SGCL: Semantic-aware Graph Contrastive Learning with Lipschitz Graph Augmentation

Jinhao Cui[1,*], Heyan Chai[1,*], Xu Yang[1], Ye Ding[2], Binxing Fang[1,3] and Qing Liao[1,3✉]
[1]Harbin Institute of Technology Shenzhen, Shenzhen, China
[2]Dongguan University of Technology, Dongguan, China
[3]Peng Cheng Laboratory, Shenzhen, China
{cuijinhao, chaiheyan, xuyang97}@stu.hit.edu.cn, dingye@dgut.edu.cn, fangbx@cae.cn, liaoqing@hit.edu.cn

2024 IEEE 40th International Conference on Data Engineering (ICDE) | 979-8-3503-1715-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICDE60146.2024.00235

***Abstract*—Graph contrastive learning (GCL) has gained increasing interest as a solution for graph representation learning. In GCL, graph augmentation is essential to generate contrastive samples used for contrastive learning. Recently, most existing methods employ learnable graph view generators to augment graphs based on the node probability distribution adaptively. However, these methods cannot ensure that semantic-related nodes are preserved during graph augmentation, leading to performance degradation. To tackle this issue, we propose a novel approach called Semantic-aware Graph Contrastive Learning (SGCL), which can generate high-quality contrastive samples by only augmenting semantic-unrelated nodes so as to facilitate the performance of GCL on downstream tasks. Specifically, we first design a Lipschitz constant generator to compute the Lipschitz constants that measure the semantic relevance of each node. Then, we propose the Lipschitz graph augmentation to augment graphs while only dropping these semantic-unrelated nodes with small Lipschitz constants. Furthermore, we propose semantic-aware contrastive learning to obtain more refined representations by contrasting the graph-level representation of anchor graphs and high-quality generated samples. Experimental results on unsupervised learning and transfer learning demonstrate the effectiveness of SGCL compared to state-of-the-art methods.**

***Index Terms*—Graph Contrastive Learning, Graph Classification, Graph Neural Networks, Self-supervised Learning**

I. Introduction

Graph neural networks (GNNs) [1]–[3], which capture the structural information of graphs by aggregating the neighborhood [4], have shown great power in various domains such as drug-drug interaction prediction [5], recommended systems [6], [7], and sentiment analysis [8]–[10]. Due to their outstanding graph-representational power, GNNs can model complex real-world entities, including molecules [11] and social networks [12]. However, most GNN methods are trained in a supervised manner, limited by the number of true annotations [13].

More recently, self-supervised graph contrastive learning (GCL) [14], [15], which does not rely on finely annotated data to obtain a refined graph representation, has received a surge of interest. GCL is a powerful method that pre-trains a GNN in a self-supervised manner, pulling positive pairs closer and pushing negative pairs farther apart [16], thereby facilitating the model fine-tuning on downstream tasks [17]. One of the essential components in the GCL is graph augmentation, employed to generate contrastive samples [18]. The generated sample and its corresponding anchor graph are treated as a positive pair. The anchor graph and the contrastive samples generated by other graphs are treated as negative pairs. Most previous methods [19]–[21] randomly utilize graph perturbations as graph augmentation, such as node dropping, edge perturbations, and attribute masking. However, random perturbations lead to the corruption of discriminative semantics, which makes huge label distribution differences between generated samples and anchor graphs [22]. These differences misguide contrastive learning and lead to performance degradation.

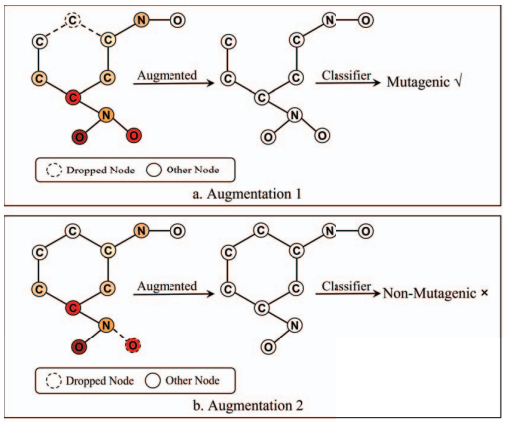


Figure 1. The effect of different graph augmentation presets on generating samples. Augmentation 1 drops a semantic-unrelated node, but Augmentation 2 drops a semantic-related node in the same node probability distribution.

To address the problem, some methods [20], [21] pick graph augmentation per dataset by tedious trial-and-errors, which increases the labor cost and limits the framework's generality. Recently, some state-of-the-art GCL methods [12], [23], [24] develop learnable graph view generators to reduce non-trial efforts. These generators select different augmentations based on the probability distribution of nodes. They translate the probability distribution into node weights and then choose a corresponding graph augmentation operation for each node. However, the probability distribution of nodes does not always equate to semantic relevance. Some nodes in a graph may

* The first two authors contributed equally to this work.
✉ Corresponding authors.

2375-026X/24/$31.00 ©2024 IEEE
DOI 10.1109/ICDE60146.2024.00235

have different semantic relevance despite having a similar probability distribution. The fact that augmentation depends on nodes' probability distribution leads to a significant variation in generated samples for different graph augmentation presets.

As shown in Figure 1, we use different graph augmentations to generate contrastive samples from one anchor graph with the same node probability distribution. The first graph augmentation identifies the semantic-related nodes and preserves the discriminative semantics of the original graph, enabling the classifier to identify its salient attribute accurately. Conversely, the second graph augmentation misjudges semantic structure and generates a sample that corrupts the salient semantic structure, resulting in the classifier misjudging its label. However, an ideal augmentation preset is not prior knowledge, and choosing the better augmentation takes non-trial efforts based on the node probability distribution. Therefore, during graph augmentation, it is crucial to prioritize the semantic relevance of the nodes rather than the probability distribution. An ideal graph augmentation method should preserve these semantic-related nodes in the contrastive samples, thereby maintaining the discriminative semantics of the original graphs.

Inspired by the above insight, we propose a novel **S**emantic-aware **G**raph **C**ontrastive **L**earning (SGCL) to generate high-quality samples and improve the performance of graph contrastive learning on downstream tasks. Specifically, we first design a Lipschitz constant generator to calculate the node Lipschitz constants to determine the semantic relevance of each node in anchor graphs. We further propose the Lipschitz graph augmentation, which combines Lipschitz constants to obtain the augmentation probability and only augments semantic-unrelated nodes with smaller Lipschitz constants to generate contrastive samples. We also propose semantic-aware contrastive learning to improve the capture of refined representations by contrasting the graph-level representation of anchor graphs and high-quality generated samples. In addition, we conduct extensive experiments to demonstrate that SGCL outperforms the previous state-of-the-art methods. We highlight the major contribution of this paper as follows:

- We propose a novel **S**emantic-aware **G**raph **C**ontrastive **L**earning (SGCL), which can generate high-quality contrastive samples with more discriminative semantic information and improve the performance of GCL on downstream tasks.
- We define the Lipschitz constant of GNNs under a certain graph augmentation, demonstrating that a graph augmentation operation with a smaller Lipschitz constant leads to less semantic corruption.
- We propose the Lipschitz graph augmentation, an effective mechanism to generate high-quality contrastive samples by only dropping semantic-unrelated nodes with smaller Lipschitz constants.
- We conduct extensive experiments on various datasets to evaluate our proposed SGCL. Experimental results demonstrate the effectiveness of SGCL compared to state-of-the-art methods.

II. Related Work

A. Graph Classification

Graph classification [25]–[27] is a fundamental task in various domains, including chemistry [28], biology [5], and social networks [29]. Early graph kernel methods [30]–[32] employed atoms as vertices and bonds as edges to learn the representation of molecular graphs. Recently, Graph Neural Networks (GNNs) [33]–[35] have achieved greater success in graph classification tasks through message passing and information propagation between neighbor nodes [1]–[3]. After obtaining the node representation, GNNs normally use a pooling function to aggregate information [36], [37] and obtain the graph representation used to solve graph classification. However, the performance of these supervised methods depends on a large amount of fine-annotated graphs [38], which may not be available in real-life scenarios. To tackle this problem, our work focuses on self-supervised graph contrastive learning, which can overcome the scarcity of fine annotations.

B. Graph Contrastive Learning

Inspired by the success of contrastive learning in computer vision domains [39], [40], self-supervised graph contrastive learning has received tremendous attention. Graph Contrastive Learning (GCL) [14], which pulls positive pairs closer and pushes negative pairs apart, can be categorized into two groups. One group aims to capture abundant information by contrasting local and global representation. For example, InfoGraph [41] is designed to maximize the mutual information between graph-level representation and node-level representation in different granularity to obtain expressive representation for graphs. Another group [19]–[21] aims to capture the discriminative semantics between different views, learning a refined representation tolerant to graph augmentation. Although our proposed method also follows this previously established framework, we improve the framework with the Lipschitz constant generator to make our model more effective.

C. Graph Augmentation

The quality of augmented samples has a crucial influence on GCL performance, making graph augmentation significant. Although GraphCL [20] proposes four types of graph augmentation operations, including Node Dropping, Edge Perturbation, Attribute Masking, and Sub-graph, the improvement comes at the price of tedious manual trial-and-error. To solve this problem, some GCL methods [12], [23], [24] design a learnable view generator to augment anchor graphs desirably according to the node probability distribution in different datasets. However, the node probability distribution does not necessarily equal semantic relevance, leading to the fact that methods with view generators cannot preserve semantic-related nodes and have a significant variation of augmented samples in different graph augmentation presets. Therefore, we propose Lipschitz graph augmentation, which preserves semantic-related nodes during graph augmentation, to generate high-quality samples with more discriminative semantics to improve the performance of GCL on the downstream tasks.

III. Preliminaries

In this section, we provide formal definitions for the terminology used in this paper for the sake of clarity.

A. Definitions of Graph

In this paper, we denote a graph as $G = (V, \boldsymbol{H}, \boldsymbol{A})$, where $V = \{v_1, v_2, \cdots, v_{|V|}\}$ denotes the node-set, $\boldsymbol{H} = [\boldsymbol{h}_1^{(0)}, \boldsymbol{h}_2^{(0)}, \cdots, \boldsymbol{h}_{|V|}^{(0)}] \in \mathbb{R}^{|V| \times d^{(0)}}$ denotes the initial node representation, $d^{(0)}$ is the dimension of the initial node representation, $\boldsymbol{A} \in \mathbb{R}^{|V| \times |V|}$ denotes the adjacency matrix, and $|V|$ is the number of nodes. We employ a GNN encoder $f(\cdot, \cdot; \boldsymbol{\theta})$ to obtain the final node representations, which can be described as follows:

$$\boldsymbol{H}^{(l)} = f(\boldsymbol{H}, \boldsymbol{A}; \boldsymbol{\theta}), \tag{1}$$

where $\boldsymbol{H}^{(l)} = [\boldsymbol{h}_1^{(l)}, \boldsymbol{h}_2^{(l)}, \cdots, \boldsymbol{h}_{|V|}^{(l)}] \in \mathbb{R}^{|V| \times d^{(l)}}$ denotes the node representation obtained by the GNN encoder $f(\cdot, \cdot; \boldsymbol{\theta})$, l is the layer of the GNN encoder $f(\cdot, \cdot; \boldsymbol{\theta})$, $d^{(l)}$ is the dimension in layer l, and $\boldsymbol{\theta}$ is parameter of the GNN encoder $f(\cdot, \cdot; \boldsymbol{\theta})$.

