition and assumptions, we adapt from the proving techniques in Agarwal et al. (2021) and propose the following theorem, which is closely related to the explanation (saliency) obtained via Eq. (4):

Theorem 1. *Let g be a function defined to be:*

$$g(\mu_p) = \lim_{V \to \infty} \frac{1}{V} \sum_{v=1}^{V} \frac{\partial L(h_{v,p})}{\partial h_{v,p}}, h_{v,p} \sim \mathcal{N}(\mu_p, \sigma_p^2), \forall v$$

then $Q(\boldsymbol{\mu}_v) = \sum_p g(\mu_p) h_{v,p}$ is $\frac{h_p^* \tilde{\mathcal{L}}_{max}}{2\sigma}$-Lipschitz w.r.t. $\ell 2$ norm, h_p^* is a constant depending on $\tilde{\mathcal{L}}_{max}$ and h_{max}.

Note if loss $L(h_{v,p}) = I(h'_G, h_G)$, then $g(\mu_p)$ equals to α_p by omitting the ReLU (Mo et al., 2021) in Eq. (3), and by assuming infinite number of nodes ($V \to \infty$) are present in graph. Function $Q(\boldsymbol{\mu}_v)$ is correspondingly defined on top of $g(\mu_p)$, with the goal to approximate the term inside the ReLU operation inside Eq. (4), which is guaranteed to be C-Lipschitz continuous with constant $C = \frac{h_p^* \tilde{\mathcal{L}}_{\max}}{2\sigma}$. This directly justifies that $Q(\boldsymbol{\mu}_v)$ used for computing the explanation in Eq. (4) is provably robust to small changes as long as Assumption 1 holds, showing the exclusive merits of our proposed augmentation in the context of graph SSL. We defer the proof to Appendix H.

6. Experiments

This section is dedicated to the empirical evaluation of the proposed GCS-based GCL. Extensive experiments are conducted to address the following two research questions: **RQ1:** Do the proposed GCS-based augmentations improve the performance of pre-trained backbones on downstream tasks? **RQ2:** How effective is the GCS in capturing semantically important structures in graphs? A summary of the datasets and training details for specific experiments, as well as the experimental analysis of hyperparameter sensitivity, are provided in the appendix.

6.1. Comparison with State-of-the-Art Methods (RQ1)

Unsupervised Learning For the unsupervised graph-level classification task, we trained graph encoders with different GCL methods using unlabeled data, then fixed the representation model and trained the classifier using labeled data. We used datasets from TU Dataset (Morris et al., 2020) for evaluations. Following GraphCL (You et al., 2020), we used a 5-layer GIN (Xu et al., 2019) with a hidden size of 128 as the graph encoder and utilized an SVM as the classifier. The GIN was trained with a batch size of 128 and a learning rate of 0.001. We conducted a 10-fold cross-validation on each dataset, and each experiment was repeated 5 times. We compared our method with kernel-based methods like Graphlet Kernel (GL) (Shervashidze et al., 2009), Weisfeiler-Lehman sub-tree kernel (WL) (Shervashidze et al., 2011) and Deep Graph Kernel (DGK) (Yanardag & Vishwanathan, 2015). Other unsupervised graph learning methods like node2vec (Grover & Leskovec, 2016), sub2vec (Adhikari et al., 2018), and graph2vec (Narayanan et al., 2017), and recent advanced GCL methods like InfoGraph (Sun et al., 2020), GraphCL (You et al., 2020), JOAO (You et al., 2021), AD-GCL (Suresh et al., 2021), and RGCL (Li et al., 2022b), were also included. All reported values for baseline methods are taken directly from Hu et al. (2020b) and their original papers. Summaries of datasets and baselines are provided in Appendix D.

Table 1 presents a comparison of different models for unsupervised learning. Our proposed GCS-based method achieved the best results on the MUTAG, IMDB-binary, REDDIT-binary, and REDDIT-Multi-5K datasets and the second-best results on the COLLAB dataset. The performance of GCS on the PROTEINS dataset was also comparable with the best results. Furthermore, GCS achieved the best average performance (77.39%), surpassing current state-of-the-art contrastive learning methods, including GraphCL (75.71%), AD-GCL (75.02%), and RGCL (76.15%).

Transfer Learning In this study, we evaluated the transfer learning performance. We first conducted self-supervised pre-training on the pre-processed ChEMBL dataset (Mayr

Table 1. Unsupervised representation learning classification accuracy (%) on TU datasets. **Bold** denotes the best performance while Underline represents the second best performance.

Model	MUTAG	PROTEINS	DD	NCI1	COLLAB	IMDB-B	REDDIT-B	RDT-M5K	avg.
GL	81.66±2.11	-	-	-	-	65.87±0.98	77.34±0.18	41.01±0.17	-
WL	80.72±3.00	72.92±0.56	-	80.01±0.50	-	72.30±3.44	68.82±0.41	46.06±0.21	-
DGK	87.44±2.72	73.30±0.82	-	**80.31±0.46**	-	72.30±3.44	78.04±0.39	41.27±0.18	-
node2vec	72.63±10.20	57.49±3.57	-	54.89±1.61	-	-	-	-	-
sub2vec	61.05±15.80	57.49±3.57	-	52.84±1.47	-	55.26±1.54	71.48±0.41	36.68±0.42	-
graph2vec	83.15±9.25	73.30±2.05	-	73.22±1.81	-	71.10±0.54	75.78±1.03	47.86±0.26	-
InfoGraph	89.01±1.13	74.44±0.31	72.85±1.78	76.20±1.06	**82.00±0.29**	73.03±0.87	82.50±1.42	53.46±1.03	75.43
GraphCL	86.80±1.34	74.39±0.45	78.62±0.40	77.87±0.41	71.36±1.15	71.14±0.44	89.53±0.84	55.99±0.28	75.71
JOAOv2	-	71.25±0.85	66.91±1.75	72.99±0.75	70.40±2.21	71.60±0.86	78.35±1.38	45.57±2.86	-
AD-GCL	88.62±1.27	**75.04±0.48**	75.73±0.51	75.86±0.62	74.89 ± 0.9	71.49±0.90	85.52±0.79	53.00±0.82	75.02
RGCL	87.66±1.01	75.03±0.43	**78.86±0.48**	78.14±1.08	70.92±0.65	71.85±0.84	90.34±0.58	56.38±0.40	76.15
GCS	**90.45±0.81**	75.02±0.39	77.22±0.30	77.37±0.30	75.56±0.41	**73.43±0.38**	**92.98±0.28**	**57.04±0.49**	77.39

