Applied Intelligence
https://doi.org/10.1007/s10489-022-03654-6

Deep subspace image clustering network with self-expression and self-supervision

Chao Chen [1] · Hu Lu [1,2] · Hui Wei [3] · Xia Geng [1]

Accepted: 19 April 2022

Abstract
The subspace clustering algorithms for image datasets apply a self-expression coefficient matrix to obtain the correlation between samples and then perform clustering. However, such algorithms proposed in recent years do not use the cluster labels in the subspace to guide the deep network and do not get an end-to-end feature extraction and trainable clustering framework. In this paper, we propose a self-supervised subspace clustering model with a deep end-to-end structure, which is called Deep Subspace Image Clustering Network with Self-expression and Self-supervision (DSCNSS). The model embeds the self-supervised module into the subspace clustering. In network model training, alternating iterative optimization is applied to realize the mutual promotion of the self-supervised module and the subspace clustering module. Additionally, we design a new self-supervised loss function to improve the overall performance of the model further. To verify the performance of the proposed method, we conducted experimental tests on standard image datasets such as Extended Yale B, COIL20, COIL100, and ORL. The experimental results show that the performance of the proposed method is better than the existing traditional subspace clustering algorithm and deep clustering algorithm.

Keywords Subspace clustering · Neural networks · Self-supervised learning · Self-expression coefficient matrix

1 Introduction

With the advent of the big-data era, clustering algorithms are increasingly urgently needed to deal with a large amount of unlabeled data. However, traditional clustering algorithms [1–5] are not only time-complex but also the clustering results are not satisfactory because the samples are often massive and high-dimensional, especially image data.

Recently, many works have been devoted to studying deep learning clustering algorithms, which extract the potential features of samples through unsupervised deep learning methods, and then transform the samples into low-dimensional space for clustering. Unlike the general deep clustering algorithms, deep subspace clustering [6–9] focuses on acquiring similarity information of the whole sample in the low-dimensional space rather than the feature acquisition of a single sample. The concept of self-expression layer is proposed in subspace clustering, which approximately replaces the original samples with the linear combination of other samples in the subspace. As a result, we can obtain a self-expression coefficient matrix, describing the similarity between samples and supporting the subsequent clustering algorithms. However, in the existing deep subspace clustering algorithms, most models pay attention to learning the self-expression coefficient matrix rather than making full use of the clustering results to update the model. In addition, these existing models do not form an end-to-end framework, where model training and clustering algorithms are separated and executed step by step, resulting in poor coupling.

In response to the above phenomenon, a novel deep subspace clustering model, DSCNSS (Deep Subspace Image Clustering Network with Self-expression and Self-supervision), is devised in this paper. We incorporate a self-supervised module into the original subspace clustering

✉ Hu Lu
luhu@ujs.edu.cn

[1] School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China

[2] Jiangsu Province Big Data Ubiquitous Perception and Intelligent Agricultural Application Engineering Research Center, Zhenjiang, China

[3] School of Computer Science, Fudan University, Shanghai, China

Published online: 16 June 2022

algorithm and designs a specific loss function for the self-supervised module in the proposed model. As a result, the clustering results and self-expression coefficient matrix of the subspace clustering module are fully utilized.

In addition, due to the different emphasis on optimization between the self-supervision module and the subspace clustering module, we design a new self-supervised training method for the proposed model and uses the alternating optimization strategy to optimize the whole model for the two modules. On the one hand, the subspace clustering module focuses more on the training of the self-expression coefficient matrix so that the results of spectral clustering are better. On the other hand, the self-supervision module focuses more on the feature extraction capabilities of the model, which will better optimize the encoder part of the auto-encoder. These two parts are not independent and share the same feature extractor, to jointly improve the deep clustering ability of the model through alternate optimization.

The main contributions of this paper can be summarized as follows:

- We devise a new self-supervised module in our proposed model. It combines with the original subspace clustering module, which can make full use of the self-expression coefficient matrix and clustering results of the model. Then together with the pseudo-labels obtained from clustering, we can pick out the positive and negative samples corresponding to each sample so that the model can be continuously optimized by using triplet loss and cross-entropy loss as loss functions for the downstream model training.
- This paper proposes a new mutual alternate optimization strategy for our model. Unlike the traditional joint optimization strategy, this approach can either focus on learning the self-expression coefficient matrix in the pretext task and thus assist the self-supervised learning in the downstream task, or focus on learning the classification module in the downstream task to reverse the optimization of the feature extraction part of the model. This strategy can take into account the individual functions of each module. It is also possible to jointly optimize the common parts of the modules.
- A large number of experiments show that our proposed model DSCNSS has a good performance in four large image datasets, which proves the effectiveness of the proposed model.

The rest of this paper is organized as follows. Section 2 presents the related works. Section 3 provides a self-supervised subspace clustering model with a deep end-to-end structure, i.e., DSCNSS. We evaluate the performance of the proposed method in Section 4. Section 5 concludes the paper.

2 Related work

In this section, we will show the related works of subspace clustering and self-supervised learning in recent years.

2.1 Subspace clustering

As one of the unsupervised algorithms, subspace clustering has been widely used in image data clustering and analysis in recent years. Most subspace clustering algorithms use the self-expression layer to learn the weight coefficient matrix of each sample by the linear combination of other samples in a specific space. This matrix is also known as the self-expression coefficient matrix. It is often represented by the symbol C. C not only show the linear combination weight of each sample with other samples but also carries the similarity information between samples. For example, if the sample x_i is approximately composed of the sample$\{x_a, x_b, x_c\}$

linearly, it means that there is a high similarity between the sample x_i and the sample $\{x_a, x_b, x_c\}$. Based on this point, the model can indirectly obtain the adjacency matrix information of samples by learning the self-expression coefficient matrix. Then we input it into spectral clustering [10, 11] to realize sample clustering.

So far, many subspace clustering algorithms have been proposed. For example, Elhamifar [12] and vidal [13] proposed the Sparse Subspace Clustering (SSC) algorithm, which successfully learns the linear expression between samples. However, minimizing C not only takes a long time to calculate but also is sensitive to noise in SSC. In order to solve this problem, Liu [14] proposed the Low Rank Representation (LRR) method, which uses kernel regularization based on SSC to minimize C. You et al. [15] proposed the elastic net subspace clustering (ENSC) method, which combines l_1 regularization and l_2 regularization, and balances the weight of both to optimize C.

After finding that clustering in the original space does not perform well, many researchers apply subspace clustering to the feature space instead of the original space[16–20], looking for a better linear combination expression for each sample in the feature space.

There are two basic methods: (1) One is to use the kernel method to map the original sample into the kernel sparse subspace, and then pass it into the self-expression layer to obtain the self-expression coefficient matrix, such as the Kernel Sparse Subspace Clustering algorithm proposed by Patel [21]; (2) The other is to use the deep neural network model. The sample can be non-linearly reduced to the feature space through feature extraction and then cluster in the subspace [22, 23]. Such as SSC with the pre-trained convolutional auto-encoder features and the classic DSC-Net [24]. AE + SSC reduces the dimensionality of features first and then subspace clustering. In comparison, DSC-Net integrates subspace

clustering and feature extraction by auto-encoder together. Valanarasu [25] proposed Overcomplete Deep Subspace Clustering Networks (ODSC) algorithm based on DSC-Net. This algorithm obtains the features of samples through different convolution layer parameters, which effectively solves the feature information loss.

Although the above subspace clustering algorithms have achieved good clustering results, improvements remain. For example, these models are not end-to-end trainable frameworks and do not use the results of subspace clustering in the process of optimization. We perform spectral clustering on the self-expression coefficient matrix obtained from the auto-encoder and combine the results of the spectral clustering with the self-expression coefficient matrix to select the corresponding positive and negative samples, which effectively utilizes the clustering results of the subspace and provides reliable supervised information for the downstream tasks.