Furthermore, we denote the edge connecting nodes v_i and v_j as e_{ij}. Following the previous methods [42], [43], we reasonably assume that the probability of the edge e_{ij} only depends on the representations of v_i and v_j.

Definition 1 (Edge Probability). *We denote the probability of edge e_{ij} as $P(e_{ij}|(\boldsymbol{h}_i^{(l)}, \boldsymbol{h}_j^{(l)}))$, which is calculated by the representations of nodes v_i and v_j,*

$$P(e_{ij}|(\boldsymbol{h}_i^{(l)}, \boldsymbol{h}_j^{(l)})) = \delta((\frac{\boldsymbol{h}_i^{(l)}}{d_i} + \frac{\boldsymbol{h}_j^{(l)}}{d_j})\boldsymbol{w}_{ij}^T), \tag{2}$$

where $\boldsymbol{h}_i^{(l)}$ and $\boldsymbol{h}_j^{(l)}$ are the representations of v_i and v_j, d_i and d_j are the degree of v_i and v_j, $\boldsymbol{w}_{ij} \in \mathbb{R}^{1 \times d^{(l)}}$ is the weight parameter, and $\delta(\cdot)$ is the logistic function.

As edges are unique, we assume that all edges in a graph are conditionally independent [44] so that the probability of the graph G can be defined as follows:

Definition 2 (Graph Probability). *We denote the probability of graph G as $P(G|\boldsymbol{H}^{(l)})$, which is calculated by the probability of each edge,*

$$P(G|\boldsymbol{H}^{(l)}) = \prod_{i,j} p(e_{ij}|(\boldsymbol{h}_i^{(l)}, \boldsymbol{h}_j^{(l)})), \tag{3}$$

where $\boldsymbol{H}^{(l)} = [\boldsymbol{h}_1^{(l)}, \boldsymbol{h}_2^{(l)}, \cdots, \boldsymbol{h}_{|V|}^{(l)}] \in \mathbb{R}^{|V| \times d^{(l)}}$ denotes the representation of nodes in graph G.

B. Graph Augmentation and GNNs Stability

Contrastive samples are generated by applying graph augmentation operations such as node dropping, edge perturbation, and attribute masking to an anchor graph. It is argued that edge perturbation and attribute masking can be regarded as a special case of node dropping. Furthermore, previous studies [20], [21] have shown that node dropping has more advantages for downstream tasks compared to the other two methods. Therefore, this paper only focuses on node dropping as a graph augmentation. The contrastive sample generated from G is denoted as $\hat{G} = (\hat{V}, \hat{\boldsymbol{H}}, \hat{\boldsymbol{A}})$, and we present the specific definition of graph augmentation as follows,

Definition 3 (Graph Augmentation). *Given a Graph G, the graph augmentation operation is denoted as $\Phi(\cdot, \cdot, \cdot)$, and the generated sample $\hat{G}$ can be denoted as,*

$$\hat{G} = \Phi(G, \rho|V|, P(V)), \tag{4}$$

where G is the input anchor graph, $|V|$ is the number of nodes in the graph G, ρ is a predefined augmentation ratio, $\rho|V|$ is the number of dropping nodes, and $P(V) = \{P(v_1), P(v_2), \cdots, P(v_{|V|})\}$ is the set of augmentation probability.

Based on the definition, we further represent graph augmentation in the following three cases:

(1) When generating a sample by dropping one specific node v_r, the graph augmentation is denoted as $\hat{G} = \Phi(G, 1, v_r)$.
(2) When generating a sample by dropping nodes randomly, the graph augmentation is denoted as $\hat{G} = \Phi(G, \rho|V|, 1)$.
(3) When generating a sample by dropping a set of nodes with augmentation probability, the specific graph augmentation is denoted as $\hat{G} = \Phi(G, \rho|V|, P(V))$.

During graph augmentation, the differences between anchor graphs and contrastive samples can be measured by topology distance and representation distance. The topology distance can be equipped with a distance metric $D_T(\cdot, \cdot)$, which can be described as follow:

$$D_T(G, \hat{G}) = \|\boldsymbol{A} - \hat{\boldsymbol{A}}\|, \tag{5}$$

where $\boldsymbol{A}$ is the adjacency matrix of an anchor graph G, and $\hat{\boldsymbol{A}}$ is the adjacency matrix of the contrastive sample $\hat{G}$.

We also define that the representation distance under a GNN $f(\cdot, \cdot; \boldsymbol{\theta})$ can be equipped with a distance metric $D_R(\cdot, \cdot)$, which can be described as follow:

$$D_R(G, \hat{G}) = \|f(\boldsymbol{H}, \boldsymbol{A}; \boldsymbol{\theta}) - f(\hat{\boldsymbol{H}}, \hat{\boldsymbol{A}}; \boldsymbol{\theta})\|, \tag{6}$$

where $G = (V, \boldsymbol{H}, \boldsymbol{A})$ is the anchor graph, $\hat{G} = (\hat{V}, \hat{\boldsymbol{H}}, \hat{\boldsymbol{A}})$ is the contrastive sample, $\boldsymbol{H}$ and $\hat{\boldsymbol{H}}$ are the initial node representation, $\boldsymbol{A}$ and $\hat{\boldsymbol{A}}$ are the adjacency matrices. Following the previous study [45], we then present the definition of GNN stability corresponding to a specific graph augmentation based on the above definition:

Definition 4 (GNNs Stability to Graph Augmentation). *A GNN $f(\cdot, \cdot; \boldsymbol{\theta})$ is K-stable to a certain graph augmentation $\Phi(\cdot, \cdot, \cdot)$ with respect to a graph set $\mathcal{G} = \{G_1, G_2, \cdots, G_N\}$:*

$$\sup_{G \in \mathcal{G}} \|f(\boldsymbol{H}, \boldsymbol{A}; \boldsymbol{\theta}) - f(\hat{\boldsymbol{H}}, \hat{\boldsymbol{A}}; \boldsymbol{\theta})\| \leq K \cdot D_T(G, \hat{G}), \tag{7}$$

where $\mathcal{G} = \{G_1, G_2, \cdots, G_N\}$ is a set of anchor graphs and $D_T(G, \hat{G})$ is the topology distance between anchor graphs and contrastive samples calculated by Eq.(5).

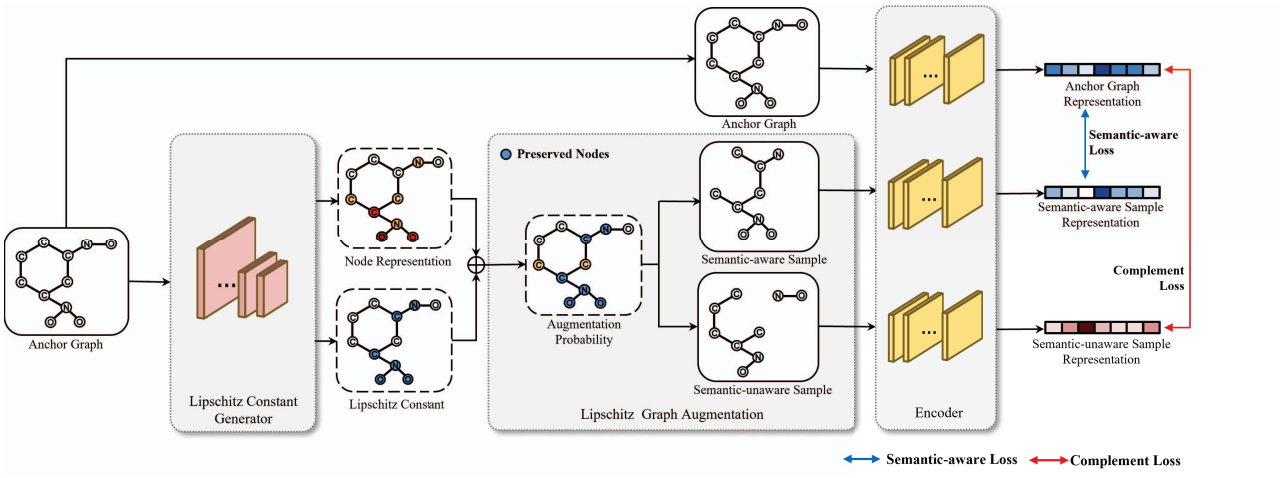


Figure 2. The overall illustration of the SGCL framework.

C. The Lipschitz Constant and Lipschitz Continuous

The Lipschitz constant is closely related to the robustness of a model. In particular, a model with a small Lipschitz constant is more likely to be robust to perturbations in the input data. [46]. A function $f(\cdot) : \mathcal{X} \to \mathcal{Y}$ is Lipschitz continuous if there exists a constant $K \geq 0$ such that for all $x, y \in \mathcal{X}$:

$$\|f(x) - f(y)\| \leq K\|x - y\|, \tag{8}$$

where $\mathcal{X}$ is the domain of definition, $\mathcal{Y}$ is the domain of value, $f(x), f(y) \in \mathcal{Y}$, and $\|\cdot\|$ denotes the norm of a matrix. The smallest constant K is the Lipschitz constant of $f(\cdot)$.

IV. Methodology

In this section, we propose the novel **S**emantic-aware **G**raph **C**ontrastive **L**earning (SGCL) to improve the quality of generated contrastive samples and facilitate the efficiency of GCL on graph classification tasks. We first present the definition of the Lipschitz constant about GNNs under a certain graph augmentation and demonstrate the correctness of combining graph augmentation with the Lipschitz constant. Next, we present the overall framework of SGCL as shown in Figure 2: Lipschitz constant generator, Lipschitz graph augmentation, and the final semantic-aware contrastive learning.

A. The Lipschitz Constant of GNNs

Following the definition of the Lipschitz constant and the stability of GNNs, we present the definition of the Lipschitz constant about GNNs under a certain graph augmentation:

Definition 5 (The Lipschitz Constant of GNNs). *Under a certain graph augmentation $\Phi(\cdot,\cdot,\cdot)$, the Lipschitz constant of a GNN $f(\cdot,\cdot;\boldsymbol{\theta})$ is defined as follows:*

$$K_G = \sup_{G \in \mathcal{G}} \frac{D_R(G, \hat{G})}{D_T(G, \hat{G})}, \tag{9}$$

where $D_R(G, \hat{G})$ is the representation distance calculated by Eq.(6), $D_T(G, \hat{G})$ is the topology distance calculated by Eq.(5), and $\mathcal{G}$ is the anchor graph set. If K_G is finite, the GNN $f(\cdot,\cdot;\boldsymbol{\theta})$ is Lipschitz continuous.

