Self-supervised Graph Learning with Segmented Graph Channels

Hang Gao[1,2], Jiangmeng Li[1,2], and Changwen Zheng[2]✉

[1] University of Chinese Academy of Sciences, Zhongguancun East Road. 80, Haidian District, 100081 Beijing, China
https://www.ucas.ac.cn/

[2] Science & Technology on Integrated Infomation System Laboratory, Institute of Software Chinese Academy of Sciences, Zhongguancun South Fourth Street. 4, Haidian District, 100083 Beijing, China 80, Haidian District, 100081 Beijing, China
first@iscas.ac.cn
http://www.iscas.cn/

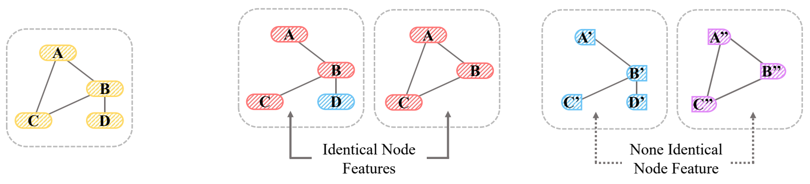
Abstract. Self-supervised graph learning adopts self-defined signals as supervision to learn representations. This learning paradigm solves the critical problem of utilizing unlabeled graph data. Conventional self-supervised graph learning methods rely on graph data augmentation to generate different views of the input data as self-defined signals. However, the views generated by such an approach contain amounts of identical node features, which leads to the learning of redundant information. To this end, we propose *Self-Supervised Graph Learning with Segmented Graph Channels* (SGL-SGC) to address the issue. SGL-SGC divides the input graph data across the feature dimensions as Segmented Graph Channels (SGCs). By combining SGCs with data augmentation, SGL-SGC can generate views that vastly reduce the redundant information. We further design a feature-level weight-sensitive loss to jointly accelerate optimization and avoid the model falling into a local optimum. Empirically, the experiments on multiple benchmark datasets demonstrate that SGL-SGC outperforms the state-of-the-art methods in contrastive graph learning tasks. Ablation studies verify the effectiveness and efficiency of different parts of SGL-SGC.

Keywords: Graph neural network · Self-supervised learning · Unsupervised learning · Contrastive learning · Node classification.

1 Introduction

Graph representation learning (GRL) aims to learn effective representations of graph-structured data. Such representations play an important role in a variety of real-world applications, including knowledge graphs [33], molecules [5], social networks [12], physical processes [19], and codes [1]. Recently, Graph Neural Networks (GNNs) emerged as a powerful approach to conducting graph representation learning. Various GNNs, including Graph Convolutional Networks (GCN) [12], Graph Attention Networks (GAT) [27], and Graph Isomorphism

Networks (GIN) [31], achieve eye-catching success in graph representation learning. These approaches require labeled graph data for training. However, labeling graph data is a rather challenging task as it requires large amounts of onerous work, particularly with large-scale graphs.



(a) Original input graph data (b) Views generated with data augmentation alone (c) Views generated with SGCs and data augmentation

Fig. 1. Examples of views generated using different methods. View generated with graph data augmentation contain identical node features. SGCs help eliminate them

To reduce the dependence on labeled data, recent research efforts are dedicated to developing self-supervised learning for GNNs. In computer vision (CV), self-supervised learning utilizing unlabeled data has already made significant progress [4,10,7]. Viewing its success in CV, some researchers combine self-supervised learning with graph learning and propose a variety of powerful self-supervised graph learning (SGL) methods [8,17,28]. SGL methods rely on views, i.e., human-defined data transformations that preserve the invariance of intrinsic properties of graph data, as training signals to conduct representation learning [34]. Previous works leverage the mutual information maximization principle (InfoMax) [15] and obtain graph representations by maximizing the mutual information between representations of different views. However, the InfoMax principle can be risky. It only encourages the maximization of mutual information while this mutual information may contain redundancy. Based on the information bottleneck principle[25,26], [30] points out that when the task-related information contained in the views is not damaged, the redundant mutual information between views should be minimized. To minimize such redundancy, the choice of views is critical [34,30].

In recent years, researchers proposed various view generation methods in graph self-supervised learning, including node dropping, edge perturbation, attribute masking, and subgraph [35]. These methods can be summarized as graph data augmentation that generates different views by making minor changes to the graph data without damaging the task-related information of the graph. We analyze the graph data augmentation methods and propose that they can be expressed as perturbing the original graph data with a specific form of noise. Views generated with such a mechanism contain a large number of identical node features, which will lead to learning redundant information. Considering

Figure 1, Figure 1(b) demonstrates the different views of an input graph (Figure 1(a)) generated with data augmentation. The data augmentation drop nodes and delete edges but leaves the features of node features A, B, and C unmodified, leaving identical node features (marked with red) between the views. However, we cannot reduce such redundant information with more perturbation, i.e., adding more noise. Otherwise, the task-related information of the original graph may be corrupted or even completely changed. The conventional graph view generation methods show limitations here.

