Unifying Visual Contrastive Learning for Object Recognition from a Graph Perspective

Shixiang Tang[1,3], Feng Zhu[3], Lei Bai[2,1,†], Rui Zhao[3,4], Chenyu Wang[1], and Wanli Ouyang[2,1]

[1] University of Sydney, Australia
[2] Shanghai AI Laboratory, China
[3] Sensetime Research
[4] Qing Yuan Research Institute, Shanghai Jiao Tong University
stan3906@uni.sydney.edu.au, baisanshi@gmail.com [†] corresponding author

Abstract. Recent contrastive based unsupervised object recognition methods leverage a Siamese architecture, which has two branches composed of a backbone, a projector layer, and an optional predictor layer in each branch. To learn the parameters of the backbone, existing methods have a similar projector layer design, while the major difference among them lies in the predictor layer. In this paper, we propose to Unify existing unsupervised Visual Contrastive Learning methods by using a GCN layer as the predictor layer (UniVCL), which deserves two merits to unsupervised learning in object recognition. First, by treating different designs of predictors in the existing methods as its special cases, our fair and comprehensive experiments reveal the critical importance of neighborhood aggregation in the GCN predictor. Second, by viewing the predictor from the graph perspective, we can bridge the vision self-supervised learning with the graph representation learning area, which facilitates us to introduce the augmentations from the graph representation learning to unsupervised object recognition and further improves the unsupervised object recognition accuracy. Extensive experiments on linear evaluation and the semi-supervised learning tasks demonstrate the effectiveness of UniVCL and the introduced graph augmentations.

1 Introduction

Self-supervised learning (SSL) [1, 36, 11, 23, 28, 62, 54, 50] has recently attracted much research interest in the computer vision community. Contrastive learning [20, 18, 7, 9, 5, 51, 65, 16, 59, 8], which is an important framework of recent unsupervised learning methods, aims to reduce the distance between augmented views from the same image (positive samples) and push apart views from different images (negative samples). It has shown the potential to extract powerful visual representations that are competitive with supervised learning and delivered superior performance on multiple visual tasks when models are pre-trained without labels.

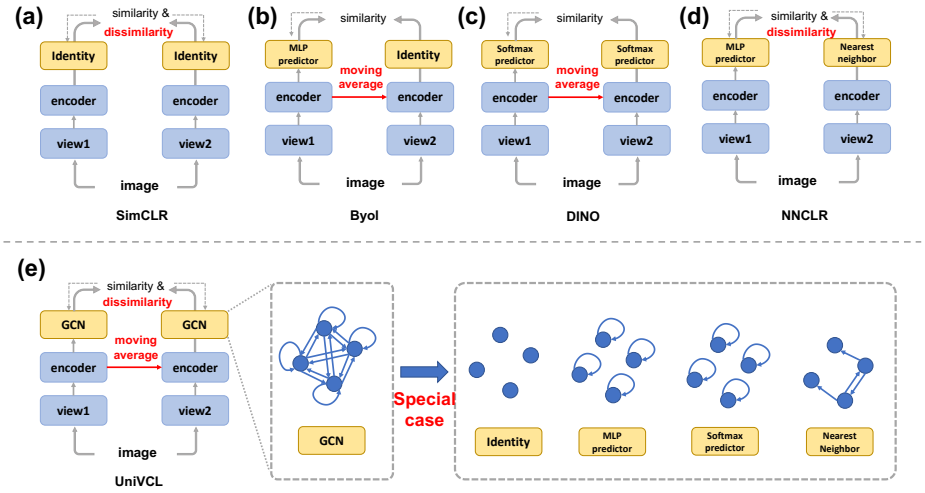

Fig. 1. (a-d): Existing self-supervised learning methods share the similar encoder designs, but have highly different designs in their predictors. The encoder in the picture includes a backbone and a projector. (e): Our UniVCL unifies different designs by a GCN layer, which has a neighborhood aggregation term and a self-loop term. The arrow denotes the aggregation operation. Specifically, the MLP predictor and Softmax predictor are the self-loop terms with different activation functions. The nearest neighbor retrieval can be viewed the neighborhood aggregation term in GCN layer.

Recent contrastive-based SSL methods leverage a Siamese architecture, which has two branches, i.e., an online branch and a target branch. Each branch is composed of a backbone, a projector layer, and an optional predictor layer. While the processes in the backbone and projector layer are similar for these works, there are major difference in their predictor layer designs as shown in Fig. 1(a-d). Specifically, there are four types of predictor layers.1) SimCLR [5], MOCOv1 [18], MOCOv2 [7] do not have any operation after the projector layer, which is mathematically equivalent to using an identity layer as the predictor. 2) BYOL [16], SimSiam [8], and MOCOv3 [9] use a MLP predictor on the online branch; 3) DINO [4], SEED [14], and TWIST [49] leverage a softmax layer on both online branch and target branch; 4) recent states-of-the-art methods, *e.g.,* NNCLR [13] and MSF [27], retrieve the K nearest neighboring samples (K-nearest neighbor layer) in the feature space on the target branch for contrastive learning. While these designs seem to be highly different in how to learn representations, we advocate that they can be viewed as a unified framework by simply modifying the predictor design from the perspective of graph neural network.

In this work, we propose a Unified Vison Contrastive Learning (UniVCL) framework that is a unified representation of the aforementioned four typical types of contrastive-based SSL methods from a graph perspective. By modeling the projection feature and its K nearest neighbors in the feature space as the

graph nodes, Graph Convolution Network (GCN) layer [24, 46, 55], which consists of a self-loop term and a neighborhood aggregation term, can formulate different predictor designs as its special variant (see the second row of Fig. 1). Specifically, the identity mapping, the MLP layer, and the softmax layer can be viewed as the self-loop term only with different activation functions, and the K-nearest neighbor retrieval can be viewed as the neighborhood aggregation term in the GCN layer. From this perspective, we benchmark different predictor designs in existing methods under the same learning schedules, data augmentations, and objective functions. Our detailed and fair experiments lead to three interesting observations. Firstly, the neighborhood aggregation term in the GCN layer can significantly improves the linear evaluation performances by **+2.1%**. Secondly, the activation function can fairly influence the linear evaluation performance. The non-linear activation function can boost about **+0.4%** performance gain than identity activation function. Thirdly, the performance difference between different non-linear activation functions are quite small if other GCN layer components are well-designed, indicating the non-linear activation function selection to be a less important factor.

With this unified framework from the graph perspective, we can further link vision SSL with graph SSL, another active SSL research topic, and explore the effectiveness of different pretext tasks in graph SSL for vision SSL. Specifically, the new data augmentation, i.e. graph augmentation, found to be effective in graph SSL can be leveraged for vision SSL. Graph augmentation adds variations to features according to the graph structure to regularize the network optimization. Our studies on ImageNet-1K uncover that the graph augmentation that uses message passing throughout the network can improve the self-supervised learning methods in image classification by **+1.0%**. Our in-depth analysis by the purity metric [27] verifies that these augmentations can add the edge noise in the GCN predictor, and regularize the encoder to learn more robustness image features.

