Simple Unsupervised Graph Representation Learning

Yujie Mo[*], Liang Peng[*], Jie Xu, Xiaoshuang Shi[†], Xiaofeng Zhu
School of Computer Science and Engineering,
University of Electronic Science and Technology of China, Chengdu 611731, China
moyujie2017@gmail.com, larrypengliang@gmail.com, jiexuwork@outlook.com, xsshi2013@gmail.com,
seanzhuxf@gmail.com

Abstract

In this paper, we propose a simple unsupervised graph representation learning method to conduct effective and efficient contrastive learning. Specifically, the proposed multiplet loss explores the complementary information between the structural information and neighbor information to enlarge the inter-class variation, as well as adds an upper bound loss to achieve the finite distance between positive embeddings and anchor embeddings for reducing the intra-class variation. As a result, both enlarging inter-class variation and reducing intra-class variation result in small generalization error, thereby obtaining an effective model. Furthermore, our method removes widely used data augmentation and discriminator from previous graph contrastive learning methods, meanwhile available to output low-dimensional embeddings, leading to an efficient model. Experimental results on various real-world datasets show the effectiveness and efficiency of our method, compared to state-of-the-art methods. Our code is released at https://github.com/YujieMo/SUGRL.

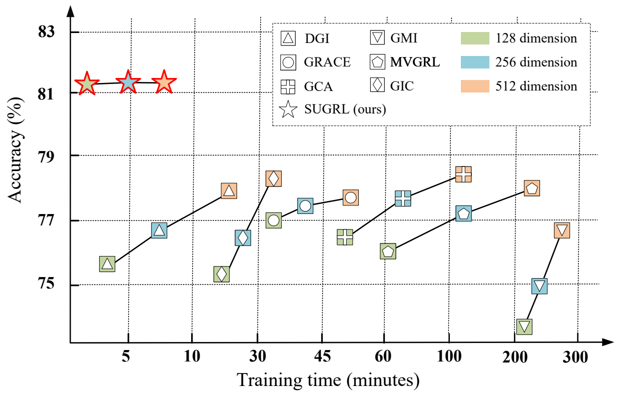


Figure 1: The variations of accuracy and training time of our method and previous methods with different dimensions of embeddings on dataset Ogbn-products.

1 Introduction

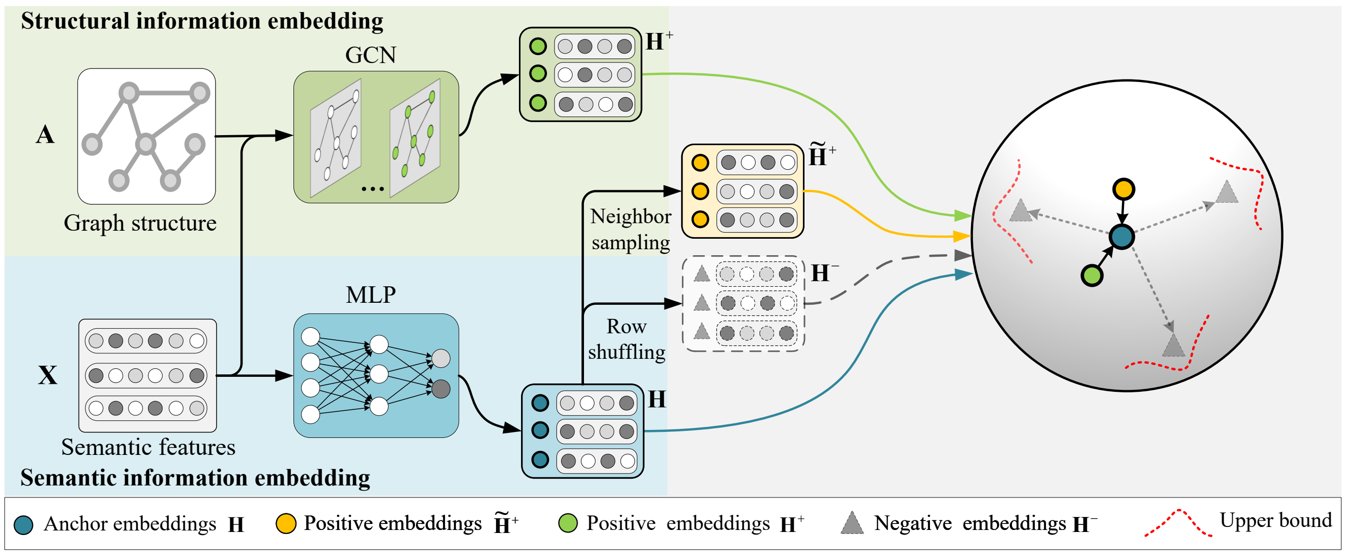
Since the widespread applications of Graph Neural Networks (GNN) (Xu et al. 2021c; Zhou et al. 2020; Zhu et al. 2017, 2019), Unsupervised Graph Representation Learning (UGRL) has also recently received a great deal of attention, which does not require abundant labeled nodes for training. UGRL is able to output discriminative representation by simultaneously learning representations and preserving the local structure of samples (Park et al. 2019; Sun et al. 2019). The discriminative representation ensures downstream tasks to output effective models so that UGRL has shown remarkable performance in real applications (Chen et al. 2020a; Hassani and Khasahmadi 2020). As one of the representative methods of the UGRL, contrastive learning was proposed to maximize Mutual Information (MI) between the input content and its related content (You et al. 2020; Qiu et al. 2020; Jing, Park, and Tong 2021).

The key difference among graph contrastive learning methods is the definitions of the input contents and their related contents (Jiao et al. 2020; Yu et al. 2021; Xu et al. 2021b). For instance, Deep Graph Infomax (DGI) maximizes the MI between the node representations and the summary of the graph (Velickovic et al. 2019). Graphical Mutual Information (GMI) maximizes the MI between the input graph and the output graph (Peng et al. 2020). GRACE (Zhu et al. 2020) and GCA (Zhu et al. 2021) maximize the MI between two views for each node through a variety of data augmentations, *e.g.,* attribute masking or edge perturbation.

Although previous methods are effective in many tasks of representation learning, they usually rely on data augmentation to generate both input contents and their related contents for MI maximization, resulting in expensive computation costs of the training process (Suresh et al. 2021; Jin et al. 2021). Hence, these previous methods are generally inefficient, especially for large-scale datasets, as shown in Figure 1, where previous methods have a drastic increase of computation costs with the increase of either the sample number or the dimensionality of the embeddings.