et al., 2018) for 100 epochs. Subsequently, we fine-tuned the backbone model on 8 benchmark multi-task binary classification datasets in the biochemistry domain, which are included in MoleculeNet (Wu et al., 2018). We evaluated the mean and standard deviation of ROC-AUC scores of 10 runs with different random seeds on each downstream dataset, which is consistent with the baselines. The baselines used in this study included Infomax, EdgePred, AttrMasking, and ContextPred which are manually designed pre-training strategies from Pretrain-GNN (Hu et al., 2020b), which is considered as a strong baseline for graph transfer learning. We also involved other state-of-the-art pre-training methods, such as GraphCL, JOAO, AD-GCL, and RGCL. All reported values for baseline methods are taken directly from Hu et al. (2020b) and their original papers. The network backbone of all these methods was GIN. Further details of the experimental settings can be found in Appendix E.

Table 2 presents the results of the comparison among different methods. Our proposed method achieved the best performance on 5 out of 8 datasets, and the second-best performance on the ClinTox dataset. When compared with GraphCL, our method consistently delivered better performance, indicating the effectiveness of the semantic-preserving augmentation views in transfer learning. Furthermore, when compared with the current state-of-the-art rationale-aware method, RGCL (Li et al., 2022b), our method exhibited a considerable improvement in performance, with an increase in the average score over all datasets from 73.16% to 74.72%.

Summary The experiments on unsupervised learning and transfer learning demonstrate the state-of-the-art performance of our proposed GCS-based GCL. Especially, compared with AD-GCL and RGCL, the results verify the advantage of the graph contrastive saliency. By utilizing semantic-preserving augmentation, GCS significantly improves the transferability and the generalization ability of pre-trained backbone graph encoders.

6.2. Effectiveness of GCS in Capturing Semantics (RQ2)

In this section, we present an evaluation of the effectiveness of graph contrastive saliency in capturing semantics on the MNIST-Superpixel dataset. This dataset is particularly suited for this analysis as it possesses clear semantics and is amenable to visual examination. We adopted the experimental setup described by You et al. (2020) for formulating the pre-training dataset by removing the labels of the training set in the MNIST-Superpixel dataset. The resulting dataset is then employed to train the backbone encoder using GCS-based GCL. Subsequently, we randomly select three cases and generate GCS-based augmentations. For comparative purposes, we also apply node dropping augmentation using GraphCL with an augmentation ratio of 0.2, while masking nodes with saliency scores in the bottom 20% of the graph. Additional information regarding the MNIST-Superpixel dataset can be found in Appendix B.

The visualization presented in Figure 2 illustrates the capability of GCS to effectively capture semantics, thereby facilitating semantic-preserving augmentations for GCL. These high-quality positive samples can assist graph encoders in avoiding semantic corruption during contrastive learning. In contrast, GraphCL's random augmentation approach may remove semantically essential nodes, resulting in significant semantic changes. For instance, the GraphCL approach results in the removal of the down-right part of the **9** superpixel graph, thereby altering its semantic information from **9** to **0**. Similarly, the semantics of **8** is transformed to somewhat similar semantics to **6** in the GraphCL view. Our method, on the other hand, tends to assign high saliency scores to semantic areas and is more likely to drop nodes in the environment, thus improving the diversity and robustness of graphs with the same semantics. By perturbing the

Table 2. Transfer learning ROC-AUC scores (%) on downstream graph classification tasks. **Bold** denotes the best performance while Underline represents the second best performance.

Model	BBBP	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV	BACE	avg.
No Pretrain	65.8±4.5	74.0±0.8	63.4±0.6	57.3±1.6	58.0±4.4	71.8±2.5	75.3±1.9	70.1±5.4	66.96
Infomax	68.8±0.8	75.3±0.5	62.7±0.4	58.4±0.8	69.9±3.0	75.3±2.5	76.0±0.7	75.9±1.6	70.29
EdgePred	67.3±2.4	76.0±0.6	64.1±0.6	60.4±0.7	64.1±3.7	74.1±2.1	76.3±1.0	**79.9±0.9**	70.28
AttrMasking	64.3±2.8	**76.7±0.4**	64.2±0.5	61.0±0.7	71.8±4.1	74.7±1.4	77.2±1.1	79.3±1.6	71.15
ContextPred	68.0±2.0	75.7±0.7	63.9±0.6	60.9±0.6	65.9±3.8	75.8±1.7	77.3±1.0	79.6±1.2	70.89
GraphCL	69.68±0.67	73.87±0.66	62.40±0.57	60.53±0.88	75.99±2.65	69.80±2.66	78.47±1.22	75.38±1.44	70.75
JOAOv2	71.39±0.92	74.27±0.62	63.16±0.45	60.49±0.74	80.97±1.64	73.67±1.00	77.51±1.17	75.49±1.27	72.12
AD-GCL	70.01±1.07	76.54±0.82	63.07±0.72	63.28±0.79	79.78±3.52	72.30±1.61	78.28±0.97	78.51±0.80	72.72
RGCL	71.42±0.66	75.20±0.34	63.33±0.17	61.38±0.61	**83.38±0.91**	76.66±0.99	77.90±0.80	76.03±0.77	73.16
GCS	**71.46±0.46**	76.16±0.41	**65.35±0.17**	**64.20±0.35**	82.01±1.90	**80.45±1.67**	**80.22±1.37**	77.90±0.26	74.72

Table 3. Ablation study for GCS on downstream unsupervised learning datasets.

Variation	MUTAG	PROTEINS	DD	NCI1	COLLAB	IMDB-B	REDDIT-B	RDT-M5K	avg.
GraphCL	86.80±1.34	74.39±0.45	78.62±0.40	77.87±0.41	71.36±1.15	71.14±0.44	89.53±0.84	55.99±0.28	75.71
w/o semantic-presvering aug.	87.38±0.54	74.31±0.27	78.90±0.26	77.71±0.55	72.21±0.41	72.37±0.46	90.88±0.63	55.19±0.34	76.12
w/o hard negative aug.	89.06±0.34	74.90±0.32	76.96±0.44	76.82±0.22	74.51±0.28	72.50±0.36	91.28±0.42	56.32±0.54	76.54
GCS (complete version)	90.45±0.81	75.02±0.39	77.22±0.30	77.37±0.30	75.56±0.41	73.43±0.38	92.98±0.28	57.04±0.49	77.39

structures in the environment, our method benefits the diversity of graphs with the same semantics and thus improves the robustness of the backbone models.

