This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Self-Supervised Convolutional Subspace Clustering Network

Junjian Zhang[†], Chun-Guang Li[†], Chong You[‡], Xianbiao Qi[♯],
Honggang Zhang[†], Jun Guo[†], and Zhouchen Lin[§]
[†] SICE, Beijing University of Posts and Telecommunications
[‡] EECS, University of California, Berkeley [♯] Shenzhen Research Institute of Big Data
[§] Key Laboratory of Machine Perception (MOE), School of EECS, Peking University

Abstract

Subspace clustering methods based on data self-expression have become very popular for learning from data that lie in a union of low-dimensional linear subspaces. However, the applicability of subspace clustering has been limited because practical visual data in raw form do not necessarily lie in such linear subspaces. On the other hand, while Convolutional Neural Network (ConvNet) has been demonstrated to be a powerful tool for extracting discriminative features from visual data, training such a ConvNet usually requires a large amount of labeled data, which are unavailable in subspace clustering applications. To achieve simultaneous feature learning and subspace clustering, we propose an end-to-end trainable framework, called Self-Supervised Convolutional Subspace Clustering Network (S^{2}ConvSCN), that combines a ConvNet module (for feature learning), a self-expression module (for subspace clustering) and a spectral clustering module (for self-supervision) into a joint optimization framework. Particularly, we introduce a dual self-supervision that exploits the output of spectral clustering to supervise the training of the feature learning module (via a classification loss) and the self-expression module (via a spectral clustering loss). Our experiments on four benchmark datasets show the effectiveness of the dual self-supervision and demonstrate superior performance of our proposed approach.

1. Introduction

In many real-world applications such as image and video processing, we need to deal with a large amount of high-dimensional data. Such data can often be well approximated by a union of multiple low-dimensional subspaces, where each subspace corresponds to a class or a category. For example, the frontal facial images of a subject taken under varying lighting conditions approximately span a linear subspace of dimension up to nine [11]; the trajectories of feature points related to a rigidly moving object in a video sequence span an affine subspace of dimension up to three [41]; the set of handwritten digit images of a single digit also approximately span a low-dimensional subspace [8]. In such cases, it is important to segment the data into multiple groups where each group contains data points from the same subspace. This problem is known as *subspace clustering* [42, 45], which we formally define as follows.

Problem (Subspace Clustering). *Let $X \in \mathbb{R}^{D\times N}$ be a real-valued matrix whose columns are drawn from a union of n subspaces of $\mathbb{R}^D$, $\bigcup_{i=1}^{n}\{S_i\}$, of dimensions $d_i \ll \min\{D, N\}$, for $i = 1, \ldots, n$. The goal of subspace clustering is to segment the columns of X into their corresponding subspaces.*

In the past decade, subspace clustering has become an important topic in unsupervised learning and many subspace clustering algorithms have been developed [2, 23, 4, 27, 22, 26, 5, 17, 37, 53, 51, 18]. These methods have been successfully applied to various applications such as motion segmentation [44, 40], face image clustering [3], genes expression microarray clustering [28, 20] and so on.

Despite the great success in the recent development of subspace clustering, its applicability to real applications is very limited because practical data do not necessarily conform with the linear subspace model. In face image clustering, for example, practical face images are often not aligned and often contain variations in pose and expression of the subject. Subspace clustering cannot handle such cases as images corresponding to the same face no longer lie in linear subspaces. While there are recently developed techniques for joint image alignment and subspace clustering [21], such a parameterized model is incapable of handling a broader range of data variations such as deformation, translation and so on. It is also possible to use manually designed invariance features such as SIFT [25], HOG [1] and PRICoLBP [38] of the images before performing subspace clustering, e.g., in [36, 35]. However, there has been neither theoretical nor practical evidence to show that such features follow the linear subspace model.

