

Deep Clustering with Intra-class Distance Constraint for Hyperspectral Images

Jinguang Sun, Wanli Wang, Xian Wei, *Member, IEEE*, Li Fang, *Member, IEEE*, Xiaoliang Tang, Yusheng Xu, *Member, IEEE*, Hui Yu, and Wei Yao

***Abstract*—The high dimensionality of hyperspectral images often results in the degradation of clustering performance. Due to the powerful ability of potential feature extraction and nonlinear representation, deep clustering algorithms have become a hot topic in hyperspectral remote sensing. Different tasks often need different features. However, the current deep clustering algorithms generally separate feature extraction from clustering, which results in the extracted features are not constrained by clustering tasks. Therefore, the features extracted by these algorithms may not be suitable for clustering. To address this issue, we adopt intra-class distance as a constraint condition and proposed an intra-class distance constrained deep clustering algorithm for hyperspectral images. The proposed algorithm propagates the clustering error back to the feature mapping process of the auto-encoder network, so as to realize the constraint of clustering objective on feature extraction and make the extracted features more suitable for clustering tasks. In addition, the proposed algorithm simultaneously completes network optimization and clustering, which is more efficient. Experimental results demonstrate the intense competitiveness of the proposed algorithm in comparison with state-of-the-art clustering methods for hyperspectral images.**

***Index Terms*—Deep learning, hyperspectral images clustering, intra-class distance constraint, low-dimensional representation, remote sensing.**

I. Introduction

With the development of remote sensing technology, sensors have more and more powerful ability to collect spectral information. From multispectral images to hyperspectral images, the sensing data is sampling the electromagnetic spectrum with higher and higher details [1]–[8]. Utilizing these myriad sensors, the Earth Observation System (EOS) generates massive practical images of various land covering objects. Owing to abundant spatial and spectral information, these numerous images make it possible to extend the applications of hyperspectral remote sensing to many potential fields [9]–[15]. However, it is very arduous to annotate these massive images effectively. Due to the lack of labeled high-dimensional samples of hyperspectral images, learning appropriate low-dimensional (LD) representations of data for clustering plays a critical role in hyperspectral image annotation and understanding.

Clustering is the task of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups [16]. For remote sensing images, the task is to classify the pixels into homogeneous regions which segment every image into different partitions [1], [7], [17]. In most cases, traditional clustering algorithms, such as K-means [18], ISODATA [19], Fuzzy C-means [4], [6], can get good performance. However, when the data distribution is complex, they often fail. Some mathematical methods are proposed to extract features for complex data, such as the fast and robust recursive algorithm [20], and the unsupervised nonlinear diffusion algorithm [21]. Based on these mathematical methods, novel clustering algorithms [22], [23] are developed, which have a great improvement in computing efficiency and clustering effect. For example, [22] made a valuable contribution to the identification of outliers. However, these approaches are based on shallow models [24]–[27], which can only use the shallow features of data for classification. During the past decades, spectral-based clustering methods [5], [28]–[32] and density-based [7] clustering methods have been state-of-the-art. Let $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n] \in \mathbb{R}^{m \times n}$ be the matrix containing n independent samples arranged as its columns, the spectral-based clustering approaches perform clustering in the following two steps. At first, an affinity matrix $\boldsymbol{C}$ is built to depict the relationship of the data, where $\boldsymbol{C}_{ij}$ denotes the similarity between data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Secondly, the data is clustered through clustering the eigenvectors of the graph Laplacian $\boldsymbol{L} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$, where $\boldsymbol{D}$ is a diagonal matrix with $\boldsymbol{D}_{ii} = \sum_{j=1}^{n} \boldsymbol{C}_{ij}$ and $\boldsymbol{A} = \boldsymbol{D} - \boldsymbol{C}$. By solving the eigenvalues and eigenvectors of $\boldsymbol{L}$, the original data can be transformed into an embedded low dimensional feature space. However, how to choose the number of embedded dimensions is still an unsolved problem. Considering that spectral clustering algorithm depends on its similarity matrix, different similarity matrices will lead to different clustering results [30], [33]–[37]. The main idea of density-based clustering [38] approaches are to find high-density regions that are separated by low-density regions. The density peaks clustering algorithm (DPCA) proposed by

This work was partially supported by the Young Scientists Fund of the National Natural Science Foundation of China under Grant No. 61602226 and No. 61806186, and the CAS Pioneer Hundred Talents Program (Type C) under Grant No. 2017-122. *(Wanli Wang and Jinguang Sun contribute equally to this work.) (Corresponding author: Xian Wei.)*

J. Sun and W. Wang are with the School of Electronic and Information Engineering, Liaoning Technical University, Huludao 125105, Liaoning, China (email: Sunjinguang@lntu.edu.cn; email: wwlswj@163.com).

X. Wei, L. Fang, X. Tang and H. Yu are with the Quanzhou Institute of Equipment Manufacturing, Haixi Institute, Chinese Academy of Sciences, Quanzhou 362216, Fujian, China (email: xian.wei@fjirsm.ac.cn).

Y. Xu is with the College of Surveying and Geo-Informatics, Tongji University, 1239 Siping Road, Shanghai 200092, China.

W. Yao is with the Department of Land Surveying and Geo-Informatics, Hong Kong Polytechnic University, 181 Chatham Road South, Hung Hom, Kowloon, Hong Kong.

Alex Rodriguez [39] has brought the density-based clustering approaches to a new stage. The core idea of DPCA is that the centre of the cluster is surrounded by some points with low local density, and these points are far away from other points with high local density. The DPCA separates the clustering process of non-clustered centre points into a single process. Due to the selection of the cluster centre and the classification of the non-cluster points are separated, the clustering precision is increased.

To solve the clustering problem of more complex distributed data, the sparse subspace clustering (SSC) algorithm [13], [40] is developed. The core idea of SSC is that among the infinite number of representations of a data point in terms of other points, a sparse representation corresponds to selecting a few points from the same subspace. In fact, the SSC algorithm is a solution of sparse optimization in the framework of spectral clustering, and it evaluates the labels for every sample in the low-dimensional space. Sharing the homogeneous idea with SSC, many sparse representation and low-rank approximation based methods for subspace clustering [24], [41]–[44] have received a lot of attention in recent years. The key components of these methods are finding a sparse and low-rank representation of the data and then building a similarity graph on the sparse coefficient matrix for the separation of the data.

Although spectral-based clustering algorithms and density-based clustering algorithms can effectively cluster arbitrarily distributed data, only shallow features of the data can be used [25]. Moreover, it is difficult to further improve the clustering effect and precision. On the other hand, deep neural networks are multilayer feedforward networks, in which each layer is a nonlinear function [45]. According to [45]–[47], deep neural networks are capable of approximating any measurable function to any desired degree of accuracy. Therefore, in recent years, the subspace clustering algorithm based on deep learning has attracted more attention.

The core idea of deep neural network clustering [48]–[52] is to non-linearly map data from the original feature space to a new feature space and then complete clustering in the new feature space. Because its mapping is non-linear and multiple, the deep neural network clustering has powerful capabilities on potential deep-seated feature extraction and data representation [53]–[55]. The clustering algorithm based on auto-encoder networks [50], [56]–[58] is a popular framework for deep clustering algorithms, which utilizes a symmetric network structure to encode and represent the data. Such an algorithm consists of two important steps. Firstly, it obtains the code space of the data by reducing the data dimension and clusters the data in the obtained code space. Secondly, it performs the representation transformation from the obtained code space to a new generative feature space. Depending upon the auto-encoder network, the idea of generative adversarial [59]–[62] is introduced into clustering field, which further improves the performance of deep clustering algorithms.

Generally speaking, current deep clustering algorithms have the form [41], [57], [63], [64] as

$$\mathcal{J}_p = \mathcal{J}_{p1} + \underbrace{\lambda \left\| \boldsymbol{Z}^{\left(\frac{M}{2}\right)} - \boldsymbol{Z}^{\left(\frac{M}{2}\right)}\boldsymbol{C} \right\|^2}_{\mathcal{J}_{p2}} + \mathcal{J}_{p3}, \tag{1}$$

where $\mathcal{J}_p$ is the loss function, $\mathcal{J}_{p1}$ represents the reconstruction error, $\mathcal{J}_{p2}$ is a sparse or low-rank constraint determined by the pre-trained matrix $\boldsymbol{C}$, and $\mathcal{J}_{p3}$ is the regularization term. The λ in $\mathcal{J}_{p2}$ represents the constraint coefficient, and $\boldsymbol{Z}^{\left(\frac{M}{2}\right)}$ denotes the obtained codes or extracted features by the auto-encoder. As described in [65], if the constraint matrix $\boldsymbol{C} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_n]$ is sparse, it can be obtained by solving the following problem:

$$\begin{cases} \min\limits_{\boldsymbol{C}} \sum\limits_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{X}\boldsymbol{c}_i\|_2^2 + \lambda \|\boldsymbol{C}\|_1 \\ \text{s.t. } \boldsymbol{c}_{ii} = 0 \end{cases}, \tag{2}$$

where $\|\cdot\|_1$ denotes the ℓ_1-norm that is usually used to measure sparsity, $\boldsymbol{c}_{ii}$ is the i-th entry of the column vector $\boldsymbol{c}_i$. If the constraint matrix $\boldsymbol{C}$ is low-rank, it can be obtained by solving a nuclear norm minimization problem [66]:

$$\min_{\boldsymbol{C}} \|\boldsymbol{C}\|_F + \lambda \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{C}\|_F^2. \tag{3}$$

The data representation shown in (1) achieves both data reconstruction and dimensionality reduction. Therefore, the features extracted by such algorithms are appropriate for data dimensionality reduction. Separating feature extraction from clustering, these algorithms use final sparse or low-rank features to cluster. However, different tasks need different features, without clustering-oriented constraint, features extracted by these algorithms may not be suitable for clustering tasks. To solve this problem, we propose an embedded deep clustering algorithm with intra-class distance constraint, which aims to develop a better feature extraction approach for deep clustering algorithm and demonstrate the feasibility of using task-oriented constraints to optimize the feature extraction process, so as to improve the clustering effect of deep clustering algorithms. The proposed algorithm embeds the intra-class distance into the auto-encoder network so that it can constrain the procedure of data mapping and obtain data representation which is more conducive to clustering.

The main contribution of this paper is that we proposed an approach to dynamically adjust the feature extraction of deep clustering networks by using clustering error as a penalty. The proposed approach makes features extracted by deep clustering networks more suitable for clustering. Moreover, the proposed method is a joint learning method, which can complete clustering and feature extraction and improve efficiency.

II. Related Works

In this section, we briefly discuss some existing works in unsupervised deep learning and subspace clustering.

A. Auto-encoder Network

Hinton first proposed the concept of the auto-encoder network and gave its basic mathematical model [67]. Then he proved that the auto-encoder network has a strong ability of dimension reduction and feature extraction by [68]. Baldi [69] proposed a general mathematical framework for auto-encoder networks. Sperduti [70] proved that the problem of training an auto-encoder network has a closed-form solution. These

existing research results provide theoretical support for the auto-encoder network.

With impressive learning and characterization capabilities, auto-encoder neural networks have achieved great success in various areas, especially in the scenario of unsupervised learning [2], [58], [71]–[73], such as natural language processing [74], image processing [75], object detection [76], biometric recognition [77], and data analysis [78]. As state-of-the-art unsupervised techniques, auto-encoder and auto-encoder based neural networks also make outstanding contributions in the field of remote sensing. In this subsection, we briefly introduce the auto-encoder network.

In general, an auto-encoder [56] is a kind of network that consists of encoder and decoder, and the structure of the encoder and decoder is symmetrical. If an auto-encoder has multiple hidden layers, then the number of the hidden layer of the encoder is equal to that of the decoder. The structural model of the basic auto-encoder is shown in Fig. 1. The purpose of the basic auto-encoder is to reconstruct the input data at the output layer, and the perfect case is that the output signal $\boldsymbol{X}_{out}$ (i.e., $\boldsymbol{Z}^{(M)}$) is exactly same as the input signal $\boldsymbol{X}_{in}$ (i.e., $\boldsymbol{Z}^{(0)}$). According to the structure shown in Fig. 1, the encoding process and the decoding process of the basic auto-encoder can be described as

$$\boldsymbol{z}^{(i+1)} = \mathcal{F}_e\left(\boldsymbol{W}_i \boldsymbol{z}^{(i)} + \boldsymbol{b}_i\right)$$

and

$$\boldsymbol{z}^{(j+1)} = \mathcal{F}_d\left(\boldsymbol{W}_j \boldsymbol{z}^{(j)} + \boldsymbol{b}_j\right)$$

respectively, where $\boldsymbol{W}_i$, $\boldsymbol{b}_i$ denote the i-th encoding weight and the i-th encoding bias respectively, $\boldsymbol{W}_j$, $\boldsymbol{b}_j$ represent the j-th decoding weight and the j-th decoding bias respectively, $\boldsymbol{z}^{(i)}$ denotes the data vector in i-th layer, and $\mathcal{F}_e$ is the non-linear transformation. *Sigmoid*, *Tanh*, *Relu* are commonly used activation functions for $\mathcal{F}_e$. $\mathcal{F}_d$ is the reverse non-linear transformation to the encoding process. Therefore, the loss function of the basic auto-encoder is to minimize the error between $\boldsymbol{X}_{in}$ and $\boldsymbol{X}_{out}$. The encoder converts the input signal into codes through some non-linear mapping, and the decoder tries to remap the codes to the input signal. The parameters of the auto-encoder, i.e., weights and biases, are learned by minimizing the total reconstruction error, which could be computed by the mean square error

$$\begin{aligned}\mathcal{J}_m(\boldsymbol{W}, \boldsymbol{b}) &= \sum_{i=1}^{n} \mathcal{L}\left(\boldsymbol{z}_i^{(0)}, \boldsymbol{z}_i^{(M)}\right) \\ &= \sum_{i=1}^{n} \left\| \boldsymbol{z}_i^{(0)} - \boldsymbol{z}_i^{(M)} \right\|_2^2,\end{aligned} \tag{4}$$

or the cross entropy

$$\begin{aligned}&\mathcal{J}_c(\boldsymbol{W}, \boldsymbol{b}) = \sum_{i=1}^{n} \mathcal{L}\left(\boldsymbol{z}_i^{(0)}, \boldsymbol{z}_i^{(M)}\right) \\ &= -\sum_{i=1}^{n} \left(\boldsymbol{z}_i^{(0)} log \boldsymbol{z}_i^{(M)} + \left(\boldsymbol{1} - \boldsymbol{z}_i^{(0)}\right) log\left(\boldsymbol{1} - \boldsymbol{z}_i^{(M)}\right)\right).\end{aligned} \tag{5}$$

For the structure shown in Fig. 1, the hidden layers of the basic auto-encoder network have three different structures: a compressed structure, a sparse structure, and an equivalent-dimensional structure. When the number of the neuron in the input layer is greater than that in the hidden layer, it is called the compressed structure [65]. Conversely, when the number of the neurons of the input layer is smaller than that of the hidden layer, it is called the sparse structure. If the two are equal, it is named the equivalent-dimensional structure.