Furthermore, our GCS approach enables the generation of hard negatives by applying high-ratio dropout to the semantic nodes while retaining the environmental parts, as depicted in Figure 3. These views and the original graph possess many similar background elements, making it challenging to discern the original semantic information. By minimizing the mutual information between these examples and the original graphs, we can induce the backbone model to suppress the encoded environmental information.

6.3. Ablation Study

In this section, we conduct ablation studies to evaluate the effectiveness of the proposed semantic-preserving augmentation (Eq. (8)) and the hard negative augmentation (Eq. (10)) based on GCS. We consider the following variants of GCL with different components in the final objective (Eq. (11)):

- **w/o semantic-preserving aug.**: This variant model replaces $\mathcal{L}_{\text{sc}}$ with the GraphCL loss, which constructs positive views by randomly dropping edges or nodes.
- **w/o hard negative aug.**: This variant model removes the $\mathcal{L}_{\text{env}}$ term from the final objective $\mathcal{L}$.

Taking the unsupervised learning task as an example, Table 3 presents the experimental results. The results demonstrate that the removal of either component of the method negatively impacts the ability of the graph representation learning to perform well. These results align with our hypothesis that random augmentation may corrupt semantics and overemphasize environmental information, thereby hindering the generalization performance on downstream tasks.

6.4. GCS agaisnt varying graph size, sparsity, noise

In our main paper, we compared all the SOTA methods on the widely used public benchmarks having a fixed size, sparsity, and noise level. However, it indeed would be interesting to test GCS against varying conditions. To reach this goal, we divide the COLLAB dataset into 5 splits representative of different levels of size, sparsity, and noise levels. For training against **graph size**, we first reorder all data samples in COLLAB dataset in terms of graph sizes (node size in ascending order). We then evenly split this reordered dataset sequentially into 5 parts, where each part has 1000 training samples. In this way, different part corresponds to different average graph sizes. Similarly, to test against varying **sparsity** levels, we define $Sparsity = \frac{n_{edge}}{n_{node}^2}$, reorder all data samples in COLLAB according to $Sparsity$, and sequentially divide the reordered data (in ascending order) into 5 splits. To test COLLAB against **noise** level, we randomly flip certain proportions (noise level) of edges in the graph of COLLAB, and report the results. The three experiments are respectively presented as follows.

Varying graph size. COLLAB is ordered according to the node numbers and divided equally into 5 split/part (1000 samples each part). Average node numbers for Part 1- Part 5 are respectively 34.319, 42.248, 52.791, 74.685, 168.431.

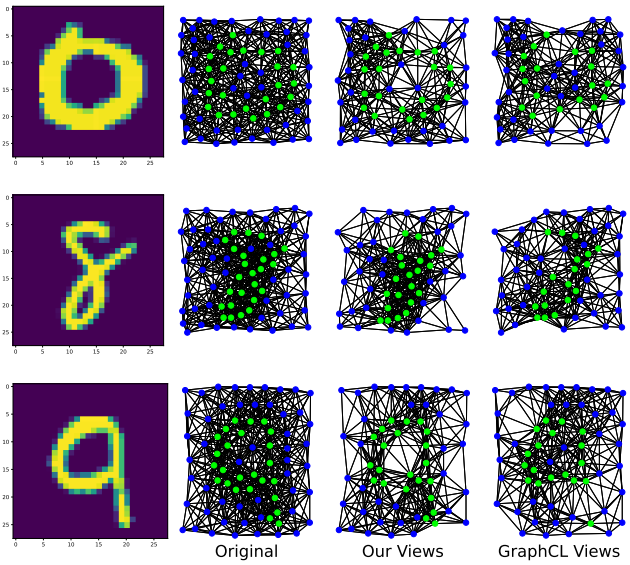

Figure 2. Positive view visualization on the MNIST-Superpixel dataset. Green nodes reflect the ground-truth saliency. Based on GCS, our views effectively preserve semantic information.

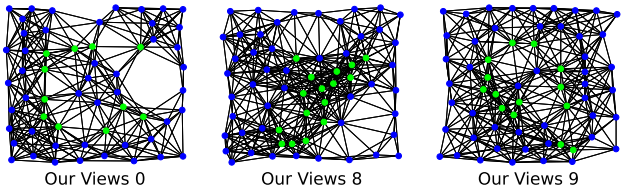

Figure 3. GCS-based hard negative views on MNIST-Superpixel. Green nodes reflect the ground-truth saliency. Identified semantics are intentionally destroyed.

The experiment results are presented in Table 4. We observe all methods are improving when graph size increases, whereas GCS performs the best in most cases.

Table 4. GCS against varying graph size.

	Part 1	**Part 2**	**Part 3**	**Part 4**	**Part 5**
ADGCL	63.3±1.1	67.1±0.5	75.1±1.3	83.1±1.0	93.6±0.8
RGCL	60.0±1.1	62.5±0.9	70.8±1.1	81.1±1.0	93.3±1.3
GCS	64.7±0.9	69.6±1.3	78.4±1.3	85.3±0.9	93.3±0.7

Varying sparsity level. COLLAB dataset is ordered according to the sparsity level and divided equally into 5 parts (1000 samples each part). The average sparsities for Part 1- Part 5 are respectively 0.16, 0.27, 0.42, 0.69, 0.96. The experiment results are presented in Table 5. It seems all methods are improving when the sparsity level increases, whereas GCS demonstrates strong advantage.

Table 5. GCS against varying sparsity level.