To conclude, our contributions are three-folds: **(1)** We propose a general framework (UniVCL) to unify recent states-of-the-art contrastive learning methods in the vision SSL domain. **(2)** We illustrate the importance of neighborhood aggregation term and the non-linear activation function of GCN layer in UniVCL by conducting fair and detailed experimental comparisons. **(3)** Owing to graph design of UniVCL, we bridge vision SSL and graph SSL, and introduce typical graph augmentations into self-supervised image classification, which is empirically verified to be beneficial to linear evaluation and semi-supervised learning performances.

2 Related work

2.1 Self-supervised Learning on Vision

A typical unsupervised learning framework for contrastive learning consists of a Siamese network. The two branches of the Siamese network are named as the online branch and the target branch [16, 5] (also named as the query branch and

the key branch in some literature [18, 7, 13]), respectively, where each includes a backbone, a projector layer and an optional predictor layer. Recent states-of-the-art vision SSL methods share similar designs of backbone network and projector layer, but they differs in the predictor designs. Typical predictor designs can be divided into four categories. The first type of the predictor layer is the identity mapping, which is adopted in SimCLR [5], MOCOv2 [7], and the target branch of MOCOv3 [9], BYOL [16], and other self-supervised contrastive learning methods [59]. The second type is the MLP predictor. The MLP predictor is initially proposed to avoid network collapse by BYOL. Some following contrastive learning based papers, *e.g.,* MOCOv3, also append the predictor layer after the projector layer to improve the self-supervised learning performance instead of avoid network collapse. The third type of the predictor is the softmax layer, which is a softmax activation function applied in methods equipped with KL-divergence loss. The representative methods include DINO [4], SEED [14], and TWIST [49]. The recent states-of-the-art methods retrieve the k-nearest sample in the feature space. Experimental results illustrate the contrastive learning by using nearest samples as the positive sample can improve the linear evaluation compared with using the different augmented view of the same sample as its positive samples. Although these methods have highly different designs for predictors, our method unifies them with a Graph Convolution Layer, which can 1) represent lots of existing works and 2) include graph self-supervised learning designs (unification of self-loop and neighborhood aggregation and graph augmentation) that are not covered in these works.

2.2 Self-supervised Learning on Graph

Recent years witness the development of deep learning on graphs [58, 33, 21, 48, 25], since the graph-structured is ubiquitous in numerous domains, including e-commence [31], traffic [52], and knowledge base [32]. The biggest challenge of self-supervised learning on graphs lies in learning the topology information in the existing network. Contrastive learning methods are also cornerstones for self-supervised learning on graphs [41, 40, 47, 2, 37]. In particular, the contrastive loss uses two different augmented views of the same graph as the positive samples to maximize their mutual information. Typical augmentation methods include Node Feature Masking [21, 64], Edge Modification [22, 57], and Graph Diffusion [26, 17]. However, self-supervised Learning on graph has not been investigated for unsupervised object recognition. In this paper, by incorporating GCN layers as the predictor in the deep models for object recognition, we can introduce and verify the effectiveness of graph augmentations that are widely adopted in the graph self-supervised learning on the unsupervised object recognition.

3 UniVCL

We are interested in using the Graph Convolution Network (GCN) to unify different predictor designs in different methods. Specifically, we maintain a support

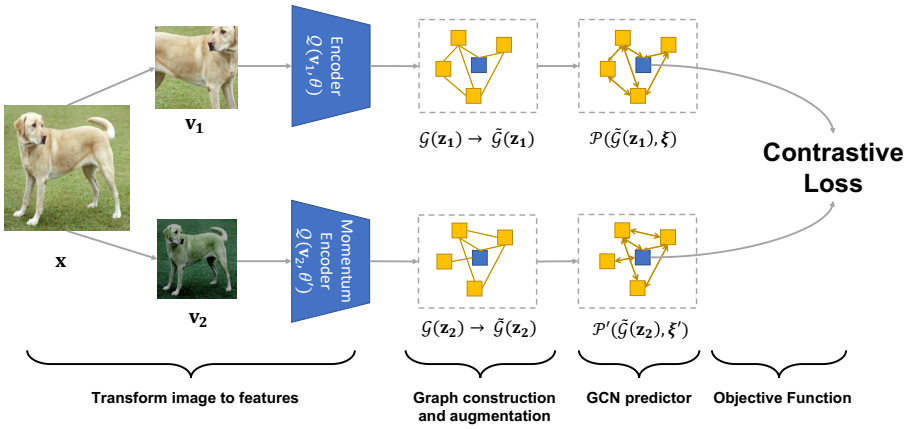

Fig. 2. The framework of UniVCL. It includes four steps. First, Given an image $\mathbf{x}$, two augmented views $\mathbf{v}_1$ and $\mathbf{v}$ are generated. Then the features $\mathbf{z}_1$ and $\mathbf{z}_2$ are extracted by encoder $\mathcal{Q}(*,\theta)$ and $\mathcal{Q}(*,\theta')$, respectively. Second, we retrieve the K nearest neighborhood samples of $\mathbf{z}_1$ and $\mathbf{z}_2$ from the support queue $\mathcal{S}$, forming graph $\mathcal{G}(\mathbf{z}_1)$ and $\mathcal{G}(\mathbf{z}_2)$ respectively. Then, we implement the graph augmentation on $\mathcal{G}(\mathbf{z}_1)$ and $\mathcal{G}(\mathbf{z}_2)$, generating augmented graphs $\widetilde{\mathcal{G}}(\mathbf{z}_1)$ and $\widetilde{\mathcal{G}}(\mathbf{z}_2)$. Third, we input the augmented graph into the GCN predictor layer, generating the predicted features $\mathbf{q}_1$ and $\mathbf{q}_2$. Last, we compute the alignment loss based on $\mathbf{q}_1$ and $\mathbf{q}_2$. The encoder denotes both the backbone network and the projector layer.

queue $\mathcal{S} = \{\mathbf{s}_i | i \in [1, \cdots, m]\} \in \mathbb{R}^{m \times d}$ in the same way as [13], where m is the size of the support queue, and d is the feature dimension. We only use embeddings from the target view to update the support set. As shown in Fig. 2, our proposed UniVCL has the following steps.

Step1: *Transform Image to Features.* Specifically, given two different augmented views $(\mathbf{v}_1, \mathbf{v}_2)$ of an image $\mathbf{x}$, the features of two views can be computed by $\mathbf{z}_1 = \mathcal{Q}(\mathbf{v}_1, \theta)$ and $\mathbf{z}_2 = \mathcal{Q}(\mathbf{v}_2, \theta')$, respectively. Following [18, 16, 7, 9], the online branch $\mathcal{Q}(*, \theta)$ is a neural network updated by backward propagation, while the target branch $\mathcal{Q}(*, \theta')$ is a network with the same architecture as the online branch but with parameters obtained from the moving average of $\mathcal{Q}(*, \theta)$.