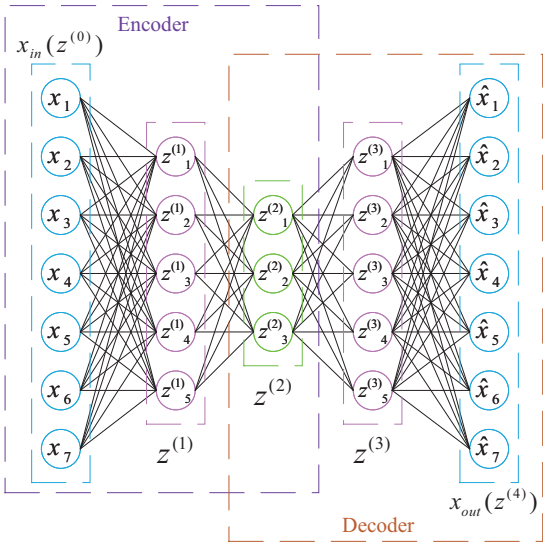

Fig. 1. The structure of the basic auto-encoder network.

B. Deep Subspace Clustering

Subspace clustering algorithms transform data from high dimensional feature space to low dimensional feature space. Elhamifar proposed a manifold clustering algorithm based on sparse representation [79]. The algorithm can automatically find the neighbors and their weights. However, the algorithm still needs to improve the theoretical guarantee of manifold control. Benefiting from the powerful capabilities of non-linear modelling and data representation of deep neural networks, some clustering approaches based on deep neural networks have been proposed in recent years. Song et al. [56] integrated an auto-encoder network with K-means to cluster the latent features extracted from original data. However, the feature mapping and clustering are two relatively independent processes in their work, and the K-means algorithm is not jointed into the feature mapping process. Therefore, the feature mapping process may not be constrained by the K-means algorithm. With the emergence of generative adversarial networks, some deep clustering algorithms [49], [62] embedded discriminant and adversarial ideas have been proposed, which further enhance the ability of deep feature extraction and data representation. On the other hand, some deep subspace clustering algorithms with prior constraints have been developed. According to different constraining conditions, these prior constraints may be sparse [65], low-rank [66], and least-square [51]. Based on the auto-encoder network, the loss functions of these algorithms share the same modality, shown in (6). These algorithms learn representation for the input data with minimal reconstruction error and incorporate prior information into the

potential feature learning to preserve the key reconstruction relation over the data. Although the features extracted by such algorithms are appropriate for dimensionality reduction and reconstruction of the data, they may not be appropriate for clustering without constraints of the relationships between classes.

$$\begin{aligned}\mathcal{J}_p\left(\boldsymbol{W}_i,\boldsymbol{b}_i\right)=&\underbrace{\left\|\boldsymbol{Z}^{(0)}-\boldsymbol{Z}^{(M)}\right\|_F^2}_{\mathcal{J}_{p1}}\\&+\underbrace{\lambda_1\left\|\boldsymbol{Z}^{\left(\frac{M}{2}\right)}-\boldsymbol{Z}^{\left(\frac{M}{2}\right)}\boldsymbol{C}\right\|_F^2}_{\mathcal{J}_{p2}}\\&+\underbrace{\lambda_2\sum_{i=1}^{M}\left(\|\boldsymbol{W}_i\|_F^2+\|\boldsymbol{b}_i\|_2^2\right)}_{\mathcal{J}_{p3}},\end{aligned}\tag{6}$$

where $\mathcal{J}_{p1}$ denotes the reconstruction error, $\mathcal{J}_{p2}$ denotes the prior constraint, $\mathcal{J}_{p3}$ denotes the regularization term and $\boldsymbol{C}$ is a pre-trained matrix. Aforementioned deep clustering algorithms have following three weaknesses:

1) Lacking prior constraints related to clustering task.
2) The matrix $\boldsymbol{C}$ needs to be pre-trained, which may not be optimal for various data to be clustered.
3) Once given, the matrix $\boldsymbol{C}$ is fixed, which cannot be optimized jointly with the network.

Different from these existing works, we propose an approach that embeds intra-class distance into an auto-encoder network. The indicator matrix and the parameters of the network are adaptively optimized simultaneously. The proposed approach utilizes intra-class distance to constrain the feature mapping process of the auto-encoder so that the deep features extracted from the source space are more conducive for clustering. More details of the proposed approach are described in the following sections.

III. The Proposed Deep Clustering Approach

In this section, we elaborate on the details of the proposed *Deep Clustering with Intra-class Distance Constraint* (DCIDC) algorithm. The framework of DCIDC is an auto-encoder network, and the specific structure may vary according to different scenarios. With the constraint of intra-class distance, DCIDC extracts potential features of the data by mapping them from the source space to a latent feature space. In the new latent feature space, objects in the same group are more similar to each other than to those in other groups. We first explain how DCIDC is specifically designed and then present the algorithm for optimizing the DCIDC model.

A. Deep Clustering with Intra-class Distance Constraint

The neural network within DCIDC consists of $M+1$ layers performing M non-linear transformations, where M is an even number. The first $\frac{M}{2}$ hidden layers are encoders to learn a set of compact representations (i.e., low-dimensional representations) and the last $\frac{M}{2}$ layers are decoders to progressively reconstruct the input. The framework of DCIDC is shown as Fig. 2. Let $\boldsymbol{Z}^{(0)}=\boldsymbol{X}_{in}\in\mathbb{R}^{N\times D}$ be one input matrix to the first layer, which denotes a hyperspectral image consisting of N image pixels (samples) and $\boldsymbol{z}^{(0)}$ be one row of the matrix, which denotes a sample in D-dimensional feature space. For the encoder, the output of the i-th layer is computed by

$$\boldsymbol{z}^{(i)}=\mathcal{F}_e\left(\boldsymbol{W}_i\boldsymbol{z}^{(i-1)}+\boldsymbol{b}_i\right)\in\mathbb{R}^{d_i},\tag{7}$$

where $i=1,2,\cdots,\frac{M}{2}$ indexes the layers of the encoder, $\boldsymbol{W}_i$ denotes the weight matrix from the $(i-1)$-th layer to the i-th layer and $\boldsymbol{b}_i$ denotes the bias of the i-th layer. $\mathbb{R}^{d_i}$ indicates that $\boldsymbol{z}^{(i)}$ belongs to a d_i-dimensional feature space. The $\mathcal{F}_e\left(\cdot\right)$ is a non-linear activation function. The $\frac{M}{2}$-th layer $\boldsymbol{z}^{\left(\frac{M}{2}\right)}\in\mathbb{R}^{d_{\frac{M}{2}}}$ is shared by the encoder and the decoder. For the purpose of reducing the dimensionality of the input data, the dimensions of the layers in the encoder are designed to be $D\geq d_{i-1}\geq d_i\geq d_{\frac{M}{2}}$. For the decoder, the output of the j-th layer can be computed by

$$\boldsymbol{z}^{(j)}=\mathcal{F}_d\left(\boldsymbol{W}_j\boldsymbol{z}^{(j-1)}+\boldsymbol{b}_j\right)\in\mathbb{R}^{d_j},\tag{8}$$

where $j=\frac{M}{2}+1,\frac{M}{2}+2,\cdots,M$ indexes the layers of the decoder. The non-linear activation function $\mathcal{F}_d\left(\cdot\right)$ can be the same as $\mathcal{F}_e\left(\cdot\right)$ or another different non-linear function. For the purpose of data reconstruction, the dimensions of the layers in the decoder are designed to be $d_{\frac{M}{2}}\leq d_{j-1}\leq d_j\leq d_M=D$. Thus, given a sample $\boldsymbol{z}^{(0)}$ (i.e., $\boldsymbol{x}_{in}$) as one input of the first layer of DCIDC, $\boldsymbol{z}^{(M)}$ (i.e., $\boldsymbol{x}_{out}$) is the reconstruction of $\boldsymbol{z}^{(0)}$, and the corresponding $\boldsymbol{z}^{\left(\frac{M}{2}\right)}$ is the representation of $\boldsymbol{x}_{in}$. Furthermore, for a data matrix $\boldsymbol{Z}^{(0)}=\left[\boldsymbol{z}_1^{(0)},\boldsymbol{z}_2^{(0)},\cdots,\boldsymbol{z}_N^{(0)}\right]^T\in\mathbb{R}^{N\times D}$ which denotes a collection of N given samples, the output matrix of the decoder $\boldsymbol{Z}^{(M)}=\left[\boldsymbol{z}_1^{(M)},\boldsymbol{z}_2^{(M)},\cdots,\boldsymbol{z}_N^{(M)}\right]^T\in\mathbb{R}^{N\times D}$ is the corresponding reconstruction of $\boldsymbol{Z}^{(0)}$, and the $\boldsymbol{Z}^{\left(\frac{M}{2}\right)}=\left[\boldsymbol{z}_1^{\left(\frac{M}{2}\right)},\boldsymbol{z}_2^{\left(\frac{M}{2}\right)},\cdots,\boldsymbol{z}_N^{\left(\frac{M}{2}\right)}\right]^T\in\mathbb{R}^{N\times d_{\frac{M}{2}}}$ is the desired low-dimensional representation of $\boldsymbol{Z}^{(0)}$.

The objective of DCIDC is to minimize the data reconstruction error and jointly constrain the non-linear transformation from $\boldsymbol{X}_{in}$ to the corresponding representation $\boldsymbol{Z}^{\left(\frac{M}{2}\right)}$ by intra-class distance. Thus, these targets can be formally stated as

$$\begin{aligned}\min_{\boldsymbol{W}_i,\boldsymbol{b}_i}\mathcal{J}\left(\boldsymbol{W}_i,\boldsymbol{b}_i\right)=&\underbrace{\left\|\boldsymbol{Z}^{(0)}-\boldsymbol{Z}^{(M)}\right\|_F^2}_{\mathcal{J}_1}\\&+\underbrace{\lambda_1\left\|\boldsymbol{Z}^{\left(\frac{M}{2}\right)}-\boldsymbol{H}\boldsymbol{S}^T\right\|_F^2}_{\mathcal{J}_2}\\&+\underbrace{\lambda_2\sum_{i=1}^{M}\left(\|\boldsymbol{W}_i\|_F^2+\|\boldsymbol{b}_i\|_2^2\right)}_{\mathcal{J}_3},\end{aligned}\tag{9}$$

where λ_1 and λ_2 are positive trade-off coefficients. The terms $\mathcal{J}_1$, $\mathcal{J}_2$, and $\mathcal{J}_3$ are respectively designed for different goals. By minimizing the first term, we get the point-to-point reconstruction of the input matrix, so that every column of the output

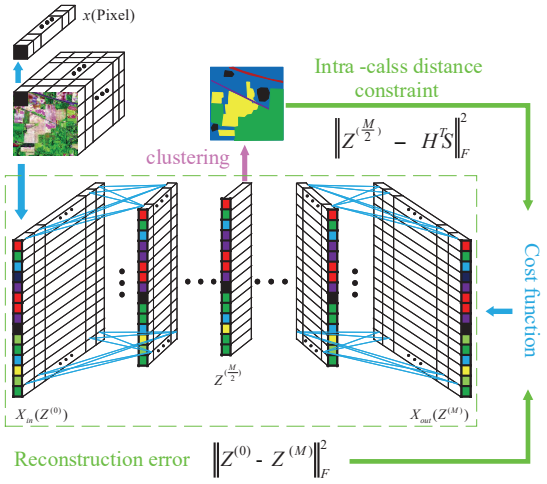

Fig. 2. The network framework of the proposed algorithm. A hyperspectral image is an input matrix. During each iteration, as parts of the cost function, intra-class distance and reconstruction error jointly constrain the procedure of feature mapping form $\boldsymbol{Z}^{(0)}$ to $\boldsymbol{Z}^{(\frac{M}{2})}$.