From Eq.(9), we know that if a specific graph augmentation brings about a larger topology distance and a smaller representation distance, we will get a smaller Lipschitz constant. The smaller representation distance can ensure that the contrastive samples inherit more discriminative semantic information from anchor graphs, and the larger topology distance can improve the diversity of samples. Therefore, a graph augmentation with a smaller Lipschitz constant is desired. Then, we give a more rigorous theorem of the above analysis. We present the following theorem to demonstrate that the semantic differences can be bounded by Lipschitz constants during graph augmentation:

Theorem 1. *Let $\mathcal{G}$ be an anchor graph set containing N graphs, $\hat{\mathcal{G}}$ be the set of contrastive samples, and $\mathcal{Y}_{\mathcal{G}}$ be the true label set of $\mathcal{G}$. $CE(\cdot,\cdot)$ is the cross-entropy function, and $K_\rho \in (0, 1)$ is the Lipschitz constant of the function $\rho(x) = log(e^x + 1)$. Under a certain graph augmentation $\Phi(\cdot,\cdot,\cdot)$, $\forall G \in \mathcal{G}$, the following inequality holds, where $\varepsilon\|\mathcal{A}\|_\infty = \max_{G\in\mathcal{G}} D_T(G, \hat{G})$ is the max topology distance, and $\mathcal{W}$ is the weight parameter matrix.*

$$|CE(\mathcal{Y}_{\mathcal{G}}, \mathcal{G}) - CE(\mathcal{Y}_{\mathcal{G}}, \hat{\mathcal{G}})| \leq K_G N(1 + K_\rho)\varepsilon\|\mathcal{A}\|_\infty \cdot \|\mathcal{W}\|, \tag{10}$$

In Theroem 1, the maximum topological distance $\varepsilon\|\mathcal{A}\|_\infty$ is finite under a specific graph augmentation. The norm of the weight parameter matrix can also be bounded by adding a regular term in the model. N and K_ρ are constants. If a smaller Lipschitz constant K_G is obtained under a graph augmentation operation, $|CE(\mathcal{Y}_{\mathcal{G}}, \mathcal{G}) - CE(\mathcal{Y}_{\mathcal{G}}, \hat{\mathcal{G}})|$ is restricted to a smaller range, indicating that the label distribution difference between anchor graphs and contrastive samples is smaller. Smaller differences in label distribution mean that generated samples are consistent with the anchor graphs regarding salient attributes, and more discriminative semantics are retained during

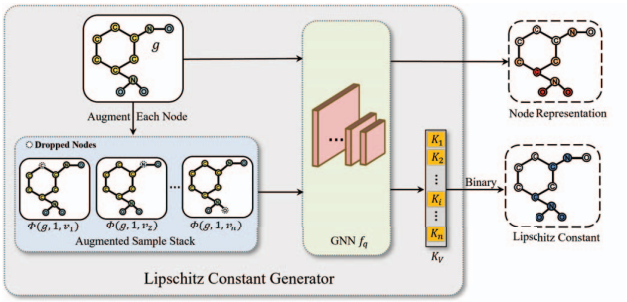


Figure 3. The architecture of Lipschitz constant generator.

graph augmentation. Therefore, the graph augmentation with a smaller Lipschitz constant can enable contrastive samples to preserve more discriminative semantic information from anchor graphs. The full proof of Theorem 1 is presented in section V.

B. Lipschitz Constant Generator

According to the Theorem 1, applying a graph augmentation with a smaller Lipschitz constant indicates that the contrastive samples can preserve more discriminative semantics from anchor graphs. Therefore, dropping a node with a smaller Lipschitz constant results in smaller semantic corruption. We refer to these nodes as the semantic-unrelated nodes in the anchor graph and others as the semantic-related nodes. In this subsection, we design the Lipschitz constant generator to compute the Lipschitz constant of each node. The architecture of the Lipschitz constant generator is shown in Figure 3. For a given node v_r, the augmented sample generated from $G = (V, \boldsymbol{H}, \boldsymbol{A})$ by dropping it can be represented as $\hat{G}_r = \Phi(G, 1, v_r) = (\hat{V}_r, \hat{\boldsymbol{H}}_r, \hat{\boldsymbol{A}}_r)$ according to Definition 3. And the Lipschitz constant of the node v_r can be computed as in Definition 5:

$$K_r = \frac{D_R(G, \hat{G}_r)}{D_T(G, \hat{G}_r)}, \tag{11}$$

where K_r is the Lipschitz constant of the node v_r, $D_R(G, \hat{G}_r)$ is the representation distance between anchor graph G and the generated sample $\hat{G}_r$ computed by Eq.(6), and $D_T(G, \hat{G}_r)$ is the topology distance computed by Eq.(5).

In Lipschitz constant generator, we utilize a GNN $f_q(\cdot, \cdot; \boldsymbol{\theta}_q)$ to obtain the representation distance $D_R(G, \hat{G}_r)$, which can be described as follows:

$$D_R(G, \hat{G}_r) = \|\boldsymbol{H}^{(l)} - \hat{\boldsymbol{H}}_r^{(l)}\|, \tag{12}$$

where $\boldsymbol{H}^{(l)} = [\boldsymbol{h}_1^{(l)}, \boldsymbol{h}_2^{(l)}, \cdots, \boldsymbol{h}_{|V|}^{(l)}]$ is the node representation of the anchor graph G, $\hat{\boldsymbol{H}}_r^{(l)} = [\hat{\boldsymbol{h}}_{r1}^{(l)}, \hat{\boldsymbol{h}}_{r2}^{(l)}, \cdots, \hat{\boldsymbol{h}}_{r|V|}^{(l)}]$ is the node representation of the augmented sample $\hat{G}_r$, and $\boldsymbol{\theta}_q$ is the parameter of the GNN $f_q(\cdot, \cdot; \boldsymbol{\theta}_q)$.

To obtain the Lipschitz constants of all nodes in the anchor graph G, we drop each node in turn to generate augmented samples. We introduce the mask mechanism to realize the node dropping instead of dropping each node by graph augmentation. Specifically, we denote $\boldsymbol{M} = [\boldsymbol{m}_1, \boldsymbol{m}_2, \cdots, \boldsymbol{m}_{|V|}] \in \mathbb{R}^{|V| \times |V|}$ as the perturbation mask matrix of the anchor graph G. Taking the vector $\boldsymbol{m}_r = [m_{r1}, m_{r2}, \cdots, m_{r|V|}]$ as an example, it can be computed as follows:

$$m_{ri} = \begin{cases} 0, i = r \\ 1, i \neq r \end{cases}, \tag{13}$$

where the i-th element m_{ri} equaling to 0 indicates the node v_i is dropped in the generation of augmented sample G_r.

Thus, in obtaining the node representation of augmented sample G_r, the node information aggregation in a single GNN layer can be described as follows:

$$\hat{\boldsymbol{h}}_{ri}^{(l)} = \sigma\left(m_{ri} \odot (\hat{\boldsymbol{h}}_{ri}^{(l-1)} + \sum_{v_j \in N_e(v_i)} m_{rj} \odot \boldsymbol{W}_q^{(l)} \hat{\boldsymbol{h}}_{rj}^{(l-1)})\right), \tag{14}$$

where $\hat{\boldsymbol{h}}_{ri}^{(l)}$ is the representation of node v_i at layer l in the augmented sample G_r, m_{ri} is the perturbation mask constant of node v_i during generating G_r, $N_e(v_i)$ is the neighbor node set of node v_i in anchor graph G, $\boldsymbol{W}_q^{(l)}$ is the weight parameter at layer l in $f_q(\cdot, \cdot; \boldsymbol{\theta}_q)$, and $\sigma(\cdot)$ is the activation function. As a result, the mask mechanism stops the dropped node from passing and aggravating messages, excluding its contribution to representation learning. However, calculating node representation with masks in each GNN layer leads to high computational costs. To optimize the time complexity of the Lipschitz Constant Generator, we use attention weight [47] to compute the dropped node's contribution to other nodes and delete that, achieving the mask mechanism in a reverse way.

With the Lipschitz constant of each node in the graph G, we construct the matrix that reflects the semantic relevance and can be applied to the graph augmentation. The Lipschitz constant matrix K_V is constructed by arranging the Lipschitz constant of each node, which can be described as follows:

$$K_V = [K_1, K_2, \cdots, K_{|V|}], \tag{15}$$

where K_i is a constant, and $K_V \in \mathbb{R}^{|V| \times 1}$.

C. Lipschitz Graph Augmentation

In this subsection, we propose the Lipschitz graph augmentation, which leads to less semantic corruption and generates high-quality contrastive samples. We propose a simple yet effective method to augment these semantic-unrelated nodes indicated by a smaller Lipschitz constant during graph augmentation. Specifically, we first calculate the mean value of the Lipschitz constant matrix K_V in the anchor graph $G = (V, \boldsymbol{H}, \boldsymbol{A})$ as a semantic threshold, which can be described as follows:

$$\bar{K} = \frac{1}{|V|} \sum_{i \in |V|} K_i, \tag{16}$$

where K_i is the Lipschitz constant of node v_i, and $|V|$ is the number of nodes in G. Then, we binary the Lipschitz constant of each node as follows:

$$C_i = \begin{cases} 0, K_i < \bar{K} \\ 1, K_i \geq \bar{K} \end{cases}, \tag{17}$$

where C_i is the binary Lipschitz constant of node v_i, $\bar{K}$ is the semantic threshold, $C = [C_1, C_2, \cdots, C_{|V|}]$ denotes the binary Lipschitz constants matrix of nodes.

We combine the node representation $\boldsymbol{h}_i^{(l)}$ and the binary Lipschitz constants C_i to compute the augmentation probability of node v_i, which can be described as follow:

$$P(v_i|(\boldsymbol{h}_i^{(l)}, C_i)) = C_i + (1 - C_i) * \delta(\boldsymbol{h}_i^{(l)} \boldsymbol{w}_i^T), \tag{18}$$

where $P(v_i) = P(v_i|(\boldsymbol{h}_i^{(l)}, C_i))$ is the augmentation probability of node v_i, $\boldsymbol{h}_i^{(l)} \in \mathbb{R}^{1\times d^{(l)}}$ is the representation of node v_i computed by GNN $f_q(\cdot, \cdot; \boldsymbol{\theta}_q)$, $\boldsymbol{w}_i \in \mathbb{R}^{1\times d^{(l)}}$ is the weight parameter, and $\delta(\cdot)$ is the logistic function.

Based on the augmentation probability, we propose the Lipschitz graph augmentation to generate semantic-aware contrastive samples. Following the Definition 3, the graph augmentation operation can be formulated as follows:

$$\hat{G} = \Phi(G, \rho|V|, P(V)), \tag{19}$$

where $P(V) = \{P(v_1), P(v_2), \cdots, P(v_{|V|})\}$ is the node augmentation probability set computed by Eq.(18), $|V|$ is the number of nodes in the anchor graph G, and ρ is the predefined dropping ratio. With the Lipschitz graph augmentation, semantic-aware nodes indicated by large Lipschitz constant are retained, and contrastive samples can preserve more discriminative semantics from the anchor graphs.