Actually, the reasons resulting in weak scalability in previous contrastive learning methods include data augmentation, high-dimensional embeddings, and contrastive loss, *etc.* First, previous works usually generate multiple views as related content by data augmentation. For example, GRACE and GCA remove edges and mask node features to generate mul-

[*]These authors contributed equally.
[†]Corresponding author.



Structural information embedding
GCN
A
Graph structure
$\mathbf{H}^+$
$\widetilde{\mathbf{H}}^+$
Neighbor sampling
$\mathbf{H}^-$
Row shuffling
MLP
X
Semantic features
H
Semantic information embedding
Anchor embeddings H
Positive embeddings $\widetilde{\mathbf{H}}^+$
Positive embeddings $\mathbf{H}^+$
Negative embeddings $\mathbf{H}^-$
Upper bound

Figure 2: The flowchart of the proposed SUGRL method. Specifically, given the semantic features $\mathbf{X}$ and its graph structure $\mathbf{A}$, the SUGRL employs a MLP network on $\mathbf{X}$ with the semantic information to generate the anchor embeddings $\mathbf{H}$, and employs GCN to generate positive embeddings $\mathbf{H}^+$ with the structural information as well as the neighbor sampling method to generate positive embeddings $\widetilde{\mathbf{H}}^+$ with the neighbor information. The SUGRL also employs the row-wise random permutation method on $\mathbf{H}$ to generate negative embeddings $\mathbf{H}^-$, and further designs a multiplet loss to achieve that the anchor embeddings are close to positive embeddings and far away from negative embeddings.

tiple views. As a result, the computation cost of data augmentation (including data generation and data encoding) takes about 20%-40% of the training time. Second, existing works increase the dimensionality of the embeddings to improve the representation quality, resulting in the increase of the training time. The reason is that the effectiveness of these methods is sensitive to the dimensionality (Liu et al. 2017, 2018). For example, DGI and GMI achieve their best accuracy with 512 dimensions. Third, previous works usually design a discriminator (contains learnable parameters) for the objective function, which is computationally prohibitive (Zhang et al. 2020). For example, DGI and MVGRL (Hassani and Khasahmadi 2020) employ a discriminator to measure the agreement of the node embedding and graph embedding, taking about 10%-30% of the training time. Obviously, it is interesting to find an UGRL method that has low computation costs for the training process and a high-quality representation.

In this paper, we propose a new contrastive learning method, namely Simple Unsupervised Graph Representation Learning (SUGRL), to achieve effectiveness and scalability for representation learning, as shown in Figure 2. Specifically, we first employ a Multi-Layer Perceptron (MLP) on the input representation with the semantic information to generate anchor embeddings, and separately adopt Graph Convolution Network (GCN) (Kipf and Welling 2017) and the neighbor sampling method to generate two different types of positive embeddings, followed by employing the row-wise random permutation method on anchor embeddings to generate negative embeddings. We further design a new multiplet loss to enforce that anchor embeddings are close to positive embeddings and far away from negative embeddings, by reducing the intra-class variation and meanwhile enlarging the inter-class variation.

Compared to previous methods, the contribution of our SUGRL is summarized as follows. First, to guarantee the effectiveness, we propose to jointly consider structural information and neighbor information to explore their complementary information, aiming at enlarging the inter-class variation, as well as designing an upper bound loss to achieve small intra-class variation. Second, to achieve efficiency, we remove data augmentation and discriminator out of contrastive learning. This makes our method easily achieve scalability on large-scale datasets. Finally, comprehensive empirical studies on 8 public benchmark datasets verifies the effectiveness and efficiency of our method, compared to 11 comparison methods, in terms of node classification.

2 Related work

2.1 Contrastive representation learning

Deep models are extensive applied with their unparalleled ability to learn representations (Liu et al. 2021; Cao et al. 2019; Xu et al. 2021a). As a part of them, self-supervised learning methods have attracted a deal of attention with their outstanding performance in areas such as computer vision (Song et al. 2018; Xu et al. 2020). Contrastive representation learning is one of the most representative, specifically, it learns discriminative representations by contrasting positive and negative samples. These methods generally encourage an encoder to learn representations by maximizing the mutual information (MI) between the input and the learned representation. For instance, Contrastive Predictive Coding (CPC) (van den Oord, Li, and Vinyals 2018) contrasts a summary of ordered local features to predict a local feature in the future while Deep InfoMax (DIM) (Devon et al. 2019) simultaneously contrasts a single global feature with all local features. Contrastive Multiview Coding (CMC) (Tian, Krishnan, and Isola 2020) presents a contrastive learning framework, which learns unsupervised representations from multiple views of a dataset. SimCLR (Chen et al. 2020a) extends the InfoMax principle to multiple views and maximizes the MI across views generated by data augmentations. MoCo (Chen et al. 2020b) presents momentum contrast for unsupervised visual representation learning, and can build a large and consistent dictionary that facilitates contrastive unsupervised learning.

2.2 Unsupervised graph representation learning

Contrastive learning methods were adapted to graph representation learning by the success in vision and other fields. DGI (Velickovic et al. 2019) extends DIM to graphs by learning node representations through contrasting local and global embeddings and obtains great performance on node classification benchmarks. GIC (Mavromatis and Karypis 2021) proposes an unsupervised graph representation learning method, which relies on maximizing the MI between the node-level representations and cluster-level representations. GMI (Peng et al. 2020) generalizes the idea of conventional MI computations from the vector space to the graph domain and measures MI from both node features and topological structure. GRACE (Zhu et al. 2020) and GCA (Zhu et al. 2021) adapt the SimCLR to graphs and achieve state-of-the-art performance. In particular, GRACE and GCA learn node representations by creating two views of the graph, pulling the representation of the same node in two views close, while pushing the representation of every other node apart. MVGRL (Hassani and Khasahmadi 2020) proposes a contrastive multi-view representation learning method by contrasting embeddings from two structural views of graphs, including first-order neighbors and a graph diffusion. However, these methods tend to rely heavily on data augmentation, which can seriously compromise the scalability of the method. Different from previous works, our work enhances effectiveness and efficiency of the model simultaneously by maximizing MI between semantic information and related information (*i.e.,* structural information and neighbor information).