To address such limitations, we look to the view generation methods in CV for inspiration. Self-supervised methods generate different views by splitting input image data across channels, e.g., an RGB image can be split into three views for R, G, and B channels [24]. The advantage of this view generation method is that there is no identical feature between different views. Furthermore, it does not introduce more noises. Since each channel provides a relatively condensed and expressive view, this method allows the neural network to pay more attention to task-related semantic information instead of redundant information. We believe that a similar approach can also be applied to graph learning. Suppose we regard each node in the graph as a pixel on the picture and artificially divide the node features into different channels. In that case, we can generate "channels" on the graph, which we denote as Segmented Graph Channels (SGCs).

With SGCs, we propose the Self-Supervised Graph Learning with Segmented Graph Channel (SGL-SGC) to enhance graph representation learning. We combine SGCs with conventional graph data augmentation methods to generate views for self-supervised learning. Figure 1(c) gives an example. Due to combining two different view generation methods, our method can generate amounts of views without introducing more noise. We design an objective function named feature-level weight-sensitive loss to train the encoders with these views. This loss function helps reduce the computational burden while avoiding the model falling into a local optimum. Furthermore, it can assign different weights to different samples according to their importance, further enhancing the learning capability of SGL-SGC.

We summarize our contributions as follows:

- We propose a novel view generate method based on segmented graph channels to generate views with less redundant information. These views strengthen the ability of our proposed method to perform representation learning.
- We design a feature-level weight-sensitive loss as an objective function for training the encoders with the generated views. Feature-level weight-sensitive loss reduces computational burden while avoiding the model falling into a local optimum. Furthermore, our loss function emphasizes the samples with more importance.
- We conduct experiments to compare our method with state-of-the-art graph self-supervised learning approaches on benchmark datasets, and the results prove the superiority of our method.

2 Related works

This section reviews some representative works on graph learning and self-supervised graph learning, as they are related to this article.

Graph Neural Networks (GNNs) GNNs learn the representation of the graph nodes through aggregating the neighboring information. The learned representations can then be applied to different downstream tasks. Varieties of GNN frameworks have been raised since the concept of GNNs was proposed. Graph Convolutional Networks (GCNs) [12] extend convolution neural networks to graph data. As a widely used GNN, GCN adopts convolution operation to aggregate the features from a node's graph neighborhood. Graph Attention Networks (GATs) [27] introduce attention mechanisms into graph learning. GATs measure the importance of the neighboring features before aggregating them. By comparing the GNNs with the WL test, [31] proposes that GNNs are most powerful as the WL test in distinguishing graphs and proposed Graph Isomorphism Networks (GIN). Our proposed SGL-SGC adopts GCNs as the basic encoder.

Self-supervised learning Self-supervised learning, which aims to learn data representations without labels, is a thriving learning approach with multiple applications. Contrastive Predictive Coding (CPC) [16] proposes a self-supervised framework that contrasts predictive features with original features. CMC [24] conducts self-supervised learning by contrasting different views of an image. Similar frameworks were later applied to graph learning. This self-supervised learning approach successfully improves the utilization of unlabeled data. Given the success of these approaches, [2,14] conduct theoretical analysis on the reason behind them. [15,11,6] elaborates on the objectives of self-supervised learning from the perspective of information theory.

With the proposal of GNNs, neural networks based on self-supervised graph learning have become a research hotspot. [35] propose a framework that adopts graph data augmentation to generate different views and maximize the agreement between different representations of different views. [32] propose an approach that adopts the EM algorithm to enhance the representation learning of local and global structures. Our method focuses on reducing the redundancy of information in the learned representations.

3 Methods

This section introduces our proposed Self-supervised Graph Learning with Segmented Graph Channels (SGL-SGC). The architecture of SGL-SGC is illustrated in Figure 2. SGL-SGC adopts a novel view generator to acquire more independent views than conventional unsupervised graph learning methods. We utilize multiple encoders for representation learning to process these views and a feature-level weight-sensitive loss function for fast and effective training.

