1357

Self-Expressive Network-Based Subspace Clustering for Deep Embedding

Tingting Leng^a, Ling Zhao^a, Xiaolong Xiong^a, Peng Cheng^a and Jun Zhou^{a;*}

^aCollege of Computer and Information Science, Southwest University, China ORCiD ID: Jun Zhou https://orcid.org/0000-0003-4353-1621

Abstract. Existing deep subspace clustering algorithms are difficult to scale to large-scale data. There are two reasons: Firstly, the existing subspace clustering algorithms almost all need to find the self-expressive coefficient matrix whose size is proportional to the square of the data set size at once. Secondly, spectral clustering needs to solve the eigenvector of the affinity matrix. These two points make the computational complexity of clustering very high when the data scale is large. This paper proposes Self-Expressive Network-Based Deep Embedded Subspace Clustering (SE-DESC), a subspace clustering method that can be applied to large-scale singleview and multi-view data. Using the idea of siamese networks, we design a self-expressive network to calculate the self-expressive coefficient between two data points, reducing the parameter amount of the self-expressive model to a constant. It can effectively avoid computational complexity. Then, we use a deeply embedded network to learn an embedding for each data point to map the data into the spectral space, avoiding the high computational complexity of spectral clustering. Extensive experiments demonstrate that SE-DESC improves the clustering performance on large-scale data compared to state-of-the-art methods.

1 Introduction

With the development of information technology and multimedia, people are generating large amounts of data all the time in their daily lives. How to mine the valuable information contained in big data has become the focus of data mining. The data generated in real life is often characterized by high dimensions and large scale, which brings new challenges to data mining [2, 8].

The clustering [10] is an important task in the field of machine learning and an important part of data analysis and mining. It has a wide range of applications in the field of big data processing [9]. However, traditional clustering algorithms such as k-means [12] and Gaussian Mixture Models [19] are often ineffective when dealing with high-dimensional data. Therefore, subspace clustering, which performs better when dealing with high-dimensional data, has gradually received extensive attention.

Due to the great success of deep neural networks in recent years, many deep subspace clustering algorithms have been proposed [20, 16, 17]. One of the more popular is the auto-encoder-based deep subspace clustering algorithm, whose main idea is to use the autoencoder to map the input data into a potentially low-dimensional space. Then, we can use a general subspace clustering algorithm such as a sparse subspace clustering algorithm to obtain the clustering results [6]. The deep subspace clustering algorithms achieve good results on nonlinear subspace data. For example, Ji et al. proposed DSC-Net [14] whose basic idea is to add a fully connected layer in the middle of the auto-encoder to perform the self-expressive process. DASC network [34] improves the performance of the algorithm by introducing an adversarial network mechanism in DSC-Net. DSSC network [1] introduces a new encoder to learn the weights of the features and finally fuses the features weighted to perform the self-expression process.

However, existing auto-encoder-based subspace clustering algorithms have limitations in accuracy. Meanwhile, the presence of the subsequent spectral clustering and the fact that the number of parameters is proportional to the square of the data size resulted in the high computational complexity of the algorithm. This leads to the fact that these algorithms cannot be applied to large-scale data. In addition, because each view provides a separate dataset, large-scale multi-view data [22] also presents new challenges to subspace clustering [32].

In this paper, we propose a Self-Expressive Network-Based Deep Embedded Subspace Clustering (SE-DESC) for large-scale singleview and multi-view data. The network is mainly divided into two parts. The first part is a self-expressive network [31], which uses the idea of Siamese Networks to calculate the self-expressive coefficients between data points. It reduces the parameter amount of the selfexpression model to a constant. The second part is a deep embedded network, which aims to solve the problem of the high computational complexity of spectral clustering. Meanwhile, SE-DESC employs a soft assignment to obtain the final results and uses target distributions to facilitate training. We conducted experiments on six public largescale datasets including four single-view datasets and two multi-view datasets to demonstrate the effectiveness of the algorithm. Our contribution can be summarised as follows:

- We propose SE-DESC, a subspace clustering method for largescale single-view and multi-view data. The problem of not being able to scale the algorithm to large-scale data due to the high computational complexity of subspace clustering is addressed using deep networks.
- We use self-expressive networks to calculate the self-expressive coefficients between data points, reducing the number of parameters while giving the network the ability to generalize to unseen new samples.
- 3. We use a deep embedded network to replace the computational process of spectral clustering, eliminating the need to compute the

^{*} Corresponding Author. Email: zhouj@swu.edu.cn

eigenvectors of the affinity matrix and solving the problem of high computational effort in spectral clustering.

2 Related Work

2.1 Sparse subspace clustering

Sparse subspace clustering (SSC) [7, 23] is based on a self-expressive model that represents each data point as a linear combination of all other data points. Therefore, we can obtain a self-expressive coefficient matrix, which is solved by imposing a regularization term constraint on the coefficient matrix. By solving, a set of sparse solutions can be obtained. That is, each data point is represented only by those data points from the same subspace as it. According to the self-expression coefficient matrix obtained, the spectral clustering algorithm is carried out to obtain the final clustering result.