	Part 1	**Part 2**	**Part 3**	**Part 4**	**Part 5**
ADGCL	67.1±0.8	69.0±0.7	73.8±0.6	79.5±1.1	75.2±0.6
RGCL	62.7±0.6	65.6±1.3	67.7±1.2	74.8±1.4	77.0±0.5
GCS	68.3±0.6	73.6±1.3	76.8±1.2	81.6±1.3	76.9±0.4

Table 6. GCS against varying noise level.

noise-level	**0**	**0.05**	**0.1**	**0.15**	**0.2**
ADGCL	74.89±0.90	69.15±0.80	65.04±1.47	58.70±1.68	49.95±0.81
RGCL	70.92±0.65	67.27±1.55	63.68±0.92	60.46±1.98	54.34±0.85
GCS	75.56±0.41	72.15±1.19	69.70±1.40	67.38±1.38	63.09±1.43

Varying noise level. We randomly flip some proportions (noise-level) of edges in the graph of the COLLAB dataset. The experiment results are presented in Table 6. We observe that all methods suffer performance drops when the noise level increases, whereas GCS is always the most robust method against noise.

7. Conclusion

GCL has demonstrated remarkable success for graph SSL. To address the issues of conventional random augmentation in GCL, we propose the graph contrastive saliency (GCS), a simple and effective gradient-based approach that utilizes contrastively trained models to capture semantics. By leveraging the semantics, we introduce two data augmentation techniques to address the aforementioned challenges. Through extensive experimentation on benchmark datasets, we demonstrate that GCS improves the transferability and generalization ability of contrastively pre-trained GNNs and achieves state-of-the-art performance. We also provide visualizations of the generated graph views to show that GCS is able to preserve discriminative structures of input graphs.

Acknowledgements

This work is supported in part by Natural Science Foundation of China (NSFC) under contract No. 62125106, 61860206003, and 62088102, in part by Ministry of Science and Technology of China under contract No. 2021ZD0109901, in part by Young Elite Scientists Sponsorship Program by CAST under contract No. 2022QNRC001.

References

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. Sanity checks for saliency maps. In *NeurIPS*, 2018.

Adhikari, B., Zhang, Y., Ramakrishnan, N., and Prakash, B. A. Sub2vec: Feature learning for subgraphs. In

PAKDD, 2018.

Agarwal, S., Jabbari, S., Agarwal, C., Upadhyay, S., Wu, S., and Lakkaraju, H. Towards the unification and robustness of perturbation and gradient based explanations. In *ICML*, 2021.

Bai, B., Liang, J., Zhang, G., Li, H., Bai, K., and Wang, F. Why attentions may not be interpretable? In *KDD*, 2021.

Belghazi, I., Rajeswar, S., Baratin, A., Hjelm, R. D., and Courville, A. C. MINE: mutual information neural estimation. *CoRR*, abs/1801.04062, 2018.

Chang, S., Zhang, Y., Yu, M., and Jaakkola, T. S. Invariant rationalization. In *ICML*, 2020.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. E. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

Fan, S., Wang, X., Shi, C., Cui, P., and Wang, B. Generalizing graph neural networks on out-of-distribution graphs. *arXiv preprint arXiv:2111.10657*, 2021.

Ghorbani, A., Abid, A., and Zou, J. Interpretation of neural networks is fragile. In *AAAI*, 2019.

Grill, J., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. Á., Guo, Z., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. Bootstrap your own latent - A new approach to self-supervised learning. In *NeurIPS*, 2020.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *KDD*, 2016.

Hamilton, W. L., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*, 2020a.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V. S., and Leskovec, J. Strategies for pre-training graph neural networks. In *ICLR*, 2020b.

Li, H., Wang, X., Zhang, Z., and Zhu, W. OOD-GNN: out-of-distribution generalized graph neural network. *IEEE Trans. Knowl. Data Eng.*, 2022a.

Li, S., Wang, X., Zhang, A., Wu, Y., He, X., and Chua, T. Let invariant rationale discovery inspire graph contrastive learning. In *ICML*, 2022b.

Linsker, R. Self-organization in a perceptual network. *Computer*, 1988.

Liu, G., Zhao, T., Xu, J., Luo, T., and Jiang, M. Graph rationalization with environment-based augmentations. In *KDD*, 2022a.

Liu, S., Wang, H., Liu, W., Lasenby, J., Guo, H., and Tang, J. Pre-training molecular graph representation with 3d geometry. In *ICLR*, 2022b.

Liu, Y., Pan, S., Jin, M., Zhou, C., Xia, F., and Yu, P. S. Graph self-supervised learning: A survey. *IEEE Trans. Knowl. Data Eng.*, 2022c.

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

Misra, I. and van der Maaten, L. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020.

Mo, S., Kang, H. B., Sohn, K., Li, C.-L., and Shin, J. Object-aware contrastive learning for debiased scene representation. In *NeurIPS*, 2021.

Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., and Bronstein, M. M. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, 2017.

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

Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., and Jaiswal, S. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.

Poole, B., Ozair, S., van den Oord, A., Alemi, A. A., and Tucker, G. On variational bounds of mutual information. In *ICML*, 2019.

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

Rong, Y., Bian, Y., Xu, T., Xie, W., Wei, Y., Huang, W., and Huang, J. Self-supervised graph transformer on large-scale molecular data. In *NeurIPS*, 2020.

Rosenfeld, E., Ravikumar, P. K., and Risteski, A. The risks of invariant risk minimization. In *ICLR*, 2021.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *ICCV*, pp. 618–626, 2017.

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Zídek, A., Nelson, A. W. R., Bridgland, A., Penedones, H., Petersen, S., Simonyan, K., Crossan, S., Kohli, P., Jones, D. T., Silver, D., Kavukcuoglu, K., and Hassabis, D. Improved protein structure prediction using potentials from deep learning. *Nat.*, 2020.

Shervashidze, N., Vishwanathan, S. V. N., Petri, T., Mehlhorn, K., and Borgwardt, K. M. Efficient graphlet kernels for large graph comparison. In *AISTATS*, 2009.

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

Shlomi, J., Battaglia, P. W., and Vlimant, J. Graph neural networks in particle physics. *Mach. Learn. Sci. Technol.*, 2021.

Slack, D., Hilgard, S., Jia, E., Singh, S., and Lakkaraju, H. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *AAAI*, 2020.

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

Subramonian, A. Motif-driven contrastive learning of graph representations. In *AAAI*, 2021.

Sui, Y., Wang, X., Wu, J., Lin, M., He, X., and Chua, T. Causal attention for interpretable and generalizable graph classification. In *KDD*, 2022.