matrix is as close as possible to the corresponding column in the input matrix. Thus, the first term $\mathcal{J}_1$ is designed to preserve the pixel-level locality by minimizing reconstruction errors. In other words, the input acts as a supervisor for the procedure of learning a low-dimensional representation $\boldsymbol{Z}^{(\frac{M}{2})}$. For the purpose that objects in the same cluster have similar features, the term $\mathcal{J}_2$ is designed to constrain the non-linear transformation from $\boldsymbol{Z}^{(0)}$ to its corresponding representation $\boldsymbol{Z}^{(\frac{M}{2})}$, by minimizing the clustering error in each iteration. The matrix $\boldsymbol{S} \in \mathbb{R}^{d_{\frac{M}{2}} \times K}$ in (11) denotes the clustering centres of which each column represents one cluster centre. The matrix $\boldsymbol{H} \in \mathbb{R}^{N \times K}$ in (12) is the indicator matrix of which each row denotes the binary label. To get the matrix $\boldsymbol{H}$, let $\boldsymbol{Z} = \boldsymbol{Z}_1 \cup \boldsymbol{Z}_2 \cup \cdots \cup \boldsymbol{Z}_K$ be the samples that are classified into K clusters and $\boldsymbol{S} = [\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_K]$ be the corresponding cluster centres. The intra-class error is calculated by

$$\begin{aligned}\mathcal{E}_{intra-class} &= \sum_{\boldsymbol{z}_j \in \boldsymbol{Z}_1} \left\| \boldsymbol{z}_j - \boldsymbol{s}_1^T \right\| + \sum_{\boldsymbol{z}_j \in \boldsymbol{Z}_2} \left\| \boldsymbol{z}_j - \boldsymbol{s}_2^T \right\| \\ &\quad + \cdots + \sum_{\boldsymbol{z}_j \in \boldsymbol{Z}_K} \left\| \boldsymbol{z}_j - \boldsymbol{s}_K^T \right\| \\ &= \sum_{i=1}^{K} \sum_{\boldsymbol{z}_j \in \boldsymbol{Z}_i} \left\| \boldsymbol{z}_j - \boldsymbol{s}_i^T \right\| = \left\| \boldsymbol{Z} - \boldsymbol{H}\boldsymbol{S}^T \right\|,\end{aligned} \tag{10}$$

thus we get the matrix $\boldsymbol{H}$, for each row of which, there is only one element is 1 and the rest are 0s. For initializing the matrix $\boldsymbol{H}$, its values are randomly generated. Once initialized, $\boldsymbol{H}$ is not fixed and it is updated during each iteration. The K in (11) denotes the number of clusters. At last, $\mathcal{J}_3$ is a regularization term to avoid over-fitting.

$$\boldsymbol{S} = \begin{bmatrix} s_{11} & \cdots & s_{1K} \\ \vdots & \ddots & \vdots \\ s_{d_{\frac{M}{2}}1} & \cdots & s_{d_{\frac{M}{2}}K} \end{bmatrix}, \tag{11}$$

and

$$\boldsymbol{H} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{12}$$

Our neural network model uses the input as self-supervisor to learn low-dimensional representations and jointly constrain the non-linear transformation by minimizing the clustering error, which is expected to enhance the deep intrinsic features extracted from the source data. The learned representations are fully adaptive andfavourable for the clustering process. Furthermore, our model completes clustering in one step without additional pre-training process, which improves the efficiency.

B. Optimization Procedure

In this subsection, we mainly demonstrate how the proposed DCIDC model can be optimized efficiently via gradient descent and the process of solving $\boldsymbol{H}$. As the optimization of parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ does not share the same mechanism with $\boldsymbol{H}$ and $\boldsymbol{S}$, we present the gradient descent and the calculation of $\boldsymbol{H}$ and $\boldsymbol{S}$ respectively. For the convenience of developing the algorithm, we rewrite (9) in the following sample-wise form:

$$\begin{aligned}\mathcal{J} &= \sum_{i=1}^{N} \left\| \boldsymbol{z}_i^{(0)} - \boldsymbol{z}_i^{(M)} \right\|_2^2 \\ &\quad + \lambda_1 \sum_{i=1}^{N} \left\| \boldsymbol{z}_i^{(\frac{M}{2})} - \boldsymbol{h}_i \boldsymbol{S}^T \right\|_2^2 \\ &\quad + \lambda_2 \sum_{m=1}^{M} \left(\|\boldsymbol{W}_m\|_F^2 + \|\boldsymbol{b}_m\|_2^2 \right).\end{aligned} \tag{13}$$

According to(7), (8) and the chain rule, the gradients w.r.t. $\boldsymbol{W}_m$ and $\boldsymbol{b}_m$ give

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{W}_m} = \left(\boldsymbol{\Delta}_m + \lambda_1 \boldsymbol{\Lambda}_m\right) \left(\boldsymbol{z}_i^{(m-1)}\right)^T + \lambda_2 \boldsymbol{W}_m, \tag{14}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{b}_m} = \boldsymbol{\Delta}_m + \lambda_1 \boldsymbol{\Lambda}_m + \lambda_2 \boldsymbol{b}_m, \tag{15}$$

where $\boldsymbol{\Delta}_m$ is defined as

$$\boldsymbol{\Delta}_m = \begin{cases} -\left(\boldsymbol{z}_i^{(0)} - \boldsymbol{z}_i^{(M)}\right) \odot \mathcal{G}'\left(\boldsymbol{y}_i^{(M)}\right) & m = M \\ \left(\boldsymbol{W}_{m+1}\right)^T \boldsymbol{\Delta}_{m+1} \odot \mathcal{G}'\left(\boldsymbol{y}_i^{(m)}\right) & \text{otherwise} \end{cases}, \tag{16}$$

and $\boldsymbol{\Lambda}_m$ is given as

$$\boldsymbol{\Lambda}_m = \begin{cases} \left(\boldsymbol{W}_{m+1}\right)^T \boldsymbol{\Lambda}_{m+1} \odot \mathcal{G}'\left(\boldsymbol{y}_i^{(m)}\right) & m = 1, \cdots, \frac{M}{2} - 1 \\ \left(\boldsymbol{z}_i^{(\frac{M}{2})} - \boldsymbol{h}_i \boldsymbol{S}^T\right) \odot \mathcal{G}'\left(\boldsymbol{y}_i^{(\frac{M}{2})}\right) & m = \frac{M}{2} \\ \boldsymbol{0} & m = \frac{M}{2} + 1, \cdots, M \end{cases}, \tag{17}$$

where the $\odot$ denotes element-wise multiplication, $\boldsymbol{y}_i^{(m)} = \boldsymbol{W}_m \boldsymbol{z}_i^{(m-1)} + \boldsymbol{b}_m$, and $\mathcal{G}'(\cdot)$ is the derivative of the activation function $\mathcal{G}(\cdot)$ defined as

$$\mathcal{G}(\cdot) = \begin{cases} \mathcal{F}_e(\cdot) & m = 1, \cdots, \frac{M}{2} \\ \mathcal{F}_d(\cdot) & m = \frac{M}{2} + 1, \cdots, M \end{cases}. \tag{18}$$

Using the gradient descent algorithm, we update $\{\boldsymbol{W}_m, \boldsymbol{b}_m\}_{m=1}^{M}$ as (19) and (20) until convergence:

$$\boldsymbol{W}_m \leftarrow \boldsymbol{W}_m - \mu \frac{\partial \mathcal{J}}{\partial \boldsymbol{W}_m}, \tag{19}$$

$$\boldsymbol{b}_m \leftarrow \boldsymbol{b}_m - \mu \frac{\partial \mathcal{J}}{\partial \boldsymbol{b}_m}, \tag{20}$$

where $\mu > 0$ is the learning rate which is typically set to a small value according to specific scenarios.

As the output of DCIDC model, the indicator matrix $\boldsymbol{H}$ is calculated via the least distance rule in each clustering step. In other words, $\boldsymbol{H}$ is updated by solving the term (21) in every iteration:

$$\min_{\boldsymbol{H},\boldsymbol{S}} \left\| \boldsymbol{Z}^{\left(\frac{M}{2}\right)} - \boldsymbol{H}\boldsymbol{S}^T \right\|_F^2. \tag{21}$$

Thus, with the accomplishment of the optimization of DCIDC model, the final $\boldsymbol{H}$ will be simultaneously obtained. Now, we demonstrate the solution of (21).

For the task of clustering, the matrix $\boldsymbol{Z}^{\left(\frac{M}{2}\right)}$ can be redefined as

$$\boldsymbol{Z}^{\left(\frac{M}{2}\right)} = \{\boldsymbol{Z}_i^{\left(\frac{M}{2}\right)}\}_{i=1}^{K}, \tag{22}$$

where $\boldsymbol{Z}_i^{\left(\frac{M}{2}\right)}$ is a component of $\boldsymbol{Z}^{\left(\frac{M}{2}\right)}$, which denotes one cluster of the data, $\bigcup_{i=1}^{K} \boldsymbol{Z}_i^{\left(\frac{M}{2}\right)} = \boldsymbol{Z}^{\left(\frac{M}{2}\right)}$ and $\boldsymbol{Z}_i^{\left(\frac{M}{2}\right)} \cap \boldsymbol{Z}_j^{\left(\frac{M}{2}\right)} = \emptyset$ w.r.t. $i \neq j \in \{1, 2, \cdots, K\}$. Then, for the convenience of solving (21), it can be transformed into

$$\min_{\boldsymbol{H},\boldsymbol{S}} \mathcal{E}_{\boldsymbol{S}} = \sum_{i=1}^{K} \sum_{\boldsymbol{z}^{\left(\frac{M}{2}\right)} \in \boldsymbol{Z}_i^{\left(\frac{M}{2}\right)}} \left\| \boldsymbol{s}_i^T - \boldsymbol{z}^{\left(\frac{M}{2}\right)} \right\|_2^2, \tag{23}$$

where $\boldsymbol{s}_i$ is a column vector of $\boldsymbol{S}$, which denotes the centre of $\boldsymbol{Z}_i^{\left(\frac{M}{2}\right)}$. Let

$$\begin{aligned} \frac{\partial \mathcal{E}_{\boldsymbol{S}}}{\partial \boldsymbol{S}} &= \frac{\partial}{\partial \boldsymbol{S}} \sum_{i=1}^{K} \sum_{\boldsymbol{z}^{\left(\frac{M}{2}\right)} \in \boldsymbol{Z}_i^{\left(\frac{M}{2}\right)}} \left\| \boldsymbol{s}_i^T - \boldsymbol{z}^{\left(\frac{M}{2}\right)} \right\|_2^2 \\ &= \sum_{i=1}^{K} \sum_{\boldsymbol{z}^{\left(\frac{M}{2}\right)} \in \boldsymbol{Z}_i^{\left(\frac{M}{2}\right)}} \frac{\partial}{\partial \boldsymbol{S}} \left\| \boldsymbol{s}_i^T - \boldsymbol{z}^{\left(\frac{M}{2}\right)} \right\|_2^2 = 0, \end{aligned} \tag{24}$$

we get

$$\boldsymbol{s}_i^T = \frac{1}{n_i} \sum_{\boldsymbol{z}^{\left(\frac{M}{2}\right)} \in \boldsymbol{Z}_i^{\left(\frac{M}{2}\right)}} \boldsymbol{z}^{\left(\frac{M}{2}\right)}, \tag{25}$$

where n_i is the number of pixel of the cluster $\boldsymbol{Z}_i^{\left(\frac{M}{2}\right)}$.

Similarly, for the purpose of calculating $\boldsymbol{H}$, the (21) can be rewritten as

$$\min_{\boldsymbol{H},\boldsymbol{S}} \mathcal{E}_{\boldsymbol{H}} = \sum_{i=1}^{K} \sum_{\boldsymbol{z}^{\left(\frac{M}{2}\right)} \in \boldsymbol{Z}_i^{\left(\frac{M}{2}\right)}} \left\| \left(\boldsymbol{z}^{\left(\frac{M}{2}\right)}\right)^T - \boldsymbol{S}\boldsymbol{h}^T \right\|_2^2, \tag{26}$$

where $\boldsymbol{h}$ is the indicator corresponding to $\boldsymbol{z}^{\left(\frac{M}{2}\right)}$, which is serialized as a row vector of $\boldsymbol{H}$. Let

$$\begin{aligned} &\frac{\partial}{\partial \boldsymbol{h}^T} \left\| \left(\boldsymbol{z}^{\left(\frac{M}{2}\right)}\right)^T - \boldsymbol{S}\boldsymbol{h}^T \right\|_2^2 \\ &= \frac{\partial}{\partial \boldsymbol{h}^T} \left\| \boldsymbol{S}\boldsymbol{h}^T - \left(\boldsymbol{z}^{\left(\frac{M}{2}\right)}\right)^T \right\|_2^2 \\ &= \frac{\partial}{\partial \boldsymbol{h}^T} [\boldsymbol{S}\boldsymbol{h}^T - \left(\boldsymbol{z}^{\left(\frac{M}{2}\right)}\right)^T]^T [\boldsymbol{S}\boldsymbol{h}^T - \left(\boldsymbol{z}^{\left(\frac{M}{2}\right)}\right)^T] \\ &= 2\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{h}^T - 2\boldsymbol{S}^T \left(\boldsymbol{z}^{\left(\frac{M}{2}\right)}\right)^T = 0, \end{aligned} \tag{27}$$

we get

$$\boldsymbol{h}^T = \left(\boldsymbol{S}^T\boldsymbol{S}\right)^{-1} \boldsymbol{S}^T \left(\boldsymbol{z}^{\left(\frac{M}{2}\right)}\right)^T. \tag{28}$$

Thus, we get

$$\boldsymbol{H} = [\mathcal{T}(\boldsymbol{h}_1), \mathcal{T}(\boldsymbol{h}_2), \cdots, \mathcal{T}(\boldsymbol{h}_N)]^T, \tag{29}$$

where

$$\mathcal{T}(\boldsymbol{h}) = \mathcal{T}\{h_i\}_{i=1}^{K} = \begin{cases} 1 & h_i = \max\left(\{h_i\}_{i=1}^{K}\right) \\ 0 & \text{otherwise} \end{cases}. \tag{30}$$

So far, the detailed procedure for optimizing the proposed model DCIDC can be summarized as Algorithm 1.