Moreover, we intentionally keep semantic-unrelated nodes to generate semantic-unaware contrastive samples,

$$\hat{G}^c = \Phi(G, \rho|V|, 1 - P(V)), \tag{20}$$

these samples with non-semantic structures complement the contrastive learning as negative samples, improving the uniformity of feature space and making the captured discriminative semantics more stable [48].

D. Semantic-aware Contrastive Learning

In this subsection, we utilize a separate GNN encoder $f_k(\cdot, \cdot; \boldsymbol{\theta}_k)$ with the pooling layer $Pooling(\cdot)$ to obtain the graph-level representation of all graphs for semantic-aware contrastive learning. The GNN $f_q(\cdot, \cdot; \boldsymbol{\theta}_q)$ and $f_k(\cdot, \cdot; \boldsymbol{\theta}_k)$ do not share their parameters but have the same architecture. Following the previous works [20], we also use a 2-layer MLP projection head $Proj(\cdot)$ to project the representations to a latent space. For anchor graphs, we argue that nodes with higher Lipschitz constants contribute more to the graph semantics and treat the Lipschitz constant as the semantic attribute scores. The representation of the anchor graph $G = (V, \boldsymbol{H}, \boldsymbol{A})$ can be obtained as follow:

$$\boldsymbol{z}_G = Proj(Pooling(f_k(\boldsymbol{H}, \boldsymbol{A}; \boldsymbol{\theta}_k) \odot K_V)), \tag{21}$$

where $\boldsymbol{z}_G$ is the representation of graph G, $Proj(\cdot)$ is a two-layer MLP projection head, $\odot$ denotes a dot product, and $K_V \in \mathbb{R}^{|V|\times 1}$ is the Lipschitz constant matrix.

However, the graph structures of generated samples are changed to varying degrees. Therefore, we do not use the semantic attribute scores to improve the representation learning of contrastive samples. The representation of contrastive sample $\hat{G} = (V, \hat{\boldsymbol{H}}, \hat{\boldsymbol{A}})$ can be obtained as follow:

$$\boldsymbol{z}_{\hat{G}} = Proj(Pooling(f_k(\hat{\boldsymbol{H}}, \hat{\boldsymbol{A}}; \boldsymbol{\theta}_k))), \tag{22}$$

where $\hat{G}$ is the contrastive sample generated by Eq.(19).

Similarly, the representation of $\hat{G}^c = (V, \hat{\boldsymbol{H}}^c, \hat{\boldsymbol{A}}^c)$ can be obtained as follow:

$$\boldsymbol{z}_{\hat{G}^c} = Proj(Pooling(f_k(\hat{\boldsymbol{H}}^c, \hat{\boldsymbol{A}}^c; \boldsymbol{\theta}_k))), \tag{23}$$

where $\hat{G}^c$ is the contrastive sample generated by Eq.(20).

Following the previous GCL frameworks [23], our proposed model maximizes the representation agreement between positive pairs and minimizes the agreement between negative pairs. Specifically, we consider the contrastive sample $\hat{G}_i$ and its corresponding anchor graph G_i as a positive pair $(G_i, \hat{G}_i)$. For a given anchor graph G_i, we treat the contrastive samples in the set $\hat{\mathcal{G}}$ generated from other graphs as negative samples. We minimize the following semantic-aware loss function followed InfoNCE [49] to update the model network,

$$\mathcal{L}_s(G_i) = -\log \frac{\exp\left(\boldsymbol{z}_{G_i}^T \boldsymbol{z}_{\hat{G}_i}/\tau\right)}{\sum_{\hat{G}_j \in \hat{\mathcal{G}}, j \neq i} \exp\left(\boldsymbol{z}_{G_i}^T \boldsymbol{z}_{\hat{G}_j}/\tau\right)}, \tag{24}$$

where $\tau \in (0, 1]$ is the temperature hyperparameter. The function $\mathcal{L}_s(G_i)$ realizes contrastive learning by promoting agreement between positive pairs $(G_i, \hat{G}_i)$ while enforcing divergence between negative pairs $(G_i, \hat{G}_j)$.

In addition to the contrastive sample set $\hat{\mathcal{G}}$, we intentionally retained some non-semantic nodes to generate $\hat{\mathcal{G}}^c$, used as an additional negative set to make captured discriminative semantics more stable to unneeded noise factors. We consider the anchor graph G_i and the samples in $\hat{\mathcal{G}}^c$ as negative pairs. The complement loss function $\mathcal{L}_c(\cdot)$ is then defined as follows:

$$\mathcal{L}_c(G_i) = -\log \frac{\exp\left(\boldsymbol{z}_{G_i}^T \boldsymbol{z}_{\hat{G}_i}/\tau\right)}{\exp\left(\boldsymbol{z}_{G_i}^T \boldsymbol{z}_{\hat{G}_i}/\tau\right) + \sum_{\hat{G}^c} \exp\left(\boldsymbol{z}_{G_i}^T \boldsymbol{z}_{\hat{G}^c}/\tau\right)}, \tag{25}$$

where $\hat{G}_i$ is the contrastive sample generated from G by Eq.(19). Moreover, according to Theorem 1, the norm of the weight parameter matrix also influences the Lipschitz graph augmentation. Thus, we further restrict the label distribution difference by introducing the regular loss of the weight parameters, which can be described as follows:

$$\Theta_{\mathcal{W}} = \|\mathcal{W}\|, \tag{26}$$

where $\mathcal{W}$ is the weight parameter matrix in Theorem 1. Therefore, our final objective loss function conflates two losses and the regular loss of the weight parameter matrix, $\mathcal{L}_s(\cdot)$, $\mathcal{L}_c(\cdot)$ and $\Theta_{\mathcal{W}}$, which can be described as follow:

$$\mathcal{L} = \mathbb{E}_{G_i \in \mathcal{G}}[\mathcal{L}_s(G_i) + \lambda_c \mathcal{L}_c(G_i)] + \lambda_{\mathcal{W}} \Theta_{\mathcal{W}}, \tag{27}$$

where λ_c and $\lambda_{\mathcal{W}}$ are the predefined hyperparameters to control the tradeoff among these loss functions.

V. Theoretical Analysis

In this section, we first present a rigorous proof of Theorem 1 and four lemmas used for the proof, then give the time complexity analysis of the pre-training process.

Lemma 1. *A function with bounded first-order derivatives must have a Lipschitz constant over its domain of definition.*

Lemma 2. *Let $\rho(x) = \log(e^x + 1)$. This function has a Lipschitz constant $K_\rho \in (0, 1)$ over its domain of definition $\mathcal{X}$, and $\forall x_1, x_2 \in \mathcal{X}$ the following inequality holds,*

$$|\rho(x_1) - \rho(x_2)| \leq K_\rho |(x_1 - x_2)|. \tag{28}$$

Proof. The first-order derivative function of $\rho(x)$ is $\rho^{'}(x) = \frac{e^x}{e^x+1}$, whose value domain is $[0, 1]$. According to Lemma 1, $\rho(x)$ must has a Lipschitz constant $K_\rho \in (0, 1)$. □

Lemma 3. *In $G = (V, \boldsymbol{H}, \boldsymbol{A})$, the following equation holds, where $\boldsymbol{h}_i^{(l)}$, d_i are the representation and degree of node v_i, $\sum_{i,j} \frac{\boldsymbol{h}_i^{(l)}}{d_i} + \frac{\boldsymbol{h}_j^{(l)}}{d_j} = \sum_i \boldsymbol{h}_i^{(l)}$*

Proof.

$$\begin{aligned}\sum_{i,j} \frac{\boldsymbol{h}_i^{(l)}}{d_i} + \frac{\boldsymbol{h}_j^{(l)}}{d_j} &= \underbrace{\frac{\boldsymbol{h}_1^{(l)}}{d_1} + \cdots + \frac{\boldsymbol{h}_1^{(l)}}{d_1}}_{d_1} + \cdots + \underbrace{\frac{\boldsymbol{h}_n^{(l)}}{d_n} + \cdots + \frac{\boldsymbol{h}_n^{(l)}}{d_n}}_{d_n} \\ &= \sum_i d_i \frac{\boldsymbol{h}_i^{(l)}}{d_i} \\ &= \sum_i \boldsymbol{h}_i^{(l)}. \end{aligned} \tag{29}$$

□

Lemma 4. *Let $\varepsilon\|\mathcal{A}\|_\infty = \max_{G \in \mathcal{G}} D_T(G, \hat{G})$. Based on the Definition 5, the following inequality holds,*

$$\max_{G \in \mathcal{G}} D_R(G, \hat{G}) \leq K_G \cdot \varepsilon \|\mathcal{A}\|_\infty. \tag{30}$$

Proof.

$$\begin{aligned}\max_{G \in \mathcal{G}} D_R(G, \hat{G}) &= \max_{G \in \mathcal{G}} [\frac{D_R(G, \hat{G})}{D_T(G, \hat{G})} D_T(G, \hat{G})] \\ &\leq \max_{G \in \mathcal{G}} \frac{D_R(G, \hat{G})}{D_T(G, \hat{G})} \max_{G \in \mathcal{G}} D_T(G, \hat{G}) \\ &= K_G \cdot \varepsilon \|\mathcal{A}\|_\infty. \end{aligned} \tag{31}$$

□

Theorem 1. *Let $\mathcal{G}$ be an anchor graph set containing N graphs, $\hat{\mathcal{G}}$ be the set of contrastive samples, and $\mathcal{Y}_\mathcal{G}$ be the true label set of $\mathcal{G}$. $CE(\cdot, \cdot)$ is the cross-entropy function, and $K_\rho \in (0, 1)$ is the Lipschitz constant of the function $\rho(x) = log(e^x + 1)$. Under a certain graph augmentation $\Phi(\cdot, \cdot, \cdot)$, $\forall G \in \mathcal{G}$, the following inequality holds, where $\varepsilon\|\mathcal{A}\|_\infty = \max_{G \in \mathcal{G}} D_T(G, \hat{G})$ is the max topology distance, and $\mathcal{W}$ is the weight parameter matrix.*

$$|CE(\mathcal{Y}_\mathcal{G}, \mathcal{G}) - CE(\mathcal{Y}_\mathcal{G}, \hat{\mathcal{G}})| \leq K_G N (1 + K_\rho) \varepsilon \|\mathcal{A}\|_\infty \cdot \|\mathcal{W}\|, \tag{32}$$

