Information Sciences 610 (2022) 235–245

Contents lists available at ScienceDirect

Information Sciences

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

Self-supervised deep geometric subspace clustering network

Sangwon Baek [a,1], Gangjoon Yoon [b,1], Jinjoo Song [a], Sang Min Yoon [a,*]

[a] *HCI Lab., College of Computer Science, Kookmin University, 77 Jeongneung-ro, Seoul 02707, Republic of Korea*
[b] *National Institute for Mathematical Sciences, 70, Yuseong-daero 1689 beon-gil, Yuseong-gu, Daejeon 34047, Republic of Korea*

A R T I C L E I N F O

Article history:
Received 18 March 2022
Received in revised form 31 July 2022
Accepted 3 August 2022
Available online 5 August 2022

Keywords:
Subspace clustering
Graph mining
Deep geometric learning

A B S T R A C T

Graph mining has been widely studied to analyze real-world graph properties and applied to various applications. In particular, graph subspace clustering performance, defined as partitioning high-dimensional graph data into several clusters by finding minimum weights for the edges, has been consistently improved by exploiting deep learning algorithms with Euclidean features extracted from Euclidean domains (image datasets). Most subspace clustering algorithms tend to extract features from the Euclidean domain to identify graph characteristics and structures, and hence are limited for real-world data applications in non-Euclidean domains. This paper proposes a self-supervised deep geometric subspace clustering algorithm optimized for non-Euclidean high-dimensional graph data by emphasizing spatial features and geometric structures while simultaneously reducing redundant nodes and edges. Quantitative and qualitative experimental results verified the proposed approach is effective for graph clustering compared with previous state-of-the-art algorithms on public datasets.

© 2022 Elsevier Inc. All rights reserved.

1. Introduction

Subspace clustering [1,4,8,13,32,33], an unsupervised learning methodology, tackles the curse of dimensionality by finding relevant dimensions spanning a subspace for each cluster. Deep neural network (DNN) based subspace clustering algorithms [1,48,29,7] have significantly improved clustering performance with deep subspace clustering (DSC) network [13] developments, compared with previous hand-crafted feature based subspace clustering approaches. However, most deep learning based subspace clustering algorithms focus on designing the deep neural network to extract meaningful features and remove redundant information; using exclusively spatial or serial datasets with limited geometric properties in Euclidean domains such as images, speech signals, and texts. Thus, a generalized deep learning based subspace clustering approach is required for various-shaped data beyond the Euclidean domain to enhance applicability and feasibility for real-world applications.

Graphs comprise a set of nodes and edges, providing an ubiquitous data representation. Hence graph clustering could offer a solution for both discrete or finite data and continuous or infinite data, such as social networks, molecular graph structures, biological protein–protein networks, or recommendation systems. Graph based machine learning remains challenging because it requires incorporating information about graph structure into the machine learning model. However, understanding graph structures using hand-crafted features is hardly applicable to real-world scenarios, where label infor-

* Corresponding author.
E-mail address: smyoon@kookmin.ac.kr (S.M. Yoon).
[1] Both authors contributed equally to this work.

https://doi.org/10.1016/j.ins.2022.08.006
0020-0255/© 2022 Elsevier Inc. All rights reserved.

mation is expensive, limited, or often unavailable. Self-supervised learning enables deep neural networks to utilize unlabeled graph data by removing excessive annotated labels, and provides stable performance by effectively extracting meaningful features from the unlabeled graph data. Graph convolutional neural networks [15] have been employed as alternative solutions to non-Euclidean variables defined over geometrical domains, but few previous studies considered geometry-dependent relationships between nodes. Therefore, we propose a deep generalized subspace clustering algorithm with effective data representation and optimization for high-dimensional unlabeled data clustering by successfully finding deep latent self-representative features in non-Euclidean geometric domains [2].

The proposed approach simultaneously reduces dimensionality and emphasizes geometric and structural features from high-dimensional non-Euclidean graph data. Self-supervision and random walk regularization for unlabeled data support the proposed subspace clustering method to preserve spatial and geometric locality in deep geometric networks. Non-Euclidean graph data is converted into embedded feature space by simultaneously measuring spatial and geometric similarities to find relevant meaningful features. These meaningful features can then be utilized to represent relationships between nodes by supervising raw data combining graph convolutional networks and random walk regularization.

Significant contributions from the current paper can be summarized as follows.

- We propose a self-supervised subspace clustering methodology to effectively search and separate subspaces by forcing the network to predict and send important features to subsequent network layers.
- The proposed generalized subspace clustering methodology considers spatial and geometric localities and then applies generalized deep neural-network architectures to non-Euclidean domains to enable learning local, stationary, and compositional task-specific features.
- We experimentally verified performance improvements for the proposed approach using public datasets compared with current state of the art approaches.

The remainder of this paper is organized as follows. Section 2 reviews related works, and Section 3 explains the proposed self-supervised deep geometric subspace clustering network. Section 4 demonstrates and analyzes the proposed approach quantitatively and qualitatively, and Section 5 summarizes and conclude the paper.

2. Related work

Graph representation and clustering algorithms have been intensively researched to understand relational knowledge regarding interacting entities. This section reviews previous remarkable graph clustering and subspace clustering approaches.

2.1. Graph clustering

Graph clustering aims to partition graph nodes into several disjoint groups. Early methods adopted classical clustering algorithms, such as K-means [18] or spectral clustering [22]. Spectral based clustering was subsequently introduced with solid theoretical foundations and guaranteed performance, but spectral graph clustering requires considerable computational cost due to eigenvalue decomposition.

Centrality indices or belief propagation based approaches have been proposed to find community boundaries, a common practical graph clustering problem [9,11]. Finding a person’s social network using profile similarity has also been studied as the problem of circle detection within clustering [20]. Zhou et al. proposed a graph clustering algorithm that exploited structural and attribute similarities [49]. Yang and Leskovec proposed a model based large-scale community detection algorithm using nonnegative matrix factorization [42], and Wang et al. proposed a modularized nonnegative matrix factorization model combining network embedding and clustering [39]. Perozzi et al. proposed a social latent-representation learning algorithm, DEEPWALK, exploiting unsupervised deep learning and modeling a series of random walks [28]. Yang et al. subsequently extended DEEPWALK to text-associated DEEPWALK (TADW) by incorporating vertex information into network representation learning [41]. Zhou et al. proposed a graph clustering algorithm that exploited both structural and attribute similarities [49]. Wang et al. proposed a modularized nonnegative matrix factorization model combining network embedding and clustering [39]. Various relational topic models [6] and co-clustering methods [10] have been proposed, using content and structure information for clustering. However, these approaches suffer from only considering specific network parts or superficial relationships between content and structure.