Let $X \in \mathbb{R}^{n \times d}$ be a data set with n data points, each with d dimensions. Then, the self-expressive model can be expressed as:

$$x_i = c_{i1}x_1 + c_{i2}x_2 + \dots + 0 \cdot x_i + \dots + c_{in}x_n = \sum_{j \neq i}^n c_{ij}x_j \quad (1)$$

where c_{ij} is the self-expressive coefficient between x_i and x_j .

When a data point belongs to the same subspace as other data points, the self-expression coefficient between each data point is nonzero and as dense as possible. Instead, their self-expression coefficients with other data points are zero and as sparse as possible. To achieve this, we add a regularization term constraint on the selfexpressive coefficients:

$$\min ||C||_p \quad \text{s.t. } X = XC, \quad \operatorname{diag}(C) = 0 \tag{2}$$

where C is self-expressive coefficient matrix and p is the regularization term. After deriving the self-expressive coefficient matrix, the affinity matrix W is constructed as follows:

$$W = \frac{|C| + \left|C^{T}\right|}{2} \tag{3}$$

Finally, a spectral clustering algorithm is performed based on the affinity matrix W to obtain the final clustering results.

2.2 Spectral Clustering

The main idea of spectral clustering[24, 18] is to consider each sample as a node in the graph, and the weights of edges between nodes as the similarity between samples. Then, the graph is cut into multiple subgraphs according to graph cut theory. The weights of edges between nodes within each subgraph should be as large as possible, while the weights of edges between nodes in different subgraphs should be as small as possible. Finally, each subgraph can be considered as a cluster.

 $X = \{x_1, x_2, x_3, \dots, x_n, \}$ denotes a data set with *n* data points, and w_{ij} represents the similarity between each sample:

$$w_{ij} = \begin{cases} 0 & others\\ \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) & x_i \in KNN(x_j) \end{cases}$$
(4)

The Laplacian matrix L = D - W can be constructed from W, where $D_{ii} = \sum_{i} W_{ij}$. Spectral clustering is to divide the

nodes in graph G into multiple subsets $A_1, A_2, A_3, \ldots, A_k$ satisfying $A_1 \cup A_2 \cup A_3 \cup \ldots \cup A_K = V$ and $A_i \cap A_j = \emptyset (i \neq j, i, j \in \{1, 2, 3, \ldots, K\})$. Define the *cut* function as:

$$\operatorname{cut}\left(A_{1}, A_{2}, \dots, A_{k}\right) = \frac{1}{2} \sum_{i=1}^{k} f\left(A_{i}, \overline{A_{i}}\right)$$
(5)

 $f(A, B) = \sum_{i \in A, j \in B} w_{ij}$ represents the sum of the weights of all connected edges between two subgraphs. In order to achieve the objective of having the largest possible edge weights between nodes in the same subset and the smallest possible edge weights between different subsets, the *cut* function is minimized. To avoid the influence of outlier points, the optimization scheme is *RatioCut* cut. Minimize the cut function and the number of nodes in each subgraph to avoid the situation where there are particularly few nodes in a particular subgraph. That is:

RatioCut
$$(A_1, A_2, ..., A_k) = \frac{1}{2} \sum_{i=1}^k \frac{f(A_i, \overline{A_i})}{|A_i|}$$
 (6)

 $|A_i|$ denotes the number of nodes of subgraph A_i . Since the above problem is an NP-Hard puzzle, define k indicator vectors $h_j = \{h_{1j}, h_{2j}, \ldots, h_{nj}\}, j = 1, 2, \ldots, k$, and the following conditions are satisfied:

$$h_{ij} = \begin{cases} 0 & v_i \notin A_j \\ \frac{1}{\sqrt{|A_j|}} & v_i \in A_j \end{cases}$$
(7)

When the *i*-th node does not belong to the *j*-th subgraph, the value of the corresponding position of the indicator vector is 0, otherwise, the value is $\frac{1}{./[A_{..}]}$.

According to the properties of the Laplacian matrix, there are:

$$h_i^T L h_i = \frac{1}{2} \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} w_{mn} (h_{im} - h_{in})^2$$
$$= \frac{\operatorname{cut} (A_i, \overline{A_i})}{|A_i|}$$
$$= \operatorname{RatioCut} (A_i, \overline{A_i})$$
(8)

At this point, h_i indicates that the indicator vectors are arranged in rows, and from the definition of the indicator vectors. We know that they are orthogonal to each other. It is as follows:

RatioCut
$$(A_1, A_2, ..., A_k) = \sum_{i=1}^k h_i^T L h_i$$

$$= \sum_{i=1}^k \left(H^T L H \right)_{ii}$$
$$= \operatorname{Tr} \left(H^T L H \right)$$
(9)

Then, the problem of solving the minimum value of RatioCut (A_1, A_2, \ldots, A_k) converts to

$$\underset{H}{\operatorname{argmin}}\operatorname{Tr}\left(H^{T}LH\right) \quad \text{s.t. } H^{T}H = I \qquad (10)$$

The maximum value of $h_i^T L h_i$ is the maximum eigenvalue of the matrix L and the minimum value is the minimum eigenvalue of the

matrix L. Thus, we convert the problem of cutting the graph into k groupings into finding the eigenvectors h_1, h_2, \ldots, h_k corresponding to the first k smallest eigenvalues of the graph Laplacian matrix L and finding the quadratic sum of L (the cut consumes the least energy), i.e., $\sum_{i=1}^{k} h_i^T L h_i$. The matrix $H = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k]$ is then treated as a new dataset with k dimensional features n samples for k-means clustering. We can see that it is clustered for each sample and the clustering of the number of categories is k, i.e., clustering for each row of H (assumed to be denoted as $h_{ri}, i = 1, 2, \ldots, n$).