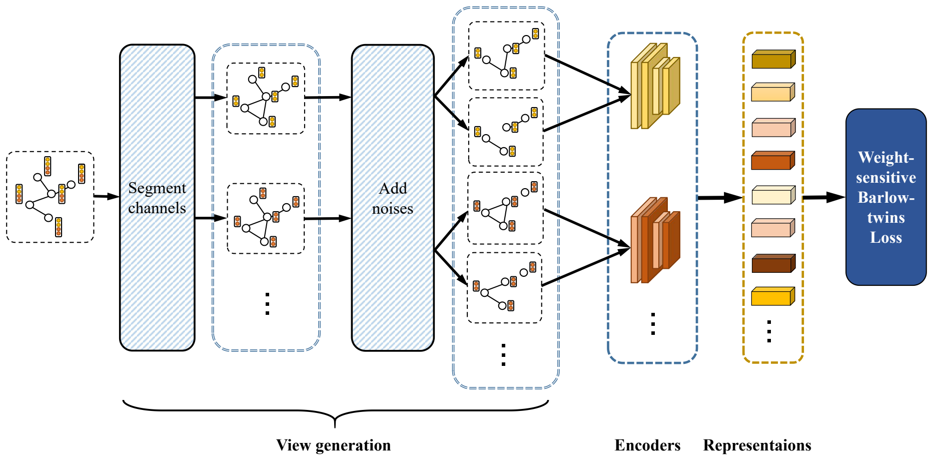


Fig. 2. The structure of SGL-SGC. SGL-SGC can be divided into three parts, including view generation, encoders, and loss function. In conventional self-supervised graph learning, the view generation part usually only consists of data augmentation operations. We, on the other hand, adopt a two-phase method, including segmenting channels and adding noise. SGL-SGC generates multiple expressive views with less redundancy. We also use a feature-level weight-sensitive loss to train the encoders to learn better representations of these views.

3.1 Preliminary

We first recap some preliminary concepts and notations for further exposition. In graph learning, the input attribute graphs can be denoted as $G = (V, E)$, where V is a node set and E is an edge set. V have attributes $\{X_v \in \mathbb{R}^F | v \in V\}$. For each node v, its neighbors are denoted as $\mathcal{N}_v$.

Learning Graph Representations. Given a set of graphs G_i, $i = 1, 2, ..., n$, in some universe $\mathcal{G}$, our objective is to learn the latent representation z_i. z_i preserves the network structures and node attributes of G_i. It can be further used for downstream tasks such as label predicting. Typically, the graph data is fed into graph neural networks (GNNs) to acquire z_i:

$$z_i = GNNs(G_i). \tag{1}$$

Graph Neural Networks. As described earlier, GNNs developed multiple variants, yet their structures still share large similarity. For a graph $G = (V, E)$, a graph neural network layer can be expressed as:

$$h_v^{(k+1)} = combine^{(k)}\Big(h_v^k, aggregate^{(k)}(h_u^k, \forall u \in \mathcal{N}_v)\Big),$$

where $h^{(k+1)}$ is the representation of node v, acquired by passing the initial node features of v through k layers of graph neural networks. *update*$(\cdot)$ and

$aggregate(\cdot)$ are trainable functions. The graph representation z can be obtained by pooling the node representations of the last layer:

$$z = pool(h_v^k, v \in V), \tag{2}$$

Mutual Information Theory. Graph contrastive learning, one of the most popular self-supervised graph learning approaches, defines its learning objective as maximizing the mutual information between the graph and its representation, which is known as the mutual information maximization principle:

$$\max_f I(G; f(G)), \ \textit{where } G \sim \mathbb{P}_{\mathcal{G}}. \tag{3}$$

$I(\cdot)$ denotes the mutual information between variables. In general, graph contrastive learning achieves mutual information maximization by maximizing the mutual information between different views generated with data augmentation [35,9,34]. Such learning objectives can be expressed as follows:

$$\max_{f_1, f_2} I\Big(f_1(V_1); f_2(V_2)\Big), \ \textit{where } V_1, V_2 \textit{ are different views of } G, \ G \sim \mathbb{P}_{\mathcal{G}} \tag{4}$$

$f_1(\cdot)$ and $f_2(\cdot)$ are encoders corresponding to each view. In some methods, the encoders share the same set of parameters. We follow the same learning objective as graph contrastive learning.

3.2 Segmented Graph Channels

We follow [35] and categorize the data augmentation approaches for view generation into four different kinds. **Node dropping** drops a certain amount of nodes along with the edges linked to them. **Edge perturbation** changes the connectivity of the graph by deleting or adding some edges. **Attribute masking** masks are part of the node features. **Subgraph sampling** samples a subgraph from the original graph.

These augmentation methods can be summarized as changing the graph structures or node features. They can be seen as imposing some noise signal S on the original graph data. Depending on the specific content, S could lead to node dropping, edge perturbation, some parts of the features being masked, and making the influenced graph a subgraph of the original one.