Recently, Convolutional Neural Networks (ConvNets) have demonstrated superior ability in learning useful image representations in a wide range of tasks such as face/object classification and detection [15, 31]. In particular, it is shown in [16] that when applied to images of different classes, ConvNets are able to learn features that lie in a union of linear subspaces. The challenge for training such a ConvNet, however, is that it requires a large number of labeled training images which is often unavailable in practical applications.

In order to train ConvNet for feature learning without labeled data, many methods have been recently proposed by exploiting the self-expression of data in a union of subspaces [36, 14, 35, 54]. Specifically, these methods supervise the training of ConvNet by inducing the learned features to be such that each feature vector can be expressed as a linear combination of the other feature vectors. However, it is difficult to learn good feature representations in such an approach due to the lack of effective supervision.

Paper contribution. In this paper, we develop an end-to-end trainable framework for simultaneous feature learning and subspace clustering, called Self-Supervised Convolutional Subspace Clustering Network (S^{2}ConvSCN). In this framework, we use the current clustering results to self-supervise the training of feature learning and self-expression modules, which is able to significantly improve the subspace clustering performance. In particular, we introduce the following two self-supervision modules:

1. We introduce a spectral clustering module which uses the current clustering results to supervise the learning of the self-expression coefficients. This is achieved by inducing the affinity generated from the self-expression to form a segmentation of the data that aligns with the current class labels generated from clustering.

2. We introduce a classification module which uses the current clustering results to supervise the training of feature learning. This is achieved by minimizing the classification loss between the output of a classifier trained on top of the feature learning module and the current class labels generated from clustering.

We propose a training framework where the feature representation, the data self-expression and the data segmentation are jointly learned and alternately refined in the learning procedure. Conceptually, the initial clustering results do not align exactly with the true data segmentation, therefore the initial self-supervision incurs errors to the training. Nonetheless, the feature learning is still expected to benefit from such self-supervision as there are data with correct labels that produce useful information. An improved feature representation subsequently helps to learn a better self-expression and consequently produce a better data segmentation (i.e., with less wrong labels). Our experiments on four benchmark datasets demonstrate superior performance of the proposed approach.

2. Related Work

In this section, we review the relevant prior work in subspace clustering. For clarity, we group them into two categories: a) subspace clustering in original space; and b) subspace clustering in feature space.

2.1. Subspace Clustering in Original Space

In the past years, subspace clustering has received a lot of attention and many methods have been developed. Among them, methods based on spectral clustering are the most popular, e.g., [2, 23, 4, 27, 3, 22, 26, 5, 17, 37, 51, 53, 18, 50]. These methods divide the task of subspace clustering into two subproblems. The first subproblem is to learn a data affinity matrix from the original data, and the second subproblem is to apply spectral clustering on the affinity matrix to find the segmentation of the data. The two subproblems are solved successively in one-pass [2, 23, 4, 27, 26, 51] or solved alternately in multi-pass [5, 17, 7, 53, 18].

Finding an informative affinity matrix is the most crucial step. Typical methods to find an informative affinity matrix are based on the self-expression property of data [2, 45], which states that a data point in a union of subspaces can be expressed as a linear combination[1] of other data points, i.e., $\mathbf{x}_j = \sum_{i \neq j} c_{ij}\mathbf{x}_i + \mathbf{e}_j$, where $\mathbf{e}_j$ is used to model the noise or corruption in data. It is expected that the linear combination of data point $\mathbf{x}_j$ uses the data points that belong to the same subspace as $\mathbf{x}_j$. To achieve this objective, different types of regularization terms on the linear combination coefficients are used. For example, in [2] the ℓ_1 norm is used to find sparse linear combination; in [23] the nuclear norm of the coefficients matrix is used to find low-rank representation; in [46, 51] the mixture of the ℓ_1 norm and the ℓ_2 norm or the nuclear norm is used to balance the sparsity and the denseness of the linear combination coefficients; and in [48] a data-dependent sparsity-inducing regularizer is used to find sparse linear combination. On the other hand, different ways to model the noise or corruptions in data have also been investigated, e.g., the vector ℓ_1 norm is used in [2], the $\ell_{2,1}$ norm is adopted in [23], and the correntropy term is used in [9].

