Pattern Recognition 146 (2024) 110065

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Deep image clustering with contrastive learning and multi-scale graph convolutional networks

Yuankun Xu [a], Dong Huang [a,b,*,1], Chang-Dong Wang [c,d], Jian-Huang Lai [c]

[a] College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China
[b] Key Laboratory of Smart Agricultural Technology in Tropical South China, Ministry of Agriculture and Rural Affairs, China
[c] School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[d] Guangdong Provincial Key Laboratory of Intellectual Property and Big Data, Guangzhou, China

ARTICLE INFO

Keywords:
Data clustering
Deep clustering
Image clustering
Graph convolutional network
Multi-scale structure learning

ABSTRACT

Deep clustering has shown its promising capability in joint representation learning and clustering via deep neural networks. Despite the significant progress, the existing deep clustering works mostly utilize some distribution-based clustering loss, lacking the ability to unify representation learning and multi-scale structure learning. To address this, this paper presents a new deep clustering approach termed **I**mage **c**lustering with contrastive **le**arning and multi-scale **G**raph **C**onvolutional **N**etworks (IcicleGCN), which bridges the gap between convolutional neural network (CNN) and graph convolutional network (GCN) as well as the gap between contrastive learning and multi-scale structure learning for the deep clustering task. Our framework consists of four main modules, namely, the CNN-based backbone, the Instance Similarity Module (ISM), the Joint Cluster Structure Learning and Instance reconstruction Module (JC-SLIM), and the Multi-scale GCN module (M-GCN). Specifically, the backbone network with two weight-sharing views is utilized to learn the representations for the two augmented samples (from each image). The learned representations are then fed to ISM and JC-SLIM for joint instance-level and cluster-level contrastive learning, respectively, during which an auto-encoder in JC-SLIM is also pretrained to serve as a bridge to the M-GCN module. Further, to enforce multi-scale neighborhood structure learning, two streams of GCNs and the auto-encoder are simultaneously trained via (i) the layer-wise interaction with representation fusion and (ii) the joint self-adaptive learning. Experiments on multiple image datasets demonstrate the superior clustering performance of IcicleGCN over the state-of-the-art. The code is available at https://github.com/xuyuankun631/IcicleGCN.

1. Introduction

Deep learning has achieved remarkable success in many supervised learning applications, which typically requires a considerable amount of training samples with true labels. To alleviate the probably labor-intensive task of data annotation, the unsupervised learning techniques [1] have recently attracted increasing attention, among which the clustering analysis plays a fundamental role [2–4].

The traditional clustering methods [2–4] generally rely on hand-crafted features, which lack the ability of feature representation learning and may result in sub-optimal clustering performance for high-dimensional complex data. Recently some efforts have been made to incorporate the deep learning technique into the unsupervised clustering task. For example, Yang et al. [5] proposed the Deep Clustering Networks (DCN) method, where a reconstruction loss of the auto-encoder and a K-means based clustering loss are jointly optimized. Chang et al. [6] presented the Deep Adaptive image Clustering (DAC) method by formulating the clustering problem as a binary pairwise classification problem and enforcing the learned labels features to be one-hot vectors which can be used for image clustering. Xie et al. [7] developed the Deep Embedded Cluster (DEC) method, which aims to map the learned features in the data space to a low-dimensional feature space with a Kullback–Leibler (KL) divergence based clustering loss. Yang et al. [8] devised the Joint Unsupervised LEarning (JULE) method, which iteratively updates a convolutional neural network (CNN) by performing the agglomerative clustering in the forward propagation and learning the feature representations in the backward propagation.

[*] Correspondence to: College of Mathematics and Informatics, South China Agricultural University, Guangzhou, Guangdong Province, 510642, China.
E-mail addresses: ykxu@stu.scau.edu.cn (Y. Xu), huangdonghere@gmail.com (D. Huang), changdongwang@hotmail.com (C.-D. Wang), stsljh@mail.sysu.edu.cn (J.-H. Lai).
[1] Google Scholar: https://scholar.google.com/citations?user=dtW41bkAAAAJ.

https://doi.org/10.1016/j.patcog.2023.110065
Received 8 May 2023; Received in revised form 10 September 2023; Accepted 15 October 2023
Available online 16 October 2023

Although these deep clustering works [5–11] have made significant progress, there are still some critical questions that remain to be addressed. Especially, in this paper, we focus on the following three key questions.

Q1: Many of previous works tend to utilize some clustering loss (usually related to the label distributions) to guide the unsupervised training, which often overlook the sample-wise relationships in their learning process. With the contrastive learning recently showing its promising ability in self-supervised learning via positive and negative sample pairs [12], the first question arises as to ***how to incorporate the contrastive learning into the deep clustering process for better representation learning and clustering***.

Q2: The conventional contrastive learning only considers the direct sample-wise relationships, e.g., the relationships between the positive pairs and the negative pairs. Regarding this, the second question arises as to ***how to go beyond the direct sample-wise relationship to explore the rich information in neighborhood structures, or even enforce neighborhood structure learning for deep clustering***.

Q3: Starting from the first two questions, the third question emerges as to ***how to extend the neighborhood structure learning (in Q2) from single-scale to multi-scale, and jointly leverage multi-scale neighborhood structure learning and contrastive learning in a unified framework.***

More recently, several attempts have been carried out to address some of the above three questions. Li et al. [13] proposed the Contrastive Clustering (CC) method to incorporate contrastive learning into deep clustering, which takes into account the sample-wise relationships (between positive sample-pairs and negative sample pairs) but still overlooks the sample-wise neighborhood structure. To investigate the neighborhood structure, van Gansbeke et al. [14] presented a two-stage deep clustering method termed Semantic Clustering by Adopting Nearest neighbors (SCAN), where the first stage employs the contrastive learning to learn the feature representation for constructing a k-nearest neighbor (k-NN) graph and the second stage aims to maximize the similarity between each sample and its k-NNs. Zhong et al. [15] designed the Graph Contrastive Clustering (GCC) method, which also utilizes a k-NN graph to provide more structure information for contrastive representation learning and clustering. Although SCAN [14] and GCC [15] have gone one step further to exploit the neighborhood structure information, yet they are still restricted to the static neighborhood connections and lack the ability to dynamically explore the higher-order connections via neighborhood structure learning.

In terms of neighborhood structure learning, the Graph Convolutional Network (GCN) provides an alternative and powerful tool [16]. As an early attempt, Bo et al. [17] devised the Structural Deep Clustering Network (SDCN) method, which first pretrains an auto-encoder and then simultaneously trains the encoder and the GCN with a KL divergence loss. With the incorporation of GCN [17], the neighborhood structure learning can be enforced in SDCN. However, on the one hand, SDCN takes vectorized feature representations as input, which lacks convolutional layers to extract spatial information from complex image data. On the other hand, SDCN typically uses a single k-NN graph (which represents a specific scale of the neighborhood structure) as the initial graph for GCN, but ignores the possibilities of extending the neighborhood structure learning from a single scale to multiple scales, so as to explore more comprehensive structure information. It remains a challenging problem how to bridge *the gap between CNN and GCN* as well as *the gap between contrastive learning and neighborhood structure learning*, and further, how to go *from single-scale to multi-scale* neighborhood structure learning in a unified deep image clustering framework.

To address the above problem, in this paper, we present a novel deep clustering approach termed **I**mage **c**lusteri**n**g with **c**ontrastive **le**arning and multi-scale **G**raph **C**onvolutional **N**etworks (**IcicleGCN**). The overall architecture of IcicleGCN consists of four modules, namely, the backbone network, the Instance Similarity Module (ISM), the Joint Cluster Structure Learning and Instance reconstruction Module (JC-SLIM), and the Multi-scale GCN module (M-GCN) (as shown in Fig. 1). Specifically, with two types of data augmentations randomly performed on each sample image, we utilize a weight-sharing CNN to learn the representations of the sample pairs, which are then fed to the later modules for contrastive learning and multi-scale neighborhood structure learning (via GCNs). In ISM, we utilize a two-layer multilayer perceptron (MLP) with an instance-level contrastive loss is to maximize the similarity between positive pairs and minimize the similarity between negative pairs. In JC-SLIM, an auto-encoder is incorporated, which simultaneously exploits a cluster-level contrastive loss to learn the cluster structures and an instance reconstruction loss to pretrain the encoder that also serves as a bridge to the M-GCN module. In M-GCN, two k-NN graphs are first built to represent two different scales of neighborhood structures, upon which two streams of GCNs and the auto-encoder (which is shared with the JC-SLIM module) are iteratively trained for the joint multi-scale neighborhood structure learning and clustering. Extensive experiments are conducted on four image datasets, which demonstrate the superiority of the proposed IcicleGCN approach over the state-of-the-art deep clustering approaches.

For clarity, the main contributions of this paper are summarized as follows:

1. This paper for the first time, to the best of our knowledge, enables multi-scale neighborhood structure learning for the image clustering task by taking advantage of multi-scale GCNs with joint self-adaptive learning.
2. This paper presents a novel deep image clustering approach termed IcicleGCN with instance-level contrastive learning, global cluster structure learning, and multi-scale neighborhood structure learning jointly enforced, which notably tackles the aforementioned three key questions (Q1, Q2, and Q3) in a unified framework.
3. Extensive experiments have confirmed the superior clustering performance of our IcicleGCN approach on several challenging image datasets in comparison with the state-of-the-art deep clustering approaches.

The rest of the paper is organized as follows. Section 2 reviews the related works on deep clustering and GCN. Section 3 describes the overall framework of IcicleGCN. Section 4 reports the experimental results. Finally, Section 5 concludes the paper.