2.2 Self-supervised learning

Self-supervised learning is an algorithm that uses auxiliary tasks to learn. It trains the network by constructing supervisory information using the model itself to assist downstream tasks to learn better feature representation. To better use the subspace clustering labels, Sun [26] proposed the self-supervised deep multi-view subspace clustering (S2DMVSC) algorithm, which uses the results of subspace clustering for self-supervised classification and learns the potential subspace representation of multiple views through clustering loss. Similar to S2DMVSC, Zhang [18] proposed the Self-Supervised Convolutional Subspace Clustering Network (S2ConvSCN) algorithm, which trains the classification module and subspace clustering module together to get a deep model with a better clustering effect.

In comparison, our proposed model not only adopts the fusion of self-supervision and subspace clustering but also designs a new loss function for the self-supervision module. Triple loss and cross-entropy loss are used as the classification loss to optimize the model, which makes full use of the output results of subspace clustering. In addition, to solve the problem of inconsistency in the optimization objectives of the two modules, we design an alternate optimization strategy to optimize the entire model.

3 Proposed method

More details about the proposed model will be shown in this section. First, we will introduce the overall model framework, as shown in Fig. 1. Then the subspace clustering module and self-supervision module will be introduced one after another. Finally, we will present how to train the whole model.

3.1 Model framework

Our proposed model can be divided into two modules, refer to Fig. 1: 1). The subspace clustering module comprises a convolutional auto-encoder, the self-expression layer, and the spectral clustering. The main focus of this module is to train a reliable self-expression coefficient matrix C so that the subsequent spectral clustering can achieve better clustering accuracy. 2). The self-supervision module is mainly composed of a feature extraction layer and a classification layer, in which the feature extraction layer shares the structure information and weight information of the network with the encoder layer of the auto-encoder [27] in the subspace clustering. This module focuses on the feature extraction of samples. After the self-supervised training, the feature extraction ability of the encoder layer in the convolution auto-encoder is enhanced, which promotes the learning of the self-expression layer in subspace clustering.

3.2 Subspace clustering module

As shown in Fig. 1, the subspace clustering module uses a convolutional auto-encoder to nonlinearly reduce the dimension of data X to subspace Z. In the low-dimensional subspace, each sample can be linearly combined with its approximate samples into a similar sample to replace the original sample to the maximum extent. Each sample can be combined by the weight coefficient of the linear combination of other samples to form a self-expression coefficient matrix C, where $C \in R^{n \times n}$.

The convolutional auto-encoder is used to extract the features of the sample. It first reduces the sample to a low-dimensional subspace and then restores the feature Z C obtained by the self-expression layer through the decoder to ensure that the dimensionality reduction of the auto-encoder is reliable. Among them, the feature $Z = Encoder(X)$ is extracted by the encoder layer of the auto-encoder, and the reconstructed sample $\widehat{X} = Decoder(ZC)$ is decoded by the decoder. The encoder of the autoencoder consists of three convolutional layers. The input image decreases in length and width with each encoder layer, but the number of channels increases. The self-expression layer consists of two fully connected layers, but only weight terms are involved in the forward propagation of neurons. No bias terms or activation functions are involved for linear representation between samples. The decoder is composed of three transposed convolution layers, which approximate the self-expressed sample features back to the original image. The loss function of the auto-encoder uses the mean square error as the reconstruction loss, that is,

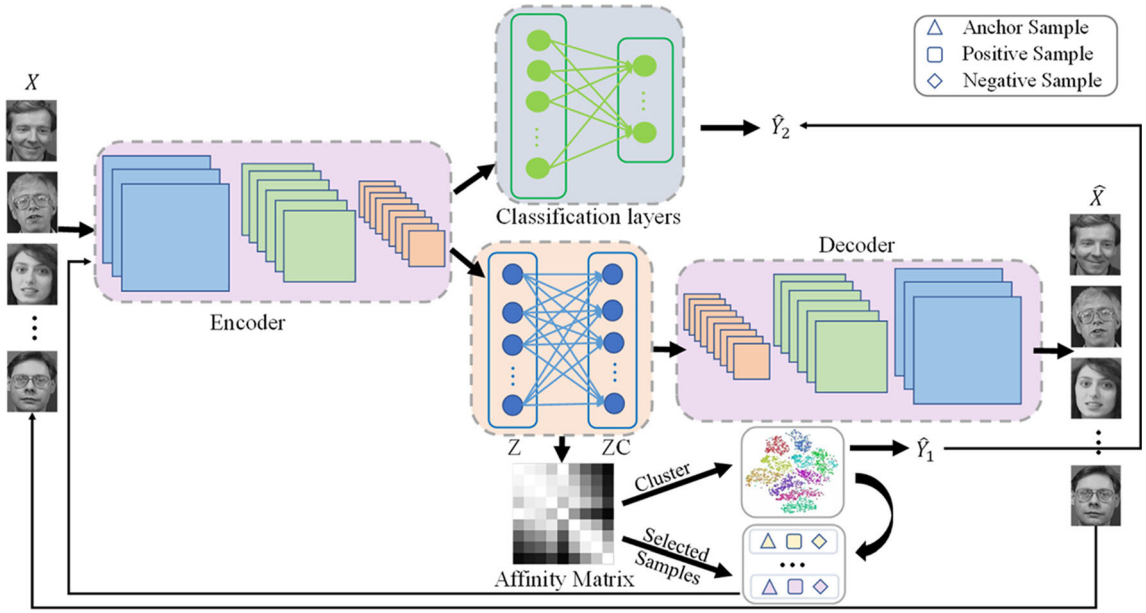


Fig. 1 Illustration of the overall architecture of the proposed model. The model obtains the self-expression coefficient matrix C through the convolutional auto-encoder and the self-expression layer. Then the model transforms C into adjacency matrix A and performs spectral clustering on A. Next, we use the obtained A to select positive and negative samples, and perform self-supervised training with the pseudo-label information after clustering to optimize the entire model

$$L_0 = \frac{1}{2N}\sum_{i=0}^{N} \| x_i - \widehat{x}_i \|_2^2 = \frac{1}{2N} \| X - \widehat{X} \|_F^2, \tag{1}$$

where N is the total number of samples.

It has been shown in [28] that when the subspaces are independent of each other, the diagonal structure of C can be achieved by minimizing some norms of C. Therefore, in order to train a better C, we use the following two sub-loss functions to optimize the self-expression layer,

$$L_1 = \| C \|_p, (diag(C) = 0) \tag{2}$$

$$L_2 = \frac{1}{2} \| Z - ZC \|_F^2, \tag{3}$$

where $\| \cdot \|_p$ denotes the regularization constraint of the matrix; $diag(C) = 0$ is a constraint set to prevent the model from falling into a trivial solution, that is, to prevent the occurrence of $C = I$. The purpose of the sub-loss function L_2 is to make the new features of the linearly combined samples approximate to the original features.

After training the above auto-encoder, the model can output a matrix C which is obtained from the self-expression coefficient matrix, so that the i -th row and j -th column in the C can effectively represent the similarity of the i -th sample to the j -th sample. We get a symmetric matrix $A = \frac{C^T + C}{2}$. At this point, the A can be used as the similarity matrix of all samples, i.e., the adjacency matrix. Then we further obtain the degree matrix D and Laplacian matrix through A, and regularize the Laplacian matrix. All samples can be clustered by spectral clustering [10] using the Laplacian matrix, and the corresponding pseudo-label $S_{cluster}$ can be obtained to provide supervision information for the following self-supervision module.

3.3 Self-supervision module

The self-supervision module takes the subspace clustering task as the pretext task. It uses the pseudo labels obtained from the clustering results to supervise the feature extraction ability of the training model. The convolutional layer shares the structure and weights with the encoder of the auto-encoder. The fully connected layer has two layers. The number of neurons in the last fully connected layer is the number of classes of the samples. Considering the collaboration between two modules, we propose a novel idea about subspace clustering, "triplet loss + cross-entropy loss" is used as the loss function of self-supervision to optimize the model. The matrix A which carries the correlation information between samples and the labels obtained by clustering, can help select positive and negative samples, making further use of the sample similarity information in matrix A.