2.2. Subspace Clustering in Feature Space

For subspace clustering in feature space, we further divide the existing methods into two types. The first type uses *latent feature space*, which is induced via a Mercer kernel,

[1] If data points lie in a union of affine subspaces [19], then the *linear* combination will be modified to *affine* combination.

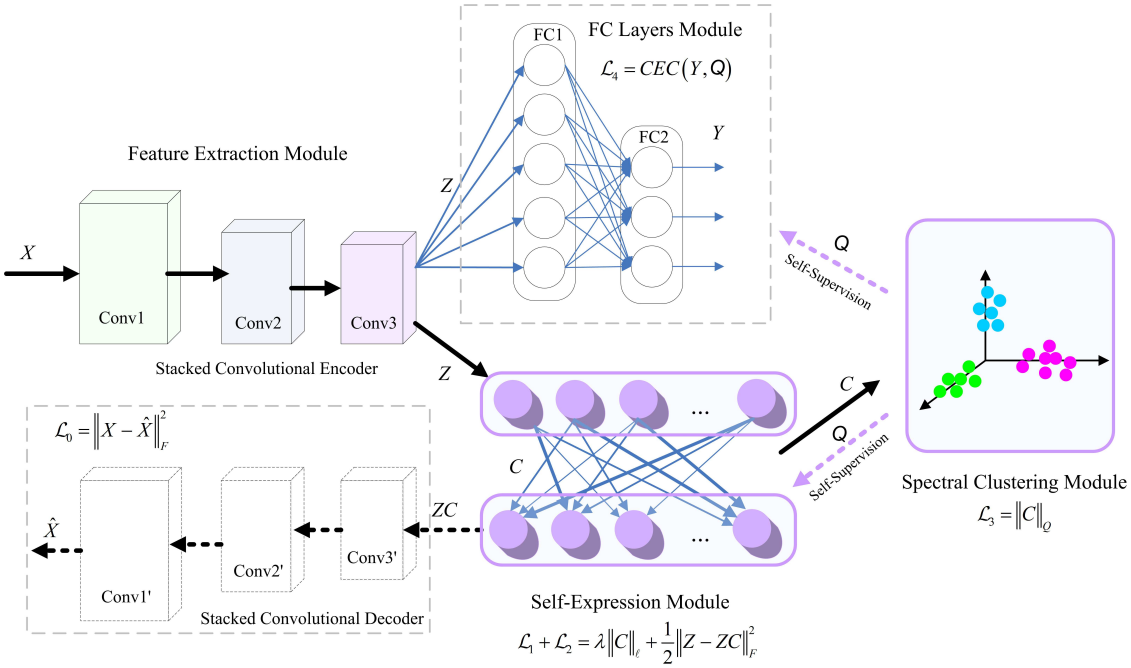


Figure 1. Architecture of the proposed Self-Supervised Convolutional Subspace Clustering Network (S^{2}ConvSCN). It consists of mainly five modules: a) stacked convolutional encoder module, which is used to extract convolutional features; b) stacked convolutional decoder module, which is used with the encoder module to initialize the convolutional module; c) self-expression module, which is used to learn the self-expressive coefficient matrix and also takes the self-supervision information from the result of spectral clustering to refine the self-expressive coefficients matrix; d) FC-layers based self-supervision module, which builds a self-supervision path back to the stacked convolutional encoder module; e) spectral clustering module, which provides self-supervision information to guide the self-expressive model and FC-layers module. The modules with solid line box are the backbone components; whereas the modules in dashed box are the auxiliary components to facilitate the training of the whole network.

e.g., [34, 32, 49, 47], or constructed via matrix decomposition, e.g., [24], [33]. The second type use *explicit feature space*, which is designed by manual feature extraction, e.g., [36], or is learned from data, e.g., [14, 54].