Step2: Graph Construction and Augmentation (Sec. 3.3). Give $\mathbf{z}_1$ and $\mathbf{z_2}$ from Step 1, we respectively construct the fully connected graph $\mathcal{G}(\mathbf{z}_1)$ and $\mathcal{G}(\mathbf{z}_2)$, where nodes in $\mathcal{G}(\mathbf{z}_1)$ and $\mathcal{G}(\mathbf{z}_2)$ are K nearest neighbors of $\mathbf{z}_1$ and $\mathbf{z}_2$ in support queue $\mathcal{S}$, respectively. Then we implement typical graph augmentations (Sec. 3.3) to generate the augmented graphs $\widetilde{\mathcal{G}}(\mathbf{z}_1)$ and $\widetilde{\mathcal{G}}(\mathbf{z}_2)$.

Step 3: *GCN Predictor (Sec. 3.1).* The augmented graphs $\widetilde{\mathcal{G}}(\mathbf{z}_1)$ and $\widetilde{\mathcal{G}}(\mathbf{z}_2)$ are respectively transformed to prediction features $\mathbf{q}_1$ and $\mathbf{q}_2$ through GCN predictors $\mathcal{P}(*, \xi)$ and $\mathcal{P}'(*, \xi')$.

Step 4: *Backward propagation using the alignment loss.* Given the prediction features $\mathbf{q}_1$ and $\mathbf{q}_2$, the parameters θ in $\mathcal{Q}(*, \theta)$ and the parameters ξ, ξ' in

$\mathcal{P}(*,\xi), \mathcal{P}'(*,\xi')$ are learned by using alignment loss. In our design, the alignment loss is implemented as the contrastive, *i.e.,*

$$\mathcal{L} = -\log \frac{\exp(\mathbf{q}_1^\top \mathbf{q}_2)}{\exp(\mathbf{q}_1^\top \mathbf{q}_2) + \sum_{i=1}^{m} \exp(\mathbf{q}_1^\top \mathbf{s}_i)}, \tag{1}$$

where $\mathbf{s}_i$ is the feature stored in the support queue $\mathcal{S}$.

3.1 General Predictor Layers as GCNs

Since UniVCL appends the GCN predictor after the encoder $\mathcal{Q}$ and $\mathcal{Q}'$, we only analyze the GCN predictor $\mathcal{P}$ on the online branch. All analysis below is applicable to the GCN predictor $\mathcal{P}'$ on the target branch.

Given the feature $\mathbf{z}_1$ obtained by $\mathcal{Q}$, the input of the GCN layer is defined as

$$\mathcal{G}(\mathbf{z}_1) = (\mathbf{z}_1, \mathcal{N}_1(\mathbf{z}_1), \mathcal{N}_2(\mathbf{z}_1), ..., \mathcal{N}_K(\mathbf{z}_1)) \tag{2}$$

where K is the number of samples retrieved from $\mathcal{S}$, and $\mathcal{N}_i(\mathbf{z}_1)$ denotes the features of the i-th nearest neighbor in the support queue $\mathcal{S}$. The GCN predictor $\mathbf{q}_1 = \mathcal{P}(\mathcal{G}(\mathbf{z}_1))$ can be represented as a stack of graph convolution layers $\mathcal{F}_l$, where l is the layer index, *i.e.,* $\mathcal{P}(\mathcal{G}(\mathbf{z}_1), \theta) = \mathcal{F}_L(\mathcal{F}_{L-1} \cdots (\mathcal{F}_2(\mathcal{F}_1(\mathcal{G}(\mathbf{z}_1)))))$, where L is the number of stacked GCN layers. Here, $\mathcal{F}_l$ is presented as

$$\mathbf{F}_{l+1} = \mathcal{F}_l(\mathbf{F}_l) = \sigma_l(\underbrace{\mathbf{W}_l \mathbf{A} \mathbf{F}_l}_{\text{neighborhood aggregation}} + \underbrace{\mathbf{W}'_l \mathbf{F}_l}_{\text{self-loop}}), \tag{3}$$

where the affinities $\mathbf{A} = \{a_{i,j}\} \in \mathbb{R}^{(K+1)\times(K+1)}, 0 \le i,j \le K$ are defined as

$$a_{i,j} = \begin{cases} \mathcal{N}_i(\mathbf{z}_1)^\top \mathcal{N}_j(\mathbf{z}_1), & i \neq j, \\ 0, & i = j, \end{cases} \tag{4}$$

where $e_{i,j}$ is the affinity between $\mathcal{N}_i(\mathbf{z}_1)$ and $\mathcal{N}_j(\mathbf{z}_1)$, and we denote $\mathcal{N}_0(\mathbf{z}_1) = \mathbf{z}_1$.

3.2 Unifying unsupervised contrastive learning methods in UniVCL

As UniGrad [42] has explored the equivalence of different objective functions in the existing methods both theoretically and experimentally, we focus on the predictor designs among these different self-supervised learning methods.

As shown in Tab. 1 and Fig. 1(a-d), the predictor designs of different self-supervised learning methods can be categorized into four types: the identity predictor, the MLP predictor, the Softmax predictor, and the nearest neighbor predictor. Based on the the formal formulation of a graph convolution layer in Eq. 3, Tab. 2 shows that these different designs are special cases of Eq. 3. The detailed derivation for Tab. 2 is presented below.

The Identity Predictor (Fig. 1(a)). SimCLR [5] and MOCOv2 [7] do not append an explicit predictor after the projector, which is mathematically equivalent

Table 1. The implementation of predictor layer the existing self-supervised learning methods. We omit the comparison of objective functions in different methods because they are not the focus of our paper. The type number here denotes one of the four types described in Sec. 1 and Fig. 1(a-d).

Method	Venue	Type	Online Branch	Target Branch
MOCOv2	Arxiv'21	a	identity	identity
SimCLR	ICML'20	a	identity	identity
Barlow Twins	ICML'21	a	identity	identity
MOCOv3	ICCV'21	b	MLP	identity
BYOL	NeurIPS'20	b	MLP	identity
SimSiam	CVPR'21	b	MLP	identity
DINO	ICCV'21	c	softmax	softmax
SEED	ICLR'21	c	softmax	softmax
TWIST	Arxiv'21	c	softmax	softmax
NNCLR	ICCV'21	d	MLP	K nearest
MSF	ICCV'21	d	MLP	K nearest

Table 2. Illustrate the simplification to different predictor layers based the formal formulation of Graph Convolution predictor in Eq. 3.

	activation	neighborhood aggregation			self-loop	
Method	σ	existence	$\mathbf{A}$	$\mathbf{W}$	existence	$\mathbf{W}'$
Identity	$\mathbf{I}$	x	-	$\mathbf{0}$	✓	$\mathbf{I}$
Percepton	ReLU(BN)	x	-	$\mathbf{0}$	✓	train
fc	$\mathbf{I}$	x	-	$\mathbf{0}$	✓	train
softmax	Softmax	x	-	$\mathbf{0}$	✓	$\mathbf{I}$
nearest neighbors	$\mathbf{I}$	✓	$\mathbf{1}$	$\mathbf{I}$	x	$\mathbf{0}$

to appending an identity predictor. In this case, the predictor can be formulated as

$$\mathbf{F}_{l+1} = \mathbf{F}_l. \tag{5}$$

The formulation above can be obtained by setting $\sigma_l = \mathbf{I}$, $\mathbf{W}_l = \mathbf{0}$, $\mathbf{W}'_l = \mathbf{I}$ in Eq. 3 for our unified graph convolution predictor. In this case, the neighborhood aggregation term is ignored, and the self-loop term is exactly the identity mapping.