Sun, F., Hoffmann, J., Verma, V., and Tang, J. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *ICLR*, 2020.

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

van den Oord, A., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

van Erven, T. and Harremoës, P. Rényi divergence and kullback-leibler divergence. *IEEE Trans. Inf. Theory*, 2012.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR*, 2018.

Wu, Y., Wang, X., Zhang, A., He, X., and Chua, T. Discovering invariant rationales for graph neural networks. In *ICLR*, 2022.

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

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *ICLR*, 2019.

Yanardag, P. and Vishwanathan, S. V. N. Deep graph kernels. In *KDD*, 2015.

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

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

Yu, J., Liang, J., and He, R. Finding diverse and predictable subgraphs for graph domain generalization. *CoRR*, abs/2206.09345, 2022.

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

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

A. Algorithm

Algorithm 1 Graph Contrastive Saliency for Graph Contrastive Learning

Input: Dataset $\{G_i \ : \ i = 1, 2, \ldots, n\}$, initial encoder $f(\cdot)$, projection head $g(\cdot)$, iteration number T, hard negative efficient λ, binary threshold ϕ_{node} and ϕ_{edge}.
Output: Optimized encoder $f(\cdot)$

for sampled minibatch of data $\{G_i \ : \ i = 1, 2, \ldots, N\}$ **do**
  for $i = 1$ to N **do**
    Define: $I(z'_i, z''_i) = \log \frac{\exp(z'^T_i z''_i/\tau)}{\sum_{j=1, j\neq i}^{m} \exp(z'^T_i z''_j/\tau)}$
    Initialize: $G_i^{\text{mask}} = G_i, \overline{\omega} = 0$
    for $t = 1$ to T **do**
      $h_i = f(G_i) \ ; \ h_i^{\text{mask}} = f(G_i)$
      $z_i = g(h_i) \ ; \ z_i^{\text{mask}} = g(h_i^{\text{mask}})$
      Compute $I(z_i, z_i^{\text{mask}})$
      $\omega^t = \text{GCS}(I(z_i, z_i^{\text{mask}}), h_i)$ (Algorithm 2)
      $\overline{\omega} = \text{Max}(\omega^t, \overline{\omega})$ if $t > 1$ else ω^t (Eq. (5))
      $\mathbf{M}_V^t, \mathbf{M}_E^t = \text{Binarize}(\overline{\omega})$
      $G_i^{\text{mask}} = (V^t \odot (\mathbf{1} - \mathbf{M}_V^t), E^t \odot (\mathbf{1} - \mathbf{M}_E^t))$
    end for
    Perform semantic-preserved augmentation: $G_i^{1,+}$ and $G_i^{2,+}$; (Eq. (7))
    Perform environment-distilled augmentation: G_i^- (Eq. (9))
    Define: $\mathcal{L}_i = -I(f(G_i^{1,+}), f(G_2^{2,+})) + \lambda I(f(G_i), f(G_i^-))$
  end for
  $\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}_i$
  Update encoder $f(\cdot)$ and $g(\cdot)$ to minimize $\mathcal{L}$
end for
Return: $f(\cdot)$

Algorithm 2 Graph Contrastive Saliency (GCS)

Input: Mutual information $I(z, z^{\text{mask}})$, graph representation h.
Output: Graph contrastive saliency score ω_u and ω_{uv} of graph $G = (V, E)$, where $v_u \in V$ and $e_{uv} \in E$.

grad $= \text{Autograd}(I(z, z^{\text{mask}}), h)$ (Eq. (2));
weight $= \text{Average\_Pooling}(\text{grad})$ (Eq. (3));
$\omega_u = \text{Normalize}(\text{Sum}(\text{weight}, h))$ (Eq. (4));
$\omega_{uv} = (\omega_u + \omega_v)/2$;
Return ω_u, ω_{uv}

B. MNIST-superpixel Dataset

We summarize the statistics of the MNIST-superpixel dataset in Table 7. The detailed information for MNIST-superpixel can be referred in (Monti et al., 2017).

Table 7. Statistics for MNIST-superpixel dataset.

Dataset	Category	# of Graphs	Avg. # of Nodes	Avg. Degree
MNIST	SuperPixel	70,000	70.57	8

C. Baselines

Here we briefly introduce some important graph self-supervised learning baseline methods.

- **InfoGraph** (Sun et al., 2020) InfoGraph maximizes the mutual information between the graph-level representation and the representations of substructures of different scales.
- **Attribute Masking** (Hu et al., 2020b) Attribute Masking learns the regularities of the node/edge attributes distributed over graph structure to capture inherent domain knowledge.
- **Context Prediction** (Hu et al., 2020b) Context Prediction predicts the surrounding graph structures of subgraphs to pre-train a backbone GNN, so that it maps nodes appearing in similar structural contexts to nearby representations.
- **GraphCL** (You et al., 2020) GraphCL learns unsupervised representations of graph data through contrastive learning with random graph augmentations.
- **JOAOv2** (You et al., 2021) JOAOv2 leverages GraphCL as the baseline model and automates the selection of augmentations when performing contrastive learning.
- **AD-GCL** (Suresh et al., 2021) AD-GCL optimizes adversarial graph augmentation strategies used in GCL to avoid capturing redundant information.
- **RGCL** (Li et al., 2022b) RGCL uses a rationale generator to reveal salient features about graph instance-discrimination as the rationale, and then creates rationale-aware views for contrastive learning.

D. Unsupervised Learning Settings

Dataset We summarize the statistics of the TU-datasets (Morris et al., 2020) for unsupervised learning in Table 8. We obtain the data from Pytorch Geometric Library[2].

Table 8. Statistics of TU-datasets for unsupervised learning.

Dataset	Category	# of Graphs	Avg. # of Nodes	Avg. Degree
NCI1	Biochemical Molecule	4,110	29.87	1.08
PROTEINS	Biochemical Molecule	1,113	39.06	1.86
DD	Biochemical Molecule	1,178	284.32	715.66
MUTAG	Biochemical Molecule	188	17.93	19.79
COLLAB	Social Network	5,000	74.49	32.99
REDDIT-binary	Social Network	2,000	429.63	1.15
REDDIT-Multi-5K	Social Network	2,000	17.93	497.75
IMDB-binary	Social Network	1,000	19.77	96.53

E. Transfer Learning Settings

Dataset We utilize MoleculeNet (Wu et al., 2018) as downstream tasks for transfer learning and summarize the statistics in Table 9.