2. Related work

In this section, we review the related works on deep clustering and GCN in Sections 2.1 and 2.2, respectively.

2.1. Deep clustering

In the past decade, the deep learning has shown its advantageous ability in learning discriminative features from complex data [18–21]. To exploit the representation learning ability of deep learning in the unsupervised scenarios, the deep clustering has recently drawn significant attention, which aims to learn high-quality feature representations for the clustering task without supervision information [5–8]. A considerable number of deep clustering methods have been designed in recent years. For example, Yang et al. [5] combined the auto-encoder network with the clustering task, where an auto-encoder is trained to reconstruct original features and a K-means clustering loss is incorporated to learn the cluster structure. Chang et al. [6] mapped the clustering problem into a binary pairwise classification framework to judge whether each image pair belongs to the same cluster, and clustered the images by the local maximum response of the label features in the deep neural

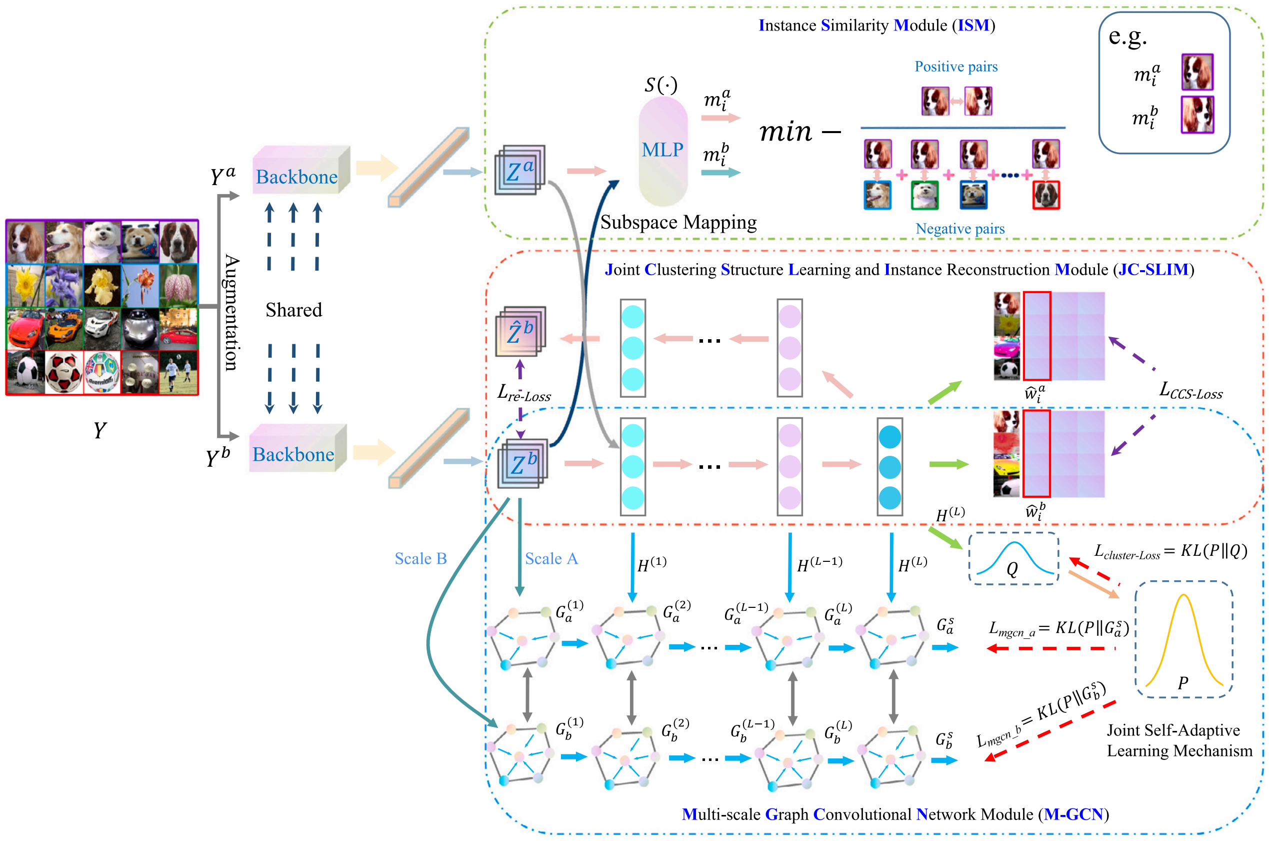

Instance Similarity Module (ISM)
e.g.
Positive pairs
Negative pairs
MLP
Subspace Mapping
min
Backbone
Shared
Augmentation
Joint Clustering Structure Learning and Instance Reconstruction Module (JC-SLIM)
Scale B
Scale A
Joint Self-Adaptive Learning Mechanism
Multi-scale Graph Convolutional Network Module (M-GCN)

Fig. 1. The network architecture of **IcicleGCN** consists of four modules, i.e., the CNN-based **backbone** (which learns the representations for two augmentation views with shared weights), the **ISM** module (which enforces the instance-level contrastive learning), the **JC-SLIM** module (which enforces cluster-level contrastive learning and meanwhile pretrains an auto-encoder as a bridge to the neighborhood structure learning in the next module), and the **M-GCN** module (which jointly leverages multiple GCNs for multi-scale neighborhood structure learning).

network. Xie et al. [7] presented the DEC method, which aims to map the learned features in the data space to a low-dimensional feature space, where the KL-divergence loss is used to iteratively optimize the cluster structure. Huang et al. [22] first utilized an auto-encoder to learn simplified representation from raw data, and then exploited a local preservation constraint to preserve the local structural properties of the data. Yang et al. [8] performed deep clustering by iteratively updating the CNN with the agglomerative clustering conducted in the forward propagation and the feature representations learned in the backward propagation.

Besides these deep clustering methods that utilize some clustering loss in the deep neural network for cluster structure learning, another popular direction in recent years is to incorporate the contrastive learning paradigm [13,14,23] for better representation learning and clustering. Specifically, Li et al. [13] presented an end-to-end deep clustering method which performs the instance-level contrastive learning and the cluster-level contrastive learning at the same time. Van Gansbeke et al. [14] developed a two-stage method termed SCAN. In the first stage, it performs contrastive learning to learn feature representation for finding the nearest neighbors of each image. In the second stage, it obtains the clustering result via a semantic clustering loss with the nearest neighbors of each image considered [14].

2.2. Graph convolutional network

The concept of graph neural network (GNN) was first proposed by Gori et al. [24] and further elaborated in GNN* [25]. Early works on GNN tend to use neural networks to transmit neighborhood information in an iterative way until reaching a stable fixed point.

The significant success of CNN in the field of computer vision has inspired some researchers to focus on convolutional operators for learning neighborhood information, which give rise to the GCN. Bruna et al. [26] designed a variant of graph convolution based on spectral graph theory, which uses the spectral domain network to generalize convolutional networks through graph Fourier transform. Henaff et al. [27] trained a graph convolutional layer that can perform forward and back propagation given a Fourier matrix, an interpolation kernel, and weights. Defferrard et al. [28] developed ChebNet to generalize CNNs to graph data thoroughly using spectral theoretical formulations, where the graph signal filtering and the graph coarsening are performed in the feature extraction stage. These spectral-based methods generally learn the entire graph structure at the same time. Since the processing of the entire graph structure is computationally expensive, these spectral-based methods are often difficult to generalize to large datasets.

Besides the spectral-based methods, there is an increasing number of spatial-based graph convolution methods in recent years. Monti et al. [29] developed the MoNet framework to generalize traditional CNNs to non-Euclidean spaces. Niepert et al. [30] proposed a general and efficient framework termed PATCHY-SAN for representation learning on arbitrary graph networks, which constructs locally connected neighborhoods in the graph and performs operations such as convolution on these neighborhoods. Gao et al. [31] presented the Large-scale learnable Graph Convolutional Networks (LGCN) method, which creates the learnable graph convolutional layers and converts general graph data into grid-structure data with regular convolution operations. These methods directly perform convolution operations on the graph domain by aggregating the information of the neighboring nodes, while ignoring the representation information of the data itself. Shi et al. [32] exploited graph convolution operations on multi-scale prototype graphs for face recognition with image sets, which relies on the prior knowledge of image sets for generating prototypes and cannot perform sample-wise contrastive learning and unsupervised clustering. Recently, to exploit the GCN for the unsupervised clustering task, Bo et al. [17] presented the SDCN method, which combines the features obtained from each layer of an encoder with the features learned by the GCN, and iteratively optimizes the network via a KL-divergence loss. Although SDCN considers the representation information of the data itself and the sample-wise structural information, yet it only takes the single-scale neighborhood information (with a single k-NN graph) as input, which overlooks the potential opportunities of jointly utilizing multi-scale neighborhood structures. Furthermore, SDCN relies on vectorized features, which restricts it application for complex images due to its lack of image-wise convolutional layers.

3. Proposed framework

In this section, we describe the proposed IcicleGCN approach, which bridges the gap between CNN and GCN and also the gap between contrastive learning and multi-scale neighborhood structure learning. As illustrated in Fig. 1, IcicleGCN consists of four modules, the CNN-based backbone, the ISM module, the JC-SLIM module, and the M-GCN module, which will be described in Sections 3.1, 3.2, 3.3, and 3.4, respectively.

3.1. Backbone

Our IcicleGCN approach utilizes a CNN (i.e., ResNet-34) as the backbone, which incorporates two augmentation views with shared weights to produce two augmented samples for each input image. Specifically, given a mini-batch of N images, denoted as $\{y_1, y_2, \dots, y_N\}$, two types of augmentations are randomly selected for each image, leading to a total of $2 \cdot N$ augmented samples for each mini-batch, which will then be fed to the following three modules (i.e., ISM, JC-SLIM, and M-GCN) for the instance-level contrastive learning, the joint cluster-level contrastive learning and auto-encoder training, and the multi-scale neighborhood structure learning, respectively.