The MLP Predictors (Fig. 1(b)). Popular unsupervised learning methods such as MOCOv3 [9], BYOL [16] and Simsiam [5] use MLP layers as predictors. The typical MLP is a stack of *fc-bn-relu* layers (perception layer), where *fc-bn-relu* layer can be formulated as

$$\mathbf{F}_{l+1} = \mathrm{ReLU}(\mathrm{BN}(\mathbf{W}'_l\mathbf{F}_l)). \tag{6}$$

The *fc-relu-bn* above can be obtained by setting $\sigma_l = \mathrm{ReLU(BN)}$ in Eq. 3. In this case, the neighborhood aggregation term is also ignored and only the self-loop term is presented. Some unsupervised learning methods such as MOCOv3,

BYOL and Simsiam uses the fully-connected layer as the last layer in the constructed MLP predictor, which can be obtained by setting the activation function as the identity matrix, *i.e.,* $\sigma_l = \mathbf{I}$.

The Softmax Predictors (Fig. 1(c)). DINO presents a softmax predictor to obtain the logits for the following KL-divergence loss. The softmax operation can be presented as

$$\mathbf{F}_{l+1} = \text{Softmax}(\mathbf{W}_l' \mathbf{F}_l). \tag{7}$$

The softmax predictor above can be achieved by setting $\sigma_l = \text{Softmax}$ for the graph convolution predictor (Eq. 3). In this case, the neighborhood aggregation term is ignored and only the self-loop term is presented.

The K Nearest Neighbors Predictors (Fig. 1(d)). Recent states-of-the-art methods treats the sample and its K nearest neighbors in the feature space as the positive samples. Given $\mathbf{F}_l = (\mathbf{f}_l^0, \mathbf{f}_l^1, ..., \mathbf{f}_l^K)$. The K nearest neighbor predictor can be presented as

$$\begin{aligned}
\mathbf{f}_{l+1}^i &= \frac{1}{K}(\mathcal{N}_1(\mathbf{f}_l^i) + \mathcal{N}_2(\mathbf{f}_l^i) + ... + \mathcal{N}_K(\mathbf{f}_l^i)) \\
&= \frac{1}{K}(0, 1, 1, ..., 1)^\top (\mathbf{f}_l^i, \mathcal{N}_1(\mathbf{f}_l^i), \mathcal{N}_2(\mathbf{f}_l^i), ..., \mathcal{N}_K(\mathbf{f}_l^i)) \\
&= \frac{1}{K}(0, 1, 1, ..., 1)^\top (\mathbf{f}_l^0, \mathbf{f}_l^1, ..., \mathbf{f}_l^K) \\
&= \frac{1}{K}(0, 1, 1, ..., 1)^\top \mathbf{F}_l.
\end{aligned} \tag{8}$$

Therefore, the output of the l-th GCN predictor can be formulated as

$$\mathbf{F}_{l+1} = \frac{1}{K}(\mathbf{0}, \mathbf{1}, \mathbf{1}, ..., \mathbf{1})^\top (\mathbf{f}_{l+1}^1, \mathbf{f}_{l+1}^2, ..., \mathbf{f}_{l+1}^K) = \frac{1}{K}(\mathbf{0}, \mathbf{1}, \mathbf{1}, ..., \mathbf{1})^\top \mathbf{F}_{l+1}. \tag{9}$$

Compared with typical graph convolution layer (Eq. 3), the K-nearest-neighbor layer can be obtained by setting $\sigma_l = \mathbf{I}$ and the affinity matrix $\mathbf{A} = \frac{1}{K}(\mathbf{0}, \mathbf{1}, \mathbf{1}, ..., \mathbf{1})^\top \in \mathbb{R}^{(K+1)\times(K+1)}$. In this case, only neighborhood aggregation term is considered and the self-loop term is ignored.

3.3 Graph Augmentations for Unsupervised Visual Learning

To better take advantage of pretext tasks in graph contrastive learning, the proposed GCN predictor layer leverages an augmented graph $\widetilde{\mathcal{G}}(\mathbf{z}_1)$ as the input [37, 63, 2, 47]. Given an input graph $\mathcal{G}(\mathbf{z})$ defined by Eq. 2, we implement the typical graph augmentations in self-supervised graph contrastive learning on $\mathcal{G}(\mathbf{z}_1)$, achieving $\widetilde{\mathcal{G}}(\mathbf{z}_1) = (\widetilde{\mathcal{V}}, \widetilde{\mathbf{A}}) = (t(\mathcal{V}), s(\mathbf{A}))$, where t and s are augmentations on the node features $\mathcal{V}$ and affinities $\mathbf{A}$, respectively. After graph augmentation, we will change the input node features and affinity as $\widetilde{\mathcal{V}}$ and $\widetilde{\mathbf{A}}$ in Eq. 3 as the input of the GCN predictor.

Node feature masking (NFM). As shown in Fig. 3, Node Feature Masking (NFM) randomly masks the features of a portion of nodes within $\mathcal{G}(\mathbf{z})$. In particular, we can completely mask seleceted feature vectors with zeros, or partially

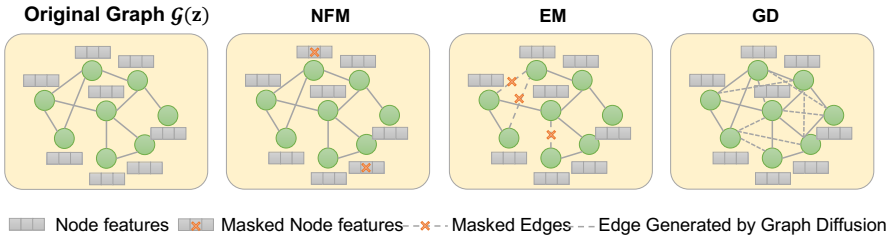

Fig. 3. Graph augmentations. There are three typical graph augmentations, *i.e.,* node feature masking (NFM), edge modification (EM), and graph diffusion (GD).

mask a number of selected feature channels with zeros. This operation can be formulated as

$$\widetilde{\mathcal{V}} = t(\mathcal{V}) = \mathbf{M}_f \circ \mathcal{V}, \quad \widetilde{\mathbf{A}} = s(\mathbf{A}) = \mathbf{A}, \tag{10}$$

where $\mathbf{M}_f$ is the feature masking matrix with the same shape of $\mathcal{V}$, and $\circ$ denotes the Hadamard (element-wise) product. The elements in $\mathbf{M}_f$ are initialized to one and masking entries are 30% randomly assigned to zero.