Definition 1. *(Graph Data Augmentation with Noise). For a graph G, $q(G, S)$ denote a graph data augmentation of G, where S is a noise signal and $q(\cdot)$ denote the function modifying G according to S. S can be randomly generated or created according to specific rules.*

As we follow the learning objective of graph contrastive learning, with Definition 1 and Equation 4, we define our learning objective as:

$$\max_{f_1, f_2} I\Big(f_1\big(q(G, S_1)\big); f_2\big(q(G, S_2)\big)\Big), \ \textit{where } G \sim \mathbb{P}_{\mathcal{G}}. \tag{5}$$

S_1 and S_2 are different noise signals corresponding to different views. e.g., S_1 and S_2 could represent the nodes and edges to be dropped, and $q(\cdot)$ could be the operation that drops them. The mechanism of data augmentation results in that there will still be a large amount of identical node features between $q(G, S_1)$ and $q(G, S_2)$. We denote the optimal choices for the noise signals as S_1^* and S_2^*. Following [23], we propose that:

$$\begin{aligned} (S_1^*, S_2^*) = \arg\min_{S_1, S_2} I\Big(q(G, S_1); q(G, S_2)\Big) \\ s.t.\ I(q(G, S_1); Y) = I(q(G, S_2); Y) = I(G; Y), \\ where\ (G, Y) \sim \mathbb{P}_{\mathcal{G}\times\mathcal{Y}}. \end{aligned} \tag{6}$$

Ideally, the values of S_1 and S_2 should be chosen to minimize the redundant mutual information between views, which means more modifications will be made to the input graph, e.g., more nodes dropped or edges deleted. Such modifications will lead to an increase in input noises, which will inevitably lead to the corruption of the original input graph. When the graphs get corrupted and changed, they may represent different things. e.g., one node feature of a graph might represent "movie." After the graph is changed, the new node feature might be the same as those representing "paper." Thus, the changed graph has a different set of labels. We denote such labels as Y'. We measure the mutual information between Y and Y' with the following theorem:

Theorem 1. *The mutual information between the original graph label Y and distorted label Y' decreases as the amount of information of input noises S increases.*

Please see Appendix B for proof. Unfortunately, in the task of self-supervised learning, Y' is not available. We have to conduct the training under the assumption that Y is almost the same as Y'. However, suppose we rely on increasing the input noises S to decrease the mutual information between views. In that case, the mutual information between Y' and Y will drop significantly, making the learning meaningless. On the other hand, If we do not increase the input noises that much, there are bound to be identical node features between different views, which will lead to learning redundant information. An alternative is required.

Inspired by contrastive learning algorithms in the computer vision domain [24], we propose the concept of the Segmented Graph Channel (SGC) to generate different views.

Definition 2. *(Segmented Graph Channel). For graph $G = (V, E)$ with attributed node features $\{X_v \in \mathbb{R}^F | v \in V\}$, we denote SGCs of G as C, $C = (V', E)$, V' is a node set that is the same as V except for attributed node features $\{X_{v'} \in \mathbb{R}^{F'} | v' \in V'\}$. $X_{v'}$ is a feature vector that consists of part of the data extracted out of X_v, $F' \leq F$. The feature vectors of graph G are split into different parts to get different SGCs. During the generation of $X_{v'}$, the extraction location on each node feature is the same.*

Different from graph augmentations, SGCs generate new views without damaging any graph information. However, SGCs alone can't serve as different views because of their lack of deformation of the graph structure. We combine the SGCs with the data augmentation method and propose our new learning objective:

$$\max_{f_1,f_2} I\Big(f_1\big(q(C_1, S_1)\big); f_2\big(q(C_2, S_2)\big)\Big), \tag{7}$$

where C_1 and C_2 are two SGCs of G. We can completely eliminated identical node features between $q(C_1, S_1)$ and $q(C_2, S_2)$ as we extract different parts of node features. The edge features can be augmented using conventional approaches. Thus, our new learning objective can effectively reduce the redundant mutual information between views compared to conventional data augmentation methods. Furthermore, we achieve such a goal without further introducing noise.

3.3 Network Structure

The network structure of SGL-SGC is demonstrated in Figure 2. We first adopt SGCs and data augmentation with noise to generate multiple views of the original graph. We will combine each SGC with multiple noise signals to generate different views. We use the same encoder to process the views generated with the same SGC. Parameters are not shared between these encoders.

With k different SGCs $\boldsymbol{C} = \{C_i\}_{i=1}^{k}$ and m different noise signals $\boldsymbol{S} = \{S_j\}_{j=1}^{m}$, we could get $k * m$ different views, these views will go through the corresponding encoders to get the representations. For single input graph $G = (V, E)$ with n nodes, we will acquire $k * m * n$ different representations for all nodes. We denote the output representation as $\boldsymbol{z}_{vij}$. $\boldsymbol{z}_{vij}$ can be formulated as:

$$\boldsymbol{z}_{v,i,j} = f_i\big(q(C_i, S_j)\big). \tag{8}$$

These representations will be processed with a loss function. In our task, we want to make use of all of them. Nevertheless, the conventional contrastive loss will require contrasting negative samples with all of the $k * m * n$ different representations for a single input graph, which will cost too much computing resources. Moreover, these representations will contribute differently to training. We want to emphasize those that contribute more.