3 Method

As shown in Figure 1, the SE-DESC network consists of SENet and DESC. The first part is a self-expressive network based on SENet [31]. It is mainly used to solve for self-expressive coefficients w_{ij} between data points x_i and x_j . The second part is a multilayer perceptron network with an orthogonal layer for mapping data points into an embedding space. Finally the probability of the data points belonging to each cluster is obtained through the fully connected layer and softmax layer.

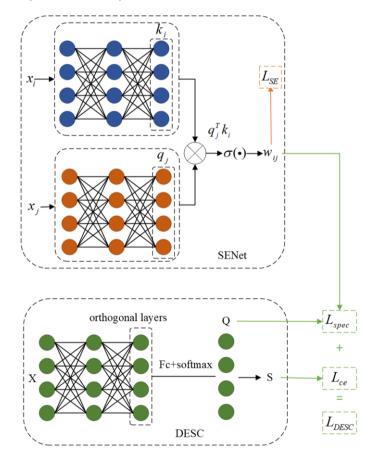


Figure 1: SE-DESC network structure. The network have consisted of SENet and DESC. SENet is mainly used to solve for self-expressive coefficients w_{ij} between data points x_i and x_j . DESC is a multilayer perceptron network with orthogonal layers for mapping data points into an embedding space.

The training of the whole network is also divided into two steps: the first step is to train the self-expressive network based on the loss L_{SE} , and the second step is to train the DESC network according to according to the loss L_{DESC} , which consists of two parts. L_{spec} is computed by embedding space and self-expressive network output to obtain the orthogonal embedding representation. L_{ce} is the crossentropy loss of the soft assignment and target distributions used to update the soft assignment, i.e. the clustering result. When the training is completed, the data to be clustered can be input into the network to get the clusters to which the data points belong. It can be found that SE-DESC has the ability to generalize to unseen new samples. The following subsections will develop a detailed description of the SE-DESC method.

3.1 Self-expressive network

The self-expressive network is based on the idea of siamese network and consists of two independent branches of a multilayer perceptron with a residual structure which is called the query network and the key network. The residual structure is designed to make network training easier. Unlike siamese network, the two branches do not share parameters and do not need to define positive and negative samples, and the loss functions are different. As shown in the left half of Figure 1, the two data points on the input side are passed through the query and key networks to obtain their respective embedding representations. Then, the two output vectors are inner-produced to obtain an output value. Finally, an activation function is used to obtain the final output, which is the self-expressive coefficient between the data points. The activation function uses a soft threshold activation function, which can filter out the smaller coefficients to achieve a sparse effect and the threshold is learnable. The loss function for the selfexpressive network training process is as follows:

$$L_{SE} = \sum_{j=1}^{n} l(x_j, X; \Theta)$$
(11)

 $l(x_j, X; \Theta)$ is the sum of the self-expressive loss and the coefficient regularization loss, specifically:

$$l(x_j, X; \Theta) = \frac{\gamma}{2} \|x_j - \sum_{i \neq j} f(x_i, x_j; \Theta) x_i\|_2^2 + \sum_{i \neq j} r(f(x_i, x_j; \Theta))$$
(12)

where the self-expressive loss expects any data point to be represented as a linear combination of other data points from the same subspace. $f(\cdot)$ denotes the self-expressive network mapping function, while Θ is a parameter of the network. Specifically:

$$f(x_i, x_j; \Theta) = \sigma(q_j^T k_i)$$
(13)

Where $\sigma(\cdot)$ denotes the soft threshold activation function that filters the smaller coefficients and retains only the higher selfexpressive coefficients to achieve sparsity. q_j denotes the embedding representation obtained by x_j after the query network and k_i denotes the embedding representation obtained by x_i after the key network.

 $r(\cdot)$ is a regular term added to the self-expression coefficients. To alleviate the problem of over-segmentation caused by the L1 norm that splits data points belonging to the same subspace, we choose the elastic net regularization proposed in [28]. It is a weighted sum of the ℓ_1 and ℓ_2 norm, where λ is a balancing parameter.

$$r(\cdot) = \lambda \|\cdot\| + \frac{1-\lambda}{2} (\cdot)^2 \tag{14}$$

After the training is completed, the network parameters are fixed and the data is input for self-expressive coefficients. The data points to be input are entered into the two branches of the network and the output is the self-expressive coefficients between them:

$$w_{ij} = f(x_i, x_j; \Theta) \tag{15}$$

To be consistent with SSC[7], w_{ij} and w_{ji} are not necessarily equal, we make the query network and the key network did not share parameters.

For large-scale data, only a fraction of it is selected for training, so the computational effort is greatly reduced. The existing deep subspace clustering algorithms feed the entire dataset directly into the model for self-expressive. The number of parameters is proportional to the square of the data size. Here, the number of parameters in the self-expressive network is a constant, avoiding the $O(n^2)$ computational complexity. At the time of inference, the self-expressive network can be extended to the entire dataset as the data distribution is consistent.