Edge modification (EM). Edge modification (EM) randomly drops the affinities, which means setting the affinities to zeros. This process is formulated as

$$\widetilde{\mathcal{V}} = t(\mathcal{V}) = \mathcal{V}, \quad \widetilde{\mathbf{A}} = s(\mathbf{A}) = \mathbf{M}_e \circ \mathbf{A}, \tag{11}$$

where $\mathbf{M}_e$ is the edge dropping matrix, and $\circ$ denotes the Hadamard product.

Graph diffusion (GD). Graph diffusion is also a type of affinity augmentations, which injects the global affinity information to the given affinity by recomputing the affinity with diffusion operations. The overall diffusion operation can be formulated as

$$\widetilde{\mathcal{V}} = t(\mathcal{V}) = \mathcal{V}, \quad \widetilde{\mathbf{A}} = s(\mathbf{A}) = \sum_{n=0}^{\infty} \Theta_n \mathbf{T}^n, \tag{12}$$

where Θ_n and $\mathbf{T}$ are weighing coefficient and transition matrix, respectively. The diffusion above have two common instantiations: heat diffusion [26, 43] and PPR diffusion [12, 15]. The heat diffusion formulates $\Theta_n = \frac{\exp(-\eta)t^n}{n!}$, and $\mathbf{T} = \mathbf{A}\mathbf{D}^{-1}$, achieving $\widetilde{\mathbf{A}} = \exp(\eta\mathbf{A}\mathbf{D}^{-1} - \eta)\mathbf{A}$, where $\mathbf{A}$ is the affinity matrix, $\mathbf{D}$ is the diagonal degree matrix, $\eta \in (0, 1)$ is the diffusion time. The PPR diffusion formulates $\Theta_n = \gamma(1-\gamma)^n$, and $\mathbf{T} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, achieving $\widetilde{\mathbf{A}} = \gamma(\mathbf{I} - (1-\gamma)\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{A}$, where $\gamma \in (0, 1)$ is the teleport probability in random walk.

4 Experiment

4.1 Implementation Details

Architecture. Our architecture is similar to MOCOv2. Specifically, we use ResNet-50 as our backbone following the common implementations in the self-supervised literature. We spatially average pool the output of ResNet-50 which

Table 3. Ablation of the importance of parameters in the GCN predictor. $\mathbf{I}^*$ denotes the centering operation proposed in DINO which is used to avoid network collapse. "train" denotes the trainable parameters. "ema" denotes exponential moving average of the parameters in the online branch.

Method	Online Branch				Target Branch				Linear
	Act.	Neigh. Agg.		Self-loop	Act.	Neigh. Agg.		Self-loop	
	σ	$\mathbf{A}$	$\mathbf{W}$	$\mathbf{W}'$	σ	$\mathbf{A}$	$\mathbf{W}$	$\mathbf{W}'$	
Baseline									
MOCOv2	$\mathbf{I}$	-	$\mathbf{0}$	$\mathbf{I}$	$\mathbf{I}$	-	$\mathbf{0}$	$\mathbf{I}$	68.6
Effectiveness of Activation Function									
Exp.(a)	$\mathbf{I}$	-	$\mathbf{0}$	train	$\mathbf{I}$	-	$\mathbf{0}$	$\mathbf{I}$	68.5
Exp.(b)	Re(BN)	-	$\mathbf{0}$	train	$\mathbf{I}$	-	$\mathbf{0}$	$\mathbf{I}$	69.3
Exp.(c)	Re(BN)	-	-	$\mathbf{I}$	$\mathbf{I}$	-	-	$\mathbf{I}$	67.5
Exp.(d)	Softmax	-	$\mathbf{0}$	$\mathbf{I}$	Softmax	-	$\mathbf{0}$	$\mathbf{I}$	69.3
Exp.(e)	Softmax	-	$\mathbf{0}$	train	Softmax	-	$\mathbf{0}$	$\mathbf{I}^*$	69.4
Effectiveness of neighborhood aggregation									
Exp.(f)	Re(BN)	-	$\mathbf{0}$	train	$\mathbf{I}$	$\mathbf{1}$	$\mathbf{I}$	$\mathbf{0}$	71.4
Exp.(g)	Re(BN)	-	$\mathbf{0}$	train	$\mathbf{I}$	$\mathbf{1}$	$\mathbf{I}$	$\mathbf{I}$	71.8
Exp.(h)	Re(BN)	$\mathbf{1}$	train	train	$\mathbf{I}$	$\mathbf{1}$	$\mathbf{I}$	ema	71.9

makes the output of the feature transformation a 2048-dimensional embedding. The projection layer is composed of 3 fully connected layers having output sizes $[2048, 4096, d]$, where d is the feature dimension applied in the loss and $d = 2048$ if not specified. Besides, batch normalization and ReLU activation function is employed in the projection layer following other SSL works [9, 8, 16]. The architecture of the predictor is the GCN layers, which is formulated in Eq. 3.

Training. For a fair comparison, we train our method on the ImageNet2012 dataset, which contains 1,281,167 images without using any annotation or class labels. In the training stage, we train for 200 and 800 epochs with a warm-up of 10 epochs and cosine annealing schedule using the LARS optimizer. The base learning rate is set to 0.3. Weight decay of 10^{-6} is applied during training. As is common practice, we do not use weight decay on the bias. The training details above are the same as MOCOv2. We also use the basic data augmentation scheme (i.e., random crop, color jittering) as MOCOv2 and do not include the multi-crop strategy [3] for a fair comparison with the most majority of works.

4.2 Ablation study

Exploring the critical factors in GCN predictor. Previous self-supervised learning methods have different predictor designs in activation functions, and the using of neighborhood aggregation. For example, MOCOv2 [7] and SimCLR [5, 6] uses the linear activation, MOCOv3 [9] and BYOL [16] use the ReLU(BN) as the activation function in the online branch, DINO [4] uses Softmax layer as the activation function. Furthermore, the recent state-of-the-art methods, *i.e.,* MSF and NNCLR, retrieve the nearest neighbors in the feature space in the target

branch, which can be mathematically viewed as the neighborhood aggregation as analyzed in Tab. 1. To strictly ablate the importance of different components in the GCN predictor layer, we keep the batch size, objective function, learning rate schedule, optimizer exactly the same, and then train the ImageNet-1K for 200 epochs. For the MLP predictor, we stack three GCN layers as the common practice with ReLU(BN) being its activation function except the last GCN layer.