3.4 Feature-level weight-sensitive loss

In order to solve the problems mentioned above, we need a loss function that can process the representations in a negative-sample-free way. Inspired by [36], we adopt a feature level learning objective so as to avoid calculating a large amount of negative samples. Given graph G with n nodes, k segmented graph channels, and m augmenters, we will acquire representations $\{\boldsymbol{z}_{1,1,1}, ..., \boldsymbol{z}_{v,i,j}, ..., \boldsymbol{z}_{n,k,m}\}$. Then, we define matrix set $\boldsymbol{M} = \{M_{1,1,2}, ..., M_{v,h,h'}, ...M_{n,k*m-1,k*m}\}$. The subscript v refers to different nodes, and h, h' refer to different views. $M_{v,h,h'}$ can be formulated as follows:

$$M_{v,h,h'} = \boldsymbol{z}_{v,i,j}^{T} \times \boldsymbol{z}_{v,i',j'},\ i \neq i'\ or\ j \neq j' \tag{9}$$

$z_{v,i,j}$ and $z_{v,i',j'}$ are representations of the same node but under different views. Our loss function can be formulated as:

$$\mathcal{L}_{fl} = \sum_{h'} \sum_{h} \sum_{v} \Big(OnDiag(M_{v,h,h'}) + \lambda\ OffDiag(M_{v,h,h'}) \Big),\ h \neq h' \tag{10}$$

Where λ is a hyperparameter that trades off the importance of two terms, $OnDiag(M_{v,h,h'})$ and $OffDiag(M_{v,h,h'})$, we define them as follows:

$$\begin{aligned} OnDiag(M) &= \sum_{a} (1 - m_{a,a})^2, \\ OffDiag(M) &= \sum_{a} \sum_{a \neq b} (m_{a,b})^2. \end{aligned} \tag{11}$$

where $m_{a,a}$ is the element on the diagonal of matrix M, $m_{a,b}$ represents the rest elements. Subscripts a and b are the coordinates. $OnDiag(M)$ implements the optimization objective we described in Equation 7 at the feature level. As we built our optimization objective without the usage of negative samples, we adopt $OffDiag(M)$ to prevent trivial solutions from optimization.

Following [20], we consider the samples that are further away from the optimal goals more crucial. We adopt weight factors $\omega_{v,h,h'}$ to measure the importance of the representations that are used to calculate $M_{v,h,h'}$. $\omega_{v,h,h'}$ can be denoted as:

$$\omega_{v,h,h'} = \Big(\big(OnDiag(M_{v,h,h'}) + \lambda\ OffDiag(M_{v,h,h'}) \big) - O \Big)^{\tau}, \tag{12}$$

where O is the optimal value of the sum of the first two terms, in our task, $O = 0$. τ is a hyperparameter that controls the effect of ω. We use $\omega_{v,h,h'}$ to help emphasize the more crucial samples. Substituting Equation 12 into Equation 10, we have:

$$\mathcal{L}_{wsfl} = \sum_{h'} \sum_{h} \sum_{v} \omega_{v,h,h'} \Big(OnDiag(M_{v,h,h'}) + \lambda\ OffDiag(M_{v,h,h'}) \Big),\ h \neq h' \tag{13}$$

The new loss function is weight-sensitive, which emphasizes representations that are considered more crucial.

4 Experiments

This section demonstrates the effectiveness of our proposed SGL-SGC by conducting extensive experiments on various benchmark datasets.

4.1 Comparison with the state-of-the-art methods

Datasets We select five widely used graph datasets, including three citation network datasets: Cora, Citeseer, and PubMed [21,3], and two relationship datasets: Amazon-Computers and Amazon-Photo [37]. We download all the datasets with DGL APIs, which can be found at `https://www.dgl.ai/`. For the experimental protocol, we follow [9,37], and adopt the same train/validation/test splits. We report the mean classification accuracy with standard deviation over ten runs of training.

Baselines For baselines, we select supervised, semi-supervised and unsupervised graph learning approaches. The supervised approaches include GCN [12] and GAT [27]. The semi-supervised approaches include CG^3 [29]. The unsupervised graph learning approaches include Deepwalk [18], GAE [13], DGI [28], MVGRL [9], GCA [37], and InfoGCL [30].

Table 1. Classification accuracy of compared methods on Cora, CiteSeer, PubMed, Amazon Computers, and Amazon Photos. According to different learning strategies, the records are divided into two groups. The records that are not associated with standard deviations due to the reason that they are directly taken from [22], which did not report their standard deviations. **Bold** denotes the best records.