3.2 Deep Embedded Subspace Clustering

The deep embedded network component consists of a multilayer perceptron with orthogonal layers designed to map the input data into the embedding space, and the objective function[21] of this component is:

$$L_{spec} = \frac{1}{n^2} \sum_{i,j=1}^{n} w_{ij} \parallel p_i - p_j \parallel^2$$
(16)

Where p_i and p_j are the embedding representations obtained after data x_i and x_j have passed through the DESC network. w_{ij} is a self-expressive coefficient matrix. By minimizing L_{spec} , the lowerdimensional representations of data points with greater similarity (i.e., greater w_{ij}) are made more similar, and the output is forced to be orthogonalized in order to avoid mapping all sample points into the same embedding space. Instead of directly using the output of the self-expressive network as w_{ij} , we use the idea from the DSESC algorithm [33]. When the data input to the self-expressive network is a Batch, the self-expressive coefficient matrix is regarded as a representation of the sample. The KNN algorithm is used to construct the affinity matrix based on the self-expressive coefficients:

$$w_{ij} = \begin{cases} 1 & ifc_i \in KNN(c_j) \\ 0 & others \end{cases}$$
(17)

Obviously, the size of the input Batch plays an important role and the experimental section will analyze the value of Batch size.

To avoid DESC from mapping all data points to the same point in the embedding space and getting nonsensical solutions, we set the last layer of the DESC network as an orthogonal layer can make the output data points orthogonalized. According to [21], it can be achieved by QR factorization. The square root method (Cholesky decomposition) is one way to achieve QR factorization, which can represent a symmetric positive definite matrix as a decomposition of the product of a lower triangular matrix L and its transpose. Specifically, let \tilde{Y} be the input of the orthogonal layer, then the square root method can be expressed as:

$$\tilde{Y}^T \tilde{Y} = L L^T \tag{18}$$

L is the orthogonal matrix, setting $\sqrt{m} (L^{-1})^T$ as the weight of the orthogonal layer can make the output vector orthogonalized. m

is the Batch size of the input data. Here, Batch size will have some influence. We will analyze the appropriate value of Batch size in our experiments.

Furthermore, to avoid post-processing, we add a set of classification layers after the orthogonal layers. The final output is the probability that a data point belongs to each cluster, which we call a soft assignment due to the lack of accuracy of this classification. Let q_{ij} denote the probability of assigning sample i to cluster j. Then we introduce a target distribution to iteratively optimise by learning from high-confidence assignments. Let s_{ij} be the target distribution corresponding to q_{ij} , then:

$$s_{ij} = \frac{\frac{q_{ij}^2}{\sum_i q_{ij}}}{\sum_j \left(\frac{q_{ij}^2}{\sum_i q_{ij}}\right)} \tag{19}$$

The soft assignment, i.e. the clustering result, is updated by minimising the cross-entropy loss between the soft assignment and the target distribution. Then the loss function for this part is:

$$L_{ce} = \text{CrossEntropy}(\boldsymbol{S} \| \boldsymbol{Q}) = -\sum_{i} q_{i} \log (s_{i})$$
(20)

This strengthens the prediction by learning from the high confidence of soft assignment. Then the loss function of the entire DESC network is:

$$L_{DESC} = L_{spec} + \beta L_{ce} \tag{21}$$

Where β is a hyperparameter to control the proportion of the loss function. The network also uses the Adam optimizer to optimize the final training loss L_{DESC} .

After the final network is trained, the data to be clustered is input to get the probability of belonging to each cluster. The cluster that corresponds to the largest of these values is the cluster to which the data point belongs. The SE-DESC network training is divided into two steps, the first step is to train the self-expressive network. Its training process is shown in Algorithm 1. The second step is to train the DESC network, and its training process is shown in Algorithm 2. Since there is no need to solve all the self-expressive coefficients at once and no need to perform the operation of solving the eigenvector, the computational complexity of the algorithm is greatly reduced and can be extended to large-scale data sets.

Input: training data X, hyperparameters γ , λ , maximum number of iterations T, t = 0

1: repeat

- 2: Randomly sample a data point x_j
- 3: x_i is passed through the query network to obtain q_i
- 4: For x_i in X and $j \neq i$:
- 5: x_i is passed through the key network to get k_i
- 6: Calculate the self-expressive coefficient $c_{ij} = q_j^T k_i$
- 7: Calculate the value of the loss function according to Eq.12

8: Use Adam optimizer for back propagation

- 9: t = t + 1
- 10: **until** $t \ge T$ or converge

Output: Trained self-expressive network

Algorithm 2 SE-DESC network training flow

Input: training data X, hyperparameter β , Batch size m, maximum number of iterations T, number of clustering k

- 1: The self-expressive network is trained according to Algorithm 1
- 2: repeat
- 3: Random sampling m samples form a Batch data input to DESC to get the output of the orthogonal layer \tilde{Y}
- 4: Calculate the cholesky decomposition: $LL^{T} = \tilde{Y}^{T}\tilde{Y};$
- 5: Set the weight of DESC network orthogonal layer as: $\sqrt{m}(L^{-1})^T$; Randomly sampling m samples to form a Batch is input to both the query and key networks and the DESC network to obtain the self-expressive coefficient matrix and the embedding representation
- 6: Construct the similarity matrix according to Eq.17
- 7: Calculate the loss function according to Eq.21
- 8: Update the network parameters using the backpropagation algorithm
- 9: t = t + 1
- 10: **until** $t \ge T$ or converge

4 Experiments

4.1 Datasets

In order to verify the effectiveness of the SE-DESC algorithm, we select six public datasets of a relatively large scale for our experiments. It includes four single-view image datasets of MNIST, FashionMNIST, CIFAR10, and CIFAR100 and two multi-view datasets of Caltech101 and NUSWIDEOBJ. The statistical information of the experimental dataset is shown in Table 1.