Table 9. Statistics of MoleculeNet dataset for downstream transfer learning.

Dataset	Category	# of Graphs	Avg. # of Nodes	Avg. Degree
BBBP	Biochemical Molecule	2,039	24.06	51.90
TOX21	Biochemical Molecule	7,831	18.57	38.58
TOXCAST	Biochemical Molecule	8,576	18.78	38.52
SIDER	Biochemical Molecule	1,427	33.64	70.71
CLINTOX	Biochemical Molecule	1,477	26.15	55.76
MUV	Biochemical Molecule	93,087	24.23	52.55
HIV	Biochemical Molecule	41,127	25.51	54.93
BACE	Biochemical Molecule	1,513	34.08	73.71

[2]https://github.com/graphdeeplearning/benchmarking-gnns/tree/master/data

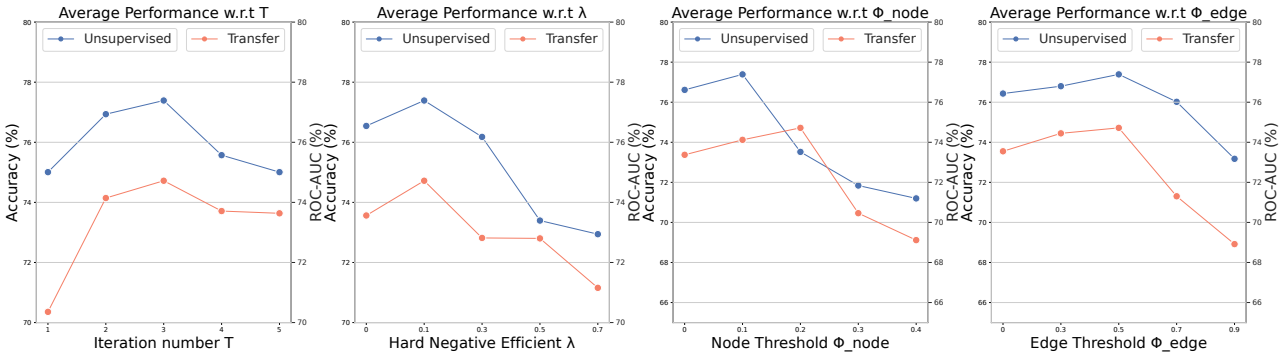

Figure 4. Influence of hyperparameter T, λ, ϕ_{node} and ϕ_{edge}.

Evaluation Protocol We follow the same evaluation protocol as outlined by Hu et al. (2020b). Our self-supervised methods are trained on the ChEMBL dataset (Mayr et al., 2018), and are later evaluated for transferability by fine-tuning them on the MoleculeNet datasets and evaluating them using the labels of these datasets. To ensure fair comparison, we use the same GIN encoder and settings by Hu et al. (2020b). During fine-tuning, an additional linear graph prediction layer is added to the encoder, which maps the representations to the task labels. This is trained end-to-end using gradient descent.

F. Model Structure and Hyperparameters

For fair comparison, we follow the backbone setting in You et al. (2020), which adopt the GIN as the graph encoder (Xu et al., 2019). We summarize the corresponding hyperparameters in Table 10.

Table 10. Model architectures and hyperparameters.

Experiment	Unsupervised Learning	Transfer Learning
Backbone Type	GIN	GIN
Backbone hidden dim	[32, 32, 32]	[300, 300, 300, 300, 300]
Projector hidden dim	[32, 32]	[300, 300]
Pooling Type	Global Add Pool	Global Mean Pool
Temperature	{0.05, **0.1**, 0.2, 0.3, 0.5}	{0.05, **0.1**, 0.2, 0.3, 0.5}

G. Hyperparameter Sensitivity

In this section, we investigate the influence of four hyperparameters on both unsupervised learning and transfer learning settings, which includes iteration number T, hard negative efficient λ, binary thresholds ϕ_{node} and ϕ_{edge}. We present the result in Figure 4.

Effect of T We vary T from 1 to 5 while keeping other parameters fixed. GCS significantly improves when the iteration number increases from 1 to 2. We attribute this improvement to the fact that a small number of iterations often only detect incomplete sub-regions of the semantic properties in a graph. Performing GCS-based augmentations on such incomplete saliency may further harm the semantic consistency of the generated positive views. Therefore, it is important to perform sufficient iterative refinement to obtain a confident mask. Figure 4 shows that increasing the iteration number substantially enhances performance, with the best results achieved at 3 iterations. Our GCS-based method also shows stable results for a large number of iterations (e.g., 5) and converges to some stationary values. This may be because the output saliency mask will include the entire graph as the saliency, resulting in the model degenerating to GraphCL when performing GCS-based augmentations.

Effect of λ We vary λ from 0 to 0.7 while keeping other parameters fixed. The environmental objective $\mathcal{L}_{\text{env}}$ serves as a regularizer in the training process, encouraging the graph model to not focus on environmental information. Note that when $\lambda = 0$, the framework degenerate to the variant model w/o hard negative aug. in Section 6.3. However, overemphasizing it

(e.g., setting a large value for λ) may also degrade performance. Therefore, we recommend tuning it around 0.1 based on the experimental results in Figure 4, which benefits contrastive learning.

Effect of Threshold ϕ_{node} and ϕ_{edge} Thresholds ϕ_{node} and ϕ_{edge} control the sensitivity of our framework to dropping nodes and edges, respectively. We vary ϕ_{node} from 0 to 0.4 and ϕ_{edge} from 0 to 0.9 to investigate their effects. The results in Figure 4 suggest choosing a relatively small number (e.g., 0.1 or 0.2) for the node dropping threshold ϕ_{node} and a number around 0.5 for the edge dropping threshold ϕ_{edge}. This may be because removing a selected node and all its connections can cause a significant change in the original graph structure, requiring caution when dropping nodes. In contrast, edges often contain less information and more noise, so we can choose a larger threshold to generate an augmentation view that is significantly different from the original graph while still preserving semantics, improving the robustness of the graph model.