Methods	Cora	CiteSeer	PubMed	Amazon Computers	Amazon Photo
Supervised & Semi-Supervised Approaches					
GCN	81.5	70.3	79.0	87.0 ± 0.3	92.6 ± 0.4
GAT	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3	86.5 ± 0.5	92.4 ± 0.2
CG^3	83.4 ± 0.7	73.6 ± 0.8	80.2 ± 0.8	79.9 ± 0.6	89.4 ± 0.5
Unsupervised Approaches					
DeepWalk	70.7 ± 0.6	51.4 ± 0.5	74.3 ± 0.9	85.7 ± 0.1	89.4 ± 0.1
GAE	71.5 ± 0.4	65.8 ± 0.4	72.1 ± 0.5	85.3 ± 0.2	91.6 ± 0.1
DGI	83.8 ± 0.5	72.0 ± 0.6	77.9 ± 0.3	84.0 ± 0.5	91.6 ± 0.2
MVGRL	83.2 ± 0.6	72.9 ± 0.3	79.8 ± 0.6	87.5 ± 0.1	91.7 ± 0.1
GCA	82.1 ± 0.4	71.7 ± 0.2	78.9 ± 0.7	87.9 ± 0.3	92.5 ± 0.2
InfoGCL	83.5 ± 0.3	73.5 ± 0.4	79.1 ± 0.2	-	-
SGL-N	83.1 ± 0.7	73.2 ± 0.5	79.5 ± 0.2	85.6 ± 0.3	91.6 ± 0.2
SGL-SGC	**84.2 ± 0.5**	**74.0 ± 0.3**	**80.8 ± 0.4**	**88.7 ± 0.2**	**93.1 ± 0.3**

Evaluation protocol For evaluation protocol, we follow [28] and pre-train the model on all the nodes in the graph without supervision. Then, we freeze the parameters and feed the acquired node representations into a logistic regression model for label prediction. We only use nodes from the training set to train the logistic regression model, and we report the classification accuracy on testing sets.

We adopt the Adam optimizer with an initial learning rate of 10^{-3} for model training. For view generation, we used a total of three SGCs. Each SGC is followed by two data augmentations with different noise signals. SGL-SGC generates six views in total. We adopt three different encoders, each corresponding to an SGC. Each encoder consists of a 2-layer GCN with a hidden dimension of 512. Their outputs are concatenated together for downstream tasks. For Cora, Citeseer, and PubMed datasets, the pre-training epochs were 100, 20, and 100. For Amazon-Computers and Amazon-Photo, the pre-training epochs were set as 60. The hyperparameter that controls the effect of ω is set to 0.2. All of our experiments were conducted on an Nvidia RTX 5000. For the ablation study, we built a network with the same structure as SGL-SGL except for the SGCs. We remove them and generate the same amount of views as SGL-SGC with conventional data augmentations. The new network is named SGL-N.

Results The classification results are reported in Table 1 . We highlight the highest records in bold. As we can see from the table, SGL-SGC outperforms all the other methods across all datasets. The results demonstrate our method's potential to outperform supervised, semi-supervised, and unsupervised methods on various datasets. We attribute this potential to the fact that SGL-SGC can generate views that contain less redundant information. Moreover, we design a feature-level weights-sensitive loss function that can be used to train the encoders better to learn from these views.

Another observable phenomenon is that SGL-N can only achieve comparable results to other methods, while SGL-SGC outperforms it. This outcome proves that only utilizing six different views generated with graph data augmentation does not help produce better performances. Furthermore, it proves the necessity of our proposed SGCs in helping increase the performance of self-supervised representation learning.

4.2 Comparison of computing resource consumption

To analyze the computational resource overhead of our method, we conduct a set of comparative experiments. For comparison, We built a graph contrastive learning framework utilizing conventional InfoNCE loss instead of feature-level weight-sensitive loss, named I-GCL. I-GCL adopts two-layer GCNs as elemental encoders, the same as SGL-SGC. We use the same augmenter for each framework.

The results are demonstrated in Table 2. It shows that SGL-SGC costs much less memory than I-GCL under six views, which proves our proposed feature-level weight-sensitive loss can vastly reduce computing costs. Another interesting phenomenon is that SGL-SGC with six views still costs less memory than I-GCL with two views. Such records prove that, in our task, SGL-SGC does not cost more computing resources than conventional graph contrastive methods. We can also see from the table that when the amount of views increases, the memory cost of I-GCL rises by 47%. On the other hand, the memory cost of SGL-SGC only rises by 16%. Such results suggest that increasing the number of segmented graph channels does not significantly increase the computational cost when using a feature-level weight-sensitive loss. Furthermore, the time cost of SGL-SGC is also less than those of I-GCL.

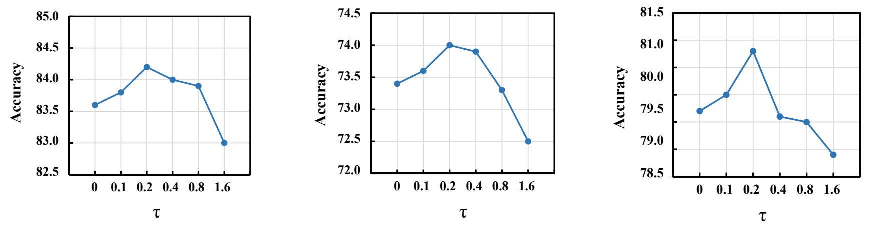
Table 2. Memory costs and time costs of different methods under different conditions. Since the network width of each layer is the same, a single column of hidden dimensions is used to represent them. Views and SGCs represent the number of graph views and SGCs we used. We conduct all the experiments on an Nvidia RTX 5000.