- MNIST: handwritten digital grayscale images, including numbers 0-9, 60,000 training data, and 10,000 test data, totaling 70,000 images. Its size is 28×28.
- FashionMNIST: Grayscale images of fashion products, including 10 products with different styles such as jackets, pants, shirts, dresses, bags, and so on. It is divided into training set and test set with a total of 70,000 images, and the size is also 28×28.
- CIFAR10/100: natural color images. Cifar10 has 10 classes with 6000 images per class. Cifar100 has 100 classes with 600 images per class. Compared to CIFAR10, CIFAR100 is more challenging due to a large number of categories.
- Caltech101: An image dataset created by the California Institute of Technology with 101 classes, each image is represented by six feature sets: Gabor, Wavelet Moments, Centrist, HOG, GIST, and LBP. a subset of which is selected for this experiment, with 2386 images and containing 20 classes.
- NUSWIDEOBJ: A web image dataset created by the Media Search Lab at the National University of Singapore, with 81 classes including 5 views. A total of 30,000 samples from 31 of these classes are selected here for the experiments.

For MNIST and FashionMNIST, we use convolution networks [3] to extract 3472-dimensional features and use the PCA algorithm to reduce to 500 dimensions. For CIFAR10 and CIFAR100, 128-dimensional features are extracted using the method of MCR^2 proposed in [30]. MCR^2 extracts the features of an image by data enhancement techniques when unlabeled, and its extracted features have the property of subspace distribution. All feature vectors are normalized using the ℓ_2 norm. In the following subsections, we present the experimental results.

Table 1: Experimental datasets statistics.

Datasets	n	size	clusters
MNIST	70000	28×28	10
FashionMNIST	70000	28×28	10
CIFAR10	60000	32×32×3	10
CIFAR100	60000	32×32×3	100
Caltech101	2386	48,40,254,1984,512,928	20
NUSWIDEOBJ	30000	65,226,145,74,129	31

4.2 Experimental results

4.2.1 Contrast Method

Since the auto-encoder-based subspace clustering cannot be extended to large-scale datasets, several traditional clustering methods and several deep clustering methods are selected for the comparison method. It includes two traditional clustering algorithms, k-means, and spectral clustering, EnSC [28] and SSC-OMP [29], two scalable sparse subspace clustering algorithms, and DEC [26], IDEC [11], VaDE [15], JULE [27], DAC [5], DCCM [25], PICA [13], SENet [31], and EDESC [4] nine deep clustering methods, where SENet and EDESC are deep subspace clustering algorithms. Here, we introduce nine deep clustering methods.

- DEC [26]: the most classical deep clustering algorithm. It learns the embedding representation of the samples by auto-encoder and learns a set of centroids to compute a soft assignment. It uses the KL divergence of the soft distribution and the target distribution as the clustering loss to optimize the clustering centroids.
- IDEC [11]: an improved deep embedding clustering algorithm. It builds on DEC to optimize the clustering label assignment feature by integrating the clustering loss and the auto-encoder reconstruction loss.
- VaDE [15]: a VAE-based deep clustering method that models the process of data generation through Gaussian mixture models and deep neural networks. Optimization is performed using stochastic gradient variational inference Bayesian and reparameterization techniques.
- JULE [27]: a deep clustering algorithm based on a specific clustering loss. It uses the agglomerative clustering results of the images as supervised signals to learn the embedding representations which in turn will benefit the image clustering. This is iterated until convergence.
- DAC [5]: a single-stage convolutional network-based image clustering method. It treats pairs of images as a binary classification problem, i.e., two images belong to one class or not. The cosine distance between the image label features is used as the similarity so that the learned label features tend to be one-hot vectors. An alternating iterative adaptive algorithm is also used to optimize the model.
- DCCM [25]: a deep integrated relevance mining clustering algorithm. Three aspects of pseudo-supervision, data augmentation, and triple mutual information are used to explore the various correlations behind unlabeled data.
- PICA [13]: a divisional confidence maximization clustering algorithm. Learn the semantically most reasonable data partitioning by maximizing the global partitioning confidence of the clustering method, which enables all clusters to be mapped one-to-one to the true class. It is implemented by introducing microdividable uncertainty metrics and their stochastic approximations and proposing

an objective loss function based on minimizing uncertainty metrics.

- SENet [31]: drawing on the idea of Siamese networks to calculate the self-expressive coefficients between data points, training this network only requires sampling part of the data. Thus, it avoids the higher computational complexity.
- EDESC [4]: This algorithm learns a set of vector bases for each subspace, satisfying that each data point only has a larger inner product with the vector bases belonging to that subspace and a smaller inner product with the vector bases of other subspaces. Based on this, the subspace to which each data point belongs can be found.