3.2. ISM

The purpose of contrastive learning is to train the neural network by maximizing the similarity between the positive pairs while minimizing the similarity between the negative pairs [12]. In this section, we describe the ISM module, which enforces the instance-level contrastive learning.

For an input image y_i, two augmented samples are generated, denoted as y_i^a and y_i^b, respectively. Then the two augmented samples from the same input image are regarded as a positive sample pair, while the other $2 \cdot (N-1)$ sample pairs are regarded as the negative pairs. For an augmented sample y_i^u, its feature representation learned by the backbone is denoted as z_i^u (for $u \in \{a, b\}$).

In the ISM module, a neural network with two fully-connected layers is leveraged to map the representations z_i^a and z_i^b to a low-dimensional space, denoted as $m_i^a = S\left(z_i^a\right)$ and $m_i^b = S\left(z_i^b\right)$. Thereby, the similarity between a pair of samples can be computed by the cosine similarity, that is

$$s\left(m_i^{u_1}, m_j^{u_2}\right) = \frac{\left(m_i^{u_1}\right)\left(m_j^{u_2}\right)^\top}{\left\|m_i^{u_1}\right\| \left\|m_j^{u_2}\right\|}, \quad (1)$$
$$u_1, u_2 \in \{a, b\}, i, j \in [1, N].$$

To maximize the similarity between positive pairs and minimize the similarity between negative pairs, the contrastive instance similarity loss for a sample y_i^a is defined as [12]

$$\ell_i^a = -\log\left(\frac{e^{s(m_i^a, m_i^b)/\tau_I}}{\sum_{j=1}^N \left(e^{s(m_i^a, m_j^a)/\tau_I} + e^{s(m_i^a, m_j^b)/\tau_I}\right)}\right), \quad (2)$$

where τ_I is the temperature parameter. With the two augmented samples of each image considered, we have the contrastive instance similarity loss (for the N images in a mini-batch) as

$$\mathcal{L}_{CIS-Loss} = \frac{1}{2N} \sum_{i=1}^N \left(\ell_i^a + \ell_i^b\right). \quad (3)$$

Thereby, with the instance-level contrastive learning enforced, we proceed to incorporate the cluster-level contrastive learning and the multi-scale neighborhood structure learning in the following.

3.3. JC-SLIM

In this section, we describe the JC-SLIM module, whose technical role is two-fold. On the one hand, it enforces the cluster-level contrastive learning. On the other hand, it also serves as a bridge between the contrastive learning via CNN and the neighborhood structure learning via GCN by sharing the pretrained auto-encoder with the M-GCN module.

Specifically, an auto-encoder is utilized in the JC-SLIM module, where the encoder maps the representations learned by the backbone to a low-dimensional space for the cluster structure learning (i.e., the cluster-level contrastive learning) and meanwhile collaborates with the decoder for the instance reconstruction. In the following, the contrastive cluster structure loss and the instance reconstruction loss will be presented in Sections 3.3.1 and 3.3.2, respectively.

3.3.1. Contrastive cluster structure loss

Besides the contrastive learning at the instance-level, it is also expected to learn the cluster structures by considering the consistency of the two augmentation views. The intuition is to maximize the similarity between the cluster distributions of the two augmentation views, which gives rise to the cluster-level contrastive learning [13].

Given the feature representations (of an augmented pair y_i^a and y_i^b) learned by the backbone, i.e., z_i^a and z_i^b, we respectively map them to the K-dimensional representations w_i^a and w_i^b via the encoder (with softmax operation) of the auto-encoder. Here, the K-dimensional vector w_i^a can be viewed as the probability of the sample y_i^a belonging to each of the K clusters.

By mapping the N pairs of augmented samples to the K-dimensional space via the encoder, we can obtain the representation matrices for the two augmentation views, denoted as $W^a \in \mathbb{R}^{N\times K}$ and $W^b \in \mathbb{R}^{N\times K}$, respectively. Here, w_i^u is the ith row in W^u (for $u \in \{a, b\}$). Let $\hat{w}_i^u$ denote the ith column in W^u, which indicates the probability of the N samples belonging to the ith cluster. Specifically, $\hat{w}_i^a$ and $\hat{w}_i^b$ are regarded as a positive cluster pair, while the other $2 \cdot (K-1)$ cluster pairs are regarded as the negative pairs. Then the contrastive cluster structure loss for $\hat{w}_i^a$ is defined as

$$\hat{\ell}_i^a = -\log\left(\frac{e^{s(\hat{w}_i^a, \hat{w}_i^b)/\tau_C}}{\sum_{j=1}^K \left(e^{s(\hat{w}_i^a, \hat{w}_j^a)/\tau_C} + e^{s(\hat{w}_i^a, \hat{w}_j^b)/\tau_C}\right)}\right), \quad (4)$$

where τ_C is the temperature parameter. With two augmentation views $\hat{w}_j^a$ and $\hat{w}_j^b$ considered, the contrastive cluster structure loss for the K clusters is defined as

$$\mathcal{L}_{CCS-Loss} = \frac{1}{2K} \sum_{i=1}^K \left(\hat{\ell}_i^a + \hat{\ell}_i^b\right) - H(W^a) - H(W^b), \quad (5)$$

where $H(W^u) = -\sum_{i=1}^K \left(P\left(\hat{w}_i^u\right) \log P\left(\hat{w}_i^u\right)\right)$ (for $u \in \{a, b\}$) is an entropy term that is incorporated to avoid the trivial solution of assigning all samples to a single cluster, and $P\left(\hat{w}_i^u\right) = \frac{1}{N}\sum_{j=1}^N w_{ji}^u$, where w_{ji}^u is the (j, i)th entry in W^u (for $u \in \{a, b\}$).

3.3.2. Instance reconstruction loss

In JC-SLIM, the encoder of the auto-encoder is exploited for both the contrastive cluster structure learning and the neighborhood structure learning (for the next module of M-GCN). Thus, besides the contrastive cluster structure loss, we also incorporate the instance reconstruction loss to aid the training of the encoder, that is

$$\mathcal{L}_{re-Loss} = \frac{1}{2N} \sum_{i=1}^{N} \sum_{u=\{a,b\}} \|\mathbf{z}_i^u - \hat{\mathbf{z}}_i^u\|_2^2 = \frac{1}{2N} \sum_{u=\{a,b\}} \|\mathbf{Z}^u - \hat{\mathbf{Z}}^u\|_F^2, \tag{6}$$

where $\hat{\mathbf{z}}_i^u$ is the representation reconstructed by the decoder. Before the neighborhood structure learning in the M-GCN module, we first jointly train the backbone, the ISM module, and the JC-SLIM module via the loss function $\mathcal{L}_1$, that is

$$\mathcal{L}_1 = \mathcal{L}_{CIS-Loss} + \mathcal{L}_{CCS-Loss} + \mathcal{L}_{re-Loss}. \tag{7}$$

It is noteworthy that, empirically, a linear combination of the three losses without additional hyper-parameters to adjust their influences can already lead to quite robust performance. By optimizing the loss function $\mathcal{L}_1$, the instance-level and cluster-level contrastive learning and the pretraining of the auto-encoder (for the next module) are simultaneously enforced. In the next section, the auto-encoder and the multiple GCNs will collaboratively perform the multi-scale neighborhood structure learning for exploring more indepth and versatile sample-wise relationships for deep clustering.

3.4. M-GCN

In this section, we describe the M-GCN module in detail, which exploits the auto-encoder pretrained in the JC-SLIM module, takes advantage of the multiple GCNs for multi-scale neighborhood structure learning, and unifies the these components via joint self-adaptive learning.

3.4.1. Multi-scale neighborhood structures

The graph structure is widely used to capture the sample-wise relationship in data, where a k-NN graph (with a fixed k) can reflect the sample-wise neighborhood structure of a specific scale [14,15,17]. Due to variations of different datasets or even the variations of local structures within a given dataset, a proper scale (with an optimal k) is not easy to be determined for the k-NN graph construction in practice. Though some previous deep clustering methods attempt to explore the neighborhood structure [14,15,17], yet they generally rely on a single k-NN graph and cannot go beyond the single-scale neighborhood structure to explore the diversity in multiple scales. In light of this, in the M-GCN module of our IcicleGCN framework, multiple scales of neighborhood structures are jointly investigated via multiple GCNs.

Specifically, in the M-GCN module, two k-NN graphs are initialized to capture the information of different scales of neighborhood structures, which are then jointly propagated and refined by two streams of GCNs. Note that M-GCN can also be extended to three or more streams of GCNs. Yet we empirically find that two scales of neighborhood structure have brought in sufficient improvement and thus we formulate the M-GCN module with two-stream GCNs (coupled with the auto-encoder from the JC-SLIM module).