Effectiveness of activation function σ. We have three findings from Tab.. 3. First, comparing Exp. (a) and Exp. (b,e), we can see that with a learnable transformation matrix $\mathbf{W}'$, the non-linear activation is better than the identity activation by about +0.9%, which is consistent with the finding in MOCOv3 [9]. Second, the trainable transformation $\mathbf{W}'$ plays an important role when the activation function is ReLU(BN), but plays an unimportant role when the activation function is Softmax and Identity mapping. We consider the difference may result from the information loss of ReLU. Third, comparing Exp. (b) and Exp. (e), we find the performance between the GCN layer with different activation functions are quite small if other components are well-designed, which indicates the activation function is a less important factor.

Effectiveness of neighborhood aggregation. Different from self-supervised learning methods that do not use supervision from other samples, neighborhood based methods uses K nearest neighbors as their positive samples. This can be achieved by using the neighborhood aggregation term in Eq. 3 in GCN predictor. As shown in Tab. 3, we have three findings. First, the linear evaluation performances of using neighboring information on the target branch are significantly higher than those self-supervised learning methods by a considerable 2.1% gain by comparing Exp.(b) and Exp. (f) . Second, comparing Exp.(f) and Exp.(g), we can see adding the self-loop term in the target branch can only boost the performance by 0.4%. Third, when adding the neighborhood aggregation and self-loop term in both online branch and the target branch, the performance can be further improved by 0.1%, which is not significant by comparing Exp.(g) and Exp.(h).

Evaluating graph augmentation in unsupervised image classification. Owing to the GCN predictor, we can naturally bridge the vision SSL with graph SSL, which benefit us to introduce the graph augmentations on the constructed graph $\mathcal{G}(\mathbf{z})$ before the GCN predictor. Refer to Tab. 3, we use the Exp (h) as the baseline (using a GCN predictor in both online branch and target branch) in this part and extend it with diverse graph augmentations. Specifically, we explore the effectiveness of three common graph augmentations in the following.

Node feature masking (NFM) and Edge masking (EM). According to the mechanism of NFM and EM described in [33], we randomly remove the node features or edges in both $\mathcal{G}(\mathbf{z})$ and $\mathcal{G}(\mathbf{z}')$ with different drop probabilities. As shown in Tab. 4, the performance first increases from 71.9 to 72.1 with NFM and 72.2 with EM, respectively, when we increase the dropping probability from 0 to 0.3. Further increasing the drop probability (e.g., to 0.7) will harm the performance. The results demonstrate that both node and edge masking graph augmentations are beneficial for vision SSL with a proper drop probability.

drop probability	NFM	EM
0	71.9	71.9
0.1	72.1	72.0
0.3	72.1	**72.2**
0.5	71.3	71.7
0.7	70.1	70.5

Table 4. Ablation study of node feature masking and edge modification with different drop probabilities.

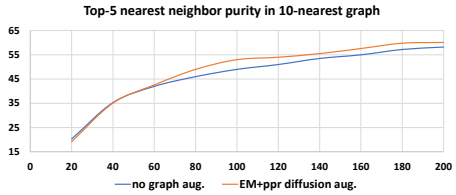

Fig. 4. Top-5 neighbor purity evolution by graph augmentations.

Table 5. Ablation study of different graph diffusion methods on the online branch and the target branch.

Method	online branch	target branch	Linear eval
Exp. (i)	No	No	72.2
Exp. (j)	No	Heat Diffusion	72.6
Exp. (k)	No	PPR Diffusion	72.8
Exp. (l)	Heat Diffusion	Heat Diffusion	72.4
Exp. (m)	PPR Diffusion	PPR Diffusion	72.5
Exp. (n)	Heat Diffusion	PPR Diffusion	**72.9**
Exp. (o)	PPR Diffusion	Heat Diffusion	72.8

Graph diffusion (GD). The graph diffusion propagates the global information in the graph to affinities by diffusion. Based on the results about masking-based augmentations, we further integrate the diffusion-based augmentations with EM (drop probability is 0.3) and explore the influence of heat diffusion and PPR diffusion on both online and target branches. The experimental results are presented in Tab. 5. We can observe that using graph diffusion to incorporate the graph information can significantly improve the the performance baseline by 0.9%. In detail, both heat diffusion and PPR diffusion can benefit vision SSL above the EM augmentation. Besides, using graph diffusion in both branches is more powerful than only using the diffusion in the target branch.

Analysis. In this section, we explain why the graph diffusion operation can improve the unsupervised learning performance empirically. We find the graph diffusion operation can correct the affinity of some visually different but semantically same samples in $\mathcal{G}(\mathbf{z})$. To better illustrate this, we utilize the setting in Exp.(e). Inspired by [39], we compare the **top-5** purity in different epochs between the original 10-nearest graph $\mathcal{G}(\mathbf{z}_2)$ and the 10-nearest augmented graph $\widetilde{\mathcal{G}}(\mathbf{z}_2)$. The purity for a single feature $\mathbf{z}$ is the percentage of $\mathcal{N}_1$ to $\mathcal{N}_K$ in the top-K nearest neighbors which have the same class as $\mathbf{z}$. Final purity is calculated by averaging the purities of all samples. The results are presented in Fig. 4. We find the purity of top-5 nearest neighbor in the augmented graph is higher than that in the original graph. By using affinity as the aggregation weight in GCN layer, we can conclude that the features can be aggregated with more accurate neighbors by using graph augmentations and therefore provided better target predictions for the online branch to learn.

Table 6. Comparison with other self-supervised learning methods under the linear evaluation protocol [18] on ImageNet. We omit the result for SwAV with multi-crop for fair comparion with other methods.

Method	Architecture	epochs	Top1	Top5
ODC [61]	ResNet-50	100	57.6	-
InstDisc [51]	ResNet-50	200	58.5	-
LocalAgg [65]	ResNet-50	200	58.8	-
MOCOv2 [7]	ResNet-50	200	68.6	-
MSF [39]	ResNet-50	200	71.4	-
MSF w/s [39]	ResNet-50	200	72.4	-
CPC v2 [19]	ResNet-50	200	63.8	85.3
DINO [19]	VIT-S/16	300	72.5	-
CMC [44]	ResNet-50	240	66.2	87.0
Adco [36]	ResNet-50	200	68.6	-
NNCLR [13]	ResNet-50	200	70.7	-
UniVCL	ResNet-50	200	72.9	-
PIRL [34]	ResNet-50	800	63.6	-
MOCOv2 [7]	ResNet-50	800	71.1	-
SimSiam [8]	ResNet-50	800	71.3	90.7
SimCLR [5]	ResNet-50	800	69.3	89.0
SwAV [3]	ResNet-50	800	71.8	-
InfoMin Aug. [45]	ResNet-50	800	73.0	91.1
BYOL [16]	ResNet-50	1000	74.3	91.6
Adco [36]	ResNet-50	800	72.8	-
Barlow Twins [59]	ResNet-50	1000	73.2	91.0
MoCov3 [9]	ResNet-50	800	73.8	-
NNCLR [13]	ResNet-50	800	75.4	92.4
UniVCL	ResNet-50	800	75.7	93.1

4.3 Comparison with State-of-the-art Methods

In this section, we utilize the optimal hyperparameters explored in the previous sections. Specifically, we apply the edge masking, followed by heat diffusion on the online brach and PPR diffusion on the target branch. For GCN predictor, we apply the setting in Exp. (h), where we add a GCN predictor in both online branch and target branch, and the parameters of GCN layers in the target branch are updated from the online branch in a momentum update manner. The results of transfer experiments are presented in supplementary materials.