4.2.2 Experimental setup

The self-expressive network is composed of three fully-connected layers with residual structure and dimensions of 1024-1024-1024. The deep embedded network uses three fully-connected layers with dimensions of 1024-512-512, and the activation function also adopts ReLU.Since the large-scale training task is more demanding on the hardware of the machine, we use an RTX 3080 graphics card to accelerate the training of the network. The hyperparameter λ is set to 0.9 and γ is set to 200 for the self-expressive network, and the hyperparameter β is set to 200 for the deep embedded network (the optimal parameter value is not fixed for each dataset, so we need to manually adjust the parameter value size in order to find the optimal parameter for each dataset), and the learning rate is set to 0.0001. The random sampling size is set to 0.0001 for MNIST, FashionM-NIST, and CIFAR10, and the random sampling size is set to 0.0001 for MNIST, FashionMNIST, and CIFAR10. For MNIST, Fashion-MNIST and CIFAR10, 5000 samples are randomly sampled as the training set, 2000 epochs are trained for the self-expressive network, and 1000 epochs are trained for the deep embedded network. for the CIFAR100 dataset, due to the large number of categories, 10,000 samples are randomly sampled as the training set. The evaluation metrics are also selected as ACC and NMI values, meanwhile, since the size of the training set has a direct impact on the training process as well as the training results, we also analysed the training time of the networks under different sizes of training sets.

4.2.3 Analysis of experimental results

Firstly, we conducted a comparison experiment, and the results are shown in Table 2 (- indicates that the code of the method is not opensource and there is no corresponding data in the paper). The experimental results in Table 2 show that: on the MNIST dataset, the accuracy of SE-DESC is only more than 60%, which is a big gap compared with the comparison algorithms, and we will analyse the reason for this in our future work; on the FashionMNIST dataset, SE-DESC has a big gap compared with SENet but is still competitive compared with most other comparison algorithms; on the CIFAR10 dataset, SE-DESC has a big gap compared with SENet but is still competitive compared with most other comparison algorithms. On the CI-FAR10 dataset, SE-DESC achieves good results, second only to the SENet algorithm; on the CIFAR100 dataset, one possible reason for the poor results of SE-DESC is that this dataset has a large number of categories, and SE-DESC does not work well with datasets that have a large number of categories. However, although the experiments on the CIFAR100 dataset did not achieve the expected results, the results are still competitive. In summary, the SE-DESC algorithm achieved some results on the four large-scale single-view datasets

and achieved scalability compared to other selfencoder-based subspace clustering algorithms. But the results were not good on the MNIST dataset, which needs to be further analysed. In addition, the SE-DESC algorithm is not effective when encountering datasets with many categories such as CIFAR100, so how to deal with highdimensional datasets with a large number of categories will be a valuable research direction in the future.

Table 2: Compare experimental results on a single-view dataset.

Methods	MNIST		FashionMNIST		CIFAR10		CIFAR100	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
k-means	0.541	0.507	0.505	0.578	0.525	0.589	0.130	0.084
SC	0.728	0.856	0.625	0.700	0.455	0.574	0.136	0.090
EnSC	0.980	0.945	0.672	0.705	0.613	0.601	0.347	0.362
SSC-OMP	0.928	0.842	0.274	0.421	0.326	0.498	0.051	0.209
DEC	0.865	0.837	0.518	0.546	0.301	0.257	0.185	0.136
IDEC	0.881	0.867	0.592	0.604	0.316	0.273	0.191	0.140
VaDE	-	-	0.578	0.630	0.156	0.036	_	_
JULE	0.964	0.913	0.563	0.608	0.272	0.192	0.137	0.103
DAC	0.978	0.935	0.615	0.632	0.522	0.396	0.238	0.185
DCCM	-	-	-	-	0.623	0.496	0.327	0.285
PICA	_	-	-	-	0.696	0.591	0.337	0.310
SENet	0.968	0.918	0.697	0.663	0.765	0.655	0.280	0.423
EDESC	0.913	0.862	0.631	0.670	0.627	0.464	0.385	0.370
SE-DESC	0.589	0.590	0.576	0.617	0.710	0.626	0.286	0.426

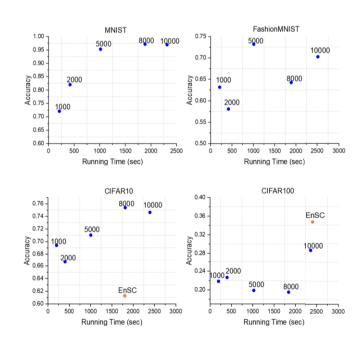


Figure 2: Training time for different sizes of training datasets.

Next, we conducted an experimental analysis on the impact of Batch size on the clustering results. The size starts from 100 and increases to 700 at an interval of 100, and observes the change of the ACC value. The experimental results are shown in Table 3. The values in the table represent the ACC values of each data set. It can be observed that as the Batch size increases from 100 to 700, the clustering accuracy on all datasets fluctuates. We can see an overall trend of increasing and then decreasing, while the best results are achieved

Table 3: Experimental results of Batch size analysis

Batch Size	MNIST	FashionMIST	CIFAR10	CIFAR100
100	0.4881	0.4375	0.4603	0.1565
200	0.5079	0.4786	0.5954	0.2163
300	0.5103	0.4914	0.6267	0.2538
400	0.5389	0.4870	0.6630	0.2640
500	0.5624	0.5651	0.7102	0.2858
600	0.4807	0.5762	0.6644	0.2703
700	0.5889	0.5169	0.6856	0.2731

on all datasets when the value is between 500 and 700. Therefore, we set 500 as the best Batch size value for training. We can observe the overall trend of increasing and then decreasing on most of the datasets in Figure 3. This also indicates that 500 is a relatively suitable Batch size value.