Latent Feature Space. Many recent works have employed the kernel trick to map the original data into a high-dimensional latent feature space, in which subspace clustering is performed, e.g., [34, 32, 49, 47]. For example, predefined polynomial and Gaussian kernels are used in the kernel sparse subspace clustering method [34, 32] and the kernel low-rank representation method [49, 47, 12]. Unfortunately, it is *not* guaranteed that the data in the latent feature space induced with such predefined kernels lie in low-dimensional subspaces.[2]

On the other hand, the latent feature space has also been constructed via matrix decomposition, e.g., [24], [33]. In [24], a linear transform matrix and a low-rank representation are computed simultaneously; in [33], a linear transform and a sparse representation are optimized jointly. However, the representation power of the learned linear transform is still limited.

[2]In [12], while the data matrix in the latent feature space is encouraged to be low-rank, it is not necessary that the data in feature space are encouraged to align with a union of linear subspaces.

Explicit Feature Space. Deep learning has gained a lot of research interests due to its powerful ability to learn hierarchical features in an end-to-end trainable way [10, 15]. Recently, there are a few works that use techniques in deep learning for feature extraction in subspace clustering. For example, in [36, 35], a fully connected deep auto-encoder network with hand-crafted features (e.g., SIFT or HOG features) combined with a sparse self-expression model is developed; in [14], a stacked convolutional auto-encoder network with a plus-in self-expression model is proposed. While promising clustering accuracy has been reported, these methods are still suboptimal because *neither* the potentially useful supervision information from the clustering result has been taken into the feature learning step *nor* a joint optimization framework for fully combining feature learning and subspace clustering has been developed. More recently, in [54], a deep adversarial network with a subspace-specific generator and a subspace-specific discriminator is adopted in the framework of [14] for subspace clustering. However, the discriminator need to use the dimension of each subspace, which is usually unknown.

In this paper, we attempt to develop a joint optimization framework for combining feature learning and subspace clustering, such that the useful self-supervision information

from subspace clustering result could be used to guide the feature learning and to refine the self-expression model. Inspired by the success of Convolutional Neural Networks in recent years for classification tasks on images and videos datasets [15] and the recent work [14], we integrate the convolutional feature extraction module into subspace clustering to form an end-to-end trainable joint optimization framework, called Self-Supervised Convolutional Subspace Clustering Network (S^{2}ConvSCN). In S^{2}ConvSCN, both the stacked convolutional layers based feature extraction and the self-expression based affinity learning are effectively self-supervised by exploiting the feedback from spectral clustering.

3. Our Proposal: Self-Supervised Convolutional Subspace Clustering Network

In this section, we present our S^{2}ConvSCN for joint feature learning and subspace clustering. We start with introducing our network formulation (see Fig. 1), then introduce the self-supervision modules. Finally, we present an effective procedure for training the proposed network.

3.1. Network Formulation

As aforementioned, our network is composed of a feature extraction module, a self-expression module and self-supervision modules for training the former two modules.

Feature Extraction Module. A basic component of our proposed S^{2}ConvSCN is the feature extraction module, which is used to extract features from raw data that are amenable to subspace clustering. To extract localized features while preserving spatial locality, we adopt the convolutional neural network which is comprised of multiple convolutional layers. We denote the input to the network as $\mathbf{h}^{(0)} = \mathbf{x}$ where $\mathbf{x}$ is the image. A convolutional layer ℓ contains a set of filters $\mathbf{w}_i^{(\ell)}$ and the associated biases $\mathbf{b}_i^{(\ell)}$, $i = 1, \cdots, m^{(\ell)}$, and produces $m^{(\ell)}$ feature maps from the output of the previous layer. The feature maps $\{\mathbf{h}_i^{(L)}\}_{i=1,\cdots,m^{(L)}}$ in the top layer L of the network are then used to form a representation of the input data $\mathbf{x}$. Specifically, the $m^{(L)}$ feature maps $\{\mathbf{h}_i^{(L)}\}_{i=1}^{m^{(L)}}$ are vectorized and concatenated to form a representation vector $\mathbf{z}$, i.e.,