Deep learning graph clustering has developed rapidly. Bruna et al. extended convolutional neural networks (CNNs) into graph Laplacian spectra for clustering low-dimensional graphs [3]; and various autoencoders have been employed for high-dimensional graphs, since autoencoders have similar theoretical basis to spectral clustering but are more flexible than spectral clustering in practice. Tian et al. proposed the GraphEncoder sparse encoder for graph clustering [34], and Cao et al. incorporated random surfing into a denoising autoencoder [5]. Kipf and Welling proposed a variational graph autoencoder for unsupervised graph clustering [16].

Wang et al. propose a marginalized graph autoencoder for graph clustering, employing stacked graph autoencoder and marginalizing process to effectively learn graph feature representations [38]. Pan et al. proposed two graph embedding frameworks incorporating adversarial regularization into graph and variational graph autoencoders [25], and an adversarial regularizing method was subsequently proposed employing a graph convolutional network as an encoder and including topological and content information [24].

Park et al. introduced a symmetric graph convolutional autoencoder exploiting graph structure and vertex information [26]. Wang et al. proposed a goal-directed graph attentional autoencoder to overcome two-step graph-embedding approaches, learning embedding using autoencoders and applying clustering methods to the learned embedding [37].

Graph mining and understanding can be applied to various areas, including social science, biology, physics, and medicine. Community detection, i.e., understanding and evaluating large and complex network structures, is essential to effectively analyze edge properties in graphs or networks.

2.2. Subspace clustering

Subspace clustering offers solutions for high-dimensional data clustering [36]. Most current subspace clustering methods incorporate two steps: building an affinity matrix by imposing various constraints to find relevant and discriminatory relationships among individual data points; and applying spectral clustering [22] to the affinity matrix to identify disjoint clusters.

Traditional linear subspace clustering approaches often use $L1$ or nuclear norms to decide cluster membership for data points; whereas sparse subspace clustering [8] and its variants [17,43] impose the $L1$ norm to find the sparsest representation; and low rank representation and its variants [31,45] employs the nuclear norm to find the lowest rank representation. However, these methods are either unable to capture sufficient global data structure or suffer from high computational cost. Xia et al. combined sparsity and low-rank for multi-view clustering [40]. Dictionary learning subspace clustering approaches have also been proposed for linear subspace clustering [30,32]. Although these approaches have provided remarkable clustering results they are challenging to choose the number of dictionary atoms hyper-parameter, and only apply for linear subspaces. Kernel approaches have been proposed to solve non-linear subspace clustering, but designing proper kernels remains difficult [27].

Deep learning based subspace clustering explores non-linear subspaces with high performance. Ji et al. designed a deep subspace clustering (DSC) network incorporating a self-expressive layer into an autoencoder, to solve non-linear subspace clustering in an unsupervised manner [13]. Adversarial learning [48], distribution consistency loss [47], and attention models [1] have been combined with deep neural networks to improve subspace clustering performance. Zhang et al. proposed an end-to-end self-supervised convolutional subspace clustering network, combining framework learning feature representation and self-coefficients [46]. Multi-view deep subspace clustering network is proposed to learn multi-view self-representation considering the inherent structure in an end-end manner; Kheirandishfard et al. employed stacked CNNs for multi-level feature representation [14]; Zhu and Peng extended the DSC network by regularizing the self-expressive layer with both sparsity and low-rank [50]; and Huang et al. added a feature leaning model in a supervised manner to the DSC network to improve clustering performance [12].

3. Proposed approach

This section builds a graph node subspace clustering model based on the proposed deep geometric network using a data self-expression approach. We use the connection structure (i.e., graph topology) and intrinsic graph node features to cluster connected graph nodes in a network structure (graph), separating the nodes using graph global structure and surrounding local neighborhood information.

We propose a deep geometric subspace clustering network, to first embed into low-dimensional latent feature space through graph convolutional layers, using graph node connection structure and content features; and then separate similar graph nodes using latent embeddings through self-expression. We attach three distinct regularization modules to the network to maximize benefits from convolutional feature extraction and self-expression modules. Fig. 1 shows that the proposed network comprises a graph convolutional module with autoencoder for feature extraction into lower dimensional latent embeddings, self-expressive layer for subspace clustering, and three regularization modules to promote clustering performance. The regularization modules include spectral clustering, self-supervision, and random walk regularization modules, which generate labels for graph node clustering and hence maximize convolutional layer performance.

First, we introduce notations and terminologies used throughout the paper, and then discuss the proposed self-supervised deep geometric subspace clustering network structure in more detail.

We use lowercase letters to denote scalars, real variables, and vectors; and bold upper case letters represent multi-dimensional terms such as tensors, matrices, and operators (networks). When denoting matrix components, we use parentheses for elements and square brackets for vectors; and braces represent sets. Table 1 shows symbols and notations used typically in this work.

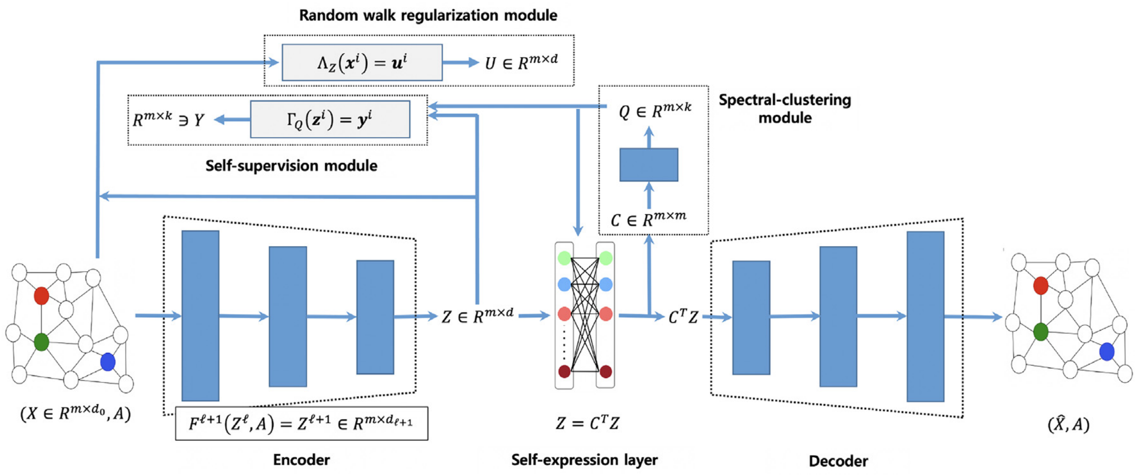


Fig. 1. Proposed self-supervised deep geometric subspace clustering network architecture with self-expression layer, fully connected layer for self-supervision, and random walk regularizer for node embedding in the graph autoencoder.