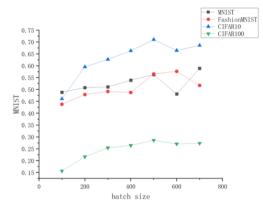


Figure 3: Batch size analysis curve.

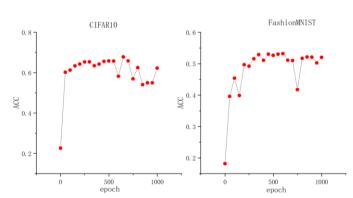


Figure 4: Online clustering plots of the FashionMNIST dataset and the CI-FAR10 dataset.

Then, when the input is batch, we compare the clustering effect of using the self-expressive coefficients directly as the affinity matrix with that of using the KNN algorithm to obtain the affinity matrix after the self-expressive coefficients. The experimental results (ACC values) are shown in Table 4.

It can be found from Table 4 that the KNN algorithm is still a relatively obvious improvement for constructing the affinity matrix for clustering on most data sets. For MNIST, using the KNN algorithm to construct the affinity matrix resulted in an improvement of around 13% for the clustering ACC metric, while for FashionMNIST it improved by around 7%, which verifies the effectiveness of it. One

possible reason for our analysis is that each sample only considers the similarity between its k nearest neighbors, allowing the network to learn the target more precisely, thus improving the performance of the algorithm.

Table 4: Results of KNN module ablation experiment (ACC).

	MNIST	FashionMNIST	CIFAR10	CIFAR100
non-KNN	0.4527	0.5074	0.6444	0.2821
KNN	0.5889	0.5762	0.7102	0.2858

Finally, we conducted online clustering experiments on the FashionMNIST and CIFAR10 datasets. We observed the online clustering effect of the algorithm by recording the changes in ACC values during the training process. The results of which are shown in Figure 4 (the left figure shows the experimental results of the FashionM-NIST dataset, and the right figure shows the experimental results of the CIFAR10 dataset). Figure 4 shows that the accuracy of the SE-DESC algorithm can steadily increase during the training process and gradually reach the convergence state. To verify the performance on multi-view large-scale data, we conducted experiments on a largescale multi-view dataset. As shown in Table 5, the SE-DESC algorithm is closer to the best result on the NUSWIDEOBJ dataset. On the Caltech101 dataset, SE-DESC was less effective compared to the comparison algorithm, a possible reason for this is that Caltech101 has a large number of categories and SE-DESC does not cluster well on data with a large number of categories.

Table 5: Experimental results on multi-view datasets.

Methods	NUSWI	IDEOBJ	Caltech101		
	ACC	NMI	ACC	NMI	
LMVSC	0.1553	0.1295	0.1449	0.3332	
MGGL	0.1204	0.0573	0.1412	0.2612	
SMVSC	0.1916	0.1217	0.2750	0.3510	
EOMSC-CA	0.1968	0.1327	0.2232	0.2470	
SE-DESC	0.1933	0.0857	0.1707	0.1214	

5 Conclusion

We propose a deep embedding subspace clustering algorithm based on self-expressive networks, which focuses on the problem of subspace clustering on large-scale data. The network consists of a selfexpressive network to solve for the self-expressive coefficients between two data points and a deep embedded network to map samples to the embedding space instead of spectral clustering. SE-DESC achieves good results on four common large-scale single-view datasets. However, the performance improvement of SE-DESC is not obvious on datasets with many categories and large-scale multi-view datasets. It will be a future research direction to design a better algorithm for large-scale multi-view datasets.

Acknowledgements

This work was financially supported by the National Natural Science Foundation of China (NSFC, No. 22274134) and the Natural Science Foundation of Chongqing, China (cstc2021jcyj-msxmX0066).