Linear evaluations Following the standard linear evaluation protocol [51, 65, 18, 7], we train a linear classifier for 90 epochs on the frozen 2048-dimensional embeddings from the ResNet-50 encoder using LARS [56] with cosine annealed learning rate of 1 with Nesterov momentum of 0.9 and batch size of 4096. Comparison with state-of-the-art methods is presented in Tab. 6. Firstly, UniVCL achieves better performance compared to the state-of-the-art methods, using a ResNet-50 backbone without multi-crops augmentations. In 200 epochs training setting, UniVCL improves MOCOv2 by 4.3%, which uses the identity layer in both online branch and target branch. UniVCL still improves the DINO by 0.4% although standard DINO [4] leverages more powerful backbone (VIT-S/16) and more training epochs (300 epochs). The significant improvements by UniVCL verifies the significance of our GCN predictor in the unsupervised vision contrastive learning. Secondly, MSF and NNCLR also leverage the neighboring information, but not in a GCN way. The results of our UniVCL is also higher

Table 7. Comparison with the state-of-the-art methods for semi-supervised learning. Pseudo Label, UDA, FixMatch and MPL are semi-supervised learning methods. † denotes using random augment [10]. We follow the exact data split in SwAV [3].

Method	ImageNet 1%		ImageNet 10%	
	Top1	Top5	Top1	Top5
Supervised baseline [60]	25.4	48.4	56.4	80.4
Pseudo label [29]	-	-	51.6	82.4
UDA [53]	-	-	68.8†	88.5†
FixMatch [38]	-	-	71.5†	89.1†
MPL [35]	-	73.5†	-	-
InstDisc [51]	-	39.2	-	77.4
PIRL [34]	-	57.2	-	83.8
PCL [30]	-	75.6	-	86.2
SimCLR [5]	48.3	75.5	65.6	87.8
BYOL [16]	53.2	78.4	68.8	89.0
SwAV (multicrop) [3]	53.9	78.5	70.2	89.9
Barlow Twins [59]	55.0	79.2	69.7	89.3
NNCLR	56.4	80.7	69.8	89.3
UniVCL	58.6	81.8	71.8	91.4

than MSF and NNCLR [13] by 0.5% and 2.2%, respectively, because of grpah formulation and the introducing of graph augmentations from the graph SSL domain with negligible additional computational cost (less than 2%).

Semi-Supervised Learning Evaluations. We conduct experiments in a semi-supervised setting on ImageNet following the standard evaluation protocol [6, 5], which fine-tunes the whole base network on 1% or 10% labeled ImageNet data without regularization after unsupervised pre-training. The experimental results are presented in Tab. 7.

5 Conclusions and Discussions

In this paper, we unify the recent state-of-the-art methods in our proposed UniVCL. Specifically, we propose the GCN predictor to unify the diverse structural designs of predictor layers in various self-supervised learning methods. Then, fairly and comprehensively experiments are conducted to explore the critical factors in the GCN predictor, revealing the key point of a good predictor is to aggregate neighboring information in the feature space. Owing to the graph perspective, we further verify the effectiveness of graph augmentations in the vision contrastive learning. In the future, we will extend UniVCL from two perspectives, 1) further link the graph self-supervised learning and vision self-supervised learning by exploring other non-contrastive frameworks with graph self-supervised learning, such as reconstruction, attribute prediction, and 2) validating the effectiveness on other vision tasks, *e.g.,* detection, segmentation.

Acknowledgement: This work was supported by the Australian Research Council Grant DP200103223, Australian Medical Research Future Fund MRFAI000085, CRC-P Smart Material Recovery Facility (SMRF) – Curby Soft Plastics, and CRC-P ARIA - Bionic Visual-Spatial Prosthesis for the Blind.

References

1. Agrawal, P., Carreira, J., Malik, J.: Learning to see by moving. In: Proceedings of the IEEE international conference on computer vision. pp. 37–45 (2015)
2. Cao, J., Lin, X., Guo, S., Liu, L., Liu, T., Wang, B.: Bipartite graph embedding via mutual information maximization. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining. pp. 635–643 (2021)
3. Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Unsupervised learning of visual features by contrasting cluster assignments. arXiv preprint arXiv:2006.09882 (2020)
4. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9650–9660 (2021)
5. 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)
6. Chen, T., Kornblith, S., Swersky, K., Norouzi, M., Hinton, G.: Big self-supervised models are strong semi-supervised learners. arXiv preprint arXiv:2006.10029 (2020)
7. Chen, X., Fan, H., Girshick, R., He, K.: Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020)
8. Chen, X., He, K.: Exploring simple siamese representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15750–15758 (2021)
9. Chen, X., Xie, S., He, K.: An empirical study of training self-supervised vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9640–9649 (2021)
10. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 702–703 (2020)
11. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: Proceedings of the IEEE international conference on computer vision. pp. 1422–1430 (2015)
12. Donoser, M., Bischof, H.: Diffusion processes for retrieval revisited. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1320–1327 (2013)
13. Dwibedi, D., Aytar, Y., Tompson, J., Sermanet, P., Zisserman, A.: With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9588–9597 (2021)
14. Fang, Z., Wang, J., Wang, L., Zhang, L., Yang, Y., Liu, Z.: Seed: Self-supervised distillation for visual representation. arXiv preprint arXiv:2101.04731 (2021)
15. Gao, Y., Yu, X., Zhang, H.: Graph clustering using triangle-aware measures in large networks. Information Sciences **584**, 618–632 (2022)
16. 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)
17. Hassani, K., Khasahmadi, A.H.: Contrastive multi-view representation learning on graphs. In: International Conference on Machine Learning. pp. 4116–4126. PMLR (2020)