The loss value of a simple triple can easily be reduced to 0. However, if the training network only learns simple triplet loss, the generalization ability of the network will be limited, and the network does not learn any features. Therefore, when

a training batch contains a large number of easy triplets will significantly reduce the convergence speed of the training network. In order to select the appropriate triple samples, this paper provides the following sample selection strategies,

$$\begin{cases} x_i^a = x_i, 0 \le i < N \\ x_i^p = x_j, \underset{j}{\operatorname{argmin}}(A_{ij}) \ and \ S_{cluster_i} = S_{cluster_j} \\ x_i^n = x_k, \underset{k}{\operatorname{argmax}}(A_{ik}) \ and \ S_{cluster_i} \neq S_{cluster_k} \end{cases}, \tag{4}$$

The selection rule of the positive sample x_i^p is to find a sample with the lowest similarity to the anchor (that is, the similarity matrix A obtains the lowest value in row i) on the premise that it belongs to the same cluster (that is, the pseudo-label of x_i^p is the same as x_i^a). The selection rule of negative sample x_i^n is to find a sample with the highest similarity to the anchor on the premise that it does not belong to the same cluster as the first sample. According to Eq. (4), we can see that although the x_i^p we selected belongs to the same class as the x_i^a, the similarity between x_i^p and x_i^a in A is not very high. In this way, when optimizing the model, the value of loss forces the model to close the distance between x_i^p and x_i^a, which has the effect of closing the distance between the samples in the same class. While the x_i^n are similar to the x_i^a according to matrix A, they do not belong to the same class after clustering. The model will pay more attention to the differences between x_i^n and x_i^a when optimizing the loss, and then let the x_i^n move away from the x_i^a in the process of training the model, which has the effect of increasing the distance between the samples in the different class.

Then, the triples $\{x_i^a, x_i^p, x_i^n\}(i = 0, 1, 2, \ldots, N)$ selected in the previous step are introduced into the self-supervision module, and the features of the samples are extracted through the feature extraction layer to obtain the corresponding triple loss L_3.

$$L_3 = \frac{1}{N}\left\{\sum_{i=0}^{N}\left[\| f(x_i^a) - f(x_i^p) \|^2 - \| f(x_i^a) - f(x_i^n) \|^2 + margin\right]_+\right\}, \tag{5}$$

where $f(x)$ represents the feature extraction of sample x; $[x]_+$ means that if $x < 0$, $[x]_+ = 0$, otherwise $[x]_+ = x$; $margin$ refers to the minimum interval between two distances, where one distance is between positive samples and the anchor and the other is between negative samples and the anchor.

In addition, the pseudo-label obtained by clustering is taken as the real label of the sample, and we use the cross-entropy to guide the classification of the last full connection layer in the self-supervision module. The cross-entropy loss function L_4 is as follows,

$$L_4 = -\frac{1}{N}\sum_{i=0}^{N} S_{cluster}(x_i) \cdot \log\left(\mathrm{S}_{\mathrm{classify}}(\mathrm{x_i})\right), \tag{6}$$

where $S_{cluster}(x_i)$ is clustering pseudo-label corresponding to the sample x_i; $S_{classify}(x_i)$ denotes the prediction label of x_i in self-supervision module.

For the self-supervision module, the advantages of using "triple loss + cross-entropy loss" as the total loss function is mainly twofold: 1) The triple loss can make the difference between positive and negative samples more evident through the comparison of positive and negative samples. It focuses on enlarging the inter-class distance. 2)The cross-entropy loss uses pseudo-labels obtained by spectral clustering to make similar samples closer, focusing on narrowing the intra-class distance. Therefore, the combined loss function can play a better clustering role in the process of self-supervised training.

3.4 Model training

The model framework proposed in this paper is mainly composed of two modules, i.e., the subspace clustering module and the self-supervision module. Because of the different emphasis of the two modules, we propose an alternating optimization strategy to iterate the optimization model. These two modules are not independent of sharing the same feature extractor. The feature extraction ability of the encoder layer in the auto-encoder is improved by alternating optimization. Therefore, the total loss function L of the model is:

$$L = \alpha_0 \cdot L_0 + \alpha_1 \cdot L_1 + \alpha_2 \cdot L_2 + \alpha_3 \cdot L_3 + \alpha_4 \cdot L_4, \tag{7}$$

where L_0 is the reconstruction loss of the auto-encoder; L_1 is the regularization constraint of the self-expression coefficient matrix; L_2 is the self-expression loss; L_3 is the triple loss; L_4 is the cross-entropy loss; α_0,α_1,α_2,α_3,α_4 are the weight parameters corresponding to the loss function of each sub-term, respectively.

We will present the steps of model training and optimization in detail.

Step 1: pre-train convolution auto-encoder

In the initial stage, we use the auto-encoder as the pre-training model to extract the preliminary features through the encoder layer of the auto-encoder. Therefore, in the pre-training stage, we use the reconstruction loss L_0 of the auto-encoder as the loss function of the model, and the auto-encoder adopts the convolution structure, which is consistent with the auto-encoder in the subsequent subspace clustering.

Step 2: update subspace clustering module

The red dotted frame in Fig. 2 shows the main training part of the module. When training this module, the model will fix the parameters of the full connection layer in the self-supervision module and optimize the encoder layer, decoder layer, and self-expression layer of the auto-encoder. The objective function of this module is,

$$J_1 = \min\{\alpha_0 \cdot \mathrm{L}_0 + \alpha_1 \cdot \mathrm{L}_1 + \alpha_2 \cdot \mathrm{L}_2\}, \tag{8}$$

where J_1 focuses on optimizing the three sub-loss items of L_0, L_1, L_2, that is, the parameters of α_3 and α_4 in the total loss function are set to 0 for training.

In the process of training, the forward propagation process of the convolutional auto-encoder can be expressed as follows:

$$a^l = \sigma(h^l) = \sigma\left(\sum_{k=1}^{M} h_k^l\right) = \sigma\left(\sum_{k=1}^{M} a_k^{l-1} * W_k^l + b^l\right), l = \{2, 3, \ldots, \mathrm{L}\}, \tag{9}$$

where a^l represents the output of the l layer; $\sigma(\cdot)$ denotes the activation function; M represents the convolution of M submatrices; W is the weight of the convolution kernel, and b is the bias.

For the middle self-expression layer, there are

$$Z = Flatten(a^{mid}), \tag{10}$$

$$a^{mid+1} = Z \cdot C s.t. ZC - Z = 0 \; and \; diag(C) = 0, \tag{11}$$

where a^{mid} is the middle layer of the auto-encoder, that is, the output of the last convolution layer in the encoder; a^{mid+1} is the input to the decoder; $ZC - Z = 0$ is to ensure that the self-expression layer learns the linear expression between samples; $Flatten(\cdot)$ represents a flattening operation.

Step 3: update self-supervision module

The green dotted frame in Fig. 2 shows the main training part of the self-supervision module. Before training this module, we need to use A obtained in the previous step for sample selection and clustering. During the training, the model fixes the decoder layer and self-expression layer of the auto-encoder and optimizes the encoder layer of the auto-encoder and the full connection layer in the self-supervision module. At this point, the objective function of the module is,

$$J_2 = \min\{\alpha_3 \cdot \mathrm{L}_3 + \alpha_4 \cdot \mathrm{L}_4\}, \tag{12}$$

where J_2 focuses on optimizing the two sub-loss items of L_3, L_4 i.e., the parameters of α_0, α_1 and α_2 in the total loss function are set to 0 for training.

In the process of training, the forward propagation process of the feature extraction layer is consistent with the subspace clustering module and shares the weight. The classification layer consists of two fully connected layers. The propagation of the first fully connected layer, $a^{Linear1}$, can be expressed as:

$$a^{Linear1} = ReLU(h^l) = ReLU\left(W^l \cdot a^{l-1} + b^l\right), \tag{13}$$

where a^{l-1} is the output of the convolution layer.