With the backbone and the auto-encoder trained via the loss function $\mathcal{L}_1$ with the instance-level and cluster-level contrastive learning enforced, this section focuses on the neighborhood structure learning and does not involve the contrastive learning, which means that the augmentation pairs are no longer required here. Therefore, for each input image y_i, it is augmented once, and then the backbone obtains its representation z^i. The representations of N samples can be stacked as a feature matrix $Z^b \in \mathbb{R}^{N\times d}$, where d is the output dimension of the backbone. To initialize the multi-scale neighborhood structures, we need to construct k-NN graphs of different scales from the feature matrix, and then aggregate the multi-scale neighborhood information from different k-NN graphs via the propagation of GCNs. Here, to define the similarity between two samples, say, z_i and z_j, the heat kernel [17] is utilized, that is

$$\mathrm{S}_{ij} = e^{-\frac{\|z_i - z_j\|^2}{t}}, \tag{8}$$

where t is the time parameter in the heat kernel. In IcicleGCN, we use $t = 1$ for all experiments. Further, we construct two different k-NN graphs to reflect two different scales of neighborhood structures, whose numbers of nearest neighbors are set to k_a and k_b (with $k_a \neq k_b$), respectively. Let A_a and A_b denote the adjacency matrices of the two constructed k-NN graphs. In the next section, we will utilize these two k-NN graphs and the feature embedding learned by the backbone as the input to the trident network with GCNs.

3.4.2. Trident network architecture with GCNs

In the M-GCN module, three sources of input are taken, namely, the feature embedding which is learned by the backbone and fed to the auto-encoder, and the two scales of k-NN graphs which are fed to the two streams of GCNs. Thereby, three streams of networks (including an auto-encoder and two GCNs) are exploited in M-GCN, which is called a trident network architecture for our multi-scale neighborhood structure learning in the proposed IcicleGCN framework (as shown in Fig. 1).

Formally, in terms of the auto-encoder, let L be the number of layers and $\mathbf{H}^{(\ell)}$ be the learned feature embedding at the lth layer of its encoder. Thus the feedforward propagation of the encoder can be represented as

$$\mathbf{H}^{(\ell)} = \Phi\left(\mathbb{W}_e^{(\ell)}\mathbf{H}^{(\ell-1)} + \beta_e^{(\ell)}\right), \tag{9}$$

where Φ is the activation function in the encoder, such as ReLU, and $\mathbb{W}_e^{(\ell)}$ and $\beta_e^{(\ell)}$ are the weight matrix and the bias of the encoder at the ℓth layer, respectively. In particular, we use the feature matrix $\mathbf{Z}^b$ obtained from the backbone as the input $\mathbf{H}^{(0)}$.

Similarly, with the decoding layers corresponding to the encoding layers, each decoding layer reconstructs the input data as follows:

$$\mathbf{H}_d^{(\ell)} = \Phi\left(\mathbb{W}_d^{(\ell)}\mathbf{H}_d^{(\ell-1)} + \beta_d^{(\ell)}\right), \tag{10}$$

where $\mathbb{W}_d^{(\ell)}$ and $\beta_d^{(\ell)}$ represent the weight matrix and the bias of the decoder at the ℓth layer, respectively.

For convenience, we denote the output of the last decoder layer (i.e., $\mathbf{H}_d^{(L)}$) as $\hat{\mathbf{Z}}^b$. Then the reconstruction loss of the auto-encoder is represented as

$$\mathcal{L}_{mgcn\_re-Loss} = \frac{1}{N}\|\mathbf{H}^{(0)} - \mathbf{H}_d^{(L)}\|_F^2 = \frac{1}{N}\|\mathbf{Z}^b - \hat{\mathbf{Z}}^b\|_F^2. \tag{11}$$

Note that the loss (11) is similar to but in fact different from the loss (6). The loss (6) is further used to collaborate with the contrastive loss in the unified loss (7) of the ISM and JC-SLIM modules, where both augmentation views should be incorporated for contrastive learning and thus the reconstruction loss of both augmented samples (of each input sample) should also be considered. Since the contrastive learning is not involved in the M-GCN module, the necessity to consider both augmentation views no longer holds, which gives rise to the reconstruction loss (11).

Further, two streams of GCNs are incorporated in the M-GCN module to *interact* with the encoder of the auto-encoder. Before introducing the interaction (in the layer-wise update and the joint self-adaptive learning), we first describe the GCN and its learning process.

Each sample is regarded as a vertex in the graph. Let V denote the set of all the sample vertices and A denote the adjacency matrix. Then the ith sample V_i aggregates the neighborhood information as

$$\mathrm{Aggregate}(V) = \widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}V \tag{12}$$

$$\left(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}V\right)_i = \left(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\right)_i \widetilde{D}^{-\frac{1}{2}}V$$
$$= \left(\sum_k \widetilde{D}_{ik}^{-\frac{1}{2}}\widetilde{A}_i\right)\widetilde{D}^{-\frac{1}{2}}V$$
$$= \widetilde{D}_{ii}^{-\frac{1}{2}}\sum_j \widetilde{A}_{ij}\sum_k \widetilde{D}_{jk}^{-\frac{1}{2}}V_j$$
$$= \widetilde{D}_{ii}^{-\frac{1}{2}}\sum_j \widetilde{A}_{ij}\widetilde{D}_{jj}^{-\frac{1}{2}}V_j$$
$$= \sum_j \frac{1}{\sqrt{\widetilde{D}_{ii}\widetilde{D}_{jj}}}\widetilde{A}_{ij}V_j, \tag{13}$$

where $\widetilde{A} = A + I$, I is the identity diagonal matrix, and $\widetilde{D}$ is the degree matrix with $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$. The purpose of $\widetilde{A}$ is to obtain the information of the node itself in the adjacency matrix A.

Then we proceed to describe the learning process of the two-stream GCNs. Let $\mathbf{G}_a^{(\ell)}$ and $\mathbf{G}_b^{(\ell)}$ denote the representations respectively learned by the *first* and the *second* GCNs at the ℓth layer, where the graph convolution operation can be conducted as

$$\mathbf{G}_u^{(\ell)} = \Phi\left(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}_u\widetilde{D}^{-\frac{1}{2}}\mathbf{G}_u^{(\ell-1)}\mathbb{W}_u^{(\ell-1)}\right), \text{ for } u \in \{a, b\}, \tag{14}$$

where $\mathbb{W}_u^{\ell-1}$ (for $u \in \{a, b\}$) is the weight matrix of the $(\ell-1)$th layer of the corresponding GCN. Under the guidance of the symmetric normalized adjacency matrix $\widetilde{D}^{-\frac{1}{2}}\widetilde{A}_u\widetilde{D}^{-\frac{1}{2}}$, we use the representation $\mathbf{G}_u^{(\ell-1)}$ of the $(\ell-1)$th layer to generate the representation $\mathbf{G}_u^{(\ell)}$ of the ℓth layer through the graph convolution operation (14).

Besides the two streams of GCNs, the representation learned by the lth layer of the encoder (of the auto-encoder) is denoted as $\mathbf{H}^{(\ell)}$. Then we update each of the two-stream GCNs as follows:

$$\widetilde{\mathbf{G}}_a^{(\ell-1)} = \sigma\mathbf{G}_a^{(\ell-1)} + \gamma\mathbf{G}_b^{(\ell-1)} + (1-\sigma-\gamma)\mathbf{H}^{(\ell-1)}, \tag{15}$$
$$\widetilde{\mathbf{G}}_b^{(\ell-1)} = \sigma\mathbf{G}_b^{(\ell-1)} + \gamma\mathbf{G}_a^{(\ell-1)} + (1-\sigma-\gamma)\mathbf{H}^{(\ell-1)}, \tag{16}$$

where σ and γ are two balance coefficients. In this work, we set σ to 0.4 and γ to 0.2 on all experiments. Through this layer-by-layer connection, the representation learning processes of the encoder and the two-stream GCNs are jointly leveraged.

Thereafter, we feed $\widetilde{\mathbf{G}}_u^{(\ell-1)}(u \in \{a, b\})$ into the ℓth layer to generate the representation $\widetilde{\mathbf{G}}_u^{(\ell)}$ as

$$\widetilde{\mathbf{G}}_u^{(\ell)} = \Phi\left(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}_u\widetilde{D}^{-\frac{1}{2}}\widetilde{\mathbf{G}}_u^{(\ell-1)}\mathbb{W}_u^{(\ell-1)}\right). \tag{17}$$

The GCN can aggregate the neighborhood information of each node in its first layer, which is called the first-order neighborhood information. And the neighbors are also aggregating their own neighborhood information. Therefore, In the second layer, when the same node aggregates its neighborhood information again, the neighborhood information of its neighbors can be aggregated, which is called the second-order neighborhood information. Theoretically, with the number of layers increased to sufficiently large, a node can aggregate the information of all nodes in the k-NN graph. However, in practical applications, the number of GCN layers generally will not exceed 4 or 5. In fact, as the number of layers increases to a large number, the aggregation of such a large amount of information will make the nodes in the graph undistinguishable, which is called the over-smoothing and may degrade the model performance.

In our M-GCN module, we use five layers of GCN, in which the last layer is associated with a softmax operation. Note that the feature representation $\mathbf{Z}^b$ obtained from the backbone is used as the original input of the node features (to the first GCN layer), that is

$$\mathbf{G}_u^{(1)} = \Phi\left(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}_u\widetilde{D}^{-\frac{1}{2}}\mathbf{Z}^b\mathbb{W}_u^{(0)}\right), \tag{18}$$

The last layer of the GCN is associated with a softmax operation, that is

$$\mathbf{G}_u^s = \mathrm{softmax}\left(\mathbf{G}_u^{(L)}\right). \tag{19}$$

Thus, with the dimension of the softmax layer set to the number of clusters K, we obtain two probability matrices $\mathbf{G}_u^s$ (with $u \in \{a, b\}$) for the cluster assignments, where the (i, j)th entry of $\mathbf{G}_u^s$ represents the probability that the sample i should be assigned to cluster j. Therefore, $\mathbf{G}_a^s$ and $\mathbf{G}_b^s$ can be regarded as two probability distributions, which will be exploited in the joint self-adaptive learning process.