H. Proof of Theorem 1

Proof. To prove that Theorem 1 holds, we firstly refer to the supporting theorem given by Agarwal et al. (2021) :

Theorem 2. *Let g be a function defined as:*

$$g(\mu) = \mathbb{E}_a \sim N(\mu, \sigma^2)[h(a)]\,.$$

Then g is $(h_{max}/2\sigma)$ - Lipschitz w.r.t. $\ell 2$ norm.

Having Theorem 2, and the assumption that p^{th} dimension of feature $h_{v,p}$ across $v \in V$ nodes follows Gaussian distribution, i.e., $h_{v,p} \sim \mathcal{N}(\mu_p, {\sigma_p}^2), \forall v$, we may define:

$$\begin{aligned} g(\mu_p) = \alpha_p &= \frac{1}{V}\sum_{v=1}^{V}\frac{\partial L(h_{v,p})}{\partial h_{v,p}} \\ &\overset{V\to\infty}{=} \mathbb{E}_{h_{v,p}}\frac{\partial L(h_{v,p})}{\partial h_{v,p}} \\ &= \mathbb{E}_{h_{v,p}}\tilde{\mathcal{L}}(h_{v,p})\,, \end{aligned}$$

$h_{v,p}$ denotes the p^{th} dimension of the feature $\boldsymbol{h}_v$. The mean value of the p^{th} dimension of the feature $\boldsymbol{h}_v$ is denoted as μ_p. We can then write:

$$\tilde{\mathcal{L}}(h_{v,p}) = \frac{\partial L(h_{v,p})}{\partial h_{v,p}}\,.$$

According to Agarwal et al. (2021), we have the following lemma:

Lemma 1. *Let X and Y be random variables over set $S = \{a \in \mathbb{R}^d \mid \|a\|_2 \le a_{max}$ and $d \in \mathcal{N}\}$, Then:*

$$\|\mathbb{E}[X] - \mathbb{E}[X]\|_2 \le a_{max}\sqrt{\frac{KL(X||Y)}{2}}\,.$$

We now develop the analysis specifically for the proposed GCS. Since $g(\mu_p) = \mathbb{E}_{h_{v,p}}\tilde{\mathcal{L}}(h_{v,p})$ for (ours), and by virtue of the assumption set given in the main paper, we specifically have that p^{th} dimension of feature $h_{v,p}$ across $v \in V$ nodes follows Gaussian distribution, i.e., $h_{v,p} \sim \mathcal{N}(\mu_p, {\sigma_p}^2), \forall v$, and $\|\tilde{\mathcal{L}}(h_{v,p})\|_2 \le \tilde{L}_{\max}$. We define μ'_p, to be the mean value of alternative graph composed of nodes v', and $h_{v',p} \sim \mathcal{N}(\mu'_p, {\sigma_p}^2), \forall v'$. Given these conditions, we immediately have:

$$\begin{aligned} &\|g(\mu_p) - g(\mu'_p)\|_2 \\ =&\|\mathbb{E}_{h_{v,p}}\tilde{\mathcal{L}}(h_{v,p}) - \mathbb{E}_{h_{v',p}}\tilde{\mathcal{L}}(h_{v',p})\|_2 \\ \le&\tilde{\mathcal{L}}_{\max}\sqrt{\frac{\mathrm{KL}(\tilde{\mathcal{L}}(h_{v,p})||\tilde{\mathcal{L}}(h_{v',p}))}{2}} \\ \le&\tilde{\mathcal{L}}_{\max}\sqrt{\frac{\mathrm{KL}(h_{v,p}||h_{v',p})}{2}} \ \big(\text{Data processing inequality, van Erven \& Harremoës, 2012}\big) \\ \le&\tilde{\mathcal{L}}_{\max}\frac{\|\mu_p - \mu'_p\|_2}{2\sigma}\,. \end{aligned} \tag{13}$$

We now show that the function $Q(\boldsymbol{\mu}_v) = \sum_p h_{v,p} g(\mu_{v,p})$ is C-Lipschitz, which necessarily and sufficiently requires:

$$\begin{aligned}\|Q(\boldsymbol{\mu}_v) - Q(\boldsymbol{\mu}'_v)\|_2 &= \|\sum_p h_{v,p} g(\mu_{v,p}) - \sum_p \hat{h}_{v,p} g(\mu_{v',p})\|_2 \\ &\leq C\|\boldsymbol{\mu}_v - \boldsymbol{\mu}'_v\|_2.\end{aligned}$$

Where vector $\boldsymbol{\mu}_v$ collects the values of μ_p as the p^{th} dimension in vector $\boldsymbol{\mu}_v$, and $\boldsymbol{\mu}'_v$ collects the values of μ_p as the p^{th} dimension in vector $\boldsymbol{\mu}'_v$. To see this, we use the result from Eq. (13)

$$\begin{aligned}&\|\sum_p h_{v,p} g(\mu_p) - \sum_p h_{v',p} g(\mu'_p)\|_2 \\ \leq &\sum_p \|h_{v,p}\, g(\mu_p) - h_{v',p}\, g(\mu'_p)\|_2 \text{ (triangle inequality)} \\ \leq &\sum_p h^*_p \|g(\mu_p) - g(\mu'_p)\| \\ \leq &\sum_p h^*_p \tilde{\mathcal{L}}_{\max} \frac{\|\mu_p - \mu'_p\|_2}{2\sigma} \text{ (using Eq. (13))} \\ \leq &h^*_p \tilde{\mathcal{L}}_{\max} \sum_p \frac{\|\mu_p - \mu'_p\|_2}{2\sigma} \\ = &\frac{h^*_p \tilde{\mathcal{L}}_{\max}}{2\sigma} \|\boldsymbol{\mu}_v - \boldsymbol{\mu}_{v'}\|_2 \\ = &C\|\boldsymbol{\mu}_v - \boldsymbol{\mu}_{v'}\|_2. \end{aligned} \tag{14}$$

Here $h^*_p = \max\left(-h_{v',p} + |h_{v,p} - h_{v',p}|\tilde{\mathcal{L}}_{\max}/\epsilon, h_{v,p} + |h_{v,p} - h_{v',p}|\tilde{\mathcal{L}}_{\max}\right)/\epsilon$, which is a value depending on $\tilde{\mathcal{L}}_{\max}$ and $h_{\max}$. Here $|h_{v,p} - h_{v',p}| \geq \epsilon$, and if $\epsilon = 0$, the equality in Eq. (14) directly holds.