Proof. For convenience, we denote $P(G|\boldsymbol{H}^{(l)})$ as $P(G)$, denote $P(e_{ij}|(\boldsymbol{h}_i^{(l)}, \boldsymbol{h}_j^{(l)}))$ as $P(e_{ij})$, and denote $|CE(\mathcal{Y}_\mathcal{G}, \mathcal{G}) - CE(\mathcal{Y}_\mathcal{G}, \hat{\mathcal{G}})|$ as $|\Delta CE|$,

$$\begin{aligned}|\Delta CE| &= |\sum_{G \in \mathcal{G}} P_\mathcal{Y}(G) \log P(G) - \sum_{G \in \mathcal{G}} P_\mathcal{Y}(G) \log P(\hat{G})| \\ &\leq |\sum_{G \in \mathcal{G}} \log P(G) - \log P(\hat{G})|, \end{aligned}$$

substituting Eq.(2) and Eq.(3) into the above gives,

$$\begin{aligned}|\Delta CE| = |\sum_{G \in \mathcal{G}} [\sum_{i,j} \log \delta((\frac{\boldsymbol{h}_i^{(l)}}{d_i} + \frac{\boldsymbol{h}_j^{(l)}}{d_j}) \boldsymbol{w}_{ij}^T) \\ - \sum_{i,j} \log \delta((\frac{\hat{\boldsymbol{h}}_i^{(l)}}{d_i} + \frac{\hat{\boldsymbol{h}}_j^{(l)}}{d_j}) \boldsymbol{w}_{ij}^T)]|, \end{aligned}$$

let $q = (\frac{\boldsymbol{h}_i^{(l)}}{d_i} + \frac{\boldsymbol{h}_j^{(l)}}{d_j}) \boldsymbol{w}_{ij}^T$, $\hat{q} = (\frac{\hat{\boldsymbol{h}}_i^{(l)}}{d_i} + \frac{\hat{\boldsymbol{h}}_j^{(l)}}{d_j}) \boldsymbol{w}_{ij}^T$, and the logistic function can be transformed into $\delta(x) = e^x \frac{1}{1+e^x}$,

$$\begin{aligned}|\Delta CE| \leq \|\mathcal{W}\| \cdot \sum_{G \in \mathcal{G}} |\sum_{i,j} [(\frac{\boldsymbol{h}_i^{(l)}}{d_i} + \frac{\boldsymbol{h}_j^{(l)}}{d_j}) - (\frac{\hat{\boldsymbol{h}}_i^{(l)}}{d_i} + \frac{\hat{\boldsymbol{h}}_j^{(l)}}{d_j})]| \\ + \sum_{G \in \mathcal{G}} \sum_{q, \hat{q}} |\log(e^{\hat{q}} + 1) - \log(e^q + 1)|. \end{aligned}$$

Using Lemma 2, Lemma 3 and Lemma 4, we have the following,

$$\begin{aligned}|\Delta CE| &= \|\mathcal{W}\| \cdot \sum_{G \in \mathcal{G}} |\sum_{i,j} [(\frac{\boldsymbol{h}_i^{(l)}}{d_i} + \frac{\boldsymbol{h}_j^{(l)}}{d_j}) - (\frac{\hat{\boldsymbol{h}}_i^{(l)}}{d_i} + \frac{\hat{\boldsymbol{h}}_j^{(l)}}{d_j})]| \\ &\quad + \sum_{G \in \mathcal{G}} K_\rho \sum_{q, \hat{q}} |q - \hat{q}|. \\ &\leq \|\mathcal{W}\| \cdot \sum_{G \in \mathcal{G}} |\sum_{i,j} [(\frac{\boldsymbol{h}_i^{(l)}}{d_i} + \frac{\boldsymbol{h}_j^{(l)}}{d_j}) - (\frac{\hat{\boldsymbol{h}}_i^{(l)}}{d_i} + \frac{\hat{\boldsymbol{h}}_j^{(l)}}{d_j})]| \\ &\quad + \|\mathcal{W}\| \cdot \sum_{G \in \mathcal{G}} K_\rho | \sum_{i,j} [(\frac{\boldsymbol{h}_i^{(l)}}{d_i} + \frac{\boldsymbol{h}_j^{(l)}}{d_j}) - (\frac{\hat{\boldsymbol{h}}_i^{(l)}}{d_i} + \frac{\hat{\boldsymbol{h}}_j^{(l)}}{d_j})] \\ &= \|\mathcal{W}\| \cdot \sum_{G \in \mathcal{G}} (1 + K_\rho) |\sum_i (\boldsymbol{h}_i^{(l)} - \hat{\boldsymbol{h}}_i^{(l)})| \\ &= \|\mathcal{W}\| \cdot \sum_{G \in \mathcal{G}} (1 + K_\rho) D_R(G, \hat{G}) \\ &\leq \|\mathcal{W}\| \cdot N (1 + K_\rho) \max_{G \in \mathcal{G}} D_R(G, \hat{G}). \\ &\leq K_G N (1 + K_\rho) \varepsilon \|\mathcal{A}\|_\infty \cdot \|\mathcal{W}\| \end{aligned} \tag{33}$$

□

Time complexity analysis. we present the time complexity analysis of the pre-training process. The computational

complexity of the proposed method mainly depends on four parts: augmentation probability computation, Lipschitz Graph Augmentation, Graph representation learning, and Contrastive Loss. For computing augmentation probability, its time expense is mainly from the Lipschitz Constant Generator. We adopt attention weight [47] to approximate the mask mechanism, which can decrease the time complexity of the Lipschitz Constant Generator from $O((|V||E|^2 + |V|)l_q B)$ to $O((|E|^2+|V|^2+|V|)l_q B)$, where $|E|$ is edge number, l_q is the layer number of f_q, and B is batch size. The Lipschitz Graph Augmentation costs $O(2B\rho|V|\log|V|)$, which is equivalent to the time cost of randomly dropping $\rho|V|$ nodes. The Graph representation learning process with a three-tower structure costs $O((3|\mathcal{E}|^2+|\mathcal{V}|)l_k B)$. For Contrastive Loss, the time cost is $O(2B^2 d)$, where d is the latent space dimension.

VI. Experiments

This section presents experiments to demonstrate our SGCL's effectiveness for graph classification. We aim to answer the following research questions:

- **RQ1** How does SGCL perform compared to SOTA?
- **RQ2** How do model's different components contribute to the performance?
- **RQ3** How do hyper-parameters impact the performance?
- **RQ4** How do the different encoder architectures and the label rate impact the performance?
- **RQ5** How effectively is the SGCL capturing semantic-aware nodes to construct contrastive samples?

A. Experimental Setups

1) Datasets: We evaluate the effectiveness of the proposed SGCL for unsupervised learning tasks and transfer learning tasks on sixteen real-world datasets.

In the unsupervised experiments, we follow the previous works [20] to use eight well-known TU datasets [50], which include four bioinformatics datasets (i.e., DD, PROTEINS, NCI1, MUTAG) and four social network datasets (i.e., COLLAB, RDT-B, REDDIT-M-5k, IMDB-B). Table I summarizes the detailed statistics of the TU datasets.

In the transfer learning experiments, we first pre-train our model backbone on Zinc-2M [51] dataset, which includes 2 million unlabeled molecule graphs sampled from the ZINC15 [51] database. Then we finetune our model on 8 benchmark multi-task binary classification datasets in the biochemistry domain, which are contained in MoleculeNet [52]. The detailed statistics of the Zinc-2M and MoleculeNet datasets are summarized in Table II.

2) Baseline: In the unsupervised learning experiments, we compare our SGCL with three state-of-the-art traditional graph kernel methods, GL [30], WL [31], and DGK [32], as well as with seven other self-supervised learning methods: InfoGraph [41], GraphCL [20], JOAOv2 [21], AD-GCL [53], SimGRACE [54], RGCL [23], and AutoGCL [24]. Among them, RGCL and AutoGCL each design a learnable graph view generator and directly select different graph augmentations based on the probability distribution of nodes. To provide rigorous and fair comparative analysis, we use the same underlying architecture (i.e., GIN [3]) when comparing self-supervised learning methods. In the transfer learning experiments, we compare our model with AttrMasking [55], ContextPred [55], GraphCL, JOAOv2, AD-GCL, RGCL, and AutoGCL, which are state-of-the-art models in this field.

TABLE I
STATISTICS FOR UNSUPERVISED LEARNING DATASETS.

Category	Datasets	#Graphs	#Avg.Nodes	#Avg.Edges
Molecules	DD	1,178	284.32	715.66
	PROTEINS	1,113	39.06	72.82
	NCI1	4,110	29.87	32.30
	MUTAG	188	17.93	19.79
Social Networks	COLLAB	5,000	74.49	2457.78
	RDT-B	2,000	429.63	497.75
	RDT-M	4,999	508.52	594.87
	IMDB-B	1,000	19.77	96.53

TABLE II
STATISTICS FOR TRANSFER LEARNING DATASETS.

Utilization	Datasets	#Graphs	#Avg.Nodes	#Avg.Degree
Pre-training	ZINC-2M	2,000,000	26.62	57.72
Finetuning	BBBP	2,039	24.06	51.90
	TOX21	7,831	18.57	38.58
	TOXCAST	8,576	18.78	38.52
	SIDER	1,427	33.64	70.71
	CLINTOX	1,477	26.15	55.76
	MUV	93,087	24.23	52.55
	HIV	41,127	25.51	54.93
	BACE	1,513	34.08	73.71

3) Parameter Settings: In our SGCL, we train the model with a learning rate of 0.001 in each iteration. We tuned all hyperparameters of the validation set by manually searching. The sampling ratio ρ in the Eq.(19) is searched in $\{0.5, 0.6, 0.7, 0.8, 0.9\}$, and the best $\rho = 0.9$ is chosen as the final sampling ratio. In the Eq.(27), the hyperparameter λ_c is searched in $\{0.0001, 0.001, 0.005, 0.01, 0.05, 0.1\}$, and $\lambda_{\mathcal{W}}$ is searched in $\{0.001, 0.01, 0.05, 0.1, 0.2, 0.5\}$. We choose $\lambda_c = 0.01$ and $\lambda_{\mathcal{W}} = 0.01$ as the final loss weights. The temperature parameter τ is searched in $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. $\tau = 0.2$ is the final temperature parameter, where SGCL achieves the best average performance.

In the Lipschitz constant generator, we utilize the three-layer GIN [3] as the encoder f_q whose encoder architecture is consistent with the GNN f_k to compute the Lipschitz constants of nodes. But f_q and f_k do not share their parameters. The embedding dimension of each hidden layer is 32, and the batch size is 16 in Lipschitz constant generator.