Methods	Hidden Dimension	Views	SGCs	Memory Costs (GB)	Time Costs per Epoch (second)
I-GCL	512	2	1	2.82	0.13
I-GCL	512	6	3	5.35	0.16
SGL-SGC	512	2	1	1.78	0.08
SGL-SGC	512	6	3	2.12	0.09
I-GCL	256	2	1	1.96	0.10
I-GCL	256	6	3	3.89	0.11
SGL-SGC	256	2	1	1.34	0.06
SGL-SGC	256	6	3	1.73	0.08

4.3 Evaluation of the weight factors

In this part, we further evaluate the weight factor ω that we introduced in our proposed loss function. We modify the value of hyperparameter τ that controls the effectiveness of ω and observe how the performance changes. We perform such experiments on multiple datasets. The results are shown in Figure 3.

As we can see, the performance peaks when the value of τ is 0.2 on all three datasets. As τ decreases, ω will hold less influence on training. It is observable that the performance of SGL-SGC drops when τ decreases from 0.2 to 0, which indicates that the influence of ω does improve the representation learning ability of SGL-SGC. We believe ω help emphasize the samples that contribute more to training, thus improving the overall performance. Another observable phenomenon is that when τ takes a larger value than 0.2, the performance of SGL-SGC also drops. This phenomenon shows that the effect of ω cannot be expanded indefinitely, and it is necessary to use the hyperparameter τ to limit it.



(a) Cora (b) CiteSeer (c) PubMed

Fig. 3. Results of SGL-SGC's performance on Cora, CiteSeer, and PubMed datasets with different values of τ. The ordinates in the figure represent different accuracy rates, while the abscissas represent different values of τ .

4.4 Representation capability analysis

For further analysis of the representation capability of our proposed method, we visualize the outputs to make an intuitive observation. For comparison, we adopt a deformation of SGL-SGC with InfoNCE loss and only two different views without SGCs. The new framework is named IN-GCL.

Figure 4 demonstrates the results. We can see that the untrained encoder output features without much distinguishability. We can observe many vertical lines running through multiple blocks of different labels. These lines represent similar representations, indicating there are common features shared between different classes. The output of SGL-SGC under ten epochs of training shows some interesting developments. The vertical lines of each block become clearer than in the previous column, and there is still not much distinguishability. For the outputs of SGL-SGC after 100 epochs of training, we can see much difference between each class. We believe that SGL-SGC will first increase the independence of each dimension of the feature during the training process, making the output more expressive. After that, the distinction between the various classes appears, indicating that the encoders have learned meaningful information.

The last column of Figure 4 shows the output of IN-GCL without feature-level weight-sensitive loss and SGCs. It can be seen that after the same 100 rounds of training, SGL-SGC can learn more discriminative features than IN-GCL, which demonstrate the superiority of our method.

5 Conclusions

This paper proposed a self-supervised graph learning method with segmented graph channels. We enhance the conventional view generation with segmented graph channels to reduce the redundant mutual information between multiple views while avoiding introducing more noises. We also proposed a feature-level weight-sensitive loss as our training objective. This loss function can emphasize samples with more contribution to training and reduce consumption of computing resources. We conducted multiple experiments to prove the superiority of our proposed method.

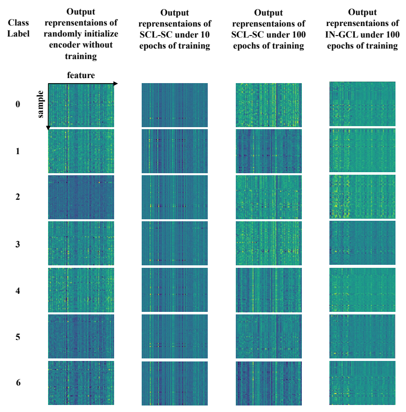


Fig. 4. Visualized output representations of different frameworks. Each horizontal line in each small block represents an output representation. In each small block, the horizontal axis represents different dimensions of the output representation, and the vertical axis represents different samples. All representations are grouped by category.

Acknowledgements This work is supported by the National Key Research and Development Program of China (No. 2019YFB1405100).