3 Method

Notations Letting $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph, where $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represent the node set and the edge set, respectively. We denote the feature matrix and the adjacency matrix as $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ and $\mathbf{A} \in \{0,1\}^{N \times N}$, respectively, where $\mathbf{x}_i$ is the feature of the node v_i, and $a_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$ otherwise $a_{ij} = 0$. In addition, we assume the existence of a set of latent classes $\mathcal{C}$ over the representation space $\mathcal{H}$, since neither the embeddings nor the samples are labeled in unsupervised learning.

The proposed SUGRL focuses on MI maximization. In the literature, considering the computation cost of MI maximization (Paninski 2003; Belghazi et al. 2018), the MI maximization in the SUGRL is transferred to contrastive learning, which involves the definitions of anchor embeddings, positive and negative embeddings, as well as the contrastive loss.

3.1 Anchor and negative embedding generation

Existing works generally treat node representations or the graph summary as anchors (Velickovic et al. 2019; Zeng and Xie 2021; Ren, Bai, and Zhang 2021; Cao et al. 2021; Sun et al. 2019). For instance, DGI and MVGRL treat the graph summary as anchors, which is first convolved by GCN and then summarized by a readout function. GRACE and GCA regard the node embedding generated in one view as anchors. However, these methods basically need to conduct GCN, which is time consuming. To achieve scalability, in this paper, we employ the MLP on the input $\mathbf{X}$ to generate anchor embeddings with the semantic information, *i.e.,*

$$\mathbf{X}^{(l+1)} = \mathit{Dropout}\left(\sigma\left(\mathbf{X}^{(l)}\mathbf{W}^{(l)}\right)\right), \tag{1}$$

$$\mathbf{H} = \mathbf{X}^{(l+1)}\mathbf{W}^{(l+1)}, \tag{2}$$

where $\mathbf{X}^{(0)} = \mathbf{X}$, σ is an activation function, and $\mathbf{W}^{(l)}$ is the weight of the l^{th} layer.

For the generation of negative embeddings, popular methods (*e.g.,* DGI, GIC and MVGRL) are to obtain a corrupted graph from the original graph, and then to process it with GCN (Velickovic et al. 2019; Park et al. 2020; Mavromatis and Karypis 2021). By contrast, we directly row-shuffle anchor embeddings to obtain negative embeddings, and further reducing the training time, *i.e.,*

$$\mathbf{H}^{-} = \mathit{Shuffle}\left([\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_N]\right). \tag{3}$$

In conclusion, our proposed method reduces the computation cost by removing the GCN for the generation of both anchors and negative embeddings, while keeping its effectiveness (verified in Section 4.2).

3.2 Positive embedding generation

Existing works generally treat the structural information as positive embeddings, *e.g.,* DGI, MVGRL, GRACE, GCA and GIC. In addition, previous works often employ data augmentation to obtain diverse information for effective contrastive learning, *e.g.,* stochastic graph augmentation in GRACE and GCA, and graph diffusion in MVGRL. By contrast, in this paper, we propose to obtain diverse information by generating two kinds of positive embeddings, *i.e.,* structural embeddings and neighbor embeddings. Specifically, we employ the GCN and the neighbor sampling method to generate them.

- **Structural information**
  To obtain the structural information of the graph, we apply the widely used GCN as the base encoder:

$$\mathbf{H}^{+^{(l+1)}} = \sigma\left(\widehat{\mathbf{A}}\mathbf{H}^{+^{(l)}}\mathbf{W}^{(l)}\right), \tag{4}$$

  where $\mathbf{H}^{+(0)} = \mathbf{X}$ and $\mathbf{H}^{+(l)}$ means the l^{th} layer features. $\widehat{\mathbf{A}} = \hat{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\hat{\mathbf{D}}^{-1/2} \in \mathbb{R}^{N \times N}$ is a symmetrically normalized adjacency matrix, $\hat{\mathbf{D}} \in \mathbb{R}^{N \times N}$ is the degree matrix of $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$, $\mathbf{I}_N$ is the identity matrix. It is noteworthy that our method directly shares the weights (*i.e.,* $\mathbf{W}^{(l)}$) between the MLP and GCN encoders to further reduce the time costs.

- **Neighbor information**

  To obtain the positive embeddings with the neighbor information, we first store the neighbor's embedding index of all nodes and then sample it, followed by calculating the average of the samples. In this way, the neighbor information of the node can be obtained efficiently:

$$\widetilde{\mathbf{h}}_i^{+} = \frac{1}{m}\sum_{j=1}^{m}\left\{\mathbf{h}_j \mid v_j \in \mathcal{N}_i\right\}, \tag{5}$$

  where m is the number of sampled neighbors, $\mathcal{N}_i$ represents 1-hop neighborhood set of node v_i.

In summary, the structural embedding and the neighbor embedding focus on all neighbors and a certain part of neighbors, respectively. That is, the structural embedding is general representation while the neighbor embedding is specific. Hence, they explain samples from different perspectives, and thus considering them together to possibly obtain their complementary information (verified by Section 3.3).

3.3 Multiplet loss

Given anchor embeddings, positive embeddings and negative embeddings, contrastive learning aims to make positive pairs (*i.e.,* anchor and positive embeddings) close while keeping negative pairs (*i.e.,* anchor and negative embeddings) far apart. Many contrastive learning methods (*e.g.,* DGI, GMI, MVGRL and GIC) design a discriminator (*e.g.,* a bilinear layer) to distinguish positive pairs from negative pairs, but the discriminator is time-consuming, as shown in the right side of Figure 6. Additionally, reducing generalization error is also important for UGRL as a small generalization error in the training process might improve the generalization capability of contrastive learning (Xuan et al. 2019). Moreover, either reducing the intra-class variation or enlarging the inter-class variation has been demonstrated to be an effective solution to reduce generalization error (Wen et al. 2016).

In SUGRL, we consider the triplet loss as the basis and design an upper bound loss to remove the discriminator from our method (efficiency), and reduce intra-class variation as well as enlarge inter-class variation (effectiveness). Specifically, the triplet loss with respect to each sample can be formulated as:

$$\alpha + d\left(\mathbf{h}, \mathbf{h}^{+}\right) < d\left(\mathbf{h}, \mathbf{h}^{-}\right), \tag{6}$$

where $d(\cdot)$ is a similarity measurement (*e.g.,* ℓ_2-norm distance) and α is a non-negative value to ensure a "safe" distance between positive and negative embeddings. By summing the loss of all negative embeddings, Eq. (6) can be extended to:

$$\mathcal{L}_{triplet} = \frac{1}{k}\sum_{i=1}^{k}\left\{d(\mathbf{h}, \mathbf{h}^{+})^2 - d\left(\mathbf{h}, \mathbf{h}_i^{-}\right)^2 + \alpha\right\}_{+}, \tag{7}$$

where $\{\cdot\}_{+} = \max\{\cdot, 0\}$, and k is the number of negative samples.