Algorithm 1 Algorithm of Deep Clustering with Intra-class Distance Constraint

Input: The data matrix $\boldsymbol{X}_{in}$ (i.e., $\boldsymbol{Z}^{(0)}$) and the number of cluster K.

Output: The indicator matrix $\boldsymbol{H}$.

```
Initialize a matrix H according to (12) and the given K.
for i = 1 to M do
    Initialize W_i, b_i by random generation.
end for
while not convergence do
    for i = 1 to M/2 do
        z^(i) ← F_e(W_i z^(i-1) + b_i).
    end for
    for j = M/2 + 1 to M do
        z^(j) ← F_d(W_j z^(j-1) + b_j).
    end for
    Calculate J_1 in (9).
    Calculate s_i by (25) and S ← [s_1, s_2, ..., s_K].
    Calculate J_2 in (9).
    Calculate J_3 in (9).
    Update H by (29).
    for i = 1 to M do
        Update W_i by (19).
        Update b_i by (20).
    end for
end while
return H.
```

IV. Experimental Results

In this section, we compare the proposed DCIDC approach with popular clustering methods on four image datasets in

terms of clustering purity [12] and normalized mutual information(NMI) [12], [16], [49], [56], [63], [80]. For clustering algorithms, the number of clusters is an important factor affecting the clustering results. Therefore, we set a series of cluster numbers for each dataset. Given the same cluster numbers, the experiment results illustrate the superiority of the DCIDC algorithm, compared with the popular clustering methods. The network structure affects the performance of the DCIDC algorithm, which is discussed by setting different network layers and the number of nodes in the middle layer. We also discuss the influence on DCIDC with different activation functions and different coefficient values of λ_1 and λ_2. Comparing with popular clustering methods in terms of time consumption, we also demonstrate the higher efficiency of the proposed algorithm. The main computing resources used in the experiments include an Intel Core i7-6700 CPU, a 16G RAM(Random-Access Memory), an NVIDIA GeForce GTX1080Ti graphics card. The software platforms used are Python 3.6 and TensorFlow.

A. Experimental Settings

1) Datasets: We carry out our experiments using four hyperspectral image datasets: Indian Pines, Pavia, Salinas, and Salinas-A. The Indian Pines dataset was gathered by the 224-band AVIRIS sensor over the Indian Pines test site in North-western Indiana. It consists of 145×145 pixels and 224 spectral reflectance bands in the wavelength range of $0.4 \sim 2.5 \times 10^{-6}$ meters. The number of bands of the Indian Pines dataset used in our experiment is reduced to 200 by removing bands covering the region of water absorption. The ROSIS sensor acquired the Pavia dataset during a flight campaign over Pavia, northern Italy. The used Pavia dataset in our experiment is a 1096×1096 pixels image with 100 spectral bands. The AVIRIS sensor collected the Salinas dataset over Salinas Valley, California, characterized by the high spatial resolution. The area covered comprises 512 lines by 217 samples. As with Indian Pines scene, 24 water absorption bands are discarded. A small sub-scene of Salinas image, denoted as Salinas-A, is adopted too.

2) Evaluation Criteria: We adopt two metrics to evaluate the clustering quality: purity and NMI. Higher values of these metrics indicate better performance. Let $\boldsymbol{\Omega} = \{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \cdots, \boldsymbol{\omega}_K\}$ be the set of clustering results and $\boldsymbol{G} = \{\boldsymbol{g}_1, \boldsymbol{g}_2, \cdots, \boldsymbol{g}_{\widehat{K}}\}$ be the set of ground truth. The purity and NMI are defined as:

$$purity\left(\boldsymbol{\Omega}, \boldsymbol{G}\right) = \frac{1}{N} \sum_{i=1}^{K} \max_{j \in \{1, \cdots, \widehat{K}\}} |\boldsymbol{\omega}_i \cap \boldsymbol{g}_j|, \tag{31}$$

and

$$\mathrm{NMI}\left(\boldsymbol{\Omega}, \boldsymbol{G}\right) = \frac{\mathrm{I}\left(\boldsymbol{\Omega}, \boldsymbol{G}\right)}{\left[\mathrm{H}\left(\boldsymbol{\Omega}\right) + \mathrm{H}\left(\boldsymbol{G}\right)\right]/2}, \tag{32}$$

where $|\cdot|$ denotes the number of elements in the set, $\mathrm{I}\left(\cdot\right)$ represents the mutual information and $\mathrm{H}\left(\cdot\right)$ is the entropy. They are defined as follows:

$$\begin{aligned} \mathrm{I}\left(\boldsymbol{\Omega}, \boldsymbol{G}\right) &= \sum_{i=1}^{K} \sum_{j=1}^{\widehat{K}} \mathcal{P}\left(\boldsymbol{\omega}_i \cap \boldsymbol{g}_j\right) \log \frac{\mathcal{P}\left(\boldsymbol{\omega}_i \cap \boldsymbol{g}_j\right)}{\mathcal{P}\left(\boldsymbol{\omega}_i\right) \mathcal{P}\left(\boldsymbol{g}_j\right)} \\ &= \sum_{i=1}^{K} \sum_{j=1}^{\widehat{K}} \frac{|\boldsymbol{\omega}_i \cap \boldsymbol{g}_j|}{N} \log \frac{N|\boldsymbol{\omega}_i \cap \boldsymbol{g}_j|}{|\boldsymbol{\omega}_i||\boldsymbol{g}_j|} \end{aligned}, \tag{33}$$

and

$$\begin{aligned} \mathrm{H}\left(\boldsymbol{\Omega}\right) &= -\sum_{i=1}^{K} \mathcal{P}\left(\boldsymbol{\omega}_i\right) \log \mathcal{P}\left(\boldsymbol{\omega}_i\right) \\ &= -\sum_{i=1}^{K} \frac{|\boldsymbol{\omega}_i|}{N} \log \frac{|\boldsymbol{\omega}_i|}{N} \end{aligned}, \tag{34}$$

where $\mathcal{P}\left(\cdot\right)$ is the probability that a given sample belongs to the set.

3) Baseline Algorithms: For the sake of fairness, we compare DCIDC with clustering algorithms that carried out experiments with the same four datasets: Indian Pines, Pavia, Salinas, and Salinas-A. These algorithms are the discriminative embedding based on set-to-set and sample-to-sample distances (LDE) [81], the deep subspace clustering with sparsity prior (PARTY) [65], the auto-encoder based subspace clustering (AESSC) [66], the fast spectral clustering (FSC) [82], the sparse subspace clustering (SSC) [31], the latent subspace sparse subspace clustering (LS3C) [42], the low-rank representation based clustering (LRR) [43], the low-rank based subspace clustering (LRSC) [83], the smooth representation clustering (SMR) [84], the spectral clustering (SC) [30], the gaussian mixture model clustering (GMM) [12] and the K-means clustering (KC). Among these methods, FSC, LDE, PARTY and AESSC are deep clustering methods.

For each dataset, we set 5 groups of orthogonal hyper-parameter: a group of cluster number Ks, a group of λ_1s, a group of λ_2s, a group of node numbers of the middle layer and a group of layer numbers of the network. The Ks are designed to evaluate the performance of the DCIDC algorithm under the condition of different cluster numbers. On the other hand, they are used to compare the performance of the proposed approach and the other methods under the condition of the same cluster number. The λ_1 and the λ_2 are aimed to evaluate the influence of the trade-off coefficients on the performance of DCIDC, and the other two groups of hyper-parameters are for demonstrating the influence of network structure on the proposed algorithm. For the reference algorithms, we repeat each of them 100 times in terms of each K on each dataset. Finally, we take the averages of the purity and NMI values as the experimental results of theses algorithms on these datasets. For the proposed DCIDC algorithm, we repeat the experiment 100 times in terms of each hyper-parameter on each dataset, then, we obtain the optimal network structure based on the best combination of the 5 kinds of hyper-parameter. We take the average purity and NMI of the 100 experiments using the optimal network structure as the final results of DCIDC.

B. Comparison with the Evaluated Methods

In this subsection, we show the clustering results of the proposed algorithm and the baseline algorithms on four datasets, respectively. The clustering purity and NMI are displayed in separate tables, and the data in the tables quantitatively show the scores of different clustering algorithms. Each row in the tables describes the scores of the same clustering algorithm with different cluster numbers, and each column shows the scores of different algorithms with the same cluster number. The boldface number represents the highest score of the column, which represents the best algorithm with the given cluster number. The following graphs qualitatively describe the changing trend of these approaches with different cluster numbers. For expressing the clustering effect intuitively, we also show the clustering result images below.

1) Indian Pines: Tab. I and the Tab. II show the clustering purity and NMI on the dataset Indian Pines respectively. As can be seen from Tab. I, when $K = 3$ and 4, the algorithm LDE gets the highest scores. When K takes the other values, the algorithm DCIDC reaches the best scores. When $K = 14$, the DCIDC performs best with a 87.2% score, which is 2.43% higher than the algorithm LDE that ranks second. In general, the proposed algorithm DCIDC performs better than the other algorithms, which is also verified by the NMI in Tab. II.

Fig. 3(a) and Fig. 3(b) show that most of the algorithms access their best results around $K = 14$, so does the DCIDC. Fig. 3(a) and Fig. 3(b) also show that the larger the deviation between the cluster number K and 14, the worse the performance of the algorithms.

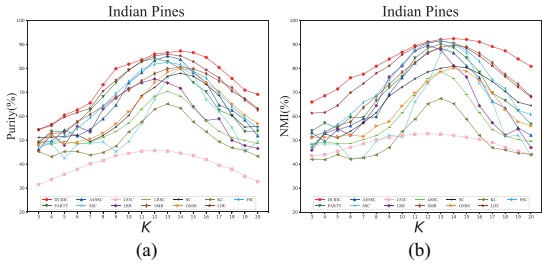

Fig. 3. The changing trend of different algorithms' clustering performance with cluster numbers on the dataset Indian Pines. (a) shows the changing trend of the clustering purity and (b) is the changing trend of NMI.

Fig. 4 intuitively shows the clustering results of the algorithms on the Indian Pines dataset.

2) Pavia: Tab. III and Tab. IV show that the proposed DCIDC algorithm gets the best clustering results on the dataset Pavia in most cases. Its best Purity and NMI are 89.81% and 92.45% respectively. In Tab. III, when $K = 20$, the shallow model algorithm LRR gets the second place, whose score is 12.2%, 8.12%, and 4.9% higher than the deep-learning-based algorithms PARTY, LDE, and FSC respectively. This maybe because that their matrix $\boldsymbol{C}$ cannot be dynamically adjusted, thus, they are not applicable when the distribution of the data to be clustered is different from that used for training the matrix $\boldsymbol{C}$.

It can be seen from Fig. 5(a) and Fig. 5(b) that both the clustering purity and NMI of the DCIDC reach the maximum value when $K = 10$, and its curve is relatively smooth, which shows that the proposed algorithm has good robustness.

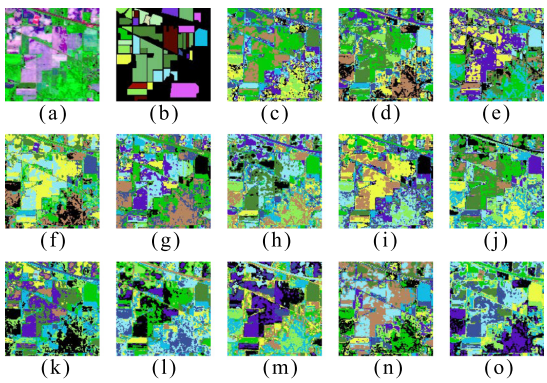

Fig. 4. The performance comparison of different algorithms on the dataset Indian Pines. (a) is the false color image of Indian Pines, (b) is the ground truth image, (c) DCIDC, (d) PARTY, (e) AESSC, (f) SSC, (g) LS3C, (h) LRR, (i) LRSC, (j) SMR, (k) SC, (l) GMM, (m) KC, (n) LDE, (o) FSC.

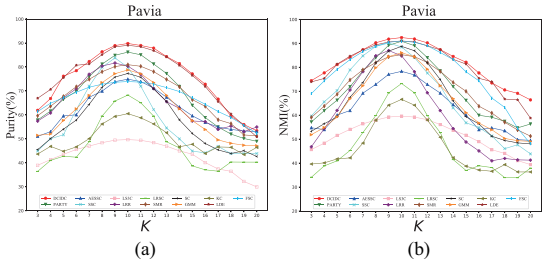

Fig. 5. The changing trend of different algorithms' clustering performance with cluster numbers on the dataset Pavia. (a) shows the changing trend of the clustering purity and (b) describes the changing trend of NMI.

Fig. 6 displays the clustering results of the algorithms on the Pavia dataset.

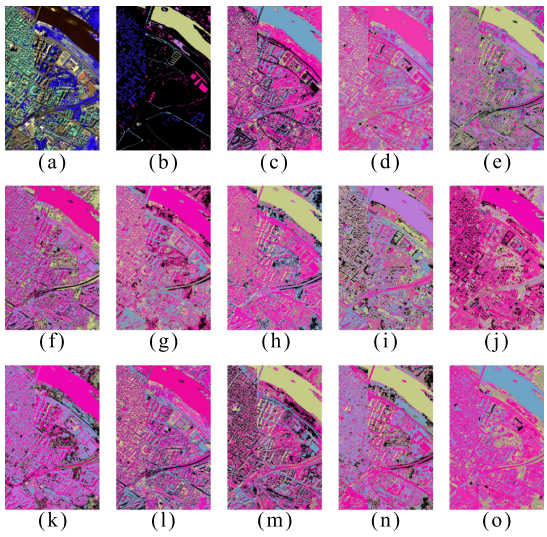

Fig. 6. The performance comparison of different algorithms on the dataset Pavia. (a) is the false color image of Pavia, (b) is the ground truth image, (c) DCIDC, (d) PARTY, (e) AESSC, (f) SSC, (g) LS3C, (h) LRR, (i) LRSC, (j) SMR, (k) SC, (l) GMM, (m) KC, (n) LDE, (o) FSC.