References

1. Allamanis, M., Brockschmidt, M., Khademi, M.: Learning to represent programs with graphs. In: International Conference on Learning Representations (2018)
2. Arora, S., Khandeparkar, H., Khodak, M., Plevrakis, O., Saunshi, N.: A theoretical analysis of contrastive unsupervised representation learning. In: 36th International Conference on Machine Learning, ICML 2019. pp. 9904–9923. International Machine Learning Society (IMLS) (2019)
3. Bojchevski, A., Günnemann, S.: Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In: International Conference on Learning Representations (2018)
4. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning. pp. 1597–1607. PMLR (2020)
5. Duvenaud, D.K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., Adams, R.P.: Convolutional networks on graphs for learning molecular fingerprints. Advances in neural information processing systems **28** (2015)
6. Federici, M., Dutta, A., Forré, P., Kushman, N., Akata, Z.: Learning robust representations via multi-view information bottleneck. In: International Conference on Learning Representations (2019)

7. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al.: Bootstrap your own latent-a new approach to self-supervised learning. Advances in Neural Information Processing Systems **33**, 21271–21284 (2020)
8. Grover, A., Zweig, A., Ermon, S.: Graphite: Iterative generative modeling of graphs. In: International conference on machine learning. pp. 2434–2444. PMLR (2019)
9. Hassani, K., Khasahmadi, A.H.: Contrastive multi-view representation learning on graphs. In: International Conference on Machine Learning. pp. 4116–4126. PMLR (2020)
10. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9729–9738 (2020)
11. Hjelm, R.D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., Bengio, Y.: Learning deep representations by mutual information estimation and maximization. In: International Conference on Learning Representations (2018)
12. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
13. Kipf, T.N., Welling, M.: Variational graph auto-encoders. In: Bayesian Deep Learning Workshop@NIPS (2016)
14. Lee, J.D., Lei, Q., Saunshi, N., Zhuo, J.: Predicting what you already know helps: Provable self-supervised learning. Advances in Neural Information Processing Systems **34** (2021)
15. Linsker, R.: Self-organization in a perceptual network. Computer **21**(3), 105–117 (1988)
16. Oord, A.v.d., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018)
17. Peng, Z., Huang, W., Luo, M., Zheng, Q., Rong, Y., Xu, T., Huang, J.: Graph representation learning via graphical mutual information maximization. In: Proceedings of The Web Conference 2020. pp. 259–270 (2020)
18. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 701–710 (2014)
19. Sanchez-Gonzalez, A., Heess, N., Springenberg, J.T., Merel, J., Riedmiller, M., Hadsell, R., Battaglia, P.: Graph networks as learnable physics engines for inference and control. In: International Conference on Machine Learning. pp. 4470–4479. PMLR (2018)
20. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 815–823 (2015)
21. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI magazine **29**(3), 93–93 (2008)
22. Sun, F.Y., Hoffman, J., Verma, V., Tang, J.: Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In: International Conference on Learning Representations (2020)
23. Suresh, S., Li, P., Hao, C., Neville, J.: Adversarial graph augmentation to improve graph contrastive learning. Advances in Neural Information Processing Systems **34** (2021)
24. Tian, Y., Krishnan, D., Isola, P.: Contrastive multiview coding. In: European conference on computer vision. pp. 776–794. Springer (2020)

25. Tishby, N., Pereira, F.C., Bialek, W.: The information bottleneck method. arXiv preprint physics/0004057 (2000)
26. Tishby, N., Zaslavsky, N.: Deep learning and the information bottleneck principle. In: 2015 ieee information theory workshop (itw). pp. 1–5. IEEE (2015)
27. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
28. Velickovic, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., Hjelm, R.D.: Deep graph infomax. ICLR (Poster) **2**(3), 4 (2019)
29. Wan, S., Pan, S., Yang, J., Gong, C.: Contrastive and generative graph convolutional networks for graph-based semi-supervised learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 10049–10057 (2021)
30. Xu, D., Cheng, W., Luo, D., Chen, H., Zhang, X.: Infogcl: Information-aware graph contrastive learning. Advances in Neural Information Processing Systems **34** (2021)
31. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: International Conference on Learning Representations (2018)
32. Xu, M., Wang, H., Ni, B., Guo, H., Tang, J.: Self-supervised graph-level representation learning with local and global structure. In: International Conference on Machine Learning. pp. 11548–11558. PMLR (2021)
33. Xu, X., Feng, W., Jiang, Y., Xie, X., Sun, Z., Deng, Z.H.: Dynamically pruned message passing networks for large-scale knowledge graph reasoning. In: International Conference on Learning Representations (2019)
34. Yang, L., Zhang, L., Yang, W.: Graph adversarial self-supervised learning. Advances in Neural Information Processing Systems **34** (2021)
35. You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., Shen, Y.: Graph contrastive learning with augmentations. Advances in Neural Information Processing Systems **33**, 5812–5823 (2020)
36. Zbontar, J., Jing, L., Misra, I., LeCun, Y., Deny, S.: Barlow twins: Self-supervised learning via redundancy reduction. In: International Conference on Machine Learning. pp. 12310–12320. PMLR (2021)
37. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Graph contrastive learning with adaptive augmentation. In: Proceedings of the Web Conference 2021. pp. 2069–2080 (2021)