To increase the inter-class variation, we should push negative pairs far away from positive pairs. To do this, we employ the triplet loss on two kinds of positive embeddings defined in Section 3.2 to have:

$$\mathcal{L}_S = \frac{1}{k}\sum_{i=1}^{k}\left\{d\left(\mathbf{h}, \mathbf{h}^{+}\right)^2 - d\left(\mathbf{h}, \mathbf{h}_i^{-}\right)^2 + \alpha\right\}_{+}, \tag{8}$$

$$\mathcal{L}_N = \frac{1}{k}\sum_{j=1}^{k}\left\{d\left(\mathbf{h}, \widetilde{\mathbf{h}}^{+}\right)^2 - d\left(\mathbf{h}, \mathbf{h}_j^{-}\right)^2 + \alpha\right\}_{+}. \tag{9}$$

As aforementioned in Section 3.2, the structural embedding (*i.e.,* $\mathbf{h}^{+}$) is different from the neighbor embedding (*i.e.,* $\widetilde{\mathbf{h}}^{+}$). According to the distance between two types of positive embeddings and the anchor embedding, two cases can be classified, *i.e.,* Case 1: $d\left(\mathbf{h}, \mathbf{h}^{+}\right)^2 \geq d(\mathbf{h}, \widetilde{\mathbf{h}}^{+})^2$ and Case 2: $d\left(\mathbf{h}, \mathbf{h}^{+}\right)^2 < d(\mathbf{h}, \widetilde{\mathbf{h}}^{+})^2$.

If $d\left(\mathbf{h}, \mathbf{h}^{+}\right)^2 \geq d(\mathbf{h}, \widetilde{\mathbf{h}}^{+})^2$, the $\{\cdot\}_{+}$ term in Eq (8) can be nonzero even if the corresponding term in Eq (9) is zero. In this scenario, $\mathcal{L}_S$ is still effective while $\mathcal{L}_N$ is ineffective. As a result, negative embeddings will continue to be pushed far away from anchor embeddings by Eq. (8), and thus the inter-class variation is enlarged. Similar to the Case 1, Case 2 can also enlarge the inter-class variation.

Based on the above analysis, either Case 1 or Case 2 can enlarge the inter-class variation. In particular, if one of them is ineffective, the other will still on effective to further enlarge the inter-class variation. Hence, Eq. (8) and Eq. (9) may obtain complementary information from the structural embeddings and the neighbor embeddings, so that they are able to the enlarge inter-class variation.

Eq. (7) requires that the distance between $d\left(\mathbf{h}, \mathbf{h}^{+}\right)^2$ and $d(\mathbf{h}, \mathbf{h}_i^{-})^2$ should be larger than α, but it ignores the distance between anchor and positive embeddings. If the distance between anchor and positive embeddings is large, the $\{\cdot\}_{+}$ term in Eq (7) can also be nonzero, However, in this scenario, the intra-class variation can be large, not benefiting the reduction of generalization error.

To address this issue, we investigate an upper bound (*i.e.,* $\alpha + \beta$) for negative pairs and positive pairs by the following objective function:

$$\alpha + d\left(\mathbf{h}, \mathbf{h}^{+}\right) < d\left(\mathbf{h}, \mathbf{h}^{-}\right) < d\left(\mathbf{h}, \mathbf{h}^{+}\right) + \alpha + \beta, \tag{10}$$

where β is a non-negative tuning parameter. The upper bound $\alpha + \beta$ guarantees that the distance between negative embeddings and anchor embeddings is finite, so the distance between positive embeddings and anchor embeddings is also finite based on Eq. (6). As a result, intra-class variation is reduced. After summing the loss for all negative embeddings, the proposed upper bound loss reducing the intra-class variation is defined as follows:

$$\mathcal{L}_U = -\frac{1}{k}\sum_{i=1}^{k}\left\{d\left(\mathbf{h}, \mathbf{h}^{+}\right)^2 - d\left(\mathbf{h}, \mathbf{h}_i^{-}\right)^2 + \alpha + \beta\right\}_{-}, \tag{11}$$

where $\{\cdot\}_{-} = \min\{\cdot, 0\}$, and $d\left(\mathbf{h}, \mathbf{h}^{+}\right)^2 + \alpha + \beta$ is a target in Eq. (11). It is noteworthy that the upper bound is not pushed on the neighbor information due to that 1) each kind

	Cora		CiteSeer		PubMed		Photo	
Method	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time
Raw Feature	47.9 ± 0.4	–	49.3 ± 0.3	–	69.1 ± 0.2	–	78.5 ± 0.2	–
Deep Walk	67.2 ± 0.2	19.2	43.2 ± 0.4	4.6	65.3 ± 0.5	144.2	89.4 ± 0.1	76.8
GCN	81.5 ± 0.2	3.1	70.3 ± 0.4	1.4	79.0 ± 0.5	6.1	91.6 ± 0.3	6.7
GAT	83.0 ± 0.2	16.9	72.5 ± 0.3	4.2	79.0 ± 0.5	62.5	91.8 ± 0.1	50.0
GAE	74.9 ± 0.4	24.5	65.6 ± 0.5	8.1	74.2 ± 0.3	165.4	91.0 ± 0.1	108.4
VGAE	76.3 ± 0.2	26.9	66.8 ± 0.2	8.7	75.8 ± 0.4	166.3	91.5 ± 0.2	107.8
DGI	82.3 ± 0.5	10.5	71.5 ± 0.4	3.1	79.4 ± 0.3	128.1	91.3 ± 0.1	54.1
GMI	83.0 ± 0.2	100.1	72.4 ± 0.2	24.3	79.9 ± 0.4	1104.2	90.6 ± 0.2	461.3
GRACE	83.1 ± 0.2	6.8	72.1 ± 0.1	2.5	79.6 ± 0.5	196.9	91.9 ± 0.3	53.4
MVGRL	82.9 ± 0.3	67.1	72.6 ± 0.4	18.3	80.1 ± 0.7	669.2	91.7 ± 0.1	272.3
GCA	81.8 ± 0.2	11.1	71.9 ± 0.4	4.2	81.0 ± 0.3	312.1	92.4 ± 0.4	65.1
GIC	81.7 ± 0.8	8.6	71.9 ± 0.9	3.6	77.4 ± 0.5	15.1	91.6 ± 0.1	15.2
SUGRL	**83.4 ± 0.5**	3.8	**73.0 ± 0.4**	0.9	**81.9 ± 0.3**	9.5	**93.2 ± 0.4**	5.6

Table 1: Classification accuracy (%) and execution time (seconds) of all methods on four datasets.