TABLE I
CLUSTERING PURITY OF THE DIFFERENT CLUSTERING ALGORITHMS FOR THE DATASET INDIAN PINES

Methods	Purity(%)																	
	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10	K=11	K=12	K=13	K=14	K=15	K=16	K=17	K=18	K=19	K=20
DCIDC	54.38	56.14	**60.49**	**62.59**	**65.52**	**73.15**	**79.83**	**81.73**	**83.56**	**85.94**	**86.57**	**87.24**	**86.55**	**84.58**	**80.30**	**75.78**	**70.87**	**69.12**
PARTY	48.97	53.71	53.65	57.46	62.52	68.15	74.04	79.22	82.80	84.17	82.82	79.99	74.07	70.18	62.47	60.24	53.61	53.78
AESSC	49.12	49.49	54.18	51.77	54.52	58.89	64.62	73.96	78.24	83.26	85.13	83.66	78.27	73.91	64.66	62.51	58.45	51.80
SSC	48.38	48.38	42.39	48.34	48.64	49.33	45.26	49.43	57.02	67.04	76.25	80.15	77.35	67.01	61.98	55.23	45.22	49.30
LS3C	31.43	33.57	35.69	37.88	40.07	41.50	43.52	44.52	45.43	45.62	45.44	44.59	43.48	41.89	39.53	37.93	34.95	32.64
LRR	45.83	52.46	47.82	55.76	53.48	63.05	67.55	71.58	73.99	75.52	73.92	71.65	63.95	58.21	58.94	49.98	47.76	46.54
LRSC	49.33	48.34	49.46	48.49	49.02	51.19	53.64	57.13	62.93	68.41	70.35	68.35	63.52	58.02	53.69	51.14	49.97	48.56
SMR	49.28	52.90	53.01	57.56	60.71	64.20	68.03	70.97	74.74	77.80	79.81	80.43	79.75	77.76	74.70	71.88	66.81	62.69
SC	51.14	51.27	51.47	52.15	49.48	51.90	55.51	60.35	68.57	72.04	76.64	77.91	76.68	73.31	68.62	60.40	55.45	55.72
GMM	47.99	52.22	48.54	49.15	50.54	52.96	56.82	61.88	67.97	74.04	78.44	80.11	78.84	74.00	70.39	65.12	60.44	56.92
KC	45.18	43.09	45.21	45.24	43.75	44.81	47.45	53.45	57.69	62.89	65.14	63.31	57.63	53.49	49.32	46.84	45.69	43.27
LDE	**54.39**	**56.65**	59.59	61.53	63.28	70.46	75.03	79.31	82.86	85.19	85.89	85.17	82.81	79.36	76.04	70.41	67.34	63.21
FSC	47.02	48.86	51.46	54.90	59.28	64.12	69.37	74.43	79.50	81.72	82.53	81.54	78.73	75.94	69.42	64.16	59.22	54.93

TABLE II
CLUSTERING NMI OF THE DIFFERENT CLUSTERING ALGORITHMS FOR THE DATASET INDIAN PINES

Methods	NMI(%)																	
	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10	K=11	K=12	K=13	K=14	K=15	K=16	K=17	K=18	K=19	K=20
DCIDC	**65.92**	**68.50**	**71.37**	**76.03**	**77.55**	**80.87**	**83.85**	**86.74**	**89.46**	**90.98**	**92.05**	**92.38**	**92.03**	**91.02**	**89.05**	**87.29**	**83.88**	**80.84**
PARTY	53.86	57.13	55.06	59.54	59.47	66.46	72.00	76.53	84.28	88.37	90.16	88.64	84.31	78.35	69.66	66.51	62.35	56.63
AESSC	53.12	49.52	54.30	56.00	58.93	59.81	69.28	76.05	82.25	86.78	88.43	86.41	80.97	76.01	66.13	63.44	55.08	52.06
SSC	48.38	48.38	48.39	42.35	42.71	49.60	52.03	51.43	66.05	73.69	86.30	90.06	86.32	73.65	66.01	51.48	51.99	43.57
LS3C	43.43	43.93	45.40	46.96	48.57	50.18	51.09	51.87	52.52	52.62	52.53	51.94	51.26	50.22	49.01	47.01	45.44	43.90
LRR	45.83	52.79	54.76	52.03	57.32	64.42	76.39	81.02	87.30	89.52	87.80	81.09	76.35	64.36	57.27	52.09	54.70	46.87
LRSC	48.33	49.34	48.48	48.55	50.24	51.87	55.32	61.36	68.52	75.77	78.69	75.72	69.11	61.41	54.41	51.82	50.20	49.62
SMR	51.28	53.38	56.06	57.63	63.63	68.25	73.30	78.41	82.34	86.29	88.88	89.71	88.90	86.26	82.30	77.41	73.36	68.28
SC	47.14	52.73	50.90	53.76	57.28	61.45	68.43	72.17	75.63	78.46	80.00	80.82	80.20	77.82	74.46	70.32	65.82	64.46
GMM	50.99	51.22	51.54	52.15	51.53	55.88	57.74	64.27	69.75	74.37	78.53	80.11	78.74	74.99	69.79	62.73	57.68	55.95
KC	41.93	41.84	43.97	42.01	42.60	43.84	46.90	53.60	58.75	65.14	67.35	65.20	58.70	51.98	46.85	45.88	44.54	44.04
LDE	61.28	61.54	64.46	69.86	73.66	77.92	80.98	85.69	88.77	90.62	91.36	90.60	88.41	85.12	81.03	76.58	72.16	68.04
FSC	47.02	53.31	56.50	60.66	65.82	68.56	75.14	81.46	87.56	90.48	91.53	90.30	87.53	82.96	75.19	71.48	62.40	60.69

TABLE III
CLUSTERING PURITY OF THE DIFFERENT CLUSTERING ALGORITHMS FOR THE DATASET PAVIA

Methods	Purity(%)																	
	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10	K=11	K=12	K=13	K=14	K=15	K=16	K=17	K=18	K=19	K=20
DCIDC	61.90	66.83	**76.15**	78.45	**82.30**	**86.35**	**88.94**	**89.81**	**88.98**	**87.87**	**84.35**	**81.41**	**77.11**	**72.80**	**66.57**	59.41	**55.98**	53.38
PARTY	57.84	61.59	66.50	69.84	75.96	81.20	84.88	86.20	85.09	81.24	77.28	71.86	64.00	58.89	54.93	51.97	50.14	48.97
AESSC	51.04	52.98	59.56	60.04	67.34	69.91	73.87	74.90	73.89	71.19	67.38	63.27	59.60	56.84	54.93	53.92	53.28	53.02
SSC	44.93	50.02	51.70	62.16	71.97	79.88	83.80	79.82	71.93	62.21	54.53	49.98	44.87	44.14	46.86	43.80	43.87	43.88
LS3C	38.94	41.34	43.51	45.56	46.98	48.40	49.45	49.67	49.38	48.43	46.94	44.98	43.57	40.08	37.41	36.42	32.20	29.93
LRR	57.39	60.81	67.16	70.90	76.95	80.40	81.65	80.34	76.92	70.95	65.51	62.70	57.45	57.19	53.92	55.34	52.96	**54.93**
LRSC	36.39	40.99	42.72	42.32	48.82	59.52	65.76	68.27	65.06	59.48	48.78	46.21	38.69	37.06	36.47	40.31	40.27	40.27
SMR	59.65	63.60	67.70	71.77	74.84	78.03	80.17	80.96	80.29	78.07	74.80	71.72	66.36	63.57	57.98	56.36	51.61	51.28
SC	45.44	50.08	54.03	57.83	64.41	71.54	75.78	77.29	75.74	70.85	65.64	57.88	52.25	48.06	45.36	43.88	45.01	42.64
GMM	51.44	52.02	57.66	62.48	68.11	73.45	77.20	78.75	77.16	73.49	68.15	62.53	57.60	53.90	49.54	48.13	47.33	47.09
KC	43.64	46.84	44.80	46.62	50.08	56.18	59.34	60.50	58.78	56.22	52.54	46.67	47.73	43.88	46.61	46.46	43.43	46.42
LDE	**66.93**	**70.62**	75.53	**80.71**	81.25	85.16	88.41	89.03	88.45	86.83	84.18	80.67	76.48	71.87	65.56	**60.36**	55.50	50.80
FSC	61.39	64.88	67.11	69.45	71.57	72.80	73.62	73.90	73.66	72.84	71.39	69.73	67.17	64.43	61.41	58.91	55.65	52.32

TABLE IV
CLUSTERING NMI OF THE DIFFERENT CLUSTERING ALGORITHMS FOR THE DATASET PAVIA

Methods	NMI(%)																	
	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10	K=11	K=12	K=13	K=14	K=15	K=16	K=17	K=18	K=19	K=20
DCIDC	**74.45**	**77.70**	81.17	84.14	**87.36**	**90.27**	**91.80**	**92.45**	**91.83**	**90.23**	**87.41**	**84.58**	**82.16**	**77.67**	73.50	**70.58**	**69.17**	**66.42**
PARTY	57.22	61.92	66.30	71.92	78.56	84.05	89.23	90.89	89.06	84.09	78.52	71.97	66.25	60.00	59.18	57.49	54.74	56.36
AESSC	54.79	54.01	60.08	61.97	69.23	72.90	77.10	78.35	76.84	72.86	69.27	64.41	60.12	53.98	54.71	53.54	49.82	49.52
SSC	59.67	65.61	70.34	78.93	84.82	88.76	90.15	88.51	84.09	77.63	72.26	63.12	56.88	54.89	51.37	46.06	47.69	43.88
LS3C	45.68	48.34	51.59	54.12	56.44	57.98	59.27	59.64	59.25	58.15	56.13	53.74	51.65	49.00	45.60	44.09	41.07	39.43
LRR	46.84	54.41	62.03	70.56	78.22	84.86	87.04	84.80	78.19	69.42	61.98	54.47	48.84	45.13	40.99	41.96	41.38	41.27
LRSC	34.11	38.85	40.98	45.30	52.80	59.89	69.33	73.26	69.35	59.85	52.75	41.51	36.95	38.92	38.19	33.98	33.93	37.93
SMR	59.11	63.81	68.79	74.75	78.98	81.99	84.42	85.31	84.46	82.03	78.32	73.73	70.02	63.78	60.94	57.08	53.88	51.28
SC	53.31	56.45	59.31	67.54	73.68	81.92	86.78	88.78	86.94	81.89	74.91	65.99	59.37	56.38	51.25	49.49	48.47	48.04
GMM	51.89	54.88	59.34	66.87	72.33	79.84	84.48	86.11	84.44	79.22	72.37	66.92	61.11	56.75	53.95	50.32	49.39	49.09
KC	39.64	40.11	41.78	42.20	48.37	57.96	64.22	66.64	64.20	57.99	50.83	42.25	38.74	37.15	36.62	39.39	36.34	36.33
LDE	73.94	75.37	**81.34**	**84.67**	87.28	89.39	90.59	91.03	90.63	89.00	87.32	83.36	81.38	75.44	**74.02**	66.76	66.43	58.80
FSC	69.03	74.36	79.13	83.23	86.64	88.81	90.49	91.00	90.48	89.03	86.19	83.28	78.14	74.39	67.21	63.29	54.62	48.07

3) Salinas: According to Tab. V, DCIDC performs best in almost all cases except when $K = 3$. Tab. VI shows that the clustering NMI of the DCIDC algorithm is 3% higher than that of the LDE algorithm. It reaches the highest score when $K = 9$. The changing trend and intuitive clustering results are shown as Fig. 7 and Fig. 8 respectively.

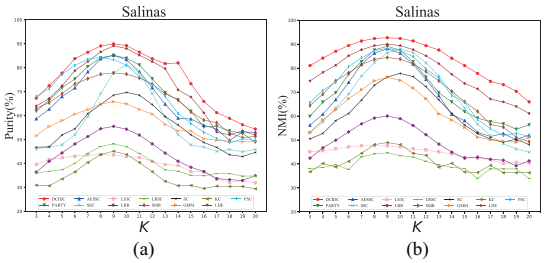

Fig. 7. The changing trend of different algorithms' clustering performance with cluster numbers on the dataset Salinas. (a) shows the changing trend of the clustering purity and (b) shows the changing trend of NMI.

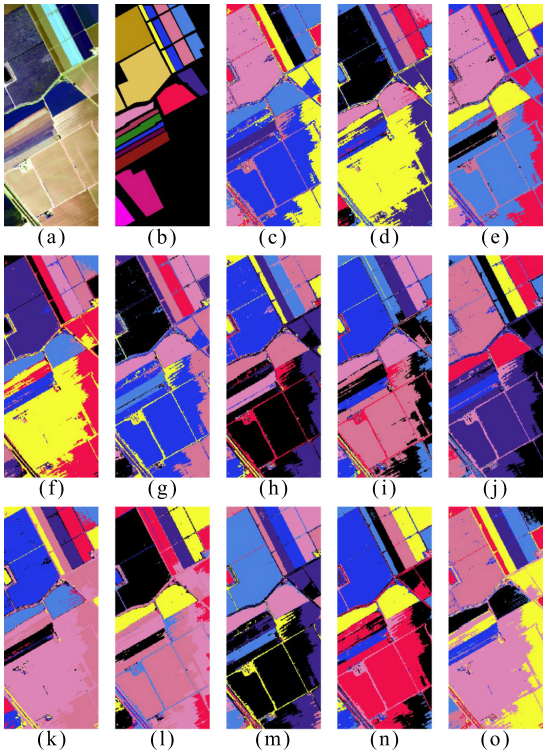

Fig. 8. The performance comparison of different algorithms on the dataset Salinas-A. (a) is the false color image of Salinas, (b) is the ground truth image, (c) DCIDC, (d) PARTY, (e) AESSC, (f) SSC, (g) LS3C, (h) LRR, (i) LRSC, (j) SMR, (k) SC, (l) GMM, (m) KC, (n) LDE, (o) FSC.