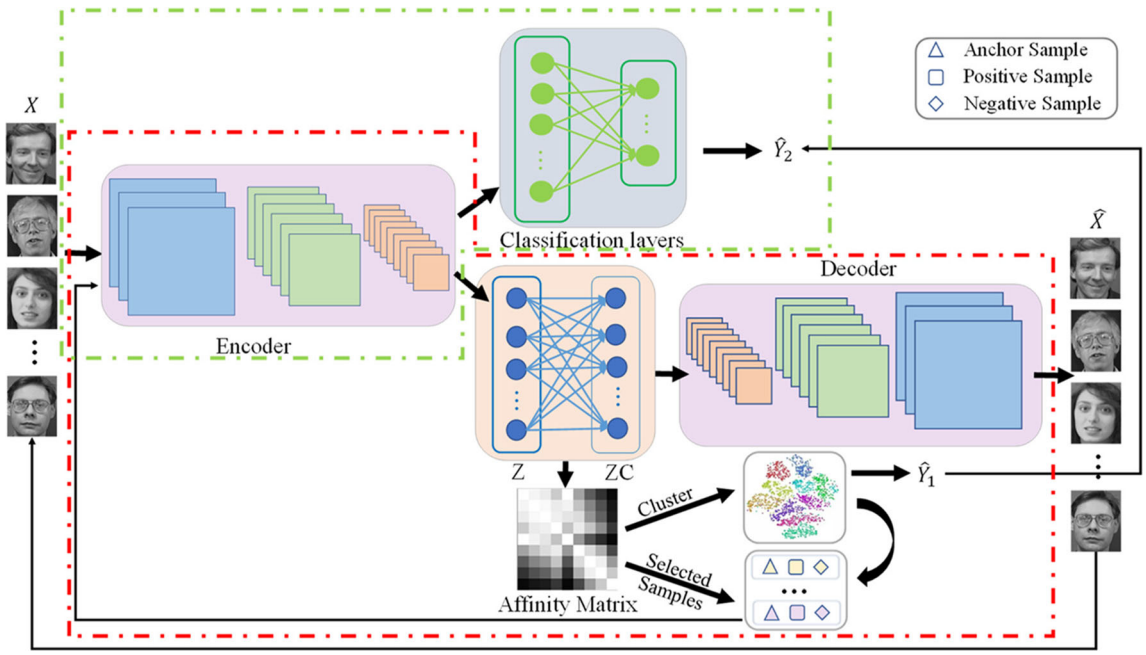


Fig. 2 Subspace clustering module and self-supervision module

The output of the fully connected layer $a^{Linear2}$ of the second layer is:

$$a^{Linear2} = softmax(h^L) = softmax(W^L \cdot a^{Linear1} + b^L), \quad (14)$$

where L represents the last layer; $softmax(.)$ is the activation function; the number of neurons in $a^{Linear2}$ layer is the total number of categories of the dataset, and the vector output by the activation function is the prediction label of the sample.

Note that the self-supervision module often needs to be trained for a long time to become stable because of the instability of the labels in the clustering results. On that account, we apply the "Hungarian algorithm" to maximize the matching between the clustering pseudo-labels obtained currently and the last clustering pseudo-labels.

After the alternating optimization, the self-expression coefficient matrix C is transformed into the adjacency matrix A, and then the final clustering result is obtained by spectral clustering.

The whole algorithm proposed in this paper is described as follows:

Algorithm 1: Deep Subspace Image Clustering Network with Self-expression and Self-supervision（DSCNSS）

Input: Input data:X; Number of the cluster: K; Tradeoff parameters:$\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4$; Maximum iterations:$T_{max}$.
Output: Labels: $S_{cluster}, S_{classify}$

```
Initialize autoencoder's weight via pre-training Convolutional autoencoder
while iter≤ T_max do
    Train Deep Subspace Clustering Networks using (1-3,8-11) and get C.
    Calculate matrix A by transforming matrix C.
    Spectral Cluster with A and get pseudo-labels: S_cluster.
    Select the triple samples with A and S_cluster using (4)
    Train Self-Supervision Modules using (5-6,12-14) and get S_classify
    if S_cluster do not change then
        Stop Training.
    end
end
```

4 Experiment

In this section, we will present the experimental results of the proposed algorithm on four commonly used datasets and compare them with the recent subspace clustering algorithms.

4.1 Dataset

To verify the effectiveness of the algorithm proposed in this paper, we select four commonly used standard image datasets as test objects in the experiment. Because our proposed model is an unsupervised clustering model, the real labels of the samples do not participate in the training of the model. The datasets we use mainly include two types. One is the face image datasets like Extended Yale B and ORL. The other is the object datasets like COIL20 and COIL100. Some of the samples in each dataset are visualized, as shown in Fig. 4.

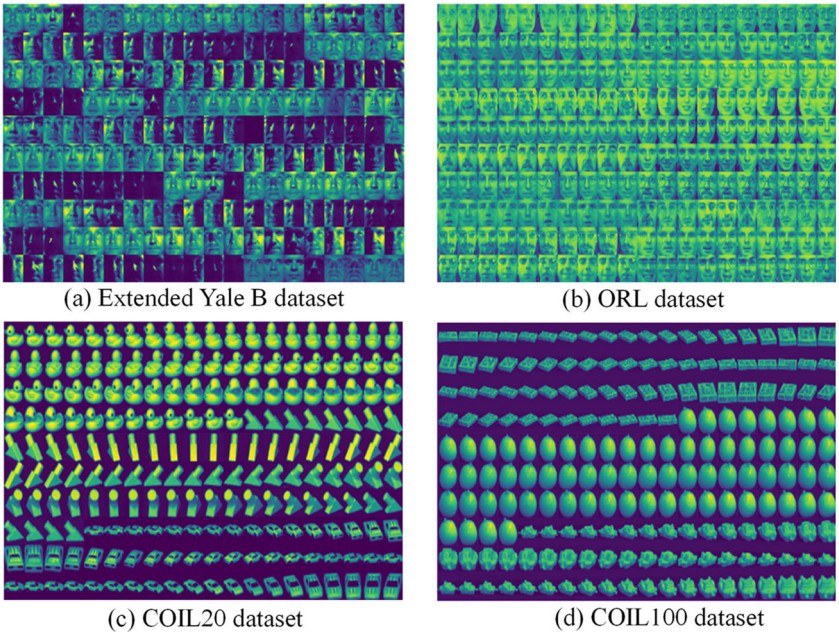
The Extended Yale B dataset [29] is one of the most commonly used datasets in subspace clustering. The dataset collects the facial information of 38 different people, each with 64 different faces, a total of 2432 pictures. Each picture consists of a single-channel grayscale image consisting of 48*42pixels, refer to Fig. 3(a).

The ORL dataset [30] consists of 40 pictures of faces of different people, each containing 10 different faces, with a total of 400 faces. The dimension of each image is 32*32*1, refer to Fig. 3(b). Compared with the Extended Yale B dataset, the ORL dataset is more challenging.

COIL20 dataset is a small object dataset that contains 20 different types of objects, such as cars, erasers, building blocks, etc. Each object has 72 different pictures, refer to Fig. 3(c). The dimension of each sample is 32*32*1.

COIL100 dataset [31] is also a small object dataset similar to the COIL20 dataset. This dataset collects 100 different kinds of objects, which increases the difficulty of clustering. Each type of object has 72 different pictures, and each picture is a 32*32 single-channel image, refer to Fig. 3(d).

The details of the four datasets are shown in Table 1.