	Computers		Ogbn-arxiv		Ogbn-mag		Ogbn-products	
Method	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time
Raw Feature	73.8 ± 0.1	–	56.3 ± 0.3	–	22.1 ± 0.3	–	59.7 ± 0.2	–
Deep Walk	85.3 ± 0.1	2.2	63.6 ± 0.4	5.1	25.6 ± 0.3	12.1	73.2 ± 0.2	30.5
GCN	84.5 ± 0.3	0.2	70.4 ± 0.3	0.1	30.1 ± 0.3	0.7	81.6 ± 0.4	2.6
GAT	85.7 ± 0.1	1.5	**70.6 ± 0.3**	2.5	30.5 ± 0.3	6.1	82.4 ± 0.4	14.6
GAE	85.1 ± 0.4	4.1	63.6 ± 0.5	9.4	27.1 ± 0.3	22.5	72.1 ± 0.1	56.8
VGAE	85.8 ± 0.3	4.0	64.8 ± 0.2	9.5	27.9 ± 0.2	22.7	72.9 ± 0.2	57.3
DGI	87.8 ± 0.2	2.0	65.1 ± 0.4	4.5	31.4 ± 0.3	10.6	77.9 ± 0.2	26.8
GMI	82.2 ± 0.4	15.3	68.2 ± 0.2	42.0	29.5 ± 0.1	123.1	76.8 ± 0.3	293.4
GRACE	86.8 ± 0.2	2.2	68.7 ± 0.4	7.2	31.5 ± 0.3	19.2	77.4 ± 0.4	43.3
MVGRL	86.9 ± 0.1	10.4	68.1 ± 0.1	24.6	31.6 ± 0.4	67.3	78.1 ± 0.1	213.7
GCA	87.7 ± 0.1	2.6	68.2 ± 0.2	7.1	31.4 ± 0.3	37.5	78.4 ± 0.3	81.2
GIC	84.9 ± 0.2	0.4	68.4 ± 0.4	1.1	31.7 ± 0.2	5.1	75.8 ± 0.2	8.8
SUGRL	**88.9 ± 0.2**	0.2	68.8 ± 0.4	0.1	31.9 ± 0.3	0.4	**82.6 ± 0.4**	2.1
SUGRL-batch	–	–	69.3 ± 0.2	0.2	**32.4 ± 0.1**	0.4	81.2 ± 0.1	2.2

Table 2: Classification accuracy (%) and execution time (minutes) of all methods on four datasets.

of information achieves similar results and 2) use them simultaneously can not significantly improve model performance in our experiments.

Finally, integrating triplet losses (*i.e.,* Eq. (8) and Eq. (9)) with the upper bound loss in Eq. (11), our proposed multiplet loss is formulated as:

$$\mathcal{L} = \omega_1 \mathcal{L}_S + \omega_2 \mathcal{L}_N + \mathcal{L}_U, \tag{12}$$

where ω_1 and ω_2 are the weights of $\mathcal{L}_S$ and $\mathcal{L}_N$, respectively.

4 Experiments

4.1 Experimental setup

Datasets In our experiments, we used 8 commonly used benchmark datasets including 3 citation networks datasets (*i.e.,* Cora, Citeseer, and Pubmed) (Yang, Cohen, and Salakhudinov 2016), 2 amazon sale datasets (*i.e.,* Photo, and Computers) (Shchur et al. 2018), 3 large-scale datasets (*i.e.,* Ogbn-arxiv, Ogbn-mag, and Ogbn-products) (Weihua et al. 2020).

Comparison methods The comparative methods include 1 traditional algorithm (*i.e.,* DeepWalk (Perozzi, Al-Rfou, and Skiena 2014)), 2 semi-supervised learning algorithms (*i.e.,* GCN (Kipf and Welling 2017) and GAT (Velickovic et al. 2018)), and 8 unsupervised learning algorithms (*i.e.,* Graph Auto-Encoders (GAE) (Kipf and Welling 2016), Variational Graph Auto-Encoders (VGAE) (Kipf and Welling 2016), DGI (Velickovic et al. 2019), GRACE (Zhu et al. 2020), GMI (Peng et al. 2020), MVGRL (Hassani and Khasahmadi 2020), and GCA (Zhu et al. 2021)), GIC (Mavromatis and Karypis 2021). In particular, raw features were directly used to conduct the node classification task.

$\mathcal{L}_S$	$\mathcal{L}_N$	$\mathcal{L}_U$	**Cora**	**CiteSeer**	**PubMed**	**Photo**	**Computers**	**Ogbn-arxiv**	**Ogbn-mag**	**Ogbn-products**
$\surd$	–	–	73.8 ± 0.6	71.7 ± 0.5	70.7 ± 0.3	91.0 ± 0.2	84.4 ± 0.2	68.1 ± 0.1	31.2 ± 0.2	82.3 ± 0.1
–	$\surd$	–	78.1 ± 0.4	71.8 ± 0.3	80.5 ± 0.3	79.7 ± 0.2	72.5 ± 0.4	67.9 ± 0.2	31.1 ± 0.1	82.1 ± 0.1
$\surd$	$\surd$	–	78.5 ± 0.4	71.9 ± 0.4	81.6 ± 0.3	91.6 ± 0.3	86.6 ± 0.4	68.5 ± 0.1	31.6 ± 0.1	82.4 ± 0.1
$\surd$	–	$\surd$	81.5 ± 0.5	72.2 ± 0.5	79.3 ± 0.4	91.9 ± 0.2	86.9 ± 0.3	68.6 ± 0.2	31.8 ± 0.1	82.5 ± 0.2
–	$\surd$	$\surd$	81.9 ± 0.4	72.1 ± 0.3	80.3 ± 0.3	82.6 ± 0.3	74.9 ± 0.4	68.0 ± 0.2	31.6 ± 0.2	82.5 ± 0.1
$\surd$	$\surd$	$\surd$	**83.4 ± 0.4**	**73.0 ± 0.3**	**81.9 ± 0.3**	**93.2 ± 0.2**	**88.9 ± 0.3**	**68.8 ± 0.1**	**31.9 ± 0.1**	**82.6 ± 0.1**

Table 3: Classification accuracy (%) of each component in our proposed method on all datasets.