3.4.3. Joint self-adaptive learning

With the auto-encoder and the two streams of GCNs connected via the layer-wise updating, the consistency of their final-layer representations will be our next focus. In this section, we design a joint self-adaptive learning mechanism to simultaneously guide the learning of the three streams of networks in M-GCN.

First, we use the Student's t-distribution [33] to convert the representations obtained by the auto-encoder into a probability distribution. That is, the probability that the ith sample is assigned to the jth cluster can be obtained as follows:

$$q_{ij} = \frac{\left(1 + \left\|\mathbf{h}_i - \boldsymbol{\mu}_j\right\|^2 / t\right)^{-\frac{t+1}{2}}}{\sum_{j'}\left(1 + \left\|\mathbf{h}_i - \boldsymbol{\mu}_{j'}\right\|^2 / t\right)^{-\frac{t+1}{2}}}, \tag{20}$$

where t is the degree of freedom of the Student's t-distribution in the probability distribution function, and $\boldsymbol{\mu}_j$ is the jth cluster center initialized by K-means through the pre-trained representations of the auto-encoder. In the encoder part, we obtain the data representation matrix $\mathbf{H}^L$, where the ith row $\mathbf{h}_i$ is the representation of the ith sample. Here, the probability of assigning sample i to cluster j (i.e., q_{ij}) is obtained by computing the similarity between $\mathbf{h}_i$ and $\boldsymbol{\mu}_j$. By considering the probability distribution that the samples assigned to different clusters, we can denote the probability distribution matrix as Q, where q_{ij} is its (i, j)th entry.

To make the representations obtained from the auto-encoder closer to the center of the corresponding cluster, we proceed to compute a target distribution P from Q, whose (i, j)th entry p_{ij} is defined as

$$p_{ij} = \frac{q_{ij}^2 / f_j}{\sum_{j'} q_{ij'}^2 / f_{j'}}, \tag{21}$$

with

$$f_j = \sum_i q_{ij}. \tag{22}$$

Since q_{ij} is a soft assignment probability, f_j can thus be regarded as the soft cluster assignment frequency. In turn, we can use the target distribution P to supervise the learning of the distribution Q:

$$\mathcal{L}_{cluster-Loss} = KL(P \parallel Q) = \sum_i\sum_j p_{ij}\log\frac{p_{ij}}{q_{ij}}. \tag{23}$$

By this formulation, we then perform self-adaptive learning of the Q distribution by minimizing the KL-divergence between the P and Q distributions.

Further, by regarding the representations $\mathbf{G}_a^s$ and $\mathbf{G}_b^s$ respectively learned by the two GCNs as probability distributions, we can also use the probability distribution P to supervise the update and learning of the two-stream GCNs by minimizing their KL-divergence losses as follows:

$$\mathcal{L}_{mgcn\_a} = KL(P \parallel \mathbf{G}_a^s) = \sum_i\sum_j p_{ij}\log\frac{p_{ij}}{g_{a,ij}^s}, \tag{24}$$
$$\mathcal{L}_{mgcn\_b} = KL(P \parallel \mathbf{G}_b^s) = \sum_i\sum_j p_{ij}\log\frac{p_{ij}}{g_{b,ij}^s}, \tag{25}$$

where $g_{a,ij}^s$ and $g_{b,ij}^s$ denote the (i, j)th entries of $\mathbf{G}_a^s$ and $\mathbf{G}_b^s$, respectively. For the joint self-adaptive learning of the auto-encoder and the two-stream GCNs, we define the overall loss function $\mathcal{L}_2$ of the M-GCN module as

$$\mathcal{L}_2 = \mathcal{L}_{mgcn\_re-Loss} + \alpha\mathcal{L}_{cluster-Loss}$$

$$+ \beta\mathcal{L}_{mgcn\_a} + \eta\mathcal{L}_{mgcn\_b}, \tag{26}$$

where α, β, and η are hyper-parameters to balance the influences of different terms. Through this formulation, the information of the data itself and the multi-scale neighborhood structure information can be adaptively aggregated, and the final clustering can be obtained from the probability distribution learned by the GCNs. For clarity, the overall process of our IcicleGCN approach is described in Algorithm 1.

Algorithm 1: Training algorithm for IcicleGCN.

```
Input: Image dataset $\mathbf{Y}$, the training epochs $E$, the batch size $N$, the number of iterations in M-GCN $N_{it}$, the temperature parameters $\tau_I$ and $\tau_C$, and the number of clusters $K$.
Output: The clustering result with $K$ clusters.
1  for each epoch do
2      Step1 : Randomly select a mini-batch of $N$ images from the image dataset;
3      Step2 : Perform two random data augmentations on each image in the mini-batch;
4      Step3 : Calculate the contrastive instance similarity loss by Eq. (3);
5      Step4 : Calculate the contrastive cluster structure loss by Eq. (5);
6      Step5 : Calculate the instance reconstruction loss by Eq. (6);
7      Step6 : Jointly update the backbone, the ISM, and the JC-SLIM by minimizing $\mathcal{L}_1$ in Eq. (7);
8  end
9  for each iteration in $N_{it}$ do
10     Step7 : Calculate the probability distribution of the two-stream GCN networks by Eq. (19);
11     Step8 : Calculate the $Q$ distribution by Eq. (20);
12     Step9 : Calculate the $P$ distribution by Eq. (21);
13     Step10 : Update the GCNs and the auto-encoder by minimizing $\mathcal{L}_2$ in Eq. (26) ;
14 end
15 Obtain the final clustering from the probability distribution learned by the M-GCN module;
```

3.5. Implementation details

Our IcicleGCN approach consists of four modules, including the backbone, the ISM module, the JC-SLIM module, and the M-GCN module. Specifically, we adopt the ResNet-34 as the backbone network, and resize the images in the dataset to the size of $224 \times 224 \times 3$. Note that there is an overlapping (or sharing) component between the JC-SLIM module and the M-GCN module, that is, the auto-encoder, which is pretrained in JC-SLIM and further participates in the updating process of the multi-scale GCNs. As for the network training, we first use the loss function $\mathcal{L}_1$ to train the backbone, the ISM, and the JC-SLIM, and then use the loss function $\mathcal{L}_2$ to train the M-GCN. The dimension of the auto-encoder is set to input-500-500-2000-K, where K is the number of clusters. The batch size is set to 128. We use the Adam optimizer with a learning rate of 1e–4. The parameters σ and γ are set to 0.4 and 0.2, respectively. The parameters α, β and η in the loss function $\mathcal{L}_2$ are tuned in the range of {0.01, 0.05, 0.1}. To construct the two different k-NN graphs, the numbers of nearest neighbors are set of 1 and 10, respectively. That is, a 1-NN graph and a 10-NN graph are initialized for M-GCN. All experiments are carried out on a machine with an Nvidia GeForce RTX 3090 GPU and a CPU with 12 cores and 2.6 GHz.

Table 1
The clustering performance w.r.t. **ACC (%)** by different clustering methods on the benchmark datasets.

Dataset	CIFAR-10	CIFAR-100	ImageNet-10	ImageNet-Dogs
K-means [38]	22.9	13.0	24.1	10.5
SC [39]	24.7	13.6	27.4	11.1
AC [40]	22.8	13.8	24.2	13.9
NMF [41]	19.0	11.8	23.0	11.8
AE [42]	31.4	16.5	31.7	18.5
DAE [43]	29.7	15.1	30.4	19.0
DCGAN [44]	31.5	15.3	34.6	17.4
DeCNN [45]	28.2	13.3	31.3	17.5
VAE [46]	29.1	15.2	33.4	17.9
JULE [8]	27.2	13.7	30.0	13.8
DEC [7]	30.1	18.5	38.1	19.5
DAC [6]	52.2	23.8	52.7	27.5
DDC [47]	52.4	–	57.7	–
DCCM [9]	62.3	32.7	71.0	38.3
IIC [48]	61.7	25.7	–	–
GATCluster [49]	62.3	32.7	73.9	32.2
PICA [10]	69.6	33.7	87.0	35.2
DRC [50]	72.7	36.7	88.4	38.9
CC [13]	79.0	42.9	89.5	34.2
CLD [51]	54.2	42.0	80.7	31.5
HCSC [52]	48.0	36.2	74.1	35.5
IcicleGCN	**80.7**	**46.1**	**95.5**	**41.5**

4. Experiments

In this section, we experimentally compare the proposed IcicleGCN algorithm against several deep and non-deep clustering algorithms on multiple image datasets.

4.1. Datasets and evaluation metrics

We conduct experiments on four image datasets. Some sample images of these datasets are shown in Fig. 2. The details of the four datasets are given below.

- **CIFAR-10** [34]**:** The CIFAR-10 dataset consists of 60,000 color images of size $32 \times 32 \times 3$. These images are divided into 10 classes, including airplanes, cars, birds, cats, trucks, etc.
- **CIFAR-100** [34]**:** The CIFAR-100 datasets consists of 60,000 color images of size $32 \times 32 \times 3$. Unlike CIFAR-10, each image in CIFAR-100 has an "elaborate" class label (corresponding to a total of 100 classes) and a "coarse" class label (corresponding to a total of 20 super-classes). In this paper, the 20 super-classes are taken as the ground-truth.
- **ImageNet-10** [6]**:** The ImageNet-10 dataset is a subset of the ImageNet dataset, containing a total of 13,000 color images with 10 classes. The size of each image is $96 \times 96 \times 3$.
- **ImageNet-Dogs** [6]**:** The ImageNet-Dogs dataset is also a subset of the ImageNet dataset, containing 19,500 images with 15 dog categories. The size of each image is $96 \times 96 \times 3$.