(a) Extended Yale B dataset (b) ORL dataset

(c) COIL20 dataset (d) COIL100 dataset

Fig. 3 Visualization of partial samples in datasets

4.2 Clustering metrics

Different from the supervised algorithm, the prediction label of clustering and the real label of the sample can not be measured by an ordinary accuracy index. Therefore, we use the clustering error rate as a metric to evaluate the performance of our model, which is commonly used in subspace clustering algorithms. The specific formula for calculating the clustering error rate is as follows,

$$Clustering\ error\ rate = 1 - \frac{\sum_{i=1}^{n} \delta(y_i, map(\widehat{y}_i))}{n}, \qquad (15)$$

where y_i and $\widehat{y}_i$ are the real label and prediction label corresponding to sample x_i; n is the total number of samples, and δ is the indicator function. Note that $\delta(\mathrm{x}, \mathrm{y}) = \begin{cases} 1, & if\ x = y \\ 0, & otherwise \end{cases}$.

Table 1 Details of the dataset

Dataset	Class	Images/class	Total	Size
Extended Yale B	38	64	2432	48×42×1
ORL	40	10	400	32×32×1
COIL20	20	72	1440	32×32×1
COIL100	100	72	7200	32×32×1

The lower the value of the clustering error rate, the better the performance of the model, and the clustering result is closer to the real category of the sample.

4.3 Implementation details

In the pre-training model stage, firstly, the input samples are enhanced, and we input the samples with Gaussian noise into the convolutional auto-encoder. On the one hand, the data enhancement can prevent the model from overfitting. On the other hand, the weights of the auto-encoder can be initialized by pre-training. Then, in the process of subspace clustering and self-supervision, the weight parameters of the sub loss function are designed as follows $\alpha_0 = 1, \alpha_1 = 1, \alpha_3 = 0.3$, $\alpha_4 = 0.7$. Note that as a super-parameter, α_2 needs to be set separately for each dataset (the parameter selection refers to the design scheme in DSC-NET[16]). In the Extended Yale B dataset, let $\alpha_2 = 1.0 \times 10^{\frac{K}{10}-3}$, and K is the number of categories of the dataset; In the ORL dataset, let $\alpha_2 = 0.2$; In the COIL20 and COIL100 datasets, let $\alpha_2 = 150, \alpha_2 = 30$. The full connectivity layer in self-supervision consists of two layers. The first layer is composed of $\frac{h}{2}$ neurons (h is the flattened dimension of the last convolution layer in the encoder), and the second layer is composed of K neurons (K is the total category of each dataset). The design of convolution kernel and channel number in convolution auto-encoder is shown in Tables 2 and 3.

The optimizer used in the subspace clustering module is Adam, and the learning rate is 10^{-3}. The activation function

Table 2 Design Scheme of auto-encoder structure for Extended Yale B and ORL datasets

Dataset		encoder-1	encoder-2	encoder-3	decoder-1	decoder-2	decoder-3
Extended Yale B	kernel Size	5×5	3×3	3×3	3×3	3×3	5×5
	channels	10	20	30	30	20	10
ORL	kernel Size	5×5	3×3	3×3	3×3	3×3	5×5
	channels	5	3	3	3	3	5

corresponding to the convolution layer in this module is the ReLU function. The self-supervision module also uses the Adam method to optimize the model, and the learning rate is 10^{-4}.

4.4 Evaluation of clustering algorithm

After testing the above algorithm in four datasets, we calculate their mean values as the final experimental results. In the comparative experiment, our algorithm is compared with the subspace clustering algorithms proposed in recent years, such as Sparse Subspace Clustering (SSC)[12], Elastic Net Subspace Clustering(ENSC)[15], Kernel Sparse Subspace Clustering (KSSC)[21], SSC by Orthogonal Matching Pursuit (SSC-OMP)[32], Efficient Dense Subspace Clustering (EDSC)[28], Low Rank Representation (LRR)[14], Low Rank Subspace Clustering (LRSC)[33], SSC with the pre-trained convolutional auto-encoder features (AE + SSC) Deep Subspace Clustering Networks (DSC-Net)[24], Finding Good Neighbors Subspace Clustering (FGNSC)[17] and the latest Deep self-representative subspace clustering network[34], Overcomplete Deep Subspace Clustering Networks (ODSC)[25] algorithms. If the compared algorithms have not been tested on the dataset, the experimental results are obtained by using the hyperparameters mentioned in the paper. If the code is not published or cannot experiment on the dataset, we replace the corresponding result with a minus sign (-).

Table 4 shows that the algorithm proposed in this paper has achieved good clustering results on Extended Yale B, ORL, COIL20, and COIL100. Except that the error rate of COIL20 and COIL100 is slightly higher than that of the DSSC algorithm, our algorithm is better than the comparison methods in experiments.

Table 3 Design Scheme of auto-encoder structure for COIL20 and COIL100 datasets

Dataset		encoder-1	decoder-3
COIL20	kernel Size	3×3	3×3
	channels	15	15
COIL100	kernel Size	5×5	5×5
	channels	50	50

4.5 Experimental discussion

4.5.1 Ablation study of each module

We perform the ablation study analysis to verify the effectiveness of the self-supervised module and the alternative training strategy proposed in this paper. We compare experimental results in three cases: (1) deep subspace clustering, (2) deep subspace clustering + self-supervised learning (joint optimization), (3) deep subspace clustering + self-supervised learning (alternating optimization), respectively corresponding to method 1 method 2 and method 3 in Fig. 4.

This comparison shows that the self-supervision module and alternate training strategy proposed in this paper further improve the performance of the model.

Table 4 Clustering error rates tested by different algorithms on four datasets. For all the metrics, the lower value is better. The best results are shown in bold.

	Yale B	ORL	COIL20	COIL100
SSC[12]	26.46	25.75	13.69	44.90
ENSC[15]	24.63	24.75	12.40	38.88
KSSC[21]	30.79	28.57	29.13	47.18
SSC-OMP[32]	26.28	29.00	35.90	67.29
EDSC[28]	11.86	29.62	16.29	38.13
LRR[14]	15.01	19.00	18.82	53.18
LRSC[33]	20.69	28.00	25.84	50.67
AE+SSC	25.20	24.37	12.89	43.93
DSC-NET-l_1 [24]	3.19	14.50	6.86	33.62
DSC-NET-l_2 [24]	2.67	14.00	6.86	30.96
FGNSC[17]	5.76	-	9.93	-
ODSC[25]	2.22	12.00	2.50	-
DSSC[34]	1.89	11.50	**2.01**	**27.48**
DSCNSS-l_1	2.08	11.32	3.94	30.34
DSCNSS-l_2	**1.85**	**10.79**	3.67	28.58

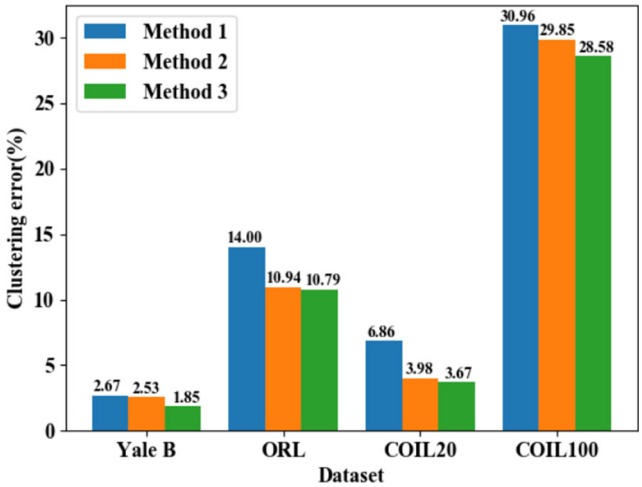


Fig. 4 Experimental analysis of ablation

4.5.2 Exploration and visualization of self-expression coefficient matrix *C*

The model in the subspace clustering module expresses the correlation between samples through the self-expression layer and indirectly transmits the similarity information between samples. To explore whether the self-expression coefficient matrix C learns this property, we tested it on ORL datasets. Figure 5 shows C trained by the model. The higher the value of row i and column j in the image is, the higher the similarity between sample i and sample j will be, and the whiter the area displayed in the figure will be. Therefore, it can be seen that each sample has a higher similarity with its adjacent points. It shows that each sample can be obtained by a linear combination of samples with high similarity, which is consistent with the results we expect. Additionally, we find that the samples have 40 different categories (each fuzzy white square in the picture is a category), which is consistent with the facial information of 40 different people in the dataset. The above analysis shows that the obtained C can indeed learn the linear combination representation between samples in the model, which provides a good source of adjacency matrix information for the following clustering.