$$\mathbf{z} = \left[\mathbf{h}_1^{(L)}(:), \cdots, \mathbf{h}_{m^{(L)}}^{(L)}(:)\right]^\top, \tag{1}$$

where $\mathbf{h}_1^{(L)}(:), \cdots, \mathbf{h}_{m^{(L)}}^{(L)}(:)$ are row vectors denoting the vectorization of the feature maps $\mathbf{h}_1^{(L)}, \cdots, \mathbf{h}_{m^{(L)}}^{(L)}$. These vectors are horizontally concatenated and then transposed to form the vector $\mathbf{z}$.

To ensure that the learned representation $\mathbf{z}$ contains meaningful information from the input data $\mathbf{x}$, the feature maps $\mathbf{h}_1^{(L)}, \cdots, \mathbf{h}_{m^{(L)}}^{(L)}$ are fed into a decoder network to reconstruct an image $\hat{\mathbf{x}}$. The loss function for this encoder-decoder network is the reconstruction error:

$$\mathcal{L}_0 = \frac{1}{2N}\sum_{j=1}^{N}\|\mathbf{x}_j - \hat{\mathbf{x}}_j\|_2^2 = \frac{1}{2N}\|X - \hat{X}\|_F^2, \tag{2}$$

where N is the number of images in the training set.

Self-Expression Module. State-of-the-art subspace clustering methods are based on the self-expression property of data, which states that each data point in a union of subspaces can be expressed as a linear combination of other data points [2, 45]. In order to learn feature representations that are suitable for subspace clustering, we adopt a self-expression module that imposes the following loss function:

$$\lambda\|C\|_\ell + \frac{1}{2}\|Z - ZC\|_F^2 \ \text{ s.t. } \ \text{diag}(C) = \mathbf{0}, \tag{3}$$

where $Z = \left[\mathbf{z}_1, \cdots, \mathbf{z}_N\right]$ is a matrix containing features from the feature extraction module as its columns, $\|C\|_\ell$ is a properly chosen regularization term, the constraint $\text{diag}(C) = \mathbf{0}$ is optionally used to rule out a trivial solution of $C = I$, and $\lambda > 0$ is a tradeoff parameter.

Self-Supervision Modules. Once the self-expression coefficient matrix C is obtained, we can compute a data affinity matrix as $A = \frac{1}{2}(|C| + |C^\top|)$. Subsequently, spectral clustering can be applied on A to obtain a segmentation of the data by minimizing the following cost:

$$\min_{Q} \ \sum_{i,j} a_{ij}\|\mathbf{q}_i - \mathbf{q}_j\|_2^2, \ \text{ s.t. } \ Q \in \mathcal{Q}, \tag{4}$$

where $\mathcal{Q} = \{Q \in \{0,1\}^{n\times N} : \mathbf{1}^\top Q = \mathbf{1}^\top \text{ and } \text{rank}(Q) = n\}$ is a set of all valid segmentation matrices with n groups, and $\mathbf{q}_i$ and $\mathbf{q}_j$ are respectively the i-th and j-th columns of Q indicating the membership of each data point to the assigned cluster. In practice, since the search over all $Q \in \mathcal{Q}$ is combinatorial, spectral clustering techniques usually relax the constraint $Q \in \mathcal{Q}$ to $QQ^\top = I$.

Observe that the spectral clustering produces a labeling of the data set which, albeit is not necessarily the correct class label for all the data points, contains meaningful information about the data. This motivates us to supervise the training of the feature extraction and self-expression modules using the output of spectral clustering. In principle, the features learned from the feature extraction module should contain enough information for predicting the class labels of the data points. Therefore, we introduce a classification layer on top of the feature extraction module which is expected to produce labels that aligns with the labels generated in spectral clustering. Furthermore, the segmentation produced by spectral clustering can also be used to construct a binary segmentation matrix, which contains information regarding which data points should be used in the expression

of a particular data point. Therefore, we incorporate the objective function of spectral clustering as a loss function in our network formulation, which has the effect of supervising the training of the self-expression module. We present the details of these two self-supervision modules in the following two subsections.