In the unsupervised learning experiments, we also utilize GIN [3] as the GNNs encoder f_k to obtain the graph-level representation, which contains three graph convolutional layers, followed by one sum-pooling layer and a two-layer MLP projection function. When we finetune the model backbone on downstream tasks, the projection head is thrown away. The embedding dimension of each hidden layer is also 32. The batch size is 128, and the number of epochs is 40 for

TABLE III
UNSUPERVISED LEARNING GRAPH CLASSIFICATION ACCURACY (%) ON TU DATASETS

Methods	MUTAG	DD	PROTEINS	NCI1	COLLAB	RDT-B	RDT-M-5K	IMDB-B	A.R. ↓
GL	81.66 ± 2.11	–	–	–	–	77.34 ± 0.18	41.01 ± 0.17	65.87 ± 0.98	10.5
WL	80.72 ± 3.00	–	72.92 ± 0.56	$\underline{80.01 \pm 0.50}$	–	68.82 ± 0.41	46.06 ± 0.21	72.30 ± 3.44	8.3
DGK	87.44 ± 2.72	–	73.30 ± 0.82	$\mathbf{80.31 \pm 0.46}$	–	78.04 ± 0.39	41.27 ± 0.18	66.96 ± 0.56	9.3
InfoGraph	$\underline{89.01 \pm 1.13}$	72.85 ± 1.78	74.44 ± 0.31	76.20 ± 1.06	70.05 ± 1.13	82.50 ± 1.42	53.46 ± 1.03	$\mathbf{73.03 \pm 0.87}$	6.0
GraphCL	86.80 ± 1.34	78.62 ± 0.40	74.39 ± 0.45	77.87 ± 0.41	71.36 ± 1.15	89.53 ± 0.84	55.99 ± 0.28	71.14 ± 0.44	5.6
JOAOv2	87.67 ± 0.79	77.40 ± 1.15	74.07 ± 1.10	78.36 ± 0.53	69.33 ± 0.34	86.42 ± 1.45	56.03 ± 0.27	70.83 ± 0.25	6.4
AD-GCL	88.74 ± 1.85	75.79 ± 0.87	73.28 ± 0.46	73.91 ± 0.77	$\underline{72.02 \pm 0.56}$	90.07 ± 0.85	54.33 ± 0.32	70.21 ± 0.68	6.4
SimGrace	89.01 ± 1.31	77.44 ± 1.11	$\underline{75.33 \pm 0.09}$	79.12 ± 0.44	71.72 ± 0.82	89.51 ± 0.89	55.91 ± 0.34	71.30 ± 0.77	4.1
RGCL	87.66 ± 1.01	$\underline{78.86 \pm 0.48}$	75.03 ± 0.43	78.14 ± 1.08	70.92 ± 0.65	$\underline{90.34 \pm 0.58}$	$\underline{56.38 \pm 0.40}$	71.85 ± 0.84	4.4
AutoGCL	88.21 ± 0.92	77.81 ± 0.49	75.12 ± 0.62	79.16 ± 0.30	71.09 ± 0.86	87.35 ± 0.94	55.51 ± 1.44	72.05 ± 0.25	4.6
SGCL (Ours)	$\mathbf{89.74 \pm 0.99}$	$\mathbf{79.71 \pm 0.34}$	$\mathbf{75.37 \pm 0.38}$	79.28 ± 0.24	$\mathbf{72.25 \pm 0.69}$	$\mathbf{90.77 \pm 0.37}$	$\mathbf{56.51 \pm 0.13}$	$\underline{72.14 \pm 0.23}$	**1.5**

pretraining on all eight TU datasets.

In the transfer learning experiments, we use a five-layer GIN as our representation learning encoder, followed by a two-layer MLP projection function. The hidden embedding dimension of GIN and projection function are both 300. The batch size is 128, and the number of epochs is 80 for pretraining.

B. Performance on Downstream Tasks (RQ1)

In this subsection, we select two experiment scenes: unsupervised learning and transfer learning to demonstrate the effectiveness of our SGCL on graph classification.

Performance on Unsupervised Learning. In unsupervised learning, we use 90% of the total unlabeled data to pretrain our SGCL and reserve 10% as labeled testing data. We use the non-linear SVM classifier and perform 10-fold cross-validation on each TU dataset. To ensure stable experimental results, we repeat each experiment five times with different random seeds and report the average as the final result. The experimental results are summarized in Table III, with the best results in bold and the second-best results underlined. AR represents the average rank, and – indicates results are unavailable in published papers. The baseline results are taken from their published papers, except for AutoGCL [24], which we reproduced on our platform, and AD-GCL [53], whose results obtained from [23]. Based on the results, we can make the following observations:

- The proposed SGCL achieves the best performance on 6 TU datasets and the best average rank, which validates the effectiveness of our methods. For molecule datasets, SGCL outperforms the second-best RGCL [23] by 1.08% on the DD dataset [50]. For social network datasets, SGCL outperforms the second-best self-supervised method, AutoGCL, by 3.03% on the COLLAB dataset. These remarkable performances further indicate that preserving semantic-related nodes during graph augmentation can improve the quality of the contrastive samples. Compared to GCL methods with view generators (i.e., AutoGCL, RGCL), SGCL allows for more accurate identification and retention of semantic-related nodes with Lipschitz constants, leading to more stable and effective results.
- It is noticeable that there are some marginal improvements between SGCL and the existing GCL methods. Compared with SGCL, SimGrace [54] utilizes an unusual framework that uses the perturbation of GNNs instead of graph augmentation, resulting in no semantic corruption but less diversity of contrastive samples. Therefore, SGCL has some marginal improvements in some sparse graph datasets(e.g., PROTEINS(0.04%↑), NCI1(0.16%↑)), which are susceptible to the semantic destruction caused by graph augmentation. By contrast, AD-GCL [53], which only uses edge dropping as its graph augmentation, greatly benefits from the heavy-dense datasets. COLLAB is the most dense dataset of all. Thus, SGCL also has a marginal improvement compared with AD-GCL in COLLAB. However, SGCL still outperforms SimGrace and AD-GCL in average rank.
- In NCI1, all the GCL methods perform worse than traditional graph kernel method DGK [32]. One of the possible reasons is that NCI1 contains a large number of graphs with very low density. The self-supervised GCL methods cannot capture enough information by contrasting these graphs. However, SGCL improves the quality of contrastive samples and achieves the best performance on NCI1 compared to other GCL methods.

Performance on Transfer Learning. In transfer learning, we first pre-train a backbone on the Zinc-2M dataset and finetune the model on 8 binary classification biochemistry datasets. To simulate the real-world scenario, we split the downstream datasets by scaffold split. The finetuning procedure is repeated 10 times with different random seeds to obtain stable results, and we evaluate the mean and standard deviation of ROC-AUC scores on each downstream dataset, which is consistent with these baselines. All transfer learning performances on the downstream tasks are presented in Table IV. We present the best results in bold and the second-best results in underlined. AR indicates the average rank. The results of the baselines are taken from their published papers, except for AutoGCL [24], which we reproduced on our platform, and AD-GCL [53],

TABLE IV
TRANSFER LEARNING ROC-AUC SCORES (%) ON DOWNSTREAM GRAPH CLASSIFICATION TASKS

Methods	BBBP	TOX21	TOXCAST	SIDER	CLINTOX	MUV	HIV	BACE	A.R. ↓
No Pre-Train	65.8 ± 4.5	74.0 ± 0.8	63.4 ± 0.6	57.3 ± 1.6	58.0 ± 4.4	71.8 ± 2.5	75.3 ± 1.9	70.1 ± 5.4	7.6
AttrMasking	64.3 ± 2.8	$\mathbf{76.7 \pm 0.4}$	$\underline{64.2 \pm 0.5}$	61.0 ± 0.7	71.8 ± 4.1	74.7 ± 1.4	77.2 ± 1.1	$\underline{79.3 \pm 1.6}$	4.5
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	$\mathbf{79.6 \pm 1.2}$	4.5
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	$\underline{78.47 \pm 1.22}$	75.38 ± 1.44	6.2
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	5.1
AD-GCL	68.26 ± 1.03	73.56 ± 0.72	63.10 ± 0.66	59.24 ± 0.86	77.63 ± 4.21	74.94 ± 2.54	75.45 ± 1.28	75.02 ± 1.88	5.5
RGCL	$\underline{71.42 \pm 0.66}$	75.20 ± 0.34	63.33 ± 0.17	61.38 ± 0.61	$\mathbf{83.38 \pm 0.91}$	$\underline{76.66 \pm 0.99}$	77.90 ± 0.80	76.03 ± 0.77	3.1
AutoGCL	68.65 ± 0.61	72.92 ± 0.52	61.01 ± 0.36	62.04 ± 0.65	$\underline{82.90 \pm 2.33}$	70.15 ± 1.98	75.1 ± 0.97	74.43 ± 1.92	6.6
SGCL (Ours)	$\mathbf{72.41 \pm 0.88}$	$\underline{76.24 \pm 0.32}$	$\mathbf{64.58 \pm 0.39}$	$\mathbf{63.02 \pm 0.55}$	81.86 ± 1.90	$\mathbf{79.81 \pm 0.89}$	$\mathbf{78.76 \pm 0.39}$	77.66 ± 0.67	**1.8**

TABLE V
ABLATION STUDY ROC-AUC SCORES (%) FOR SGCL ON DOWNSTREAM TRANSFER LEARNING DATASETS ('W/O' MEANS WITHOUT)

Models	BBBP	TOX21	TOXCAST	SIDER	CLINTOX	MUV	HIV	BACE	AVG.
w/o VG	70.85 ± 0.69	75.32 ± 0.27	63.20 ± 0.19	60.65 ± 0.60	76.80 ± 1.13	69.72 ± 1.59	78.86 ± 1.22	74.85 ± 1.57	71.29
w/o LGA	70.47 ± 0.70	74.31 ± 0.40	63.17 ± 0.32	60.92 ± 0.51	77.48 ± 1.29	76.23 ± 1.69	77.51 ± 0.39	75.37 ± 0.81	71.93
w/o SRL	71.91 ± 0.63	75.72 ± 0.46	64.21 ± 0.31	61.92 ± 0.62	78.78 ± 2.04	78.04 ± 1.30	78.51 ± 0.41	75.79 ± 1.59	73.11
w/o $\mathcal{L}_c$	71.42 ± 0.76	75.58 ± 0.44	63.22 ± 0.34	61.74 ± 0.37	81.04 ± 1.44	76.45 ± 1.27	78.78 ± 0.72	76.17 ± 1.01	73.05
w/o LW	71.67 ± 0.61	75.85 ± 0.34	64.04 ± 0.27	61.56 ± 0.49	77.38 ± 2.62	78.58 ± 0.99	78.12 ± 0.76	76.03 ± 0.98	72.90
SGCL (Ours)	72.41 ± 0.88	76.24 ± 0.32	64.58 ± 0.39	63.02 ± 0.55	81.86 ± 1.90	79.81 ± 0.89	78.76 ± 0.39	77.66 ± 1.67	74.29

which we obtained from [23]. We can make the following observations based on the results:

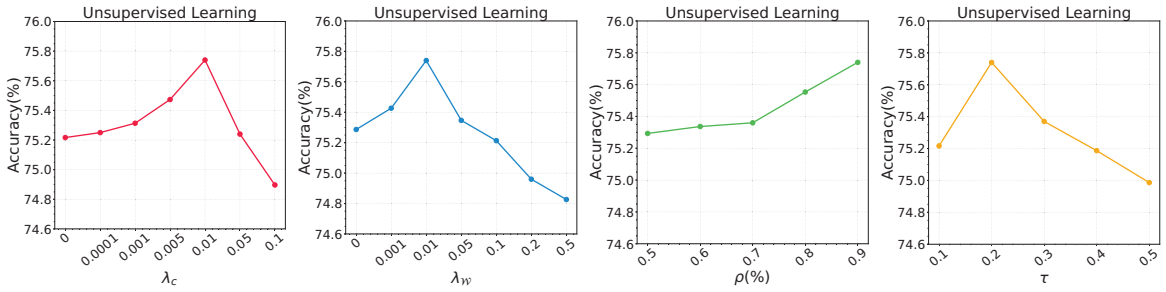
- Although there is no universally beneficial pre-training scheme, especially for the out-of-distribution scenario in transfer learning [20], our SGCL achieves the best performance on 5 out of 8 biochemistry datasets and the highest average rank compared to other methods. Guided by Lipschitz constants, our SGCL generates high-quality contrastive samples, facilitating pre-training to obtain more discriminative information.
- The proposed SGCL performs best among other GCL baselines. The possible reason for this better performance is that our semantic-aware graph augmentation can construct stable contrastive samples with more discriminative semantics. Moreover, we compare the performance of SGCL with AutoGCL to demonstrate the crucial role of preserving semantic-related nodes. Specifically, our SGCL outperforms AutoGCL by 13.77% on the MUV dataset and is 4.8 places ahead in the average ranking.
- The performance deterioration on the CLINTOX dataset can be attributed to the huge difference in Lipschitz constants between CLINTOX and the pre-training dataset ZINC15, which makes our method fail to preserve semantic-related nodes in the CLINTOX dataset. Due to the different data distributions between the pre-training and fine-tuning datasets, the Lipschitz constants generator trained by ZINC15 may not precisely capture the semantic information in the CLINTOX dataset, leading to performance degradation of Lipschitz constants. Therefore, there needs an extensive investigation in the future to explore how to improve the Lipschitz constants for such out-of-distribution scenarios in transfer learning.

C. Ablation Study (RQ2)

In this subsection, we conduct ablation experiments to demonstrate the effectiveness of each component. Specifically, we introduce each model as follows:

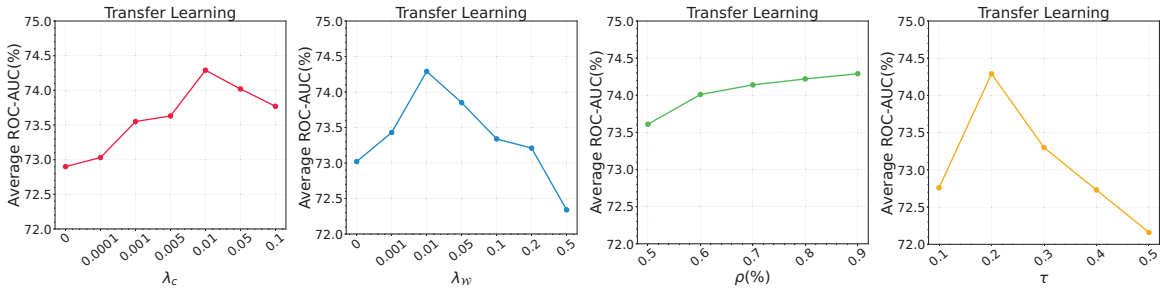
- SGCL w/o VG: We replace the Lipschitz graph augmentation with randomly dropping nodes to generate contrastive samples.
- SGCL w/o LGA: We construct contrastive samples by replacing the Lipschitz graph augmentation with a learnable view generator (i.e., the node dropping depends on its probability distribution).
- SGCL w/o SRL: For all anchor graphs and contrastive samples, we only use the GNN encoder without Lipschitz constant attributes to obtain their representation.
- SGCL w/o $\mathcal{L}_c$: We remove the complement loss in the final loss function (i.e., $\lambda_c = 0$)
- SGCL w/o LW: We remove the regular loss of parameter in the final loss function (i.e., $\lambda_{\mathcal{W}} = 0$)

The overall results of the ablation study are shown in Table V. We summarize the following findings: (1) Our experimental results show that SGCL performs better than SGCL w/o VG by 4.21% and SGCL w/o LGA by 3.28%. This demonstrates the effectiveness of our proposed Lipschitz graph augmentation (LGA) in improving the quality of contrastive samples, leading to better representation learning. (2) SGCL w/o LGA also performs better than SGCL w/o VG, which is in accordance with the findings in previous work [24]. The learnable graph view generator is superior to augmentation in a random fashion. (3) SGCL w/o SRL shows a performance gap(1.18%↓) compared



(a) Unsupervised: λ_c (b) Unsupervised: $\lambda_{\mathcal{W}}$ (c) Unsupervised: $\rho(\%)$ (d) Unsupervised: τ

Figure 4. Sensitivity w.r.t hyperparameter λ_c, $\lambda_{\mathcal{W}}$, ρ and τ on the average of PROTEINS, DD and IMDB-BINARY.



(a) Transfer: λ_c (b) Transfer: $\lambda_{\mathcal{W}}$ (c) Transfer: $\rho(\%)$ (d) Transfer: τ

Figure 5. Sensitivity w.r.t hyperparameter λ_c, $\lambda_{\mathcal{W}}$, ρ and τ in Transfer learning experiments.

to SGCL. This is because the improved representation learning using the node Lipschitz constant better captures information of semantic-related nodes. (4) SGCL w/o $\mathcal{L}_c$ performs worse than the Full SGCL, which indicates the complement loss helps the model capture more stable discriminative semantics. (5) Comparing SGCL w/o LW and SGCL, we observe that the regularization of parameters through Lipschitz weighting improves the performance of SGCL by 1.91%. This finding aligns with the theorem stated in our paper (Theorem 1).

D. Parameter Sensitivity (RQ3)

In this subsection, we evaluate the sensitivity of hyperparameters in our SGCL. Specifically, we investigate the effect of varying the objective loss function hyperparameters λ_c and $\lambda_{\mathcal{W}}$, the perturbation ratio ρ, and the temperature hyperparameter τ in different experimental scenarios. The performance on both transfer learning and unsupervised learning are respectively shown in Figure 4 and Figure 5, and we have the following observations:

- The complement loss hyperparameter λ_c adjusts the weight of the non-semantic structure involved in contrast learning. When λ_c is close to zero, it leads to insufficient non-semantic structure, which reduces model performance. However, the performance also drops when λ_c is set to a large value (i.e., 0.05, 0.1), indicating that too much non-semantic structure can misguide the model.
- The term $\lambda_{\mathcal{W}}$ works as a weight regularizer in the optimization process, so overemphasizing it leads to performance degradation. A proper setting benefits (i.e., $\lambda_{\mathcal{W}} = 0.01$) graph contrastive learning.
- The perturbation ratio ρ controls the scale of the dropped semantic-unrelated nodes. Compared with the other three hyperparameters, ρ has a minimal impact on performance. Among the reasons are our SGCL only drops the semantic-unrelated nodes during graph augmentation. And we still suggest tuning it around a comparatively large value (i.e., $\rho = 0.9$), because the semantic-unrelated nodes also contribute to the model pre-training.
- τ plays an important role in contrastive learning, which can adjust the uniformity of contrastive samples [56]. Both a too-small (i.e., $\tau = 0.1$) value and a too-large (i.e., $\tau = 0.5$) value of τ hurt the performance of SGCL. In our experiment, we finally choose $\tau = 0.2$, which is not optimal in all datasets but is optimal on average.

E. Further Analysis (RQ4)

Furthermore, we investigate the impact of different encoder architectures and label rates. Different encoder architectures are conducted in unsupervised learning, and the effect of different label rates is completed in semi-supervised learning.

Effect of Different Encoder Architectures. Here we aim to examine the impact of different encoder architectures on the performance of SGCL. We focus on four widely-used graph neural networks: GCN [1], GraphSAGE [57], GAT [2], and GIN [3]. We compare them in unsupervised experiments

on four TU datasets: MUTAG, PROTEINS, DD, and IMDB-BINARY, as shown in Figure .6. This histogram illustrates that GIN slightly outperforms the other base models and the robustness of our model to different encoder architectures.

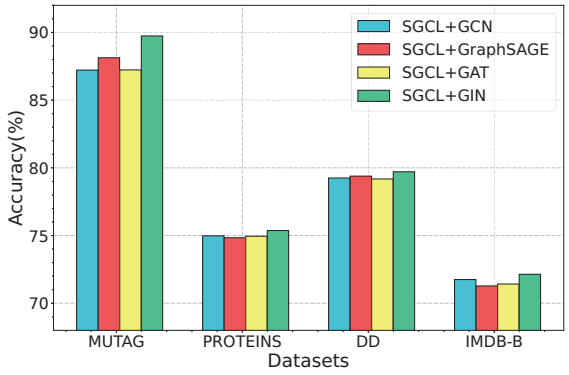


Figure 6. The accuracy (%) of SGCL with different encoder architectures on unsupervised learning graph classification.

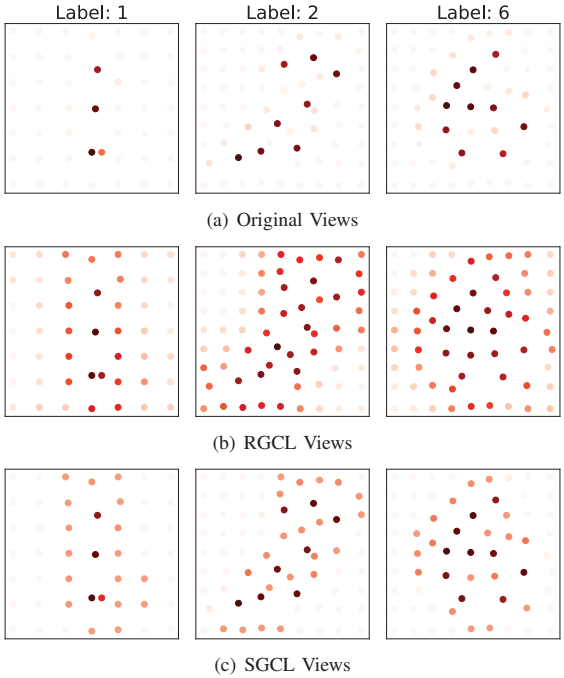
Effect of Different Label Rates. Here we further examine the impact of labeled data numbers on the performance of SGCL on graph classification. The results are shown in Table VI. The baseline results are taken from the published papers, except for the 1% label rate experiments of AutoGCL [24], which we reproduce on our platform. We report two semi-supervised tasks with 1% (i.e., NCI1(1%), COLLAB(1%)) and 10% (i.e., NCI1(10%), COLLAB(10%)) label rate respectively. The results show that our SGCL outperforms all the previous baselines in the 1% label rate setting. For the 10% label rate setting, SGCL performs comparably to the SOTA method AutoGCL [24], which proposes a joint training strategy focused on improving the performance in semi-supervised learning.

TABLE VI
SEMI-SUPERVISED LEARNING ACCURACY (%) ON TU DATASETS.