4.5.3 Discussion on the convergence of loss function

With the continuous optimization of the model, the value of the loss function will continue reducing. To study the values of five sub-loss functions in the proposed model and the changing trend of J_1 and J_2 during alternating training, we test on Extended Yale B datasets. Figure 6(a-g) shows that with the increase of the number of iterations, the loss of each term of the model will gradually decrease until it tends to be stable (except the regular term of C). Among them, C is initialized to an all-zero matrix in the program. With the increase of epoch, more and more samples participate in the linear combination, which results that the sub-loss can still learn the linear representation between samples under the premise that C is a diagonal matrix as far as possible [28]. From the experimental results of each sub-loss function, we agree that the model has good convergence.

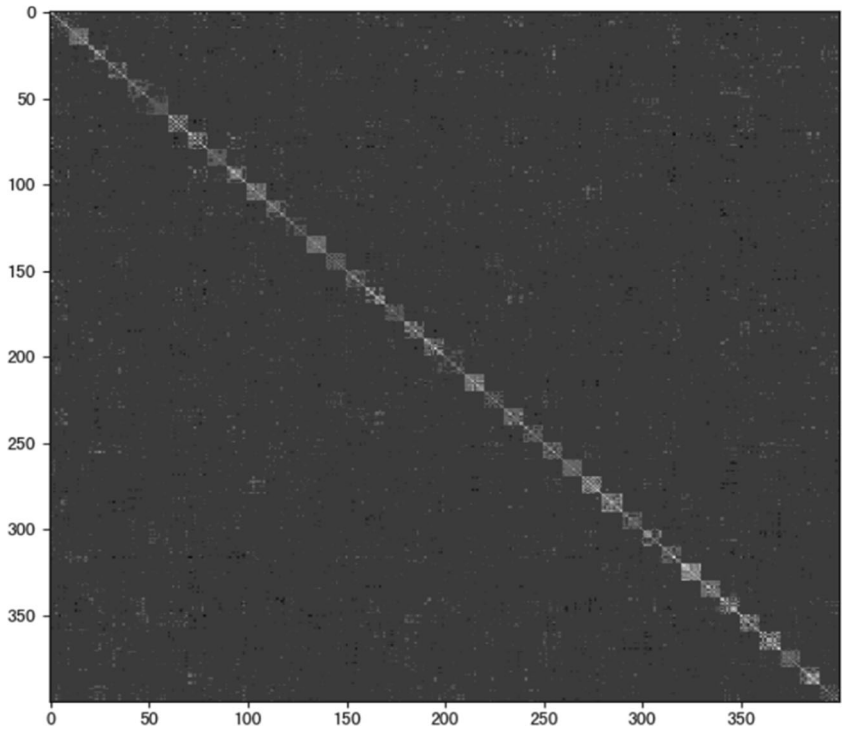
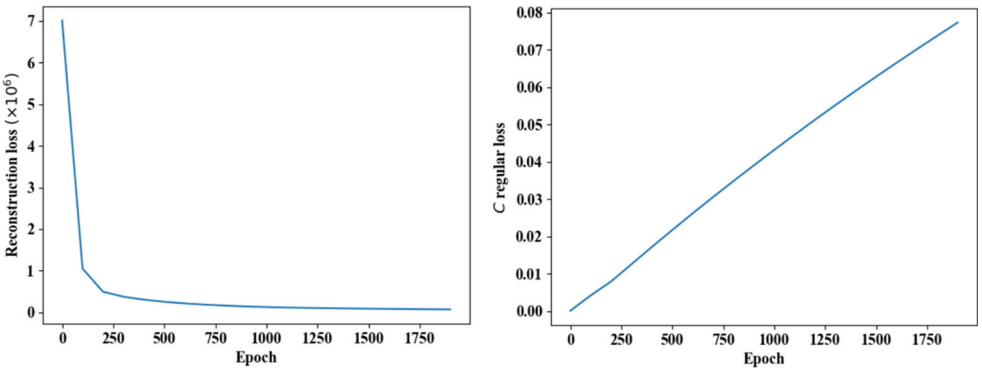
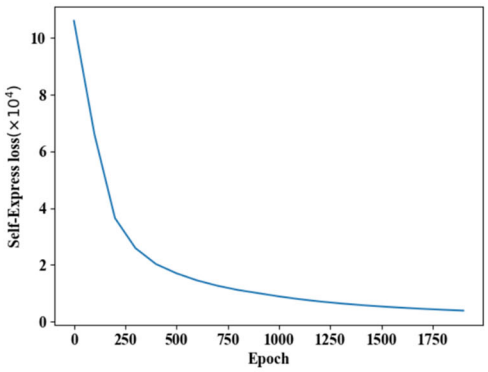


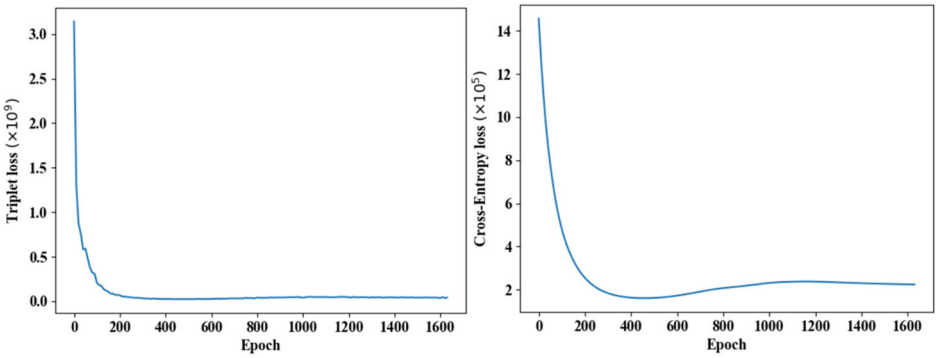
Fig. 5 Self-expression coefficient matrix C of ORL dataset



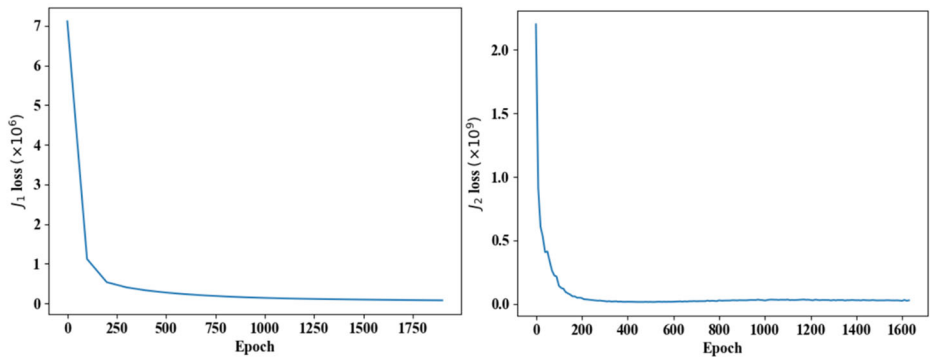
(a)Reconstruction loss (b) Regularization loss of Matrix C



(c) Self-expression loss



(d)Triplet loss (e) Cross-Entropy loss



(f) J_1 loss (g) J_2 loss

Fig. 6 The changing trend of the loss function value of each item. (**a**) Reconstruction loss (**b**) Regularization loss of Matrix C (**c**) Self-expression loss (**d**)Triplet loss (**e**) Cross-Entropy loss (**f**)J_1 loss (**g**)J_2 loss