Table 1
Notations and symbols employed throughout this paper.

Notation or symbol	Description
	Dimensions and parameters
m	Number of nodes in the graph
d_0	Node feature dimension (attributes)
$d_\ell, \ell = 1,2,3$	Dimension of ℓ-th encoder layer
$d = d_3$	Latent feature dimensions
k	Number of clusters
	Vectors, matrices, and sets
$A = (a_{ij}) \in \mathbb{R}^{m\times m}$	Input adjacency matrix
$X = [\mathbf{x}^i] \in \mathbb{R}^{m\times d_0}$	Input node feature matrix
$Z^\ell = [\mathbf{z}^{\ell,j}]_{j=1}^m \in \mathbb{R}^{m\times d_\ell}$	Feature matrix for the ℓ- encoder layer
$Z = Z^3 = [\mathbf{z}^i] \in \mathbb{R}^{m\times d}$	Latent embedding feature matrix
$W^\ell \in \mathbb{R}^{d_{\ell-1}\times d_\ell}$	Weight matrix for the ℓ- encoder layer
$\hat{X} \in \mathbb{R}^{m\times d_0}$	Output node feature matrix
$C = (c_{ij}) \in \mathbb{R}^{m\times m}$	Self-expression coefficient matrix
$Q = [\mathbf{q}^i] \in \mathbb{R}^{m\times k}$	Segmentation matrix
$Y = [\mathbf{y}^i] \in \mathbb{R}^{m\times k}$	Matrix for the self-supervision module
$U = [\mathbf{u}^i] \in \mathbb{R}^{m\times d}$	Matrix for random walk regularization
$\mathcal{G} = (V, E)$	Graph with node set V and edge set E
	Functions and operators
$F^\ell\left(Z^{\ell-1}, A\right) = Z^\ell$	Feature mappings of ℓ-th encoder layer
$G\left(C^T Z\right) = \hat{X}$	Decoder mapping
$\Lambda_Z : \mathbb{R}^{d_0} \to \mathbb{R}^d$	Random walk regularizer
$\Gamma_Q : \mathbb{R}^d \to \mathbb{R}^k$	Self-supervision mapping
$\Psi(C) = Q$	Segmentation optimizer
$\mathcal{L}, \mathcal{L}_i$	Loss functions
σ^ℓ	Activation function

3.1. Preliminaries

Consider a graph $\mathcal{G} = (V, E)$ with m nodes $V = \{\mathbf{v}_i\}_{i=1}^m$ and edge set $E = \{\mathbf{e}_{ij}\}$, such that $\mathbf{e}_{ij} \in E$ if and only if there exists an edge connecting nodes $\mathbf{v}_i$ and $\mathbf{v}_j$. Let $A = (a_{ij}) \in \mathbb{R}^{m\times m}$ be the adjacency matrix for the graph representing the topology, and $X = [\mathbf{x}^i] \in \mathbb{R}^{m\times d_0}$ be the node feature matrix, where each row represents a feature vector $\mathbf{x}^i \in \mathbb{R}^{d_0}$ for node $\mathbf{v}_i$ in the graph.

3.2. Proposed network architecture

Fig. 1 shows the proposed self-supervised deep geometric subspace clustering network comprises several modules, including a graph convolutional autoencoder, self-expressive layer, fully connected layer for self-supervision, and random walk regularizer for node embedding. The encoder helps to extract spatial features and reduce input raw graph data dimensionality. However, plain autoencoder networks are limited to precisely reconstruct graph shapes while preserving spatial and geometric locality due to complexity. Subsequent sections discuss each module's role and effects to extract meaningful spatial and geometric features for graph node clustering.

3.2.1. Graph convolutional autoencoder

The feature extraction module is critical for the autoencoder framework, extracting local features for subspace clustering and reducing raw data dimensionality. The feature extraction module comprises several graph convolutional layers,

$$Z^\ell = F^\ell\left(Z^{\ell-1}, A\right) = \sigma^\ell\left(\widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}Z^{\ell-1}W^\ell\right), \tag{1}$$

where $\widetilde{A} = I_m + A$ with identity matrix I_m of size m, $\widetilde{D}$ is the diagonal matrix obtained from $\widetilde{A}$ as $\tilde{d}_{ii} = \sum_{j=1}^m \tilde{a}_{ij}$, $W^\ell \in \mathbb{R}^{d_{\ell-1}\times d_\ell}$ is the weight matrix for the ℓ-layer, and $Z^0 = X$. We use the ReLU activation function $?^\ell$ for the first two hidden ?es $(\ell = 1, 2)$, and activate the last layer with softmax.

From (1), feature $\mathbf{z}^{\ell,i}$ of $Z^\ell = \left[\mathbf{z}^{\ell,j}\right]_{j=1}^m \in \mathbb{R}^{m\times d_\ell}$ is generated from a weighted sum of features $\mathbf{z}^{\ell-1,j}$ and $\mathbf{z}^{\ell-1,i}$ in $Z^{\ell-1}$, where $\tilde{a}_{ij}$ are nonzero. These feature mappings help maintain and supplement similarity between nodes by simultaneously utilizing node graphic information A and content similarity in X. We denote the last feature matrix $Z^3 = Z \in \mathbb{R}^{m\times d}$ with $d = d_3$, and encode graph structure (from A) and node features (from X) into latent embedding Z. Graph convolutional network and properties follow that previously reported [16], and we train the autoencoder network using the loss function

$$\mathcal{L}_1 = \frac{1}{2m}\|X - \widehat{X}\|_F^2 = \frac{1}{2m}\sum_{i=1}^{m}\sum_{j=1}^{d_0}\left(x_{ij} - \hat{x}_{ij}\right)^2, \tag{2}$$

where $\widehat{X} = \left(\hat{x}_{ij}\right) \in \mathbb{R}^{m\times d_0}$ is the reconstructed output node feature matrix.