References

- Sangwon Baek, Gangjoon Yoon, Jinjoo Song, and Sang Min Yoon, 'Deep self-representative subspace clustering network', *Pattern Recognition*, **118**, 108041, (2021).
- [2] Fatemeh Behrad and Mohammad Saniee Abadeh, 'An overview of deep learning methods for multimodal medical data mining', *Expert Systems with Applications*, 117006, (2022).
- [3] Joan Bruna and Stéphane Mallat, 'Invariant scattering convolution networks', *IEEE transactions on pattern analysis and machine intelli*gence, 35(8), 1872–1886, (2013).
- [4] Jinyu Cai, Jicong Fan, Wenzhong Guo, Shiping Wang, Yunhe Zhang, and Zhao Zhang, 'Efficient deep embedded subspace clustering', in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1–10, (2022).
- [5] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan, 'Deep adaptive image clustering', in *Proceedings of the IEEE international conference on computer vision*, pp. 5879–5887, (2017).
- [6] Chao Chen, Hu Lu, Hui Wei, and Xia Geng, 'Deep subspace image clustering network with self-expression and self-supervision', *Appl. Intell.*, 53(4), 4859–4873, (2023).
- [7] Ehsan Elhamifar and René Vidal, 'Sparse subspace clustering: Algorithm, theory, and applications', *IEEE transactions on pattern analysis* and machine intelligence, **35**(11), 2765–2781, (2013).
- [8] Aurora Esteban, Amelia Zafra, and Sebastian Ventura, 'Data mining in predictive maintenance systems: A taxonomy and systematic review', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discov*ery, **12**(5), e1471, (2022).
- [9] D Ezhilmaran and D Vinoth Indira, 'A survey on clustering techniques in pattern recognition', in *AIP Conference Proceedings*, volume 2261, p. 030093. AIP Publishing LLC, (2020).
- [10] Attri Ghosal, Arunima Nandy, Amit Kumar Das, Saptarsi Goswami, and Mrityunjoy Panday, 'A short review on different clustering techniques and their applications', *Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph 2018*, 69–83, (2020).
- [11] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin, 'Improved deep embedded clustering with local structure preservation.', in *Ijcai*, pp. 1753–1759, (2017).
- [12] John A Hartigan and Manchek A Wong, 'Algorithm as 136: A k-means clustering algorithm', *Journal of the royal statistical society. series c* (applied statistics), 28(1), 100–108, (1979).
- [13] Jiabo Huang, Shaogang Gong, and Xiatian Zhu, 'Deep semantic clustering by partition confidence maximisation', in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8849–8858, (2020).
- [14] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian Reid, 'Deep subspace clustering networks', *Advances in neural information processing systems*, **30**, (2017).
- [15] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou, 'Variational deep embedding: An unsupervised and generative approach to clustering', *arXiv preprint arXiv:1611.05148*, (2016).
- [16] Changsheng Li, Chen Yang, Bo Liu, Ye Yuan, and Guoren Wang, 'Lrsc: Learning representations for subspace clustering', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8340–8348, (2021).
- [17] Juncheng Lv, Zhao Kang, Xiao Lu, and Zenglin Xu, 'Pseudosupervised deep subspace clustering', *IEEE Transactions on Image Processing*, **30**, 5252–5263, (2021).
- [18] Andrew Ng, Michael Jordan, and Yair Weiss, 'On spectral clustering: Analysis and an algorithm', *Advances in neural information processing* systems, 14, (2001).
- [19] Ming Ouyang, William J Welsh, and Panos Georgopoulos, 'Gaussian mixture clustering and imputation of microarray data', *Bioinformatics*, 20(6), 917–923, (2004).
- [20] Zhihao Peng, Hui Liu, Yuheng Jia, and Junhui Hou, 'Adaptive attribute and structure subspace clustering network', *IEEE Transactions on Image Processing*, **31**, 3430–3439, (2022).
- [21] Uri Shaham, Kelly Stanton, Henry Li, Boaz Nadler, Ronen Basri, and Yuval Kluger, 'Spectralnet: Spectral clustering using deep neural networks', arXiv preprint arXiv:1801.01587, (2018).
- [22] Shiliang Sun, 'A survey of multi-view machine learning', Neural computing and applications, 23, 2031–2038, (2013).
- [23] Ehsan Elhamifar René Vidal, 'Sparse subspace clustering', in 2009

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol, pp. 2790–2797, (2009).

- [24] Ulrike Von Luxburg, 'A tutorial on spectral clustering', *Statistics and computing*, 17, 395–416, (2007).
- [25] Jianlong Wu, Keyu Long, Fei Wang, Chen Qian, Cheng Li, Zhouchen Lin, and Hongbin Zha, 'Deep comprehensive correlation mining for image clustering', in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8150–8159, (2019).
- [26] Junyuan Xie, Ross Girshick, and Ali Farhadi, 'Unsupervised deep embedding for clustering analysis', in *International conference on machine learning*, pp. 478–487. PMLR, (2016).
- [27] Jianwei Yang, Devi Parikh, and Dhruv Batra, 'Joint unsupervised learning of deep representations and image clusters', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5147–5156, (2016).
- [28] Chong You, Chun-Guang Li, Daniel P Robinson, and René Vidal, '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, (2016).
- [29] Chong You, Daniel Robinson, and René Vidal, 'Scalable sparse subspace clustering by orthogonal matching pursuit', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3918–3927, (2016).
- [30] Yaodong Yu, Kwan Ho Ryan Chan, Chong You, Chaobing Song, and Yi Ma, 'Learning diverse and discriminative representations via the principle of maximal coding rate reduction', *Advances in Neural Information Processing Systems*, 33, 9422–9434, (2020).
- [31] Shangzhi Zhang, Chong You, René Vidal, and Chun-Guang Li, 'Learning a self-expressive network for subspace clustering', in *Proceedings* of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12393–12403, (2021).
- [32] Ling Zhao, Yunpeng Ma, Shanxiong Chen, and Jun Zhou, 'Multi-view co-clustering with multi-similarity', *Applied Intelligence*, 1–12, (2022).
- [33] Ling Zhao, Yunpeng Ma, Shanxiong Chen, and Jun Zhou, 'Deep double self-expressive subspace clustering', in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, (2023).
- [34] Pan Zhou, Yunqing Hou, and Jiashi Feng, 'Deep adversarial subspace clustering', in *Proceedings of the IEEE conference on computer vision* and pattern recognition, pp. 1596–1604, (2018).