4.5.4 Sensitivity analysis of hyperparameters in the model

The total loss function proposed in this paper includes five hyperparameters. It should be noted that the first three hyperparameters in the loss function, namely α_0, α_1, α_2, are referring to the parameter scheme in the DSC-Net[24]. In which, $\alpha_0 = 1; \alpha_1 = 1$. In the Extended Yale B dataset, let $\alpha_2 = 1.0 \times 10^{\frac{K}{10}-3}$, and K is the number of categories of the dataset; In the ORL dataset, let $\alpha_2 = 0.2$; In the COIL20 and COIL100 datasets, let α_2 =150; α_2 =30. α_3 and α_4 are used to control the percentage of triplet loss and cross-entropy loss in the self-supervised training, which is obtained after model debugging. In the training of the downstream model, we need to set α_0, α_1 and α_2 to zero, so we only need to consider the percentage of triplet loss and cross-entropy loss for the time being, i.e., in the paper, the optimization process of $J_2 = \min\{\alpha_3 \cdot \mathrm{L}_3 + \alpha_4 \cdot \mathrm{L}_4\}$. In our experiments, we assume $\alpha_4 = 1 - \alpha_3$, then $J_2 = \min\{\alpha_3 \cdot \mathrm{L}_3 + (1 - \alpha_3) \cdot \mathrm{L}_4\}$, so that we can find the percentage value of both. We take the value interval of 0.1 for α_3. The experimental results on the ORL dataset are shown in Fig. 7.

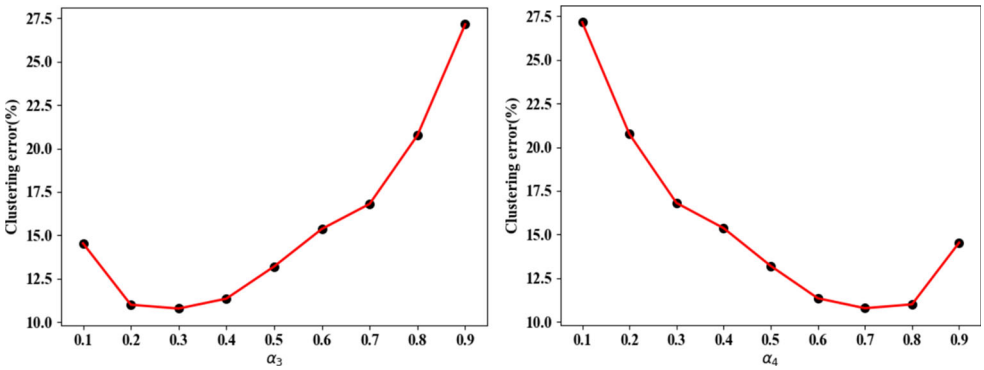
From Fig. 7, we can see that the clustering error rate reaches the minimum value when $\alpha_3 = 0.3$ and $\alpha_4 = 0.7$, and the performance of the model is the best at this time. If α_3 is smaller, then L_3 accounts for a smaller proportion, and the role of cross-entropy loss plays a larger role at this time. If α_3 is larger, the L_3 share is larger, and the role of triplet loss plays a larger role at this time. From the experimental results in the figure, it can be seen that when the ratio of triplet loss and cross-entropy loss is 3:7, the roles of both can take into account each other and promote the model to achieve the optimal performance.

4.5.5 Sample distribution of subspace clustering

To show the clustering effect of this algorithm clearly, we apply the t-SNE algorithm to reduce the dimension of the samples in the COIL20 dataset from the feature space to two-dimensional and visualize the final clustering effect in the plane coordinate system. Figure 8(a) shows the sample distribution in the original space; Fig. 8(b) shows the sample distribution after spectral clustering; Fig. 8(c) shows the sample distribution after deep subspace clustering; Fig. 8(d) shows the sample distribution after clustering with the proposed model. There are a large number of wrong labels in the sample tags obtained by spectral cluster, such as the samples in the red box in Fig. 8(b), where the samples in each box should belong to the same category but are marked into different categories. The DSC-Net algorithm has a significant improvement compared with spectral clustering, but there is also a small number of mislabeling. For example, a small number of samples in the red dotted frame in Fig. 8(c) are marked as wrong categories. The sample tags clustered by our proposed algorithm have improved the mislabeling behavior of the DSC-Net algorithm, referring to the red dotted frame in Fig. 8(d). Through the visual analysis after sample dimensionality reduction, the sample distribution obtained by our proposed algorithm is closest to the label distribution of the original spatial samples, and the clustering effect is the best.

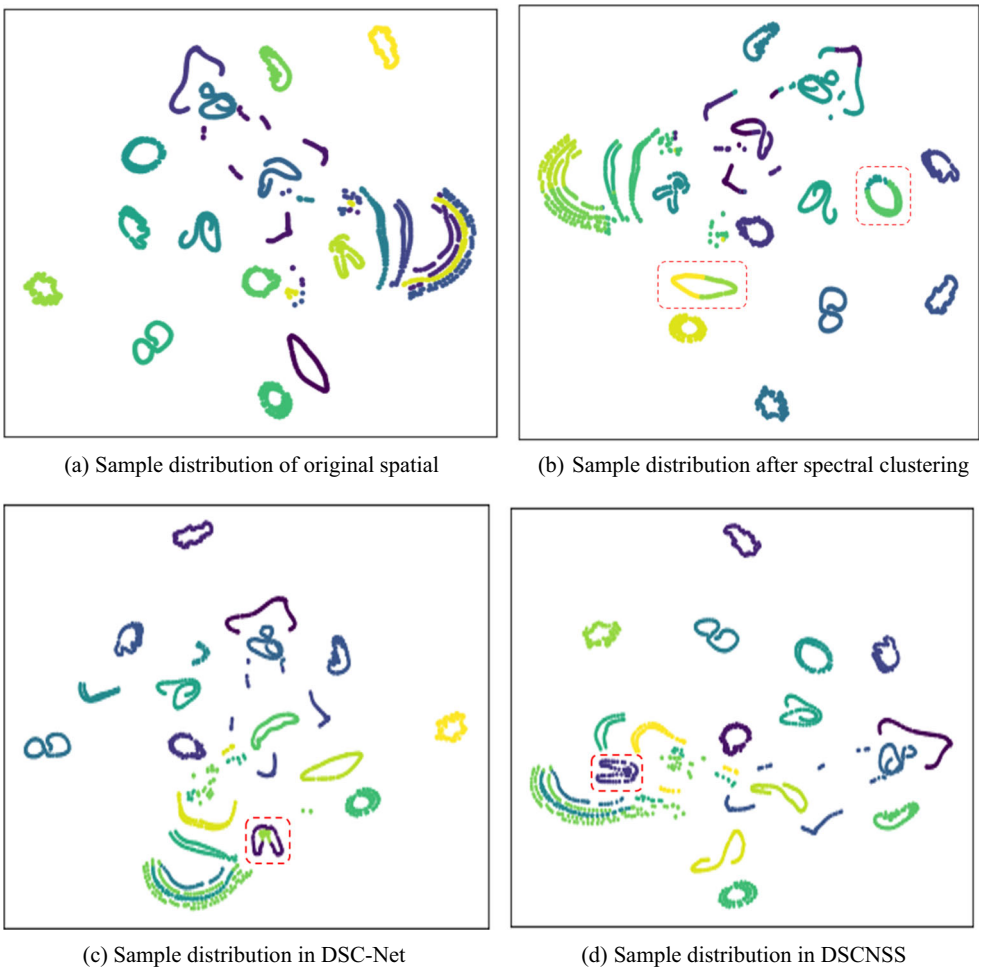
5 Conclusions

We proposed a novel model called Deep Subspace Image Clustering Network with Self-expression and Self-supervision (DSCNSS) in this paper. The algorithm combines a self-supervision mechanism with subspace clustering. In subspace clustering, the model can get the self-expression coefficient matrix, then transform it into the adjacency matrix and cluster to get the cluster id of each sample as pseudo-label. At the same time, the adjacency matrix is used to select samples. In the self-supervision module, the pseudo-label information of the sample is used for supervised training to optimize the model. The two modules are optimized alternately to improve the accuracy of the model. We conduct extensive experiments on four high-dimensional image datasets, and the model also obtains the best test results on the four



(a) Sensitivity analysis of α_3 (b) Sensitivity analysis of α_4

Fig. 7 Sensitivity analysis of hyperparameters in ORL dataset. (**a**) Sensitivity analysis of α_3 (**b**) Sensitivity analysis of α_4



(a) Sample distribution of original spatial

(b) Sample distribution after spectral clustering

(c) Sample distribution in DSC-Net

(d) Sample distribution in DSCNSS

Fig. 8 Visualization effect of samples in COIL20 dataset. (**a**) Sample distribution of original spatial (**b**) Sample distribution after spectral clustering (**c**) Sample distribution in DSC-Net (**d**) Sample distribution in DSCNSS

image datasets. Experiments demonstrate that our method achieves better performance than most state-of-the-art methods. In future work, we will further explore how to use the self-supervised subspace clustering algorithm to handle the larger number of datasets as well as datasets with more complex sample distributions [35].