4) Salinas-A: According to Tab. VII, the DCIDC algorithm reaches the highest scores with $K = 6$. When $K \geq 16$, the algorithm LDE gets the best result. Comparing their best results, algorithm DCIDC is 4.13% higher than algorithm LDE. In terms of NMI, the proposed DCIDC algorithm performs best according to the results in Tab. VIII and gets a high score of 98.45%. As described in Fig. 9, the DCIDC reaches its best performance when $K = 6$. The visualized clustering results are shown in Fig. 10.

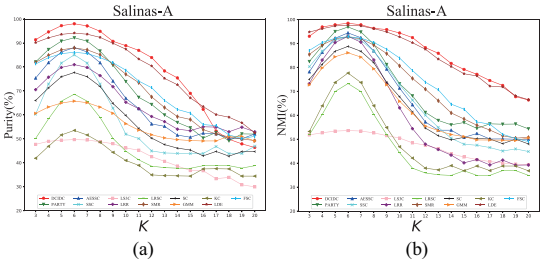

Fig. 9. The changing trend of different algorithms' clustering performance with cluster numbers on the dataset Salinas-A. (a) shows the changing trend of the clustering purity and (b) shows the changing trend of NMI.

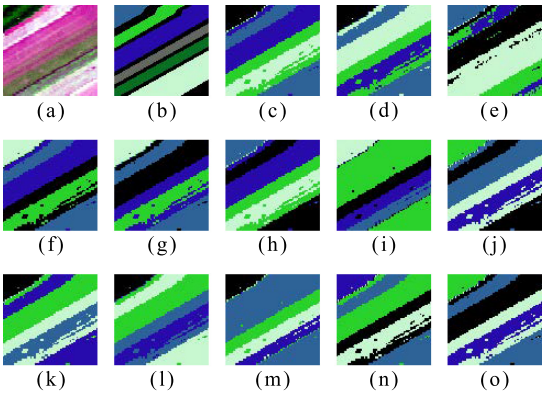

Fig. 10. The performance comparison of different algorithms on the dataset Salinas-A. (a) is the false color image of Salinas-A, (b) is the ground truth image, (c) DCIDC, (d) PARTY, (e) AESSC, (f) SSC, (g) LS3C, (h) LRR, (i) LRSC, (j) SMR, (k) SC, (l) GMM, (m) KC, (n) LDE, (o) FSC.

5) Discussion: We carry out a lot of comparative experiments by setting different cluster numbers. The experimental results show that our DCIDC algorithm performs better than state-of-the-art and traditional clustering algorithms in most cases. This indicates that it is very effective to constrain the feature mapping process by intra-class distance, which makes the data mapped from the source feature space to the new feature space that is more conducive to clustering. The ground-truth cluster numbers of Indian pines, Pavia, Salinas, Salinas-A are 15, 10, 9, 6 respectively. On these four datasets, most clustering algorithms reach their best clustering effect when K=14, 10, 9, 6, respectively. This shows that the closer the cluster number used by the clustering algorithm is to the ground-truth cluster number, the better the clustering effect will be. If the cluster number is too small or too large, it will lead to clustering errors and reduce the clustering accuracy. According to the experimental results, in most cases, the clustering effect of the deep-learning-based algorithms(DCIDC, PARTY, AESSC, LDE and FSC) is better than that of the other shallow models. However, in some cases (for example, when $K = 20$ in Tab. III), the scores of the PARTY, the AESSC, and the LDE algorithm are lower than that of the LRR, which may be because that the prior matrix $\boldsymbol{C}$ adopted by the two algorithms need to be pre-trained and cannot be adjusted dynamically in the clustering procedure, when the distribution of the data to be clustered is quite different from that of the data used for training the matrix $\boldsymbol{C}$, the two

TABLE V
CLUSTERING PURITY OF THE DIFFERENT CLUSTERING ALGORITHMS FOR THE DATASET SALINAS

Methods	Purity(%)																	
	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10	K=11	K=12	K=13	K=14	K=15	K=16	K=17	K=18	K=19	K=20
DCIDC	67.26	**72.47**	**77.73**	**83.79**	**86.43**	**89.07**	**89.92**	**89.21**	**86.47**	**83.75**	**81.69**	**81.98**	**73.31**	**65.91**	**61.24**	**58.85**	**56.18**	**54.38**
PARTY	61.88	66.36	71.61	75.32	80.48	84.13	85.20	83.95	81.15	75.36	69.59	66.42	61.81	55.24	55.51	53.62	52.60	48.97
AESSC	58.63	62.66	67.89	71.55	78.27	83.71	85.14	83.31	78.30	71.59	64.74	58.93	58.56	55.95	54.34	49.63	53.17	53.02
SSC	46.15	46.86	47.83	52.06	59.56	69.17	77.95	81.25	77.93	69.14	59.52	52.11	47.76	46.84	45.06	45.84	44.89	45.88
LS3C	39.63	41.57	42.40	43.10	43.61	43.69	43.55	42.97	42.45	40.94	39.58	39.19	36.56	36.50	33.49	32.12	32.83	31.93
LRR	36.38	40.90	44.40	48.18	51.26	54.54	55.49	54.31	51.88	48.22	44.35	40.96	38.40	36.66	33.67	33.25	32.93	34.93
LRSC	36.02	36.73	37.41	40.03	44.23	47.08	48.17	47.04	44.19	40.91	37.36	36.77	35.05	34.85	35.78	35.77	34.77	34.77
SMR	63.07	65.38	69.16	73.23	75.43	77.27	77.84	77.31	75.69	73.19	69.11	66.60	61.56	58.15	57.01	52.70	50.79	51.28
SC	46.82	46.92	51.88	54.43	60.76	64.65	68.53	69.68	68.51	64.62	59.61	56.08	51.95	48.78	46.73	43.63	42.92	44.64
GMM	51.59	55.49	57.93	60.61	62.54	64.88	65.79	64.90	62.58	60.65	56.29	53.59	53.56	50.34	49.69	49.32	49.06	49.09
KC	30.93	30.67	33.55	36.50	40.43	43.73	45.13	43.75	40.40	36.54	32.55	30.73	30.86	29.55	30.53	30.43	30.42	29.42
LDE	64.00	67.13	72.41	78.77	82.59	86.59	89.10	88.00	85.20	82.56	79.63	70.74	67.73	61.21	53.08	52.01	53.62	51.81
FSC	**68.03**	71.16	77.32	81.01	83.52	84.30	83.32	81.04	76.28	72.87	68.07	60.94	56.65	53.11	50.46	48.58	50.31	49.52

TABLE VI
CLUSTERING NMI OF THE DIFFERENT CLUSTERING ALGORITHMS FOR THE DATASET SALINAS

Methods	NMI(%)																	
	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10	K=11	K=12	K=13	K=14	K=15	K=16	K=17	K=18	K=19	K=20
DCIDC	**81.08**	**84.20**	**87.03**	**89.42**	**91.33**	**92.37**	**92.71**	**92.40**	**91.37**	**89.38**	**87.59**	**84.16**	**81.05**	**77.80**	**74.48**	**73.11**	**70.34**	**65.92**
PARTY	59.92	65.83	69.74	77.43	82.74	87.60	89.17	87.43	82.70	76.29	69.69	65.89	61.80	59.12	57.65	56.78	54.51	56.36
AESSC	56.20	60.75	66.82	74.03	81.68	86.24	88.10	86.22	81.72	75.26	68.46	60.69	58.07	53.22	51.46	52.67	50.19	52.02
SSC	52.91	57.56	63.39	69.28	76.72	82.30	86.44	87.89	86.49	82.27	76.68	69.99	63.33	56.63	51.86	48.45	46.27	44.88
LS3C	45.08	45.50	46.29	47.13	47.57	47.63	47.52	47.16	46.34	45.87	45.03	44.16	42.50	42.46	41.75	40.19	40.69	40.43
LRR	42.43	46.72	49.93	53.37	56.77	59.16	60.02	58.93	56.08	52.22	48.28	44.95	42.50	42.85	41.94	41.56	39.26	41.27
LRSC	38.07	38.54	39.62	37.74	42.90	44.22	44.64	43.45	42.86	41.12	39.57	38.58	38.10	33.98	37.93	37.93	37.93	33.93
SMR	64.42	69.14	74.73	78.04	81.48	83.66	84.45	83.78	81.74	78.60	74.68	70.35	65.99	61.96	56.52	55.43	51.13	49.28
SC	50.45	52.75	58.05	61.04	66.69	72.92	76.32	77.79	76.48	72.23	66.73	61.09	56.28	52.67	50.36	49.12	50.35	48.04
GMM	53.28	58.44	62.57	67.20	71.02	74.73	76.15	74.94	71.72	67.25	60.93	58.38	55.25	51.21	50.08	49.48	49.11	51.09
KC	36.73	40.22	38.71	41.06	44.47	48.08	48.91	48.10	44.44	43.61	38.67	40.28	36.66	36.43	39.44	36.34	36.33	36.33
LDE	74.64	78.25	81.17	85.18	87.70	89.40	89.99	89.42	87.74	85.13	81.88	77.42	73.59	71.34	67.09	65.86	64.03	61.11
FSC	65.67	70.67	74.20	80.73	84.27	87.23	88.31	87.40	84.78	79.68	75.90	68.56	65.74	58.53	54.76	51.90	52.86	51.52

TABLE VII
CLUSTERING PURITY OF THE DIFFERENT CLUSTERING ALGORITHMS FOR THE DATASET SALINAS-A

Methods	Purity(%)																	
	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10	K=11	K=12	K=13	K=14	K=15	K=16	K=17	K=18	K=19	K=20
DCIDC	**91.36**	**94.64**	**97.25**	**98.04**	**97.13**	**94.90**	**90.71**	**88.90**	**86.93**	**83.93**	**78.34**	**75.43**	**68.99**	62.31	53.23	49.82	47.88	46.38
PARTY	82.09	87.31	90.78	92.20	90.76	86.64	80.69	73.93	67.33	64.35	59.95	56.83	54.62	50.31	52.65	49.15	52.06	51.97
AESSC	75.36	81.80	86.23	88.14	86.25	81.77	74.13	66.94	62.60	56.21	55.23	51.54	50.67	52.36	52.11	50.15	50.03	52.02
SSC	60.29	72.85	81.62	85.08	81.60	72.82	62.75	51.92	50.09	44.94	44.10	43.93	43.80	43.88	46.78	43.78	43.88	46.88
LS3C	47.68	48.95	49.42	49.67	49.50	48.75	47.97	46.17	45.26	42.53	40.60	38.64	36.76	36.69	33.35	33.90	30.80	29.93
LRR	70.54	75.72	79.80	80.98	79.78	76.42	71.77	65.26	62.58	59.31	57.18	54.02	53.39	55.01	54.89	52.95	54.83	**52.93**
LRSC	50.14	58.33	65.72	68.66	65.70	58.92	50.19	44.85	41.21	38.57	37.98	37.73	37.69	38.77	38.77	38.77	37.77	38.77
SMR	81.88	84.88	87.17	87.96	87.15	85.14	81.32	76.96	73.48	67.50	63.08	59.28	58.08	53.83	52.04	50.77	49.93	49.28
SC	66.01	71.23	75.91	77.67	75.93	71.86	64.78	59.76	54.34	50.30	47.59	46.11	45.32	42.87	44.72	42.76	44.65	44.64
GMM	60.62	63.29	64.94	65.75	65.09	63.25	60.66	56.31	53.62	51.69	50.37	49.64	49.27	49.10	49.12	51.10	50.99	49.09
KC	41.90	46.87	51.61	53.50	51.59	48.56	44.36	40.95	38.91	34.93	34.63	34.54	34.43	37.42	37.52	37.42	34.42	34.42
LDE	90.19	92.16	93.61	94.15	93.67	92.36	89.90	87.42	83.29	80.40	75.23	72.58	67.05	**63.44**	**60.11**	**59.01**	**56.66**	52.51
FSC	81.25	83.88	85.56	86.14	85.60	83.91	81.75	78.67	74.12	71.33	66.03	62.25	60.76	55.95	55.48	51.61	50.12	51.02

TABLE VIII
CLUSTERING NMI OF THE DIFFERENT CLUSTERING ALGORITHMS FOR THE DATASET SALINAS-A

Methods	NMI(%)																	
	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10	K=11	K=12	K=13	K=14	K=15	K=16	K=17	K=18	K=19	K=20
DCIDC	93.02	**96.89**	**97.80**	**98.45**	**97.80**	**96.31**	**95.82**	**94.38**	**92.45**	**88.27**	**85.92**	**81.68**	**79.17**	**77.13**	**74.54**	**72.49**	67.78	66.42
PARTY	82.47	89.00	95.01	96.89	94.80	89.70	81.24	73.20	68.13	61.23	57.88	56.04	56.99	54.54	56.45	56.29	56.37	54.36
AESSC	78.17	86.28	92.19	94.41	92.40	86.90	79.31	71.35	64.40	57.24	53.79	53.80	50.78	52.41	50.12	50.15	50.03	50.02
SSC	80.31	86.80	91.26	92.80	91.30	86.77	80.27	72.83	66.35	60.06	55.00	50.44	47.92	47.50	46.55	45.10	45.98	44.88
LS3C	51.84	52.71	53.41	53.64	53.45	52.67	51.53	50.51	48.51	47.53	46.10	43.86	42.62	41.51	40.70	40.09	39.64	39.43
LRR	73.24	83.07	90.37	92.99	90.54	83.11	73.29	63.14	54.43	47.95	45.74	43.40	40.18	41.53	39.29	41.31	39.18	39.27
LRSC	51.88	60.36	69.93	73.26	69.91	60.40	51.92	44.39	37.95	35.92	35.19	34.90	36.85	34.93	34.93	36.93	36.93	34.93
SMR	85.43	89.40	91.95	92.78	91.93	89.36	85.78	80.67	75.42	70.94	66.12	61.01	58.48	55.87	53.87	51.45	50.50	50.78
SC	75.05	81.31	86.74	88.78	86.76	81.93	73.82	67.73	61.37	54.70	51.49	49.75	50.82	50.31	50.13	48.17	50.05	48.04
GMM	72.71	79.79	84.45	86.11	84.31	79.42	72.75	65.85	60.83	55.54	53.62	51.97	51.12	49.73	49.67	49.61	49.50	50.59
KC	53.26	63.96	73.72	77.64	73.70	63.99	54.89	46.87	42.15	37.95	37.21	38.98	36.85	38.84	36.93	38.83	38.83	36.83
LDE	**94.77**	96.17	97.30	97.68	97.32	96.21	94.81	92.76	90.32	87.61	83.48	80.40	77.35	76.44	72.19	71.84	**68.15**	**66.51**
FSC	87.11	90.42	92.18	92.86	92.22	90.45	87.61	83.93	78.69	73.89	70.66	64.69	62.58	57.23	56.31	52.09	50.33	49.02

algorithms cannot achieve good clustering effect.