Setting-up All experiments were implemented in PyTorch and conducted on a server with 8 NVIDIA GeForce 3090 (24GB memory each). In all experiments, we repeated the experiments five times with random seeds for all algorithms to finally report the average results, and the corresponding standard deviation (std). We obtained the author-verified codes for all comparison methods and achieved their best performance via a grid search method. For large-scale datasets, the comparison methods utilized the mini-batch strategy (Zeng et al. 2020) since they would be out of memory if employing the full-batch strategy, and we also provided the results of our method on both the mini-batch and full-batch strategies to ensure a fair comparison.

In SUGRL, all parameters were initialized by the Glorot initialization (Glorot and Bengio 2010) and optimized by the Adam optimizer (Kingma and Ba 2015). For the optimizer, we set the initial learning rate during the range of [0.001, 0.01] and the weight decay within [0, 0.0001] for all datasets, respectively. We applly the ReLU function (Nair and Hinton 2010) as a nonlinear activation for each layer and conducted the row normalization on input features. Moreover, a dropout function is applied behind each layer. In addition, for node classification task, we follow the standard linear evaluation protocol in DGI.

4.2 Results and analysis

Node classification Tables 1 and 2 summarize classification accuracy and execution time of all methods on 8 real graph-structure datasets. First, SUGRL outperforms all self-supervised methods (*i.e.,* DGI, GMI, GRACE, MVGRL, GIC and GCA) in terms of classification accuracy. For example, our method on average improves by 4.0% and 1.9%, respectively, compared to the worst method DGI and the best comparison method MVGRL. Compared to semi-supervised methods (*i.e.,* GCN and GAT) which adopt the label information in the learning process, SUGRL also achieves the superior performance. Second, our SUGRL has the best efficiency. Specifically, compared with other self-supervised methods on 8 datasets, SUGRL is on average 122.4× and 4.4× faster, respectively, compared to the slowest comparison method GMI and the fastest comparison method GIC.

Analysis In conclusion, our method outperforms the other comparison methods on almost all datasets, in terms of model performance and execution time, on node classification. The reasons can be summarized as follows. First, SUGRL jointly considers structural information and neighbor information to generate two kinds of positive embeddings and their contrastive loss (*i.e.,* $\mathcal{L}_S$ and $\mathcal{L}_N$), which can push the negative embedding further away from the anchor embedding (*i.e.,* achieving large inter-class variation). Second, SUGRL employs an upper bound to ensure that the distance between positive embeddings and anchor embeddings is finite (*i.e.,* achieving small intra-class variation). Third, SUGRL removes the step of both data augmentation and discriminator, leading to a significant reduction of training time. Finally, SUGRL is available to output low-dimensional and high-quality embeddings as well as to reduce the training time while keeping the model effectiveness.

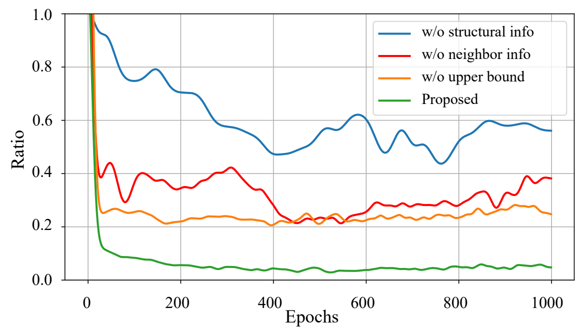


Figure 3: The ratio of intra-class variance to inter-class variation on the dataset Photo.

4.3 Ablation study

SUGRL considers three types of information, *i.e.,* semantic information, structural information and neighbor information, to generate two types of positive pairs with corresponding contrastive losses (*i.e.,* $\mathcal{L}_S$ and $\mathcal{L}_N$). To verify the effectiveness of each component of our framework, we investigate 1) the effectiveness of structural information, neighbor information and upper bound, respectively, and 2) the effectiveness of each component in our contrastive loss.

Effectiveness of the ratio of intra-class to inter-class variations. Considering different magnitudes between intra-class and inter-class variations, we follow (Li, Zhong, and Zheng 2019) to report the ratio of intra-class to inter-class variations on dataset Photo in Figure 3 by normalizing the ratio into [0, 1]. First, the method without structural or neighbor information generally outputs a larger ratio (*i.e.,* a smaller inter-class variation), compare to the proposed method. Second, the methods without upper bound (*i.e.,* w/o upper bound)

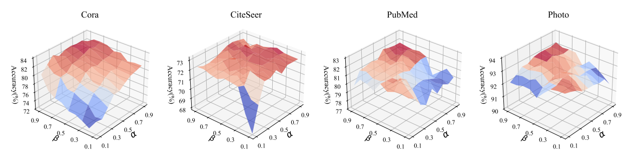


Figure 4: Classification results of our method at different parameter settings (*i.e.,* α and β).

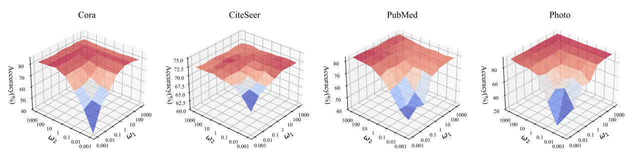


Figure 5: Classification results of our method at different parameter settings (*i.e.,* ω_1 and ω_2).

also outputs a larger ratio (*i.e.,* a larger intra-class variation), compared to our method. Thus, the effectiveness of either structural or neighbor information or upper bound is verified.

Effectiveness of each component in our contrastive loss. To analyze the effectiveness of each component (*i.e.,* $\mathcal{L}_S$, $\mathcal{L}_N$ and $\mathcal{L}_U$) on downstream tasks, we conduct experiments on all combinations of all components (except individual $\mathcal{L}_U$) for the node classification task since only using $\mathcal{L}_U$ cannot form the contrastive loss and thus cause model collapse. In Table 3, the last row using all components achieves the best performance as these components are complementary to each other. If an individual component (*i.e.,* either $\mathcal{L}_S$ or $\mathcal{L}_N$) is implemented, the performance is inferior. By contrast, the combination of any two components can help to learn better embeddings than that with only one component. Hence, the effectiveness of each component is verified.