Acknowledgements This work was supported by the National Science Foundation of China under Grant 61976108 and the Postgraduate Research & Practice Innovation Program of Jiangsu Province (Project No. SJCX20_1416).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, vol 1, no 14, pp 281–297, Oakland, CA, USA
2. Ester M, Kriegel H-P, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. kdd 96(34):226–231
3. Johnson SC (1967) Hierarchical clustering schemes. Psychometrika 32(3):241–254
4. Koga H, Ishibashi T, Watanabe T (2007) Fast agglomerative hierarchical clustering algorithm using Locality-Sensitive Hashing. Knowl Inf Syst 12(1):25–53
5. Lu H, Song Y, Wei H (2020) Multiple-kernel combination fuzzy clustering for community detection. Soft Comput 24(18):14157–14165
6. Peng X, Feng J, Xiao S, Yau W-Y, Zhou JT, Yang S (2018) Structured autoencoders for subspace clustering. IEEE Trans Image Process 27(10):5076–5086

7. Peng X, Feng J, Zhou JT, Lei Y, Yan S (2020) Deep subspace clustering. IEEE Trans Neural Netw Learn Syst 31(12):5509–5521
8. Zhou P, Hou Y, Feng J (2018) Deep adversarial subspace clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1596–1604
9. Liu M, Wang Y, Sun J, Ji Z (2021) Adaptive low-rank kernel block diagonal representation subspace clustering. Appl Intell 52(2): 2301–2316
10. Yang X, Deng C, Liu X, Nie F (2018) New l2, 1-norm relaxation of multi-way graph cut for clustering. In: Thirty-Second AAAI Conference on Artificial Intelligence
11. Ng AY, Jordan MI, Weiss Y (2002) On spectral clustering: Analysis and an algorithm. In: Advances in neural information processing systems, pp 849–856
12. Elhamifar E, Vidal R (2013) Sparse subspace clustering: Algorithm, theory, and applications. IEEE Trans Pattern Anal Mach Intell 35(11):2765–2781
13. Vidal EER (2009) Sparse subspace clustering. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol 6, pp 2790–2797
14. Liu G, Lin Z, Yu Y (2010) Robust subspace segmentation by low-rank representation. In: Icml, vol 1, p 8, Citeseer
15. You C, Li C-G, Robinson DP, Vidal R et al (2016) Oracle based active set algorithm for scalable elastic net subspace clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3928–3937
16. Lu H, Liu S, Wei H, Tu J (2020) Multi-kernel fuzzy clustering based on auto-encoder for fMRI functional network. Expert Syst Appl 159:113513
17. Yang J, Liang J, Wang K, Rosin PL, Yang M-H (2019) Subspace clustering via good neighbors. IEEE Trans Pattern Anal Mach Intell 42(6):1537–1544
18. Zhang J et al (2019) Self-supervised convolutional subspace clustering network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 5473–5482
19. Liu M, Wang Y, Sun J, Ji Z (2020) Structured block diagonal representation for subspace clustering. Appl Intell 50(8):2523–2536
20. Mi Y, Ren Z, Mukherjee M, Huang Y, Sun Q, Chen L (2021) Diversity and consistency embedding learning for multi-view subspace clustering. Appl Intell 51(10):6771–6784
21. Patel VM, Vidal R (2014) Kernel sparse subspace clustering. In: IEEE International conference on image processing (ICIP). IEEE, pp 2849–2853
22. Huang Q, Zhang Y, Peng H, Dan T, Weng W, Cai H (2020) Deep subspace clustering to achieve jointly latent feature extraction and discriminative learning. Neurocomputing 404:340–350
23. Zhang Y et al (2021) Deep multiview clustering via iteratively self-supervised universal and specific space learning. IEEE Trans Cybern (99):1–13
24. Ji P, Zhang T, Li H, Salzmann M, Reid I (2017) Deep subspace clustering networks. Advances in neural information processing systems, pp 24–33
25. Valanarasu JMJ, Patel VM (2021) Overcomplete deep subspace clustering networks. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp 746–755
26. Sun X, Cheng M, Min C, Jing L (2019) Self-supervised deep multi-view subspace clustering. In: Asian Conference on Machine Learning. PMLR, pp 1001–1016
27. Masci J, Meier U, Cireşan D, Schmidhuber J (2011) Stacked convolutional auto-encoders for hierarchical feature extraction. In: International conference on artificial neural networks. Springer, Berlin, pp 52–59
28. Ji P, Salzmann M, Li H (2014) Efficient dense subspace clustering. In: IEEE Winter Conference on Applications of Computer Vision. IEEE, pp 461–468
29. Georghiades AS, Belhumeur PN, Kriegman DJ (2001) From few to many: Illumination cone models for face recognition under variable lighting and pose. IEEE Trans Pattern Anal Mach Intell 23(6):643–660
30. Samaria FS, Harter AC (1994) Parameterisation of a stochastic model for human face identification. In: Proceedings of 1994 IEEE workshop on applications of computer vision. IEEE, pp 138–142
31. Nene SA (1996) Columbia object image library (coil-100). Technical Report 6
32. You C, Robinson D, Vidal R (2016) Scalable sparse subspace clustering by orthogonal matching pursuit. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3918–3927
33. Vidal R, Favaro P (2014) Low rank subspace clustering (LRSC). Pattern Recogn Lett 43:47–61
34. Baek S, Yoon G, Song J, Yoon SM (2021) Deep self-representative subspace clustering network. Pattern Recogn 118:108041
35. Lu H, Chen C, Wei H (2022) Improved deep convolutional embedded clustering with re-selectable sample training. Pattern Recognit 127:108611

Chao Chen received the B.E. degree from Huaiyin Institute of Technology, Jiangsu, China. He is currently working toward the M.Eng. degree with the School of Computer Science and Communication Engineering, Jiangsu University, Jiangsu, China. His research interests include unsupervised learning and deep learning.

Hu Lu received the Ph.D. degree from Fudan University in 2013. He was a postdoctoral fellow at the School of software, Fudan University from 2013 to 2015. He was a visiting scholar with the Department of Computer and Information Sciences, Temple University, USA in 2016. He is currently with the School of Computer Science and Communication Engineering, Jiangsu University, Jiangsu, China, where he serves as an Associate Professor. His main research interests include computer vision, machine learning, deep learning and brain cognitive function.

Hui Wei received the Ph.D. degree from the Department of Computer Science, Beijing University of Aeronautics and Astronautics, in 1998. From 1998 to 2000, He was a Post-Doctoral Fellow with the Department of Computer Science and the Institute of Artificial Intelligence, Zhejiang University. Since November 2000, he has been with the Department of Computer Science and Engineering, Fudan University, Shanghai, China. His current research interests include artificial intelligence and cognitive science.

Xia Geng received the Ph.D. degree in Computer Application Technology from JiangSu University, China, in 2012. She is currently an Associate Professor with the School of Computer Science and Communication Engineering, JiangSu University. Her research interests include person re-identification, computer vision, pattern recognition, and image processing.