3.2.2. Self-expressive layer

Given a set of points $\left\{\mathbf{z}^i \in \mathbb{R}^d\right\}_{i=1}^m$ drawn from an unknown K linear subspaces $\left\{S_j\right\}_{j=1}^K$, each data point (feature vector) can be expressed as a linear combination of the other points using a few vector bases within the same subspace. This property is defined as self-expressiveness and is employed by most high-performance subspace clustering approaches to enhance subspace clustering results [8,13,36]. They also use self-representation for the latent feature vectors $Z, Z = C^TZ$ with coefficient $C = \left(c_{ij}\right) \in \mathbb{R}^{m\times m}$. We adopt a similar self-expression module to learn effective feature representations, and optimize the module using the loss function

$$\arg\min_C \frac{1}{2}\|Z - C^TZ\|_F^2 + \lambda\|C\|_p \quad \text{with } \mathrm{diag}(C) = 0, \tag{3}$$

where $\|\cdot\|_F$ indicates Frobenius matrix norm, $\|\cdot\|_p$ is an arbitrary matrix norm, λ is a trade-off parameter, and constraint diag $(C) = 0$ avoids the trivial identity matrix.

Self-supervision for self-expression [46] enhances clustering performance by supervising the self-expression and feature extraction modules; and we design a spectral clustering module to obtain segmentation matrix $Q \in \mathbb{R}^{m\times k}$ by solving the optimization

$$\Psi(C) = \arg\min_Q \|C\|_Q := \sum_{i,j}|c_{ij}|\frac{\|\mathbf{q}^i - \mathbf{q}^j\|_2^2}{2} \tag{4}$$

with constraint $Q^TQ = I_k$ for the identity matrix I_k.

From (4), Q-weighted ℓ_1 norm $\|C\|_Q$ measures discrepancy between the coefficient C and segmentation Q matrices. Minimizing $\|C\|_Q$ enforces C such that $c_{ij} \neq 0$ only if the i-th and j-th nodes have the same cluster labels. Q provides feedback to the self-expression module as well as graph node labels, hence supervising the encoder for robust feature extraction. Section 3.2.3 discusses filtering and emphasizing features using a self-supervision module incorporating Q.

3.2.3. Self-supervision and random walk regularization modules

The proposed subspace clustering network was devised for unsupervised learning and extracts latent feature embeddings using a graph convolutional encoder. Although CNNs work well for feature extraction, they have some limitations in that they require large labeled datasets for training, and hence likely overfitted due to limited training data. Graph node feature embeddings are also required to capture local content information and preserve network topology structure for effective graph node segmentation performance.

Therefore, we added two regularization modules (self-supervision for feature extraction and random walk regularization) to fulfil these requirement and circumvent limitations for the proposed method. We utilize a module with two fully connected layers [46] to supervise training the graph convolutional feature extraction, generating a k-dimensional output vector $\mathbf{y}$ from the latent feature embeddings, $\Gamma_Q(\mathbf{z}) = \mathbf{y} \in \mathbb{R}^k$ for $\mathbf{z} \in: \mathbb{R}^d$. If target output for Γ_Q is $\{\mathbf{q}^i\}_{i=1}^m$ for input $\{\mathbf{z}^i\}_{i=1}^m$, then we supervise graph convolutional feature extraction module training using the cross-entropy loss function,

$$\mathcal{L}_2 = \frac{1}{m}\sum_{i=1}^{m}\left(\ln\left(1 + e^{-\tilde{\mathbf{y}}^i(\mathbf{q}^i)^T}\right)\right), \tag{5}$$

where $\tilde{\mathbf{y}}^i$ is a softmax normalization for $\Gamma_Q(\mathbf{x}^i) = \mathbf{y}^i$.

We designed a stochastic random walk module to handle overfitting, capture local content information, and preserve network topology, with the following process. First, we randomly choose a predetermined number of samples Ω from node set V, and then obtain the path connected context node set $R_\mathbf{v}$ using the Random Walk with Restarts algorithm [23]. We adopted the SkipGram model [35] using context nodes in $R_\mathbf{v}$, comprising two embedding layers corresponding to nodes and context nodes, and softmax activation after the second layer. This module generates output $\mathbf{u} \in \mathbb{R}^d$ for input graph node feature data $\mathbf{x} \in \mathbb{R}^{d_0}$, denoted by $\Lambda_Z(\mathbf{x}) = \mathbf{u}$, and we train the module using loss function

$$\mathcal{L}_3 = -\sum_{\mathbf{v}\in\Omega}\sum_{\mathbf{v}_t\in R_\mathbf{v}} \ln p(\mathbf{v}_t|Z(\mathbf{v})), \tag{6}$$

where probability $p(\mathbf{v}_t|Z(\mathbf{v}))$ is joint rather than conditional probability. Hence

$$p(\mathbf{v}_t|Z(\mathbf{v})) = \Lambda_Z(\mathbf{v}_t)^T Z(\mathbf{v}) = \sum_{j=1}^{d} u_j z_j$$

for $\Lambda_Z(\mathbf{v}_t) = (u_j)_{j=1}^d; Z(\mathbf{v}) = (z_j)_{j=1}^d$ is the row vector for Z corresponding to node $\mathbf{v}$; and $\Lambda_Z(\mathbf{v}_t)$ and $Z(\mathbf{v})$ elements are probability values due to softmax activation. This random walk regularization module emphasizes geometric similarity by maximizing co-occurrence probability for features (Z) from interconnected nodes.