4.4 Hyper-parameter analysis

We investigate the impact of hyper-parameters in SUGRL, *i.e.,* α and β in Eq. (11) as well as ω_1 and ω_2 in Eq. (12). We conduct node classification by varying the values of α and β from 0.1 to 0.9 and report the results in Figure 4. SUGRL achieves good performance while setting large values for α and β. If the values of α and β are too small, SUGRL obtains bad results as these scenarios result in a small margin between positive and negative pairs. We also conduct node classification by varying the values of ω_1 and ω_2 from 10^{-3} to 10^3, and fix the weight of $\mathcal{L}_U$ to 1, then report the results in Figure 5. The accuracy reduces if the values of ω_1 and ω_2 are too small (*e.g.,* 10^{-3} and 10^{-2}). This indicates that both $\mathcal{L}_S$ and $\mathcal{L}_N$ are important as they contribute to further push positive embeddings far away from negative embeddings.

4.5 Efficiency analysis

We divide the training process of all methods into three parts (*i.e.,* pretext, encoding and loss). Due to space limitations, we only report the time cost of each part on the dataset Photo.

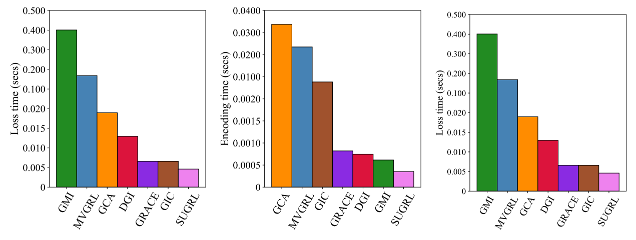


Figure 6: Execution time (seconds) of different parts in all methods on the dataset Photo.

Without data augmentation Previous works conduct data augmentation for contrastive learning while our SUGRL does not require it. Previous methods enquire generating new views as well as encoding them, while SUGRL gets diverse information by GCN and neighbors sampling. obviously, our method consumes the least time, as shown in the left sub-figure of Figure 6.

Fast encoding SUGRL generates the representations of anchors by the MLP and the representations of negative samples by row-shuffling the anchors, instead of using the GCN encoder. Moreover, our method generates the representations of positive samples containing neighbor information by neighbors sampling. Thus, SUGRL can employ fewer graph convolution operations and lead to fewer encoding time, compared to all comparison methods, as shown in the middle sub-figure of Figure 6.

Low-dimensional embeddings SUGRL obtains its best performance with a 128-dimensional embedding, while other methods generally need 512 dimensions to achieve their best performance. The dimension of embeddings has a significant impact on time costs, as shown in Figure 1.

Contrastive loss SUGRL does not need to design a discriminator for contrastive learning, but previous methods do need. Therefore, $\mathcal{L}$ in our method takes fewer time costs, as shown in the right sub-figure of Figure 6.

5 Conclusion

In this paper, we designed a simple framework, namely Simple Unsupervised Graph Representation Learning (SUGRL), to achieve effective and efficient contrastive learning. To obtain the effectiveness, we designed two triplet losses to explore complementary information between structural information and the neighbor information to enlarge the inter-class variation, as well as an upper band loss to reduce the intra-class variation. To attain the efficiency, our method was designed to remove the GCN for generating anchor and negative embeddings, as well as remove data augmentation and discriminator from previous graph contrastive learning. We conducted comprehensive experiments on various real-world datasets. Experimental results demonstrate that our method consistently outperforms state-of-the-art methods both in terms of both accuracy and scalability.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Grants No: 61836016 and 61876046) and the Guangxi "Bagui" Teams for Innovation and Research, China.

References

Belghazi, I.; Rajeswar, S.; Baratin, A.; Hjelm, R. D.; and Courville, A. C. 2018. MINE: Mutual Information Neural Estimation. *CoRR*, abs/1801.04062.

Cao, J.; Bu, Z.; Wang, Y.; Yang, H.; Jiang, J.; and Li, H. 2019. Detecting Prosumer-Community Groups in Smart Grids From the Multiagent Perspective. *IEEE Trans. Syst. Man Cybern. Syst.*, 49(8): 1652–1664.

Cao, J.; Lin, X.; Guo, S.; Liu, L.; Liu, T.; and Wang, B. 2021. Bipartite Graph Embedding via Mutual Information Maximization. In *WSDM*, 635–643.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020a. A simple framework for contrastive learning of visual representations. In *ICML*, 1597–1607.

Chen, X.; Fan, H.; Girshick, R. B.; and He, K. 2020b. Improved Baselines with Momentum Contrastive Learning. *CoRR*, abs/2003.04297.

Devon, R. H.; Alex, F.; Samuel, L.-M.; Karan, G.; Phil, B.; Adam, T.; and Yoshua, B. 2019. Learning deep representations by mutual information estimation and maximization. In *ICLR*.

Glorot, X.; and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 249–256.

Hassani, K.; and Khasahmadi, A. H. 2020. Contrastive multi-view representation learning on graphs. In *ICML*, 4116–4126.

Jiao, Y.; Xiong, Y.; Zhang, J.; Zhang, Y.; Zhang, T.; and Zhu, Y. 2020. Sub-graph Contrast for Scalable Self-Supervised Graph Representation Learning. In *ICDM*, 222–231.

Jin, M.; Zheng, Y.; Li, Y.; Gong, C.; Zhou, C.; and Pan, S. 2021. Multi-Scale Contrastive Siamese Networks for Self-Supervised Graph Representation Learning. In *IJCAI*, 1477–1483.

Jing, B.; Park, C.; and Tong, H. 2021. Hdmi: High-order deep multiplex infomax. In *WWW*, 2414–2424.

Kingma, P. D.; and Ba, L. J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Kipf, N. T.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Kipf, T. N.; and Welling, M. 2016. Variational Graph Auto-Encoders. *CoRR*, abs/1611.07308.

Li, W.-H.; Zhong, Z.; and Zheng, W.-S. 2019. One-pass person re-identification by sketch online discriminant analysis. *Pattern Recognition*, 93: 237–250.

Liu, H.; Li, X.; Li, J.; and Zhang, S. 2018. Efficient Outlier Detection for High-Dimensional Data. *IEEE Trans. Syst. Man Cybern. Syst.*, 48(12): 2451–2461.