C. Influence of Tradeoff Coefficients

The coefficient λ_1 indicates the constraint degree of intra-class distance to the DCIDC algorithm and the coefficient λ_2 is designed to prevent over-fitting. We investigate the variation of clustering purity and NMI with different values of λ_1 and λ_2 to obtain the optimal values of the two coefficients. It can be found that given different values of λ_1 and λ_2, the model achieves different clustering purities and NMIs. In most cases, the optimal values of λ_1 and λ_2 are around 0.3 and 0.0003 respectively. The variations of clustering purity and NMI with λ_1 and λ_2 are shown as Fig. 11.

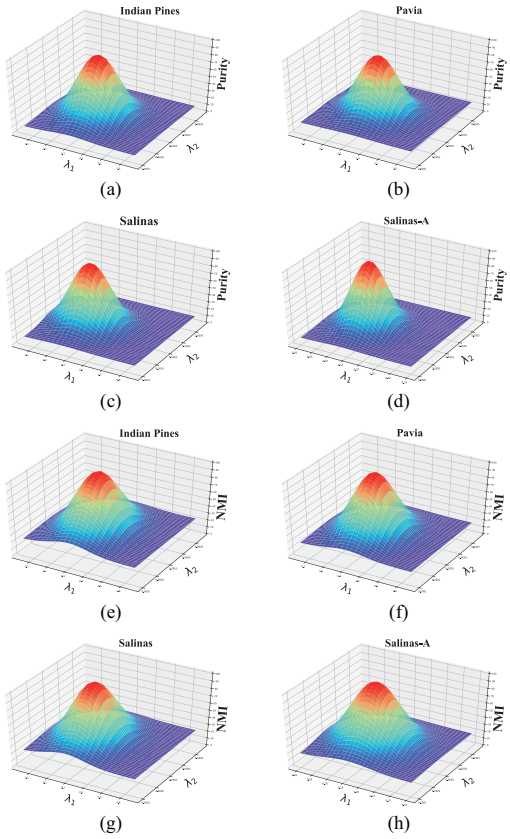

Fig. 11. The variations of clustering purity and NMI with λ_1 and λ_2. (a) $\sim$ (d) describe the variation of clustering purity on the four datasets: Indian Pines, Pavia, Salinas, Salinas-A. (e) $\sim$ (h) show the variation of NMI on the same four datasets as (a) $\sim$ (d).

D. Influence of Activation Functions

In this subsection, we report the performance of DCIDC with four different activation functions on the four datasets. The used activation functions include *Tanh*, *Sigmoid*, *Nssigmoid*, and *Softplus*. From Fig. 12, we can see that the *Tanh* function outperforms the other three activation functions in the experiments and the *Nssigmoid* function achieves the second best result which is very close to the best one.

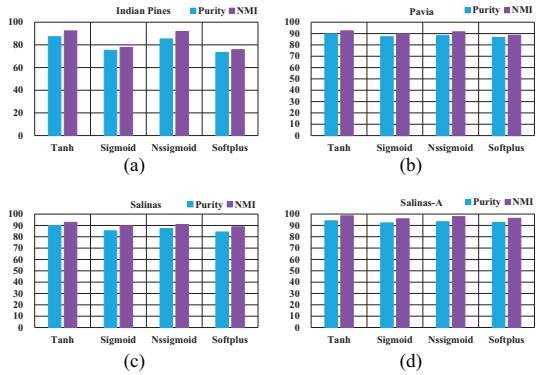

Fig. 12. The performance of DCIDC with four different activation functions. (a) Indian Pines, (b) Pavia, (c) Salinas, (d) Salinas-A.

E. Influence of Network Structure

In order to explore the influence of the network structure on the DCIDC algorithm, we investigated the variation of clustering purity and NMI by setting different numbers of the middle layer nodes and network layers, as shown in Fig. 13. It can be seen from Fig. 13 that the number of nodes in the middle layer has little impact on the DCIDC algorithm. With the number of nodes changing from 15 to 55, the clustering purity and NMI on the same dataset have a negligible change. On the other hand, too many or too few network layers will affect the performance of the DCIDC algorithm. Experimental results show that the optimal network layer number is around 7.

F. Time Consumption

In order to verify the efficiency of the DCIDC algorithm, we studied the running time of each algorithm under its own optimal hyper-parameter conditions on the four datasets, and the statistical time is the average value of 50 repeated experiments. The results are shown in Fig. 14. When the dataset size is very small, the advantage of the DCIDC algorithm is not obvious; on the contrary, when the dataset is large, the efficiency of the DCIDC algorithm is very prominent. For example, on the Salinas-A dataset, the DCIDC algorithm and K-means algorithm take less than one second; on the Salinas dataset, the DCIDC algorithm takes less than 90 seconds, and the K-means algorithm takes nearly 200 seconds.

V. Conclusions

In this paper, we have proposed an intra-class distance constrained deep clustering approach, which constrains the feature mapping procedure by intra-class distance. Compared with the current deep clustering approaches, the procedure of feature extraction of DCIDC is more purposeful, making the features learned more conducive to clustering. The proposed approach jointly optimizes the parameters of the network and the procedure of the clustering without additional pre-training process, which is more efficient. We conducted comparative experiments on four different hyperspectral datasets, and the purities of the DCIDC are at least 2.4%, 0.9%, 0.9%, 4.13%

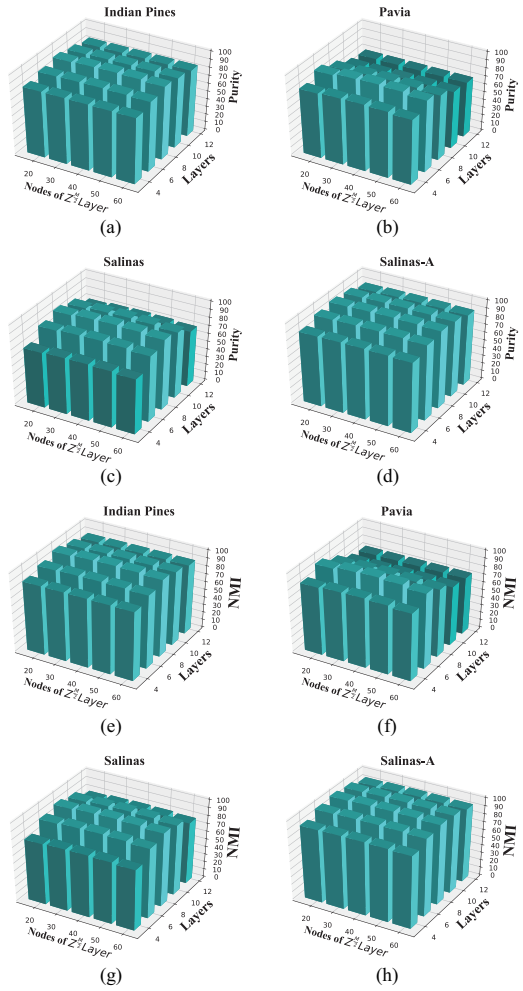

Fig. 13. The variations of clustering purity and NMI with node numbers of the middle ($\boldsymbol{Z}^{\frac{M}{2}}$) layer and network layers. (a) $\sim$ (d) describe the variation of clustering purity on the four datasets: Indian Pines, Pavia, Salinas, Salinas-A. (e) $\sim$ (h) show the variation of NMI on the same four datasets as (a) $\sim$ (d).

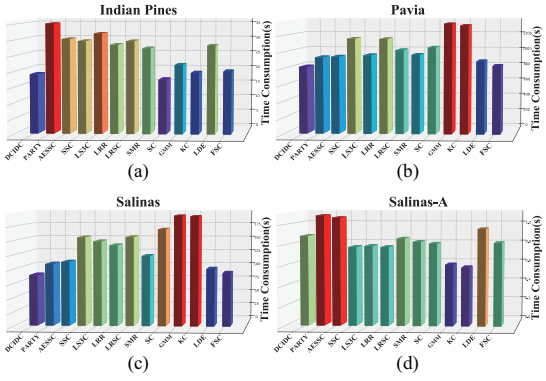

Fig. 14. Time consumption of different algorithms on the four datasets. (a) Indian Pines, (b) Pavia, (c) Salinas, (d) Salinas-A.

higher than that of the compared methods, respectively, regarding the datasets of Indian Pines, Pavia, Salinas and Salinas-A. The NMIs of the DCIDC are at least 1.9%, 1.5%, 3.3%, 0.8% higher than that of the compared algorithms respectively on the four datasets. The experimental results demonstrate that the proposed approach performs better than state-of-the-art clustering methods in terms of purity and NMI. In the future, adaptive deep clustering methods may be further explored based on this work.

REFERENCES

[1] H. Li, S. Zhang, X. Ding, C. Zhang, and P. Dale, "Performance Evaluation of Cluster Validity Indices (CVIs) on Multi/Hyperspectral Remote Sensing Datasets," *Remote Sensing*, vol. 8, no. 4, pp. 295–317, 2016.

[2] A. Romero, C. Gatta, and G. Camps-Valls, "Unsupervised deep feature extraction for remote sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 3, pp. 1349–1362, 2016.

[3] N. Yokoya, C. Grohnfeldt, and J. Chanussot, "Hyperspectral and multispectral data fusion: A comparative review of the recent literature," *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 2, pp. 29–56, 2017.

[4] J. Guo and H. Huo, "An Enhanced IT2fcm* Algorithm Integrating Spectral Indices and Spatial Information for Multi-Spectral Remote Sensing Image Clustering," *Remote Sensing*, vol. 9, no. 9, pp. 960–981, 2017.

[5] H. Zhai, H. Zhang, X. Xu, L. Zhang, and P. Li, "Kernel sparse subspace clustering with a spatial max pooling operation for hyperspectral remote sensing data interpretation," *Remote Sensing*, vol. 9, no. 4, pp. 335–350, 2017.

[6] T. Jiang, D. Hu, and X. Yu, "Enhanced IT2fcm algorithm using object-based triangular fuzzy set modeling for remote-sensing clustering," *Computers & Geosciences*, vol. 118, pp. 14–26, 2018.

[7] H. Xie, A. Zhao, S. Huang, J. Han, S. Liu, X. Xu, X. Luo, H. Pan, Q. Du, and X. Tong, "Unsupervised Hyperspectral Remote Sensing Image Clustering Based on Adaptive Density," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 4, pp. 632–636, 2018.

[8] P. Ghamisi, B. Rasti, N. Yokoya, Q. Wang, B. Hofle, L. Bruzzone, F. Bovolo, M. Chi, K. Anders, R. Gloaguen *et al.*, "Multisource and multitemporal data fusion in remote sensing: A comprehensive review of the state of the art," *IEEE Geoscience and Remote Sensing Magazine*, vol. 7, no. 1, pp. 6–39, 2019.

[9] S. Liu, L. Bruzzone, F. Bovolo, and P. Du, "Unsupervised multitemporal spectral unmixing for detecting multiple changes in hyperspectral images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 5, pp. 2733–2748, 2016.

[10] L. Yu, J. Xie, S. Chen, and L. Zhu, "Generating labeled samples for hyperspectral image classification using correlation of spectral bands," *Frontiers of Computer Science*, vol. 10, no. 2, pp. 292–301, 2016.

[11] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the art," *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 2, pp. 22–40, 2016.

[12] B. Zhao, Y. Zhong, A. Ma, and L. Zhang, "A spatial Gaussian mixture model for optical remote sensing image clustering," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 12, pp. 5748–5759, 2016.

[13] H. Zhang, H. Zhai, L. Zhang, and P. Li, "Spectral–spatial sparse subspace clustering for hyperspectral remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 6, pp. 3672–3684, 2016.

[14] K. V. Kale, M. M. Solankar, D. B. Nalawade, R. K. Dhumal, and H. R. Gite, "A research review on hyperspectral data processing and analysis algorithms," *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, vol. 87, no. 4, pp. 541–555, 2017.

[15] D. Hong, N. Yokoya, J. Chanussot, and X. X. Zhu, "An augmented linear mixing model to address spectral variability for hyperspectral unmixing," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1923–1938, 2019.

[16] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, and A. Bouras, "A survey of clustering algorithms for big data: Taxonomy and empirical analysis," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 3, pp. 267–279, 2014.