Algorithm 1: Proposed self-supervised deep geometric subspace clustering network

```
Input
  1. $X, A$, hyperparameters for Random Walk, $t = 1$
     trade-off parameters, other training parameters.
Pre-training
  1. initialize feature extraction encoder and decoder.
  2. initialize trivially the remaining modules.
  3. optimize only the feature extraction autoencoder using (2).
Fine-tuning
  1. initialize all layer weights
     except feature extraction encoder and decoder.
  2. optimize self-expression and
     spectral clustering layers to derive $Q$.
  3. while End_stop do
     (a) fixed $Q$, choose $\Omega \subset V$ randomly.
     (b) optimize the Random Walk module.
     (c) optimize other modules except self-expression
         and spectral clustering.
     (d) optimize self-expression and spectral clustering
         modules and update $Q$.
Output: coefficient matrix $C$.
```

3.2.4. Training the self-supervised subspace clustering network

The proposed subspace clustering network mainly comprises a feature extraction autoencoder and self-expression between encoder and decoder modules, with auxiliary modules, such as spectral clustering, self-supervision, and random walk regularization, designed to facilitate effective subspace clustering performance.

We trained the network using two sequential steps. The autoencoder network was pretrained using loss function $\mathcal{L}_1$ from (2) without activating the remaining modules, i.e., we set $C = I_m, Q = 0, \Omega = \emptyset$ and all self-supervision module weights = 0. The whole network was then trained including all modules using total loss function

$$\mathcal{L} = \mathcal{L}_1 + \lambda_2\mathcal{L}_2 + \lambda_3\mathcal{L}_3 + +\lambda_4\mathcal{L}_4 + \lambda_5\mathcal{L}_5 + \lambda_6\mathcal{L}_6, \tag{7}$$

where loss functions $\mathcal{L}_4 = \|Z - C^T Z\|_F^2, \mathcal{L}_5 = \|C\|_p$, and $\mathcal{L}_6 = \|C\|_Q$; and λ_i's are trade-off parameters for the loss functions. Algorithm 1 summarizes the network training process. Nodes are segmented after obtaining C using Laplacian spectral clustering to the affinity matrix $1/2\left(|C| + ||C|^T\right)$.

4. Experiments

4.1. Datasets

In contrast with conventional subspace clustering algorithm experiments, graph data is more desirable for comparative experiments in non-Euclidean domains than image datasets, such as COIL-20, COIL-100, ORL, E-YaleB, or MNIST. For fair comparison with previous approaches, we used three graph based network datasets: Cora, Citeseer, and PubMed for node clustering to verify subspace clustering generalizability for non-Euclidean data [19].

4.2. Network details

The proposed self-supervised deep geometric subspace clustering network in Fig. 1 was implemented in Python 3 using tensorflow 2.2 on a CUDA NVIDIA GPU (Tesla P100). The proposed autoencoder framework comprises encoder and decoder networks; where the encoder has three convolutional layers and the decoder configuration mirrors the encoder. Table 2 shows parameter details for the proposed subspace clustering network for each dataset. Trade-off parameters for the loss functions update the weights for optimizing the proposed network and enable meaningful feature extraction by considering both spatial and topological structures from high-dimensional raw data. In particular, trade-off parameters λ_3 and λ_6, which have larger values than the other parameters, confirm that self-supervision of local geometrical information contributed to improved performance for node clustering.

4.3. Performance evaluation

This section quantitatively and qualitatively evaluates and compares the proposed subspace clustering performance with previous approaches.

4.3.1. Qualitative analysis

Fig. 2 shows t-distributed stochastic neighbor embedding (t-SNE) results to help visualize clustering outcomes for each dataset, i.e., separating raw data into several groups. In particular, the proposed self-supervised learning approach enforces data to be grouped near the cluster centroid with increased inter-cluster distances by distinguishing meaningful features from raw data. Projecting the high-dimensional graph data verifies the proposed self-supervised network approach emphasizes spatial and geometric features, providing important input to find relevant features and consequently improve subspace clustering performance.

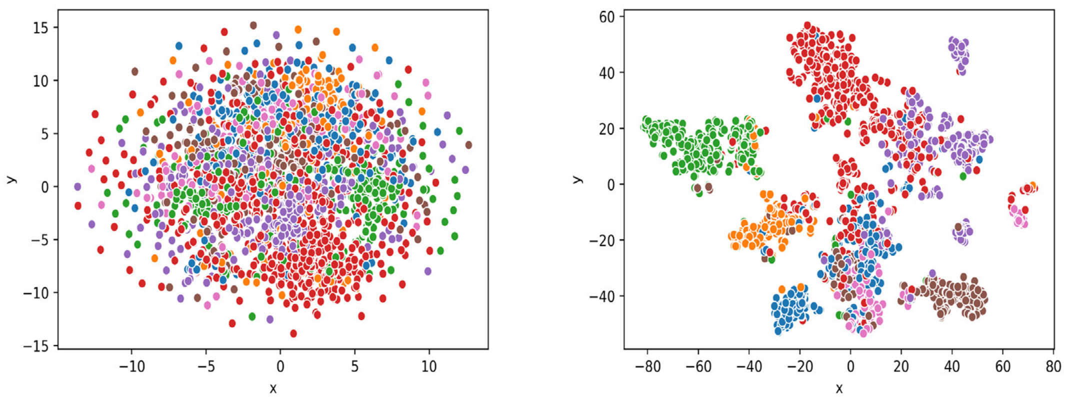
4.3.2. Quantitative analysis

The public datasets employed here have been widely used for graph node clustering, and we compared 15 representative graph node clustering approaches using accuracy (ACC), normalized mutual information (NMI), and adjusted rand index (ARI).

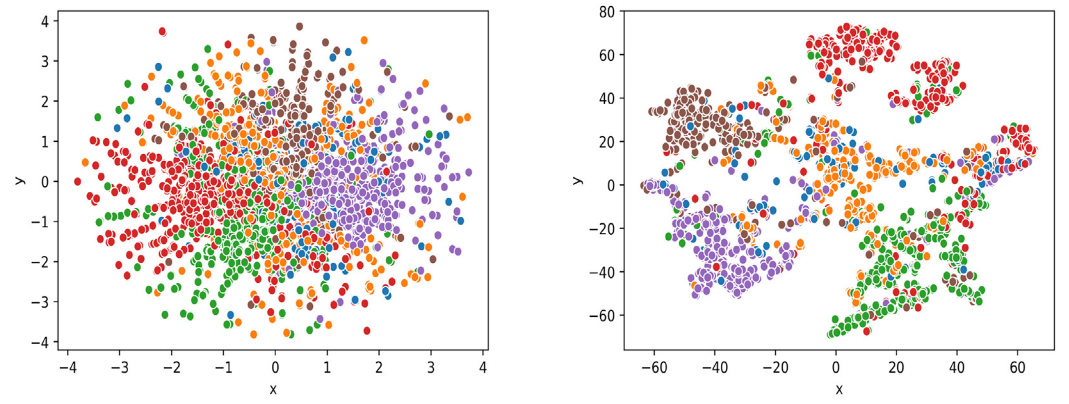
Table 3 compares node clustering outcomes for Cora, Citeseer, and PubMed datasets. The proposed algorithm outperforms all current algorithms for all datasets that have large numbers of edges and nodes. This is because jointly employing a graph convolutional network and random walk regularization extracts and emphasizes self-representative features adequately and

Table 2
Architecture information for the proposed network.