To quantitatively compare the clustering results by different clustering methods, we adopt three widely-used evaluation metrics, namely, Accuracy (ACC) [35], Normalized Mutual Information (NMI) [36], and Adjusted Rand Index (ARI) [37]. Notice that larger values of the three metrics indicate better clustering performance.

In this section, we compare our IcicleGCN method with nineteen deep and non-deep clustering methods, including K-means [38], Spectral Clustering (SC) [39], Agglomerative Clustering (AC) [40], Nonnegative Matrix Factorization (NMF) [41], Auto-Encoder (AE) [42], Denoising Auto-Encoder (DAE) [43], Deep Convolution Generative Adversarial Network (DCGAN) [44], Deconvolutional Network (DeCNN) [45], Variational Auto-Encoder (VAE) [46], Jointly Unsupervised Learning (JULE) [8], Deep Embedding Clustering (DEC) [7], Deep Adaptive image Clustering (DAC) [6], Deep Discriminative Clustering

Fig. 2. Some examples of the four image datasets.

Table 2
The clustering performance w.r.t. **NMI (%)** by different clustering methods on the benchmark datasets.

Dataset	CIFAR-10	CIFAR-100	ImageNet-10	ImageNet-Dogs
K-means [38]	8.7	8.4	11.9	5.5
SC [39]	10.3	9.0	15.1	3.8
AC [40]	10.5	9.8	13.8	3.7
NMF [41]	8.1	7.9	13.2	4.4
AE [42]	23.9	10.0	21.0	10.4
DAE [43]	25.1	11.1	20.6	10.4
DCGAN [44]	26.5	12.0	22.5	12.1
DeCNN [45]	24.0	9.2	18.6	9.8
VAE [46]	24.5	10.8	19.3	10.7
JULE [8]	19.2	10.3	17.5	5.4
DEC [7]	25.7	13.6	28.2	12.2
DAC [6]	39.6	18.5	39.4	21.9
DDC [47]	42.4	–	43.3	–
DCCM [9]	49.6	28.5	60.8	32.1
IIC [48]	51.1	22.5	–	–
GATCluster [49]	49.6	28.5	59.4	28.1
PICA [10]	59.1	31.0	80.2	35.2
DRC [50]	62.1	35.6	83.0	38.4
CC [13]	70.5	43.1	86.2	40.1
CLD [51]	44.3	42.5	67.1	27.9
HCSC [52]	40.7	36.1	64.7	35.5
IcicleGCN	**72.9**	**45.9**	**90.4**	**45.8**

Table 3
The clustering performance w.r.t. **ARI (%)** by different clustering methods on the benchmark datasets.

Dataset	CIFAR-10	CIFAR-100	ImageNet-10	ImageNet-Dogs
K-means [38]	4.9	2.8	5.7	2.0
SC [39]	8.5	2.2	7.6	1.3
AC [40]	6.5	3.4	6.7	2.1
NMF [41]	3.4	2.6	6.5	1.6
AE [42]	16.9	4.8	15.2	7.3
DAE [43]	16.3	4.6	13.8	7.8
DCGAN [44]	17.6	4.5	15.7	7.8
DeCNN [45]	17.4	3.8	14.2	7.3
VAE [46]	16.7	4.0	16.8	7.9
JULE [8]	13.8	3.3	13.8	2.8
DEC [7]	16.1	5.0	20.3	7.9
DAC [6]	30.6	8.8	30.2	11.1
DDC [47]	32.9	–	34.5	–
DCCM [9]	40.8	17.3	55.5	18.2
IIC [48]	41.1	11.7	–	–
GATCluster [49]	40.8	17.3	55.2	16.3
PICA [10]	51.2	17.1	76.1	20.1
DRC [50]	54.7	20.8	79.8	23.3
CC [13]	63.7	26.6	82.5	22.5
CLD [51]	31.9	26.4	62.6	14.1
HCSC [52]	29.5	29.5	55.9	20.9
IcicleGCN	**66.0**	**31.1**	**90.5**	**27.9**

(DDC) [47], Deep Comprehensive Correlation Mining (DCCM) [9], Invariant Information Clustering (IIC) [48], self-supervised Gaussian-ATtention network for image Clustering (GATCluster) [49], PartItion Confidence mAximisation (PICA) [10], Deep Robust Clustering (DRC) [50], Contrastive Clustering (CC) [13], Cross-Level Discrimination (CLD) [51] and Hierarchical Contrastive Selective Coding (HCSC) [52]. The clustering performances of different clustering methods w.r.t. ACC, NMI, and ARI are reported in Tables 1, 2, and 3, respectively.

4.2. *Comparisons with other clustering methods*

In terms of ACC, as shown in Table 2, our IcicleGCN method outperforms or significantly outperforms the baseline clustering methods on all the four benchmark datasets. Especially, on the CIFAR-100, ImageNet-10, and ImageNet-Dogs datasets, our IcicleGCN method achieves ACC (%) scores of 46.1, 95.5, and 41.5, respectively, while the best baseline method (i.e., CC) only obtains scores of 42.9, 89.5, and 34.2, respectively. In terms of NMI and ARI, similar advantages of IcicleGCN can also be observed. As shown in Table 2, on the four benchmark datasets, our IcicleGCN method achieves NMI (%) scores of 72.9, 45.9, 90.4, and 45.8, respectively, while the best baseline method only obtains scores of 70.5, 43.1, 86.2, and 40.1. As shown in Table 3, IcicleGCN achieves ARI (%) scores of 66.0, 31.1, 90.5, and 27.9 on the four datasets, while the best baseline method only obtains scores of 63.7, 26.6, 82.5, and 22.5, respectively. The experimental results in Tables 1, 2, and 3 confirm the superior clustering performance of the proposed IcicleGCN method over the baseline deep and non-deep clustering methods on the benchmark datasets.

4.3. *Ablation study*

In this section, we conduct ablation analysis to test the influences of different modules (and the components inside each module). We first test the clustering performance of IcicleGCN with different components in ISM and JC-SLIM or even the whole module removed in Section 4.3.1. Then we test the clustering performance with the components in M-GCN removed in Section 4.3.2. Further, we test the influence of the multi-scale GCNs and the different neighborhood combinations in Sections 4.3.3 and 4.3.4, respectively.

4.3.1. *Influences of the components in ISM and JC-SLIM*

This section conducts ablation analysis on the components in ISM and JC-SLIM. Specifically, three losses are incorporated in the ISM and JC-SLIM modules, namely, the contrastive instance similarity loss $\mathcal{L}_{CIS-Loss}$, the contrastive cluster structure loss $\mathcal{L}_{CCS-Loss}$, and the instance reconstruction loss (of the auto-encoder) $\mathcal{L}_{re-Loss}$, which jointly contribute to the self-supervised training of the CNN in our IcicleGCN framework. Notice that the auto-encoder serves as a bridge between the JC-SLIM module and the M-GCN module. Therefore, the reconstruction loss $\mathcal{L}_{re-Loss}$ will be preserved to keep this connection to M-GCN, while the other two losses will be removed and tested in this section. In fact, when the two contrastive losses are removed, the effects of the ISM module and the JC-SLIM module almost disappear, except that the auto-encoder still remains for the sake of M-GCN. As shown in Table 4, both the contrastive instance similarity loss and the contrastive cluster structure loss play a substantial role in IcicleGCN. Especially, the joint incorporation of the three losses leads to the best clustering

Table 4
Influence of three losses in the ISM module and the JC-SLIM module.

Loss function	ImageNet-Dogs		
	ACC	NMI	ARI
$\mathcal{L}_{re-Loss} + \mathcal{L}_{CIS-Loss} + \mathcal{L}_{CCS-Loss}$	**41.5**	**45.8**	**27.9**
$\mathcal{L}_{re-Loss} + \mathcal{L}_{CIS-Loss}$	25.8	23.2	12.5
$\mathcal{L}_{re-Loss} + \mathcal{L}_{CCS-Loss}$	9.2	1.0	0.3
$\mathcal{L}_{re-Loss}$	6.9	0.4	0.0

Table 5
Influence of the GCNs and the auto-encoder (AE) in the M-GCN module.

Dataset	Ablation of components	ACC	NMI	ARI
ImageNet-Dogs	*With* GCNs and AE	**41.5**	**45.8**	**27.9**
	Without GCNs	39.3	41.4	24.5
	Without GCNs and AE	34.2	40.1	22.5

performance (w.r.t. ACC, NMI, and ARI) when compared to the variants with one or two components removed.

4.3.2. Influence of the components in M-GCN

There are three streams of networks in the M-GCN module, including an auto-encoder and two streams of GCNs. In this section, we test the influence of the GCNs and the auto-encoder. Note that when the auto-encoder is removed, we will put back a nonlinear multi-layer MLP instead, so as to make the JC-SLIM module still functionable. As shown in Table 5, removing the GCNs degrades the ACC (%), NMI (%), and ARI (%) scores from 41.5, 45.8, and 27.9 to 39.3, 41.4, and 24.5, respectively, while removing both the GCNs and the auto-encoder further degrades the ACC (%), NMI (%), and ARI (%) scores to 34.2, 40.1, and 22.5, respectively, which demonstrate the contributions of the components in the M-GCN module. In the following, we will further test the influence of using different settings of the GCNs.

4.3.3. Multi-scale GCNs VS. single GCN

In this paper, the proposed IcicleGCN method is able to capture and propagate the multi-scale neighborhood information via two streams of GCNs, for which two k-NN graphs with different neighborhood sizes are constructed as the initial input. In this section, we test the clustering performance of IcicleGCN using the two-stream GCNs against using a single GCN. As shown in Table 6, the incorporation of two scales of k-NN graphs (in the M-GCN module) yields consistently better clustering performance w.r.t. ACC, NMI, and ARI than using the GCN with a single-scale k-NN graph.