18. 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)
19. Henaff, O.: Data-efficient image recognition with contrastive predictive coding. In: International Conference on Machine Learning. pp. 4182–4192. PMLR (2020)
20. Hu, Q., Wang, X., Hu, W., Qi, G.J.: Adco: Adversarial contrast for efficient learning of unsupervised representations from self-trained negative adversaries. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1074–1083 (2021)
21. Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., Leskovec, J.: Strategies for pre-training graph neural networks. arXiv preprint arXiv:1905.12265 (2019)
22. Hu, Z., Dong, Y., Wang, K., Chang, K.W., Sun, Y.: Gpt-gnn: Generative pre-training of graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1857–1867 (2020)
23. Kim, D., Cho, D., Yoo, D., Kweon, I.S.: Learning image representations by completing damaged jigsaw puzzles. In: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 793–802. IEEE (2018)
24. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
25. Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016)
26. Klicpera, J., Weißenberger, S., Günnemann, S.: Diffusion improves graph learning. Advances in Neural Information Processing Systems **32** (2019)
27. Koohpayegani, S.A., Tejankar, A., Pirsiavash, H.: Mean shift for self-supervised learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10326–10335 (2021)
28. Larsson, G., Maire, M., Shakhnarovich, G.: Learning representations for automatic colorization. In: European conference on computer vision. pp. 577–593. Springer (2016)
29. Lee, D.H., et al.: Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In: Workshop on challenges in representation learning, ICML. vol. 3, p. 896 (2013)
30. Li, J., Zhou, P., Xiong, C., Socher, R., Hoi, S.C.: Prototypical contrastive learning of unsupervised representations. arXiv preprint arXiv:2005.04966 (2020)
31. Li, Z., Shen, X., Jiao, Y., Pan, X., Zou, P., Meng, X., Yao, C., Bu, J.: Hierarchical bipartite graph neural networks: Towards large-scale e-commerce applications. In: 2020 IEEE 36th International Conference on Data Engineering (ICDE). pp. 1677–1688. IEEE (2020)
32. Liu, Q., Allamanis, M., Brockschmidt, M., Gaunt, A.: Constrained graph variational autoencoders for molecule design. Advances in neural information processing systems **31** (2018)
33. Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., Tang, J.: Self-supervised learning: Generative or contrastive. IEEE Transactions on Knowledge and Data Engineering (2021)
34. Misra, I., Maaten, L.v.d.: Self-supervised learning of pretext-invariant representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6707–6717 (2020)
35. Pham, H., Dai, Z., Xie, Q., Le, Q.V.: Meta pseudo labels. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11557–11568 (2021)

36. Qi, G.J., Zhang, L., Lin, F., Wang, X.: Learning generalized transformation equivariant representations via autoencoding transformations. IEEE Transactions on Pattern Analysis and Machine Intelligence (2020)
37. Robinson, J., Chuang, C.Y., Sra, S., Jegelka, S.: Contrastive learning with hard negative samples. arXiv preprint arXiv:2010.04592 (2020)
38. Sohn, K., Berthelot, D., Li, C.L., Zhang, Z., Carlini, N., Cubuk, E.D., Kurakin, A., Zhang, H., Raffel, C.: Fixmatch: Simplifying semi-supervised learning with consistency and confidence. arXiv preprint arXiv:2001.07685 (2020)
39. Soroush Abbasi, K., Tejankar, A., Pirsiavash, H.: Mean shift for self-supervised learning. In: International Conference on Computer Vision (ICCV) (2021)
40. Subramonian, A.: Motif-driven contrastive learning of graph representations. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 15980–15981 (2021)
41. Sun, Q., Li, J., Peng, H., Wu, J., Ning, Y., Yu, P.S., He, L.: Sugar: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism. In: Proceedings of the Web Conference 2021. pp. 2081–2091 (2021)
42. Tao, C., Wang, H., Zhu, X., Dong, J., Song, S., Huang, G., Dai, J.: Exploring the equivalence of siamese self-supervised learning via a unified gradient framework. arXiv preprint arXiv:2112.05141 (2021)
43. Thanou, D., Dong, X., Kressner, D., Frossard, P.: Learning heat diffusion graphs. IEEE Transactions on Signal and Information Processing over Networks **3**(3), 484–499 (2017)
44. Tian, Y., Krishnan, D., Isola, P.: Contrastive multiview coding. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16. pp. 776–794. Springer (2020)
45. Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., Isola, P.: What makes for good views for contrastive learning. arXiv preprint arXiv:2005.10243 (2020)
46. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
47. Wang, C., Liu, Z.: Learning graph representation by aggregating subgraphs via mutual information maximization. arXiv preprint arXiv:2103.13125 (2021)
48. Wang, C., Pan, S., Long, G., Zhu, X., Jiang, J.: Mgae: Marginalized graph autoencoder for graph clustering. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. pp. 889–898 (2017)
49. Wang, F., Kong, T., Zhang, R., Liu, H., Li, H.: Self-supervised learning by estimating twin class distributions. arXiv preprint arXiv:2110.07402 (2021)
50. Wei, C., Wang, H., Shen, W., Yuille, A.: Co2: Consistent contrast for unsupervised visual representation learning. arXiv preprint arXiv:2010.02217 (2020)
51. Wu, Z., Xiong, Y., Yu, S.X., Lin, D.: Unsupervised feature learning via non-parametric instance discrimination. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3733–3742 (2018)
52. Wu, Z., Pan, S., Long, G., Jiang, J., Zhang, C.: Graph wavenet for deep spatial-temporal graph modeling. arXiv preprint arXiv:1906.00121 (2019)
53. Xie, Q., Dai, Z., Hovy, E., Luong, M.T., Le, Q.V.: Unsupervised data augmentation for consistency training. arXiv preprint arXiv:1904.12848 (2019)
54. Xie, Z., Lin, Y., Zhang, Z., Cao, Y., Lin, S., Hu, H.: Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16684–16693 (2021)
55. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)

56. You, Y., Gitman, I., Ginsburg, B.: Large batch training of convolutional networks. arXiv preprint arXiv:1708.03888 (2017)
57. 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)
58. You, Y., Chen, T., Wang, Z., Shen, Y.: When does self-supervision help graph convolutional networks? In: international conference on machine learning. pp. 10871–10880. PMLR (2020)
59. 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)
60. Zhai, X., Oliver, A., Kolesnikov, A., Beyer, L.: S4l: Self-supervised semi-supervised learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1476–1485 (2019)
61. Zhan, X., Xie, J., Liu, Z., Ong, Y.S., Loy, C.C.: Online deep clustering for unsupervised representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6688–6697 (2020)
62. Zhang, L., Qi, G.J., Wang, L., Luo, J.: Aet vs. aed: Unsupervised representation learning by auto-encoding transformations rather than data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2547–2555 (2019)
63. Zhu, Q., Yang, C., Xu, Y., Wang, H., Zhang, C., Han, J.: Transfer learning of graph neural networks with ego-graph information maximization. Advances in Neural Information Processing Systems **34** (2021)
64. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Deep graph contrastive representation learning. arXiv preprint arXiv:2006.04131 (2020)
65. Zhuang, C., Zhai, A.L., Yamins, D.: Local aggregation for unsupervised learning of visual embeddings. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6002–6012 (2019)