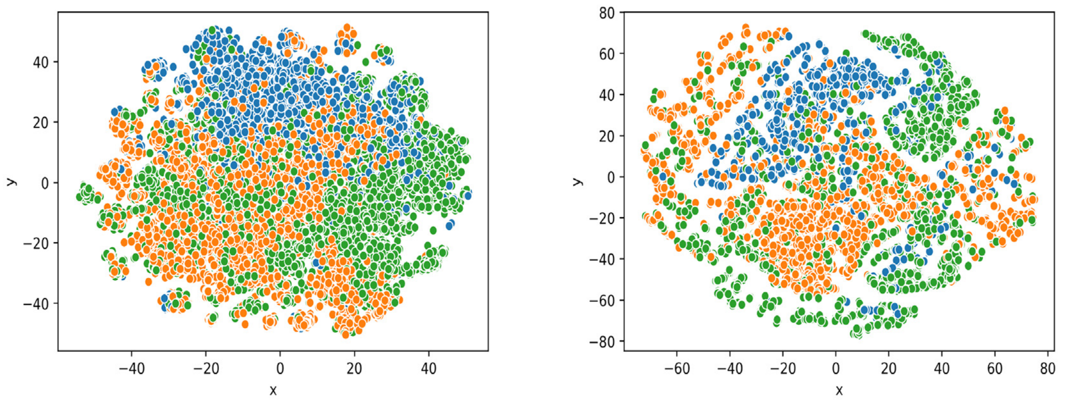
		Cora dataset	Citeseer dataset	PubMed dataset
Encoder layer dimensionality	d_1	800	2000	400
	d_2	400	1000	300
	d_3	179	462	125
Loss function parameters	λ_2	9	4	3
	λ_3	1	9	100
	λ_4	2	8	7
	λ_5	4	3	15
	λ_6	30	75	12



(a) t-SNE and the proposed approach for the raw Cora dataset



(b) t-SNE and the proposed approach for the raw Citeseer dataset



(c) t-SNE and the proposed approach fir the raw PubMed dataset

Fig. 2. t-SNE and the proposed approach for each dataset for qualitative evaluation.

removes meaningless redundant features from inputs. Current graph node clustering algorithms utilize high-dimensional information to separate the nodes into several groups, whereas the proposed subspace clustering algorithm effectively partitions nodes by reducing their dimensionality.

Table 3
Node clustering outcomes for different methods on Cora, Citeseer, and PubMed benchmark datasets. Best performance is highlighted in bold.

Method	Cora			Citeseer			PubMed		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
K-means [18]	0.4922	0.3210	0.2296	0.5401	0.3054	0.2786	0.5952	0.3152	0.2817
Spectral [22]	0.3672	0.1267	0.0311	0.2389	0.0557	0.0100	0.5282	0.0971	0.0620
Big-Calm [42]	0.2718	0.0073	0.0011	0.2500	0.0357	0.0071	–	–	–
DeepWalk [28]	0.4840	0.3270	0.2427	0.3365	0.0878	0.0922	–	–	–
GraEnc [34]	0.3249	0.1093	0.0055	0.2252	0.0330	0.0100	–	–	–
DNGR [5]	0.4191	0.3184	0.1422	0.3259	0.1802	0.0429	–	–	–
Circles [20]	0.6067	0.4042	0.3620	0.5716	0.3007	0.2930	–	–	–
RTM [6]	0.4396	0.2301	0.1691	0.4509	0.2393	0.2026	–	–	–
RMSC [40]	0.4066	0.2551	0.0895	0.2950	0.1387	0.0488	–	–	–
TADW [41]	0.5603	0.4411	0.3320	0.4548	0.2914	0.2281	–	–	–
VGAE [16]	0.5020	0.3292	0.2547	0.4670	0.2605	0.2056	0.6887	0.3108	0.3018
MGAE [38]	0.6844	0.5111	0.4447	0.6607	0.4122	0.4137	0.5932	0.2822	0.2483
ARGA [25]	0.6400	0.4490	0.3520	0.5730	0.3500	0.3410	0.6807	0.2757	0.2910
ARVGA [25]	0.6380	0.4500	0.3740	0.5440	0.2610	0.2450	0.5130	0.1169	0.0777
GALA [26]	0.7459	0.5767	0.5315	0.6932	0.4411	0.4460	0.6939	0.3273	0.3214
DMAGE [44]	0.7430	0.5820	–	0.6990	0.4350	–	0.741	0.3600	–
R-GMM-VGAE [21]	**0.7670**	0.573	**0.5790**	0.689	0.4200	0.4390	**0.7400**	0.334	**0.3790**
Proposed	0.7523	**0.5942**	0.5483	**0.7196**	**0.4521**	**0.4817**	0.7039	**0.3477**	0.3344

4.3.3. Ablation test

Most previous graph node clustering algorithms used the complete dataset, whereas the proposed deep geometric subspace clustering method separates high-dimensional graph data into several subsets. Table 4 shows ablation test outcomes for different numbers of nodes, edges, classes, and dimensions for the benchmark datasets used in the experiments; highlighting reduced dimensionality and proportion of data used for the proposed subspace clustering approach. Latent variables for Pubmed data have quite small dimensionality compared with their number of nodes. Therefore, the loss function was configured to sensitively reflect graph node feature information to ensure the latent vector inherently captured the immediate context node information. This indirectly verifies that both meaningful and meaningless information coexist within the benchmark datasets and hence subspace clustering is a good solution to clustering compressed high-dimensional data as explained in the subspace clustering definition.