4.3.4. Influence of different neighborhood combinations

In this section, we test the influence of using different combinations of k-NN graphs in the M-GCN module. In IcicleGCN, two streams of GCNs are utilized, each of which requires a k-NN graph as the input graph. As shown in Table 6, using two different scales of k-NN graphs is often beneficial for the clustering performance. Particularly, using the combination of a 1-NN graph and a 10-NN graph leads to the optimal clustering performance, probably due to the complementariness between the nearest neighborhood and a relative larger neighborhood size. In this paper, we adopt the combination of a 1-NN graph and a 10-NN graph for our multi-scale neighborhood structure learning via the GCNs on all the benchmark datasets.

4.4. Visualization of confusion matrices

In Fig. 3, we visualize the confusion matrices of the clustering results by IcicleGCN on the four datasets. The confusion matrices for CIFAR-10 and ImageNet-10 have a clear block diagonal structure that implies our IcicleGCN method successfully partitions most of the images into semantic clusters. For CIFAR-100 and ImageNet-Dogs, the block diagonal structure can still be observed, though it is not as clear as

Table 6
The clustering performance of IcicleGCN using a single GCN and using multi-scale GCNs (with different neighborhood combinations).

Neighborhood combinations in M-GCN	ImageNet-Dogs		
	ACC	NMI	ARI
1-NN	40.1	43.5	25.5
3-NN	39.8	43.4	25.6
5-NN	39.7	43.4	24.8
10-NN	39.6	43.5	25.3
1-NN + *10*-NN	**41.5**	**45.8**	**27.9**
3-NN + *10*-NN	41.1	45.3	26.0
3-NN + *5*-NN	41.4	45.3	27.0

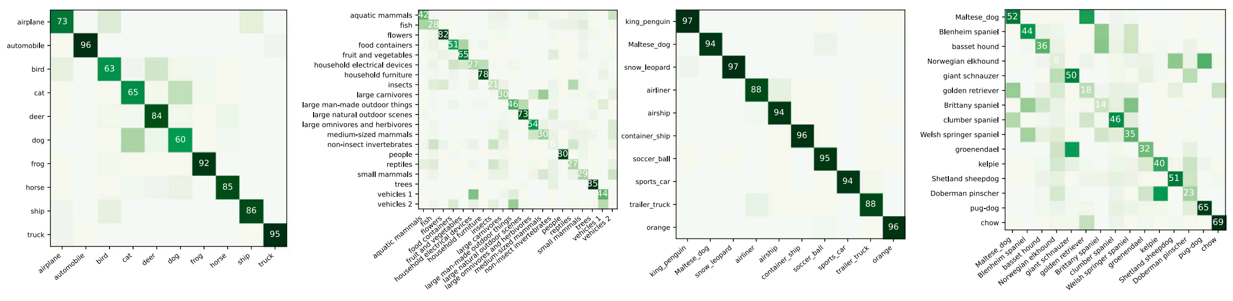
the other two confusion matrices, possibly due to the fact that (i) the images in CIFAR-100 are mostly small and blurry and that (ii) the categories of dogs in ImageNet-Dogs are sometimes difficult to be distinguished even for humans (as further illustrated in Fig. 4). However, even on the challenging datasets of CIFAR-100 and ImageNet-Dogs, the proposed IcicleGCN method still exhibits a substantial advantage over the state-of-the-art deep clustering methods (as shown in Tables 1, 2, and 3).

4.5. Case study

To gain a deeper understanding of how our IcicleGCN method performs, we investigate the success and failure cases on the ImageNet-Dogs dataset. Specifically, we study three situations of four dog categories on ImageNet-Dogs, namely, (i) the *successful* cases that correspond to the correctly clustered samples, (ii) the *false negative* failure cases that correspond to the samples which belong to this cluster but are incorrectly assigned to other clusters, and (iii) the *false positive* cases that correspond to the samples which do not belong to this cluster but are incorrectly assigned to this cluster. As shown in Fig. 4, IcicleGCN successfully assigns many images of the same category to the same cluster. However, when there are multiple objects of different categories in the images, some of them may be assigned to the incorrect clusters (as shown in the middle sub-figure of Fig. 4). Furthermore, some categories of dogs look very similar to each other, which may also contribute to the incorrect cluster assignments (as shown in the right sub-figure of Fig. 4). How to distinguish multiple objects in the same image and how to distinguish different fine-grained classes with similar appearance is still a very challenging problem for unsupervised learning and clustering.

5. Conclusion and future work

In this paper, we propose a novel deep clustering approach called IcicleGCN, which bridges the gap between CNN and GCN as well as the gap between contrastive learning and multi-scale neighborhood structure learning for the deep image clustering task. The IcicleGCN approach consists of four main modules, i.e., the backbone, the ISM, the JC-SLIM, and the M-GCN. With two augmented samples generated for each input image, the backbone with two weight-sharing views is utilized to extract their feature representations, which are then fed to the following three modules. In ISM and JC-SLIM, three types of losses are designed and jointly optimized, namely, the contrastive instance similarity loss, the contrastive cluster similarity loss, and the instance reconstruction loss (via an auto-encoder). It is worth mentioning that the auto-encoder serves as a bridge between the JC-SLIM module and the M-GCN module. In M-GCN, two streams of GCNs and the auto-encoder are iteratively and simultaneously updated with layer-wise representation fusion, upon which a joint self-adaptive learning mechanism is further incorporated to ensure the consistency of their last-layer clustering distributions. Experiments are conducted



(a) CIFAR-10 (b) CIFAR-100 (c) ImageNet-10 (d) ImageNet-Dogs

Fig. 3. Confusion matrices for the clustering results of IcicleGCN on the four image datasets.

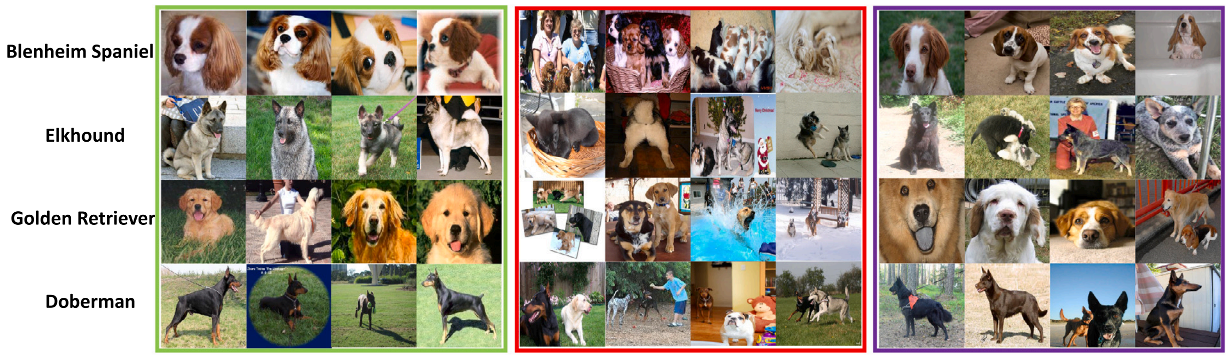


Fig. 4. Case study on ImageNet-Dogs. The ***left*** sub-figure shows the **successful** cases, the ***middle*** sub-figure shows the **false negative** cases, and the ***right*** sub-figure shows the **false positive** cases.

on four challenging image datasets, which have shown the advantageous clustering performance of IcicleGCN over the state-of-the-art deep clustering approaches.

Note that this paper mainly focuses on the deep clustering task for image data. In future work, it may be a promising direction to extend the proposed deep clustering framework to other types of data, such as time series data [53]. Besides, this paper performs the image-level clustering with each image treated as a sample, which may also be extended to the deep clustering at other levels of granularity, such as the pixel-level clustering (i.e., image segmentation [54]) in the future research.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the NSFC (61976097, 62276277 & U22A2095), the Natural Science Foundation of Guangdong Province (2021A1515012203), and Guangdong Provincial Key Laboratory of Intellectual Property and Big Data (2018B030322016).

References

[1] S.-G. Fang, D. Huang, C.-D. Wang, Y. Tang, Joint multi-view unsupervised feature selection and graph learning, IEEE Trans. Emerg. Top. Comput. Intell. (2023) http://dx.doi.org/10.1109/TETCI.2023.3306233.

[2] J. Wang, C. Tang, X. Liu, W. Zhang, W. Li, X. Zhu, L. Wang, A.Y. Zomaya, Region-aware hierarchical latent feature representation learning-guided clustering for hyperspectral band selection, IEEE Trans. Cybern. (2022) 1–14.

[3] J. Lao, D. Huang, C.-D. Wang, J.-H. Lai, Towards scalable multi-view clustering via joint learning of many bipartite graphs, IEEE Trans. Big Data (2023) http://dx.doi.org/10.1109/TBDATA.2023.3325045.

[4] D. Huang, C.-D. Wang, J.-H. Lai, Fast multi-view clustering via ensembles: Towards scalability, superiority, and simplicity, IEEE Trans. Knowl. Data Eng. (2023) http://dx.doi.org/10.1109/TKDE.2023.3236698.

[5] B. Yang, X. Fu, N.D. Sidiropoulos, M. Hong, Towards k-means-friendly spaces: Simultaneous deep learning and clustering, in: Proceedings of the International Conference on Machine Learning, 2017, pp. 3861–3870.

[6] J. Chang, L. Wang, G. Meng, S. Xiang, C. Pan, Deep adaptive image clustering, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 5879–5887.