Method	NCI1(1%)	COLLAB(1%)	NCI1(10%)	COLLAB(10%)
No pre-train	60.72 ± 0.45	57.46 ± 0.25	73.72 ± 0.24	73.71 ± 0.27
GAE	61.63 ± 0.84	63.20 ± 0.67	74.36 ± 0.24	75.09 ± 0.19
Infomax	62.72 ± 0.65	61.70 ± 0.77	74.86 ± 0.26	73.76 ± 0.29
GraphCL	62.55 ± 0.86	64.57 ± 1.15	74.63 ± 0.25	74.23 ± 0.21
JOAOv2	62.52 ± 1.16	64.51 ± 2.21	74.48 ± 0.27	75.30 ± 0.32
SimGRACE	64.21 ± 0.65	64.28 ± 0.98	74.60 ± 0.41	74.74 ± 0.28
AutoGCL	64.38 ± 0.85	65.37 ± 1.04	73.75 ± 2.25	$\mathbf{77.16 \pm 1.48}$
SGCL (Ours)	$\mathbf{64.99 \pm 0.72}$	$\mathbf{65.62 \pm 0.64}$	$\mathbf{75.64 \pm 0.68}$	75.82 ± 0.11

F. Visualization (RQ5)

In this subsection, we visualize the contrastive sample generated from our SGCL and RGCl in MNIST-Superpixel dataset. As shown in Figure 7, the nodes' colors reflect each node's augmentation probability in RGCL views and reflect the Lipschitz constant in our SGCL. The darker color indicates a higher probability of a node being preserved during the graph augmentation. RGCL and our SGCL methods both capture semantic nodes located at the center of the graph and assign them the highest probability. However, compared to the node probability distribution in RGCL, the distribution of the Lipschitz constant is much closer to the original views. This suggests that our SGCL can accurately identify the semantic-related nodes and preserve more discriminative semantics.



(a) Original Views

(b) RGCL Views

(c) SGCL Views

Figure 7. Contrastive sample visualization of 1, 2, 6 on the MNIST-Superpixel dataset. In RGCL, redness reflects the probability of each node. In our SGCL, redness reflects the Lipschitz constant of each node.

VII. CONCLUSION

In this paper, we propose a novel **S**emantic-aware **G**raph **C**ontrastive **L**earning (SGCL) approach to preserve semantic information in contrastive samples, which improves the performance of graph contrastive learning on downstream tasks. SGCL first provides an effective generator to compute the Lipschitz constants of each node. By combining Lipschitz constants, our proposed Lipschitz graph augmentation can identify semantic-related structures while augmenting semantic-unrelated nodes with smaller Lipschitz constants to generate high-quality samples for semantic-aware contrastive learning. Furthermore, we conduct extensive experiments to demonstrate the effectiveness of SGCL. Our experimental results on unsupervised and transfer learning graph classification tasks demonstrate SGCL outperforms state-of-the-art methods.

ACKNOWLEDGEMENT

This work was supported by the National Natural Science Foundation of China (No.61976051, U19A2067), and The Guangdong Major Project of Basic and Applied Basic Research (No.2019B030302002), and The Major Key Project of PCL (Grant No.PCL2022A03), and Shenzhen Science and Technology Program (Grant No.ZDSYS20210623091809029).

REFERENCES

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[2] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[3] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2019.

[4] H. ZHAO, Q. YAO, and W. TU, "Search to aggregate neighborhood for graph neural network," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2021, pp. 552–563.

[5] Y. Wang, Y. Min, X. Chen, and J. Wu, "Multi-view graph contrastive representation learning for drug-drug interaction prediction," in *Proceedings of the Web Conference 2021*, 2021, pp. 2921–2933.

[6] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui, and Q. V. H. Nguyen, "Are graph augmentations necessary? simple graph contrastive learning for recommendation," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022, pp. 1294–1303.

[7] Y. Chen, Y. Yang, Y. Wang, J. Bai, X. Song, and I. King, "Attentive knowledge-aware graph convolutional networks with collaborative guidance for personalized recommendation," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2022, pp. 299–311.

[8] Z. Lin, B. Liang, Y. Long, Y. Dang, M. Yang, M. Zhang, and R. Xu, "Modeling intra-and inter-modal relations: Hierarchical graph contrastive learning for multimodal sentiment analysis," in *Proceedings of the 29th International Conference on Computational Linguistics*, 2022, pp. 7124–7135.

[9] H. Chai, S. Tang, J. Cui, Y. Ding, B. Fang, and Q. Liao, "Improving multi-task stance detection with multi-task interaction network," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds. Association for Computational Linguistics, 2022, pp. 2990–3000.

[10] H. Chai, J. Cui, Y. Wang, M. Zhang, B. Fang, and Q. Liao, "Improving gradient trade-offs between tasks in multi-task text classification," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, A. Rogers, J. L. Boyd-Graber, and N. Okazaki, Eds. Association for Computational Linguistics, 2023, pp. 2565–2579.

[11] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in neural information processing systems*, vol. 28, 2015.

[12] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *Proceedings of the Web Conference 2021*, 2021, pp. 2069–2080.

[13] C. Tan, J. Xia, L. Wu, and S. Z. Li, "Co-learning: Learning from noisy labels with self-supervision," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 1405–1413.

[14] J. Xia, Y. Zhu, Y. Du, and S. Z. Li, "A survey of pretraining on graphs: Taxonomy, methods, and applications," in *arXiv preprint arXiv:2202.07893*, 2022.

[15] J. Zeng and P. Xie, "Contrastive self-supervised learning for graph classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 824–10 832.

[16] J. Xia, L. Wu, G. Wang, J. Chen, and S. Z. Li, "Progcl: Rethinking hard negative mining in graph contrastive learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 24 332–24 346.

[17] H. Chai, Z. Yin, Y. Ding, L. Liu, B. Fang, and Q. Liao, "A model-agnostic approach to mitigate gradient interference for multi-task learning," *IEEE Trans. Cybern.*, vol. 53, no. 12, pp. 7810–7823, 2023.

[18] H. Yang, H. Chen, S. Pan, L. Li, P. S. Yu, and G. Xu, "Dual space graph contrastive learning," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1238–1247.

[19] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4116–4126.

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

[21] Y. You, T. Chen, Y. Shen, and Z. Wang, "Graph contrastive learning automated," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 121–12 132.

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

[23] S. Li, X. Wang, A. Zhang, Y. Wu, X. He, and T.-S. Chua, "Let invariant rationale discovery inspire graph contrastive learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 13 052–13 065.

[24] Y. Yin, Q. Wang, S. Huang, H. Xiong, and X. Zhang, "Autogcl: Automated graph contrastive learning via learnable view generators," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8892–8900.

[25] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," in *Journal of molecular biology vol. 330,4 (2003)*, 2003, pp. 771–83.

[26] P. Mohapatra, S. Chakravarty, and P. Dash, "Microarray medical data classification using kernel ridge regression and modified cat swarm optimization based gene selection system," in *Swarm and Evolutionary Computation*, vol. 28. Elsevier, 2016, pp. 144–160.

[27] M. Le Mercier, D. Hastir, X. Moles Lopez, N. De Neve, C. Maris, A.-L. Trepant, S. Rorive, C. Decaestecker, and I. Salmon, "A simplified approach for the molecular classification of glioblastomas." Public Library of Science San Francisco, USA, 2012.

[28] A. Olar and K. D. Aldape, "Using the molecular classification of glioblastoma to inform personalized treatment," in *The Journal of pathology*, vol. 232, no. 2. Wiley Online Library, 2014, pp. 165–177.

[29] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The world wide web conference*, 2019, pp. 417–426.

[30] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Artificial intelligence and statistics*. PMLR, 2009, pp. 488–495.

[31] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." in *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[32] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1365–1374.

[33] G. Zhong and C.-M. Pun, "Latent low-rank graph learning for multi-modal clustering," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 492–503.

[34] D. Shimin, Y. Quanming, Y. Zhang, and C. Lei, "Efficient relation-aware scoring function search for knowledge graph embedding," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 1104–1115.

[35] Z. Wang, T. Xia, R. Jiang, X. Liu, K.-S. Kim, X. Song, and R. Shibasaki, "Forecasting ambulance demand with profiled human mobility via heterogeneous multi-graph neural networks," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 1751–1762.

[36] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *International conference on machine learning*. PMLR, 2019, pp. 3734–3743.

[37] Q. Sun, J. Li, H. Peng, J. Wu, Y. Ning, P. S. Yu, and L. He, "Sugar: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism," in *Proceedings of the Web Conference 2021*, 2021, pp. 2081–2091.

[38] X. Luo, W. Ju, M. Qu, C. Chen, M. Deng, X.-S. Hua, and M. Zhang, "Dualgraph: Improving semi-supervised graph classification via dual contrastive learning," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2022, pp. 699–712.

[39] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[40] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[41] S. F.-Y., J. Hoffmann, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," in *ICLR*, 2020.

[42] P. D. Hoff, A. E. Raftery, and M. S. Handcock, "Latent space approaches to social network analysis," in *Journal of the american Statistical association*, vol. 97, no. 460. Taylor & Francis, 2002, pp. 1090–1098.
[43] J. Ma, W. Tang, J. Zhu, and Q. Mei, "A flexible generative framework for graph-based semi-supervised learning," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.
[44] S. Wan, S. Pan, J. Yang, and C. Gong, "Contrastive and generative graph convolutional networks for graph-based semi-supervised learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 11, 2021, pp. 10 049–10 057.
[45] F. Gama, J. Bruna, and A. Ribeiro, "Stability properties of graph neural networks," in *IEEE Transactions on Signal Processing*, vol. 68. IEEE, 2020, pp. 5680–5695.
[46] P. Pauli, A. Koch, J. Berberich, P. Kohler, and F. Allgöwer, "Training robust neural networks using lipschitz bounds," in *IEEE Control Systems Letters*, vol. 6, 2022, pp. 121–126.
[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
[48] T. Wang and P. Isola, "Understanding contrastive representation learning through alignment and uniformity on the hypersphere," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9929–9939.
[49] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm, "Mutual information neural estimation," in *International conference on machine learning*. PMLR, 2018, pp. 531–540.
[50] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," in *arXiv preprint arXiv:2007.08663*, 2020.
[51] T. Sterling and J. J. Irwin, "Zinc 15–ligand discovery for everyone," in *Journal of chemical information and modeling*, vol. 55, no. 11. ACS Publications, 2015, pp. 2324–2337.
[52] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, "Moleculenet: a benchmark for molecular machine learning," in *Chemical science*, vol. 9, no. 2. Royal Society of Chemistry, 2018, pp. 513–530.
[53] S. Suresh, P. Li, C. Hao, and J. Neville, "Adversarial graph augmentation to improve graph contrastive learning," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 15 920–15 933.
[54] J. Xia, L. Wu, J. Chen, B. Hu, and S. Z. Li, "Simgrace: A simple framework for graph contrastive learning without data augmentation," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1070–1079.
[55] W. Hu*, B. Liu*, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," in *International Conference on Learning Representations*, 2020.
[56] F. Wang and H. Liu, "Understanding the behaviour of contrastive loss," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2495–2504.
[57] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.