Dimensionality reduction while maintaining performance is critical to solve the curse of dimensionality under when data dimensionality and computational complexity rapidly increases. Experiment results verify that the proposed approach effectively reduces input data dimensionality and simultaneously successfully extracts significant relevant features.

The proposed self-representative learning approach conveys essential features extracted from node attributes and topology to the next layer, enhancing clustering performance despite reduced dimensionality. In particular, learning geometric structures from non-Euclidean graph data guarantees robustness to preserving structural information from different graph scales in clustering.

We devised the loss function (7) to effectively extract and emphasize features from a given dataset. This loss function can be broadly separated into self-supervision ($\mathcal{L}_2$), random walk ($\mathcal{L}_3$), subspace clustering ($\mathcal{L}_4, \mathcal{L}_5$, and $\mathcal{L}_6$), and basic autoencoder network ($\mathcal{L}_1$) losses. Table 5 compare performance for various combinations of three loss functions. The devised loss functions help transmit self-representative features to subsequent layers while removing redundant noise and outliers.

Table 4
Dimensional reduction for high-dimensional datasets with different numbers of edges and nodes.

Data	Nodes	Edges	Classes	Dimension	Reduced dimension
Cora	2708	5429	7	1433	179 (12.49%)
Citeseer	3312	4732	6	3703	462(12.47%)
Pubmed	19717	44338	3	500	125(25%)

Table 5
Ablation tests for various loss function combinations, amongst subspace clustering, self-supervised, and random walk loss.

Method	Cora			Citeseer			PubMed		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
Subspace clustering loss	0.749	0.5767	0.5315	0.6932	0.4411	0.4460	0.6939	0.3273	0.3214
Subspace clustering and self-supervision loss	0.7463	0.5819	0.5374	0.6973	0.4428	0.4483	0.6940	0.3333	0.3298
Subspace clustering and random walk loss	0.7466	0.5931	0.5316	0.6935	0.4419	0.4683	0.6999	0.3222	0.3219
Total loss	**0.7523**	**0.5942**	**0.5483**	**0.7196**	**0.4521**	**0.4817**	**0.7039**	**0.3477**	**0.3344**

5. Discussion

This paper proposed a generalized subspace clustering approach to effectively reduce data dimensionality and successfully extract self-representative features from non-Euclidean high-dimensional data. Previous subspace clustering algorithms focused on analyzing features in Euclidean domains, whereas the proposed approach uses various loss functions to emphasize spatial and geometric features in non-Euclidean domains. The proposed autoencoder framework effectively extracts meaningful spatial information using self-representative features from high-dimensional raw data, and random walk and self-supervised regularizers successfully optimize the proposed network considering topological structure information from graph data to derive relevant geometric features in non-Euclidean domains. Quantitative and qualitative evaluations verified that the proposed self-supervised subspace clustering provides robust results despite only using small portions of the input data. The proposed algorithm can be used as a basic technique for various areas, including authorship attribution and community detection.

The limited amount of adjacent graphical data is represented as nodes, which are connected whenever they share a minimum semantic content by extracting as well as emphasizing self-representative features in the non-Euclidean space.

Large-scale subspace clustering is an important topics, but still suffers from high computational and spatial complexity, causing clustering difficulties for large datasets. Most previous large-scale subspace clustering algorithms treat the problem as an out of sample extension for subspace clustering on a few selected landmark data points, where the clustering problem on the remainder is solved via classification. However, most previous algorithms were only devised and/or tested for Euclidean datasets; whereas the proposed approach deploys various landmark feature extraction methods for large scale graph data subspace clustering.

CRediT authorship contribution statement

Sangwon Baek: Data curation, Validation, Visualization, Writing – original draft. **Gangjoon Yoon:** Methodology, Software, Writing – original draft, Writing – review & editing. **Jinjoo Song:** Investigation, Validation, Writing – review & editing, Funding acquisition. **Sang Min Yoon:** Conceptualization, Funding acquisition, Writing – review & editing.

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.

Acknowledgments

G.-J.Yoon was supported by the National Institute for Mathematical Sciences grant funded by the Korean government (grant NIMS-B22810000). J. Song was supported by the National Research Foundation of Korea (grant 2021R1F1A1059202). S.M. Yoon was supported by the Institute of Information communications Technology Planning and Evaluation (grant 2020–0-00457) and the National Research Foundation of Korea (grant NRF-2021R1A2C1008555).

References

[1] S. Baek, G. Yoon, J. Song, S.M. Yoon, Deep self-representative subspace clustering network, Pattern Recognit. 118 (2021) 108041.
[2] M.M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, Geometric deep learning: Going beyond euclidean data, IEEE Signal Process. Mag. 34 (4) (2017) 18–42.
[3] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and deep locally connected networks on graphs, Proc. Int. Conf. Learn. Represent. (2014).
[4] Y. Cai, M. Zeng, Z. Cai, X. Liu, Z. Zhang, Graph regularized residual subspace clustering network for hyperspectral image clustering, Inf. Sci. 578 (2021) 85–101.
[5] S. Cao, W. Lu, Q. Xu, Deep neural networks for learning graph representations, Proc. AAAI Conf. Artif. Intell. (2016).
[6] J. Chang, D. Blei, Relational topic models for document networks, Proc. Artif. Intell. Stat. (2009) 81–88.
[7] Z. Dang, C. Deng, X. Yang, H. Huang, Multi-scale fusion subspace clustering using similarity constraint, Proc. CVPR (2020) 6657–6666.
[8] E. Elhamifar, R. Vidal, Sparse subspace clustering: Algorithm, theory, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 35 (11) (2013) 2765–2781.
[9] M. Girvan, M.E. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. 99 (2002) 7821–7826.
[10] X. Guo, L. Gao, X. Liu, J. Yin, Improved deep embedded clustering with local structure preservation, in: Proc. IJCAI, 2017, pp. 1753–1759.
[11] M.B. Hastings, Community detection as an inference problem, Phys. Rev. E 74 (3) (2006) 035102.
[12] Q. Huang, Y. Zhang, H. Peng, T. Dan, W. Weng, H. Cai, Deep subspace clustering to achieve jointly latent feature extraction and discriminative learning, Neurocomputing 404 (2020) 340–350.
[13] P. Ji, T. Zhang, H. Li, M. Salzmann, I. Reid, Deep subspace clustering networks, Proc. NeurIPS 30 (2017) 24–33.
[14] M. Kheirandishfard, F. Zohrizadeh, and F. Kamangar, Multi-level representation learning for deep subspace clustering, in: Proc. Conf. Appl. Comput. Vis., 2020, pp. 2039–2048.
[15] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.
[16] T.N. Kipf, M. Welling, Variational graph auto-encoders. arXiv preprint arXiv:1611.07308, 2016.
[17] C.-G. Li, R. Vidal, Structured sparse subspace clustering: A unified optimization framework, Proc. CVPR (2015) 277–286.
[18] S. Lloyd, Least squares quantization in pcm, IEEE Trans. Inf. Theory 28 (2) (1982) 129–137.
[19] B. London, L. Getoor, Collective classification of network data, in: C.C. Aggarwal (Ed.), Data Classification: Algorithms and Applications, CRC Press, 2014, pp. 399–416.
[20] J.J. McAuley, J. Leskovec, Learning to discover social circles in ego networks, Proc. NeurIPS (2012) 548–556.