This completes the proof.

□

I. More Visualization Results

Following the setting in RGCL (Li et al., 2022b), we pre-trained the backbone model on Zinc-2M, a dataset of 2 million unlabeled molecule graphs extracted from ZINC15 database (Sterling & Irwin, 2015). Subsequently, we evaluate the performance of the pre-trained model on Mutag, a real-world benchmark dataset of molecules, to capture its saliency. As can be seen in Figure 5, GCS mostly captures semantically meaningful substrucutres as saliency (blue parts), such as the linking nodes between several carbon rings.

Convergence of α. It is actually non-trivial to prove the convergence of α in a principled way. But it is indeed helpful to observe the change of α as GCS evolves. We illustrate the change of α at different training iterations, and include the visualization results in in Figure 6. Here, Figure 6 presents the positive views created based on the computed saliency under different iterations. We find α beyond 6 iterations have converged to the same saliency pattern till the final iteration of 30, whereas we can observe more evident changes of this α pattern during earlier iterations. This empirically demonstrates the convergence of α.

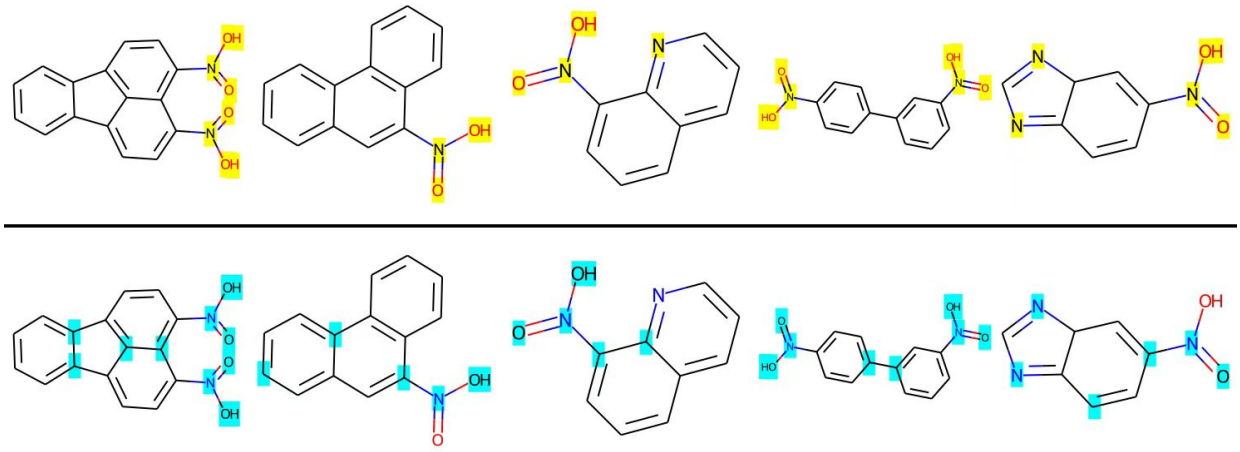

Figure 5. **Visualization of Mutag Graphs.** The first row presents the saliency labeled by chemistry experts and the second row presents those discovered by our GCS.

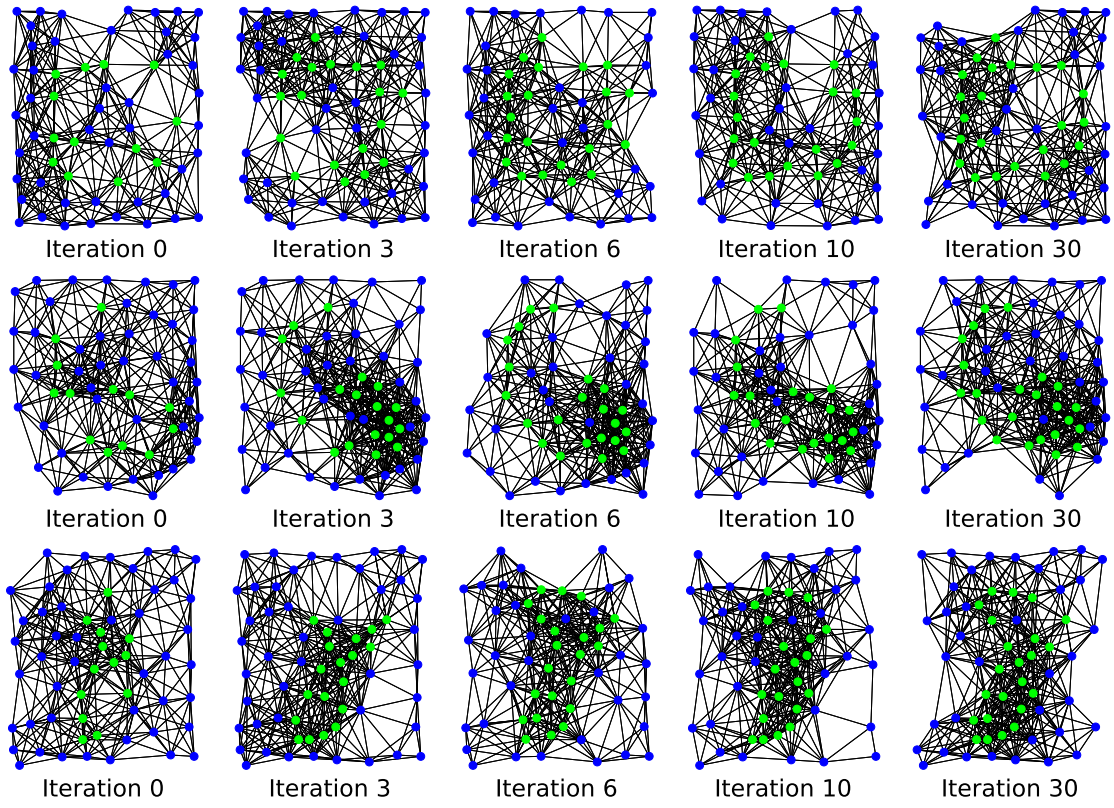

Figure 6. Positive view saliency visualization on the MNIST-Superpixel dataset. Green nodes reflect the ground-truth saliency. Based on GCS, our views effectively preserve semantic information.