[17] A. K. Alok, S. Saha, and A. Ekbal, "Multi-objective semi-supervised clustering for automatic pixel classification from remote sensing imagery," *Soft Computing*, vol. 20, no. 12, pp. 4733–4751, 2016.

[18] W. Yang, K. Hou, B. Liu, F. Yu, and L. Lin, "Two-stage clustering technique based on the neighboring union histogram for Hyperspectral remote sensing images," *IEEE Access*, vol. 5, pp. 5640–5647, 2017.

[19] S. Hemalatha and S. M. Anouncia, "Unsupervised segmentation of remote sensing images using FD based texture analysis model and ISODATA," *International Journal of Ambient Computing and Intelligence*, vol. 8, no. 3, pp. 58–75, 2017.

[20] N. Gillis and S. A. Vavasis, "Fast and robust recursive algorithmsfor separable nonnegative matrix factorization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 4, pp. 698–714, 2013.

[21] M. Maggioni and J. M. Murphy, "Learning by Unsupervised Nonlinear Diffusion," *Journal of Machine Learning Research*, vol. 20, no. 160, pp. 1–56, 2019.

[22] N. Gillis, D. Kuang, and H. Park, "Hierarchical clustering of hyperspectral images using rank-two nonnegative matrix factorization," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 4, pp. 2066–2078, 2014.

[23] J. M. Murphy and M. Maggioni, "Unsupervised Clustering and Active Learning of Hyperspectral Images With Nonlinear Diffusion," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 3, pp. 1829–1845, 2018.

[24] V. M. Patel and R. Vidal, "Kernel sparse subspace clustering," in *Image Processing of 2014 IEEE International Conference*, 2014, pp. 2849–2853.

[25] M. Yin, Y. Guo, J. Gao, Z. He, and S. Xie, "Kernel sparse subspace clustering on symmetric positive definite manifolds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5157–5164.

[26] R. Zhang, F. Nie, and X. Li, "Self-weighted supervised discriminative feature selection," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3913–3918, 2018.

[27] X. Wei, *Learning Image and Video Representations Based on Sparsity Priors*. Aachen, Germany: Shaker Verlag GmbH, 2017.

[28] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[29] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*, 2002, pp. 849–856.

[30] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[31] H. Zhai, H. Zhang, L. Zhang, P. Li, and A. Plaza, "A New Sparse Subspace Clustering Algorithm for Hyperspectral Remote Sensing Imagery," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 1, pp. 43–47, 2017.

[32] X. Wei, H. Shen, and M. Kleinsteuber, "Trace quotient meets sparsity: A method for learning low dimensional image representations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5268–5277.

[33] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in neural information processing systems*, 2002, pp. 585–591.

[34] V. R. De Sa, "Spectral clustering with two views," in *ICML workshop on learning with multiple views*, 2005, pp. 20–27.

[35] B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis, "Diffusion maps, spectral clustering and reaction coordinates of dynamical systems," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 113–127, 2006, publisher: Elsevier.

[36] B. Nadler and M. Galun, "Fundamental limitations of spectral clustering," in *Advances in neural information processing systems*, 2007, pp. 1017–1024.

[37] C. X.-y. D. Guan-zhong and Y. Li-bin, "Survey on Spectral Clustering Algorithms," *Computer Science*, vol. 7, no. 005, 2008.

[38] P. Ji, M. Salzmann, and H. Li, "Efficient dense subspace clustering," in *Applications of Computer Vision of 2014 IEEE Winter Conference*, 2014, pp. 461–468.

[39] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[40] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[41] K.-C. Lee, J. Ho, and D. J. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 5, pp. 684–698, 2005.

[42] V. M. Patel, H. Van Nguyen, and R. Vidal, "Latent space sparse subspace clustering," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 225–232.

[43] H. Zhang, Z. Lin, C. Zhang, and J. Gao, "Robust latent low rank representation for subspace clustering," *Neurocomputing*, vol. 145, pp. 369–373, 2014.

[44] X. Wei, H. Shen, and M. Kleinsteuber, "Trace Quotient with Sparsity Priors for Learning Low Dimensional Image Representations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–16, Jun. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8731748/

[45] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators." *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[46] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989, publisher: Springer.

[47] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information theory*, vol. 39, no. 3, pp. 930–945, 1993, publisher: IEEE.

[48] A. Dundar, J. Jin, and a. E. Culurciello, "Convolutional Clustering for Unsupervised Learning," in *ICLR 2016*, Feb. 2016, pp. 1–11.

[49] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International Conference on Machine Learning*, 2016, pp. 478–487.

[50] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, "Deep adaptive image clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5879–5887.

[51] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid, "Deep subspace clustering networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 24–33.

[52] M. T. Law, R. Urtasun, and R. S. Zemel, "Deep spectral clustering learning," in *International Conference on Machine Learning*, 2017, pp. 1985–1994.

[53] R. Eldan and O. Shamir, "The power of depth for feedforward neural networks," in *Conference on learning theory*, 2016, pp. 907–940.

[54] I. Safran and O. Shamir, "Depth-width tradeoffs in approximating natural functions with neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2979–2987.

[55] H. Lee, R. Ge, T. Ma, A. Risteski, and S. Arora, "On the Ability of Neural Nets to Express Distributions," in *Conference on Learning Theory*, Jun. 2017, pp. 1271–1296.

[56] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, "Auto-encoder based data clustering," in *Iberoamerican Congress on Pattern Recognition*. Springer, 2013, pp. 117–124.

[57] N. Dilokthanakul, P. A. Mediano, M. Garnelo, M. C. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, "Deep unsupervised clustering with gaussian mixture variational autoencoders," in *ICLR 2017*, Jan. 2017, pp. 1–12.

[58] X. Wei, H. Shen, Y. Li, X. Tang, F. Wang, M. Kleinsteuber, and Y. L. Murphey, "Reconstructible nonlinear dimensionality reduction via joint dictionary learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 1, pp. 175–189, 2019.

[59] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[60] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Acoustics, Speech and Signal Processing of 2016 IEEE International Conference*, 2016, pp. 31–35.

[61] J. Liang, J. Yang, H.-Y. Lee, K. Wang, and M.-H. Yang, "Sub-GAN: An Unsupervised Generative Model via Subspaces," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 698–714.

[62] P. Zhou, Y. Hou, and J. Feng, "Deep Adversarial Subspace Clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1596–1604.

[63] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A survey of clustering with deep learning: From the perspective of network architecture," *IEEE Access*, vol. 6, pp. 39 501–39 514, 2018.

[64] W. Lin, J. Chen, C. D. Castillo, and R. Chellappa, "Deep Density Clustering of Unconstrained Faces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8128–8137.

[65] X. Peng, S. Xiao, J. Feng, W. Yau, and Z. Yi, "Deep Subspace Clustering with Sparsity Prior," in *International Joint Conference on Artificial Intelligence*, 2016, pp. 1925–1931.

[66] Y. Chen, L. Zhang, and Z. Yi, "Subspace clustering using a low-rank constrained autoencoder," *Information Sciences*, vol. 424, pp. 27–38, 2018.
[67] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.
[68] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006, publisher: American Association for the Advancement of Science.
[69] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML workshop on unsupervised and transfer learning*, 2012, pp. 37–49.
[70] A. Sperduti, "Linear autoencoder networks for structured data," in *International Workshop on Neural-Symbolic Learning and Reasoning*, 2013.
[71] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
[72] F. Zhang, B. Du, and L. Zhang, "Saliency-guided unsupervised feature learning for scene classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 4, pp. 2175–2184, 2015.
[73] Y. Li, X. Huang, and H. Liu, "Unsupervised deep feature learning for urban village detection from high-resolution remote sensing images," *Photogrammetric Engineering & Remote Sensing*, vol. 83, no. 8, pp. 567–579, 2017.
[74] D. T. Grozdic and S. T. Jovicic, "Whispered speech recognition using deep denoising autoencoder and inverse filtering," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 25, no. 12, pp. 2313–2322, 2017.
[75] Y. Dai and G. Wang, "Analyzing tongue images using a conceptual alignment deep autoencoder," *IEEE Access*, vol. 6, pp. 5962–5972, 2018.
[76] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1544–1551, 2018.
[77] J. Yu, C. Hong, Y. Rui, and D. Tao, "Multitask autoencoder model for recovering human poses," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 6, pp. 5060–5068, 2018.
[78] M. Ma, C. Sun, and X. Chen, "Deep coupling autoencoder for fault diagnosis with multimodal sensory data," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 3, pp. 1137–1145, 2018.
[79] E. Elhamifar and R. Vidal, "Sparse manifold clustering and embedding," in *Advances in neural information processing systems*, 2011, pp. 55–63.
[80] R. Zhang, F. Nie, and X. Li, "Embedded clustering via robust orthogonal least square discriminant analysis," in *Acoustics, Speech and Signal Processing of 2017 IEEE International Conference*, 2017, pp. 2332–2336.
[81] Y. Qin, L. Bruzzone, and B. Li, "Learning Discriminative Embedding for Hyperspectral Image Clustering Based on Set-to-Set and Sample-to-Sample Distances," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 1, pp. 473–485, 2019, publisher: IEEE.
[82] Y. Zhao, Y. Yuan, and Q. Wang, "Fast Spectral Clustering for Unsupervised Hyperspectral Image Classification," *Remote Sensing*, vol. 11, no. 4, pp. 1–21, 2019.
[83] P. Favaro, R. Vidal, and A. Ravichandran, "A closed form solution to robust subspace estimation and clustering," in *CVPR 2011*. IEEE, 2011, pp. 1801–1807.
[84] H. Hu, Z. Lin, J. Feng, and J. Zhou, "Smooth representation clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3834–3841.

Jinguang Sun received the Ph.D. degree from Liaoning Technical University, Fuxin, China. She is currently the doctoral supervisor of Liaoning Technical University. Her research interests focus on image analysis and understanding, computer graphics. She has authored over 30 publications in refereed journals and conference proceedings. The applications include face recognition, image encryption, and remote disaster monitoring.

Wanli Wang received the M.S. degree in remote sensing from Liaoning Technical University, Fuxin, China. He is currently a Ph.D. student in Quanzhou Institute of Equipment Manufacturing, Haixi Institute, Chinese Academy of Sciences. His research interests focus on remote sensing image recognition, machine vision and pattern recognition.

Xian Wei received the M.S. degree in computer science from Shanghai Jiaotong University, Shanghai, China, and received the Ph.D. degree in Engineering from the Technical University of Munich, Munich, Germany. Recently, he joined Fujian Institute of Research on the Structure of Matter, Chinese Academy of Sciences, China, as a Leading Researcher of Machine Learning and Pattern Recognition Lab. His research interests focus on sparse coding, deep learning, geometric optimization, and time series analysis. The applications include multi-sensor fusion for intelligent car, robotic vision, data sequence or images modeling, synthesis, recognition and semantics. He has authored over 50 publications in refereed journals and conference proceedings.

Li Fang was born in 1986. He received the Bachelor's degree in Land Resources Management and Master's degree in physical geography from China University of Geoscience, Wuhan, China, in 2007 and 2009, respectively. He received the second Master's degree in geoinformatics and the Ph.D. degree in photogrammetry and remote sensing, in 2011 and 2017, respectively, from the Technical University of Munich, Munich, Germany. His main research interests include image processing and analysis, pattern recognition, and related remote sensing/environmental and industrial applications.

Xiaoliang Tang received the B.S., and Ph.D. degrees from Dalian University of Technology, Dalian, China, in 2003, and 2009, respectively. In 2010, he joined the Liaoning Technical University, Huludao, China. From 2010 to 2018, he was a lecturer with the College of Software Engineering, Liaoning Technical University, Huludao, China. In 2018, he joined Quanzhou Institute of Equipment Manufacturing Haixi Institutes, Chinese Academy of Sciences. He is currently an Associate Research Fellow with the Quanzhou Institute of Equipment Manufacturing Haixi Institutes, Chinese Academy of Sciences. His research interests include image processing, pattern recognition, and machine learning.

Yusheng Xu (S'17- M'19) was born in 1989. He received his B.S. and M.E degrees from Tongji University, Shanghai, China, in 2011 and 2014, respectively. He received his Ph.D. degree in 2019 from the Technical University of Munich (TUM) at the Chair of Photogrammetry and Remote Sensing, Munich, Germany. He has published more than 50 entries. His research interests are in 3D point cloud processing, image processing, and spaceborne photogrammetry.

Hui Yu received the B.S. degree in micro electronics from Hefei University of Technology, Hefei, China, in 2008, the M.S. and Ph.D. degrees in electronic and electrical engineering from University of Strathclyde, in 2012, and 2018, respectively. Now he is an associate researcher in Fujian Institute of Research on the Structure of Matter, Chinese Academy of Sciences. His research interests include complex dynamic system modelling, analysis and optimization.

Wei Yao received the B.S. in photogrammetry and remote sensing from Wuhan University, China, and in 2007 and 2010, he obtained Dipl.-Ing. (Univ.) degree in Geodesy and Geoinformation and the Ph.D. degree from the Technische Universität München (TUM), Germany. Since 2007, he has been a scientific collaborator and lecturer within the Institute of Photogrammetry and Cartography of TUM. Since 2011, he also worked as a senior scientist in the research cluster of computer vision, remote sensing, and navigation at Munich University of Applied Sciences.

His main research interests include active remote sensing technology towards reconstruction and analysis of spatial-temporal behaviors of objects, image processing and analysis, machine learning, and related environmental and industrial applications. He has already published nearly 100 academic articles in refereed international journals and conferences. He was the recipient of Best Presentation Award of the International Symposium on Mobile Mapping 2013 and multiple best research paper awards of IEEE GRSL/ISPRS events. Since 2016 he served as co-chair of ISPRS WG III/6 as well as guest editors of special issues for Remote Sensing Journal.