[21] N. Mrabah, M. Bouguessa, M.F. Touati, R. Ksantini, Rethinking graph auto-encoder models for attributed graph clustering. CoRR, abs/2107.08562, 2021.
[22] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, Proc. NeurIPS (2002) 849–856.
[23] J.-Y. Pan, H.-J. Yang, C. Faloutsos, P. Duygulu, Automatic multimedia cross-modal correlation discovery, Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (2004) 653–658.
[24] S. Pan, R. Hu, S.-F. Fung, G. Long, J. Jiang, C. Zhang, Learning graph embedding with adversarial training methods, IEEE Trans. Cybern. 50 (6) (2019) 2475–2487.
[25] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, Adversarially regularized graph autoencoder for graph embedding, in: Proc. IJCAI, pages 2609–2615, 2018.
[26] J. Park, M. Lee, H.J. Chang, K. Lee, and J.Y. Choi, Symmetric graph convolutional autoencoder for unsupervised graph representation learning, in: Proc. ICCV, 2019, pp. 6519–6528.
[27] V.M. Patel and R. Vidal, Kernel sparse subspace clustering, in: Proc. ICIP, 2014, pp. 2849–2853.
[28] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (2014) 701–710.
[29] J. Seo, J. Koo, T. Jeon, Deep closed-form subspace clustering, in: 2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27–28, 2019, IEEE, 2019, pp. 633–642.
[30] J. Shen, P. Li, Learning structured low-rank representation via matrix factorization, Artif. Intell. Stat. (2016) 500–509.
[31] J. Shen, P. Li, H. Xu, Online low-rank subspace clustering by basis dictionary pursuit, in: Proc. ICML, 2016, pp. 622–631.
[32] J. Song, G. Yoon, K. Hahn, S.M. Yoon, Subspace clustering via structure-enforced dictionary learning, Neurocomputing 362 (2019) 1–10.
[33] Ł. Struski, J. Tabor, P. Spurek, Lossy compression approach to subspace clustering, Inf. Sci. 435 (2018) 161–183.
[34] F. Tian, B. Gao, Q. Cui, E. Chen, T.-Y. Liu, Learning deep representations for graph clustering, Proc. AAAI Conf. Artif. Intell. (2014).
[35] Vaibhav, P.-Y. Huang, R. Frederking, Rwr-gae: Random walk regularization for graph auto encoders. arXiv preprint arXiv:1908.04003, 2019.
[36] R. Vidal, Subspace clustering, IEEE Signal Process. Mag. 28 (2) (2011) 52–68.
[37] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, C. Zhang, Attributed graph clustering: A deep attentional embedding approach. arXiv preprint arXiv:1906.06532, 2019.
[38] C. Wang, S. Pan, G. Long, X. Zhu, J. Jiang, MGAE: marginalized graph autoencoder for graph clustering, in: Proc. ACM Int. Conf. Inf. Knowl. Manag., 2017, pp. 889–898.
[39] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, S. Yang, Community preserving network embedding, Proc. AAAI Conf. Artif. Intell. (2017).
[40] R. Xia, Y. Pan, L. Du, J. Yin, Robust multi-view spectral clustering via low-rank and sparse decomposition, in: Proc. AAAI Conf. Artif. Intell., 2014, pp. 2149–2155.
[41] C. Yang, Z. Liu, D. Zhao, M. Sun, E.Y. Chang, Network representation learning with rich text information, in: Proc. IJCAI, 2015, pp. 2111–2117.
[42] J. Yang, J. Leskovec, Overlapping community detection at scale: a nonnegative matrix factorization approach, Proc. ACM Int. Conf. Web Search Data Min. (2013) 587–596.
[43] C. You, C.-G. Li, D.P. Robinson, R. Vidal, Oracle based active set algorithm for scalable elastic net subspace clustering, in: Proc. CVPR, 2016, pp. 3928–3937.
[44] Z. Zang, S. Li, D. Wu, J. Guo, Y. Xu, S.Z. Li, Unsupervised deep manifold attributed graph embedding. CoRR, abs/2104.13048, 2021.
[45] H. Zhai, H. Zhang, L. Zhang, P. Li, Laplacian-regularized low-rank subspace clustering for hyperspectral image band selection, IEEE Trans. Geosci. Remote Sens. 57 (3) (2018) 1723–1740.
[46] J. Zhang, C.-G. Li, C. You, X. Qi, H. Zhang, J. Guo, Z. Lin, Self-supervised convolutional subspace clustering network, in: Proc. CVPR, 2019, pp. 5473–5482.
[47] L. Zhou, B. Xiao, X. Liu, J. Zhou, E.R. Hancock, et al., Latent distribution preserving deep subspace clustering, in: Proc. IJCAI, 2019, pp. 4440–4446.
[48] P. Zhou, Y. Hou, J. Feng, Deep adversarial subspace clustering, Proc. CVPR (2018) 1596–1604.
[49] Y. Zhou, H. Cheng, J.X. Yu, Graph clustering based on structural/attribute similarities, Proc. VLDB Endow. 2 (2009) 718–729.
[50] W. Zhu, B. Peng, Sparse and low-rank regularized deep subspace clustering, Knowl.-Based Syst. 204 (2020) 106199.