Liu, H.; Liu, L.; Le, T. D.; Lee, I.; Sun, S.; and Li, J. 2017. Nonparametric Sparse Matrix Decomposition for Cross-View Dimensionality Reduction. *IEEE Trans. Multim.*, 19(8): 1848–1859.

Liu, S.; Xue, S.; Wu, J.; Zhou, C.; Yang, J.; Li, Z.; and Cao, J. 2021. Online Active Learning for Drifting Data Streams. *IEEE Trans. Neural Networks Learn. Syst.*

Mavromatis, C.; and Karypis, G. 2021. Graph InfoClust: Maximizing Coarse-Grain Mutual Information in Graphs. In *PAKDD*, 541–553.

Nair, V.; and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

Paninski, L. 2003. Estimation of entropy and mutual information. *Neural computation*, 15(6): 1191–1253.

Park, C.; Kim, D.; Han, J.; and Yu, H. 2020. Unsupervised Attributed Multiplex Network Embedding. In *AAAI*, 5371–5378.

Park, J.; Lee, M.; Chang, H. J.; Lee, K.; and Choi, J. Y. 2019. Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In *ICCV*, 6519–6528.

Peng, Z.; Huang, W.; Luo, M.; Zheng, Q.; Rong, Y.; Xu, T.; and Huang, J. 2020. Graph Representation Learning via Graphical Mutual Information Maximization. In *WWW*, 259–270.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*, 701–710.

Qiu, J.; Chen, Q.; Dong, Y.; Zhang, J.; Yang, H.; Ding, M.; Wang, K.; and Tang, J. 2020. GCC: Graph contrastive coding for graph neural network pre-training. In *SIGKDD*, 1150–1160.

Ren, Y.; Bai, J.; and Zhang, J. 2021. Label Contrastive Coding Based Graph Neural Network for Graph Classification. In *DASFAA*, volume 12681, 123–140.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of Graph Neural Network Evaluation. *CoRR*, abs/1811.05868.

Song, J.; Zhang, H.; Li, X.; Gao, L.; Wang, M.; and Hong, R. 2018. Self-Supervised Video Hashing With Hierarchical Binary Auto-Encoder. *IEEE Trans. Image Process.*, 27(7): 3210–3221.

Sun, F.-Y.; Hoffman, J.; Verma, V.; and Tang, J. 2019. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *ICLR*.

Suresh, S.; Li, P.; Hao, C.; and Neville, J. 2021. Adversarial Graph Augmentation to Improve Graph Contrastive Learning. *CoRR*, abs/2106.05819.

Tian, Y.; Krishnan, D.; and Isola, P. 2020. Contrastive Multi-view Coding. In *ECCV*, volume 12356, 776–794.

van den Oord, A.; Li, Y.; and Vinyals, O. 2018. Representation Learning with Contrastive Predictive Coding. *CoRR*, abs/1807.03748.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*.

Velickovic, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep Graph Infomax. In *ICLR*.

Weihua, H.; Matthias, F.; Marinka, Z.; Yuxiao, D.; Hongyu, R.; Bowen, L.; Michele, C.; and Jure, L. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *NeurIPS*.

Wen, Y.; Zhang, K.; Li, Z.; and Qiao, Y. 2016. A discriminative feature learning approach for deep face recognition. In *ECCV*, 499–515.

Xu, J.; Ren, Y.; Tang, H.; Pu, X.; Zhu, X.; Zeng, M.; and He, L. 2021a. Multi-VAE: Learning Disentangled View-Common and View-Peculiar Visual Representations for Multi-View Clustering. In *ICCV*, 9234–9243.

Xu, M.; Wang, H.; Ni, B.; Guo, H.; and Tang, J. 2021b. Self-supervised Graph-level Representation Learning with Local and Global Structure. In *ICML*, 11548–11558.

Xu, X.; Lu, H.; Song, J.; Yang, Y.; Shen, H. T.; and Li, X. 2020. Ternary Adversarial Networks With Self-Supervision for Zero-Shot Cross-Modal Retrieval. *IEEE Trans. Cybern.*, 50(6): 2400–2413.

Xu, X.; Wang, T.; Yang, Y.; Hanjalic, A.; and Shen, H. T. 2021c. Radial Graph Convolutional Network for Visual Question Generation. *IEEE Trans. Neural Networks Learn. Syst.*, 32(4): 1654–1667.

Xuan, X.; Peng, B.; Wang, W.; and Dong, J. 2019. On the Generalization of GAN Image Forensics. In *CCBR*, 134–141.

Yang, Z.; Cohen, W.; and Salakhudinov, R. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 40–48.

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

Yu, J.; Yin, H.; Gao, M.; Xia, X.; Zhang, X.; and Hung, N. Q. V. 2021. Socially-Aware Self-Supervised Tri-Training for Recommendation. In *KDD*, 2084–2092.

Zeng, H.; Zhou, H.; Srivastava, A.; Kannan, R.; and Prasanna, V. 2020. GraphSAINT: Graph Sampling Based Inductive Learning Method. In *ICLR*.

Zeng, J.; and Xie, P. 2021. Contrastive Self-supervised Learning for Graph Classification. In *AAAI*, 10824–10832.

Zhang, H.; Lin, S.; Liu, W.; Zhou, P.; Tang, J.; Liang, X.; and Xing, E. P. 2020. Iterative Graph Self-Distillation. *CoRR*, abs/2010.12609.

Zhou, X.; Shen, F.; Liu, L.; Liu, W.; Nie, L.; Yang, Y.; and Shen, H. T. 2020. Graph Convolutional Network Hashing. *IEEE Trans. Cybern.*, 50(4): 1460–1472.

Zhu, X.; Li, X.; Zhang, S.; Xu, Z.; Yu, L.; and Wang, C. 2017. Graph pca hashing for similarity search. *IEEE Trans. Multim.*, 19(9): 2033–2044.

Zhu, X.; Zhang, S.; Li, Y.; Zhang, J.; Yang, L.; and Fang, Y. 2019. Low-rank sparse subspace for spectral clustering. *IEEE Trans. Knowl. Data Eng.*, 31(8): 1532–1543.

Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; and Wang, L. 2020. Deep Graph Contrastive Representation Learning. *CoRR*, abs/2006.04131.

Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; and Wang, L. 2021. Graph Contrastive Learning with Adaptive Augmentation. In *WWW*, 2069–2080.