[7] J. Xie, R. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, in: Proceedings of the International Conference on Machine Learning, 2016, pp. 478–487.

[8] J. Yang, D. Parikh, D. Batra, Joint unsupervised learning of deep representations and image clusters, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5147–5156.

[9] J. Wu, K. Long, F. Wang, C. Qian, C. Li, Z. Lin, H. Zha, Deep comprehensive correlation mining for image clustering, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 8150–8159.

[10] J. Huang, S. Gong, X. Zhu, Deep semantic clustering by partition confidence maximisation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020, pp. 8849–8858.

[11] H. Lu, C. Chen, H. Wei, Z. Ma, K. Jiang, Y. Wang, Improved deep convolutional embedded clustering with re-selectable sample training, Pattern Recognit. 127 (2022) 108611.

[12] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, in: Proc. of International Conference on Machine Learning (ICML), 2020, pp. 1597–1607.

[13] Y. Li, P. Hu, Z. Liu, D. Peng, J.T. Zhou, X. Peng, Contrastive clustering, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2021.

[14] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, L. Van Gool, Scan: Learning to classify images without labels, in: Proceedings of the European Conference on Computer Vision, 2020, pp. 268–285.

[15] H. Zhong, J. Wu, C. Chen, J. Huang, M. Deng, L. Nie, Z. Lin, X.-S. Hua, Graph contrastive clustering, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 9224–9233.

[16] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, arXiv preprint arXiv:1609.02907.
[17] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, P. Cui, Structural deep clustering network, in: Proceedings of the Web Conference 2020, 2020, pp. 1400–1410.
[18] J. Zhang, C. Li, Y. Yin, J. Zhang, M. Grzegorzek, Applications of artificial neural networks in microorganism image analysis: a comprehensive review from conventional multilayer perceptron to popular convolutional neural network and potential visual transformer, Artif. Intell. Rev. 56 (2) (2023) 1013–1070.
[19] H. Chen, C. Li, X. Li, M.M. Rahaman, W. Hu, Y. Li, W. Liu, C. Sun, H. Sun, X. Huang, et al., IL-MCAM: An interactive learning and multi-channel attention mechanism-based weakly supervised colorectal histopathology image classification approach, Comput. Biol. Med. 143 (2022) 105265.
[20] H.-B. Ling, D. Huang, J. Cui, C.-D. Wang, HOLT-Net: Detecting smokers via human-object interaction with lite transformer network, Eng. Appl. Artif. Intel. 126 (2023) 106919.
[21] D. Huang, D.-H. Chen, X. Chen, C.-D. Wang, J.-H. Lai, DeepCluE: Enhanced image clustering via multi-layer ensembles in deep neural networks, IEEE Trans. Emerg. Top. Comput. Intell. (2023).
[22] P. Huang, Y. Huang, W. Wang, L. Wang, Deep embedding network for clustering, in: 2014 22nd International Conference on Pattern Recognition, 2014, pp. 1532–1537.
[23] X. Deng, D. Huang, D.-H. Chen, C.-D. Wang, J.-H. Lai, Strongly augmented contrastive clustering, Pattern Recognit. 139 (2023) 109470.
[24] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: Proceedings of IEEE International Joint Conference on Neural Networks, Vol. 2, 2005, pp. 729–734.
[25] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2008) 61–80.
[26] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, 2013, arXiv preprint arXiv:1312.6203.
[27] M. Henaff, J. Bruna, Y. LeCun, Deep convolutional networks on graph-structured data, 2015, arXiv preprint arXiv:1506.05163.
[28] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Adv. Neural Inf. Process. Syst. 29 (2016).
[29] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, M.M. Bronstein, Geometric deep learning on graphs and manifolds using mixture model cnns, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5115–5124.
[30] M. Niepert, M. Ahmed, K. Kutzkov, Learning convolutional neural networks for graphs, in: Proceedings of the International Conference on Machine Learning, 2016, pp. 2014–2023.
[31] H. Gao, Z. Wang, S. Ji, Large-scale learnable graph convolutional networks, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1416–1424.
[32] X. Shi, X. Chai, J. Xie, T. Sun, MC-GCN: A multi-scale contrastive graph convolutional network for unconstrained face recognition with image sets, IEEE Trans. Image Process. 31 (2022) 3046–3055.
[33] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (11) (2008).
[34] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images, 2009.
[35] S.-G. Fang, D. Huang, X.-S. Cai, C.-D. Wang, C. He, Y. Tang, Efficient multi-view clustering via unified and discrete bipartite graph learning, IEEE Trans. Neural Netw. Learn. Syst. (2023) http://dx.doi.org/10.1109/TNNLS.2023.3261460.
[36] Y. Liang, D. Huang, C.-D. Wang, P.S. Yu, Multi-view graph learning by joint modeling of consistency and inconsistency, IEEE Trans. Neural Netw. Learn. Syst. (2022) http://dx.doi.org/10.1109/TNNLS.2022.3192445.
[37] D. Huang, C.-D. Wang, J.-H. Lai, C.-K. Kwoh, Toward multidiversified ensemble clustering of high-dimensional data: From subspaces to metrics and beyond, IEEE Trans. Cybern. 52 (11) (2022) 12231–12244.
[38] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, Oakland, CA, USA, 1967, pp. 281–297.
[39] L. Zelnik-Manor, P. Perona, Self-tuning spectral clustering, in: Advances in Neural Information Processing Systems, 2004.
[40] K.C. Gowda, G. Krishna, Agglomerative clustering using the concept of mutual nearest neighbourhood, Pattern Recognit. 10 (2) (1978) 105–112.
[41] D. Cai, X. He, X. Wang, H. Bao, J. Han, Locality preserving nonnegative matrix factorization, in: International Jont Conference on Artifical Intelligence, 2009.
[42] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, Adv. Neural Inf. Process. Syst. 19 (2006).
[43] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, L. Bottou, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion, J. Mach. Learn. Res. 11 (12) (2010).
[44] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, 2015, arXiv preprint arXiv:1511.06434.
[45] M.D. Zeiler, D. Krishnan, G.W. Taylor, R. Fergus, Deconvolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 2528–2535.
[46] D.P. Kingma, M. Welling, Auto-encoding variational bayes, 2013, arXiv preprint arXiv:1312.6114.
[47] J. Chang, Y. Guo, L. Wang, G. Meng, S. Xiang, C. Pan, Deep discriminative clustering analysis, 2019, arXiv preprint arXiv:1905.01681.
[48] X. Ji, J.F. Henriques, A. Vedaldi, Invariant information clustering for unsupervised image classification and segmentation, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 9865–9874.
[49] C. Niu, J. Zhang, G. Wang, J. Liang, Gatcluster: Self-supervised gaussian-attention network for image clustering, in: Proceedings of the European Conference on Computer Vision, 2020, pp. 735–751.
[50] H. Zhong, C. Chen, Z. Jin, X.-S. Hua, Deep robust clustering by contrastive learning, 2020, arXiv preprint arXiv:2008.03030.
[51] X. Wang, Z. Liu, S.X. Yu, Unsupervised feature learning by cross-level instance-group discrimination, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 12586–12595.
[52] Y. Guo, M. Xu, J. Li, B. Ni, X. Zhu, Z. Sun, Y. Xu, HCSC: Hierarchical contrastive selective coding, in: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 9696–9705.
[53] Y. Zhong, D. Huang, C.-D. Wang, Deep temporal contrastive clustering, Neural Process. Lett. (2023) http://dx.doi.org/10.1007/s11063-023-11287-0.
[54] J. Zhang, C. Li, S. Kosov, M. Grzegorzek, K. Shirahama, T. Jiang, C. Sun, Z. Li, H. Li, LCU-Net: A novel low-cost U-Net for environmental microorganism image segmentation, Pattern Recognit. 115 (2021) 107885.

Yuankun Xu received the B.Eng. degree in software engineering from Jilin University, ChangChun, China, in 2020. He is currently pursuing the master degree in computer science with the College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China. His research interests include deep clustering and graph neural network.

Dong Huang received the B.S. degree in computer science from the South China University of Technology, Guangzhou, China, in 2009, and the M.Sc. and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, in 2011 and 2015, respectively. He joined South China Agricultural University in 2015, where he is currently an Associate Professor with the College of Mathematics and Informatics. From July 2017 to July 2018, he was a Visiting Fellow with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests include pattern recognition and machine learning. He has published more than 80 papers in international journals and conferences, such as IEEE TKDE, IEEE TCYB, IEEE TSMC-S, IEEE TBD, ACM TKDD, Pattern Recognition, SIGKDD, and AAAI. He was the recipient of the 2020 ACM Guangzhou Rising Star Award.

Chang-Dong Wang received the Ph.D. degree in computer science from Sun Yat-sen University, Guangzhou, China, in 2013, where he is currently an Associate Professor with the School of Computer Science and Engineering. His current research interests include machine learning and data mining. He has published more than 100 scientific papers in international journals and conferences such as IEEE TPAMI, IEEE TKDE, IEEE TNNLS, ACM TKDD, Pattern Recognition, and SIGKDD.

Jian-Huang Lai received the Ph.D. degree in mathematics from Sun Yat-sen University, China, in 1999. He joined Sun Yat-sen University in 1989 as an Assistant Professor, where he is currently a Professor with the School of Computer Science and Engineering. His current research interests include the areas of digital image processing, pattern recognition, multimedia communication, wavelet and its applications. He has published more than 200 scientific papers in the international journals and conferences, such as IEEE TPAMI, IEEE TKDE, IEEE TIP, IEEE TSMC-B, Pattern Recognition, ICCV, CVPR, IJCAI, ICDM and SDM.