3.2. Self-Supervision for Self-Expression

To exploit the information in the labels produced by spectral clustering, we incorporate spectral clustering as a module of the network which provides a feedback to the self-expression model (see Fig. 1).

To see how the objective function of spectral clustering in (4) provides such feedback, we rewrite (4) to a weighted ℓ_1 norm of C as in [17], that is,

$$\frac{1}{2}\sum_{i,j} a_{ij}\|\mathbf{q}_i - \mathbf{q}_j\|_2^2 = \sum_{i,j}|c_{ij}|\frac{\|\mathbf{q}_i - \mathbf{q}_j\|_2^2}{2} := \|C\|_Q, \quad (5)$$

where we have used the fact that $a_{ij} = \frac{1}{2}(|c_{ij}| + |c_{ji}|)$. It can be seen from (5) that $\|C\|_Q$ measures the discrepancy between the coefficients matrix C and the segmentation matrix Q. When Q is provided, minimizing the cost $\|C\|_Q$ has the effect of enforcing the self-expression matrix C to be such that an entry c_{ij} is nonzero only if the i-th and j-th data points have the same class labels. Therefore, incorporating the term $\|C\|_Q$ in the network formulation helps the training of the self-expression module. That is, the result of previous spectral clustering can be incorporated into the self-expression model to provide self-supervision for refining the self-expression matrix C.

3.3. Self-Supervision for Feature Learning

We also use the class labels generated from spectral clustering to supervise the training of the feature extraction module. Notice that the output of spectral clustering is an n-dimensional vector which indicates the membership to n subspaces (i.e., clusters). Thus, we design FC layers as $p \times N_1 \times N_2 \times n$, where p is the dimension of the extracted convolutional feature, which is defined as the concatenation of the different feature maps of the last convolutional layer in the encoder block, and N_1 and N_2 are the numbers of neurons in the two FC layers, respectively.

Denote $\mathbf{y}$ as the n-dimensional output of the FC layers, where $\mathbf{y} \in I\!R^n$. Note that the output $\{\mathbf{q}_j\}_{j=1}^N$ of spectral clustering will be treated as the target output of the FC layers. To exploit the self-supervision information to train the convolutional encoder, we define a mixture of *cross-entropy loss* and *center loss* (CEC) as follows:

$$\mathcal{L}_4 = \frac{1}{N}\sum_{j=1}^{N}(\ln(1 + \mathrm{e}^{-\tilde{\mathbf{y}}_j^\top \mathbf{q}_j}) + \tau\|\mathbf{y}_j - \mu_{\pi(\mathbf{y}_j)}\|_2^2), \quad (6)$$

where $\tilde{\mathbf{y}}_j$ is a normalization of $\mathbf{y}_j$ via *softmax*, $\mu_{\pi(\mathbf{y_j})}$ denotes the cluster center which corresponds to $\mathbf{y}_j$, $\pi(\mathbf{y}_j)$ is to take the index of $\mathbf{y}_j$ from the output of spectral clustering, and $0 \le \tau \le 1$ is a tradeoff parameter. The first term of $\mathcal{L}_4$ is effectively a cross-entropy loss and the second term of $\mathcal{L}_4$ is a center loss which compresses the intra-cluster variations.

An important issue in defining such a loss function is that the output of spectral clustering $\{\mathbf{q}_j\}_{j=1}^N$ provides merely *pseudo labels* for the input data. That is, the label index assigned to a cluster in the returned result of spectral clustering is up to an unknown permutation. Therefore, the class labels from two successive epochs might not be consistent. To address this issue, we propose to perform a permutation of the new pseudo labels via Hungarian algorithm [29] to find an optimal assignment between the pseudo labels of successive iterations before feeding them into the self-supervision module with the cross-entropy loss in (6).

Remark 1. Note that the output of spectral clustering is used in two interrelated self-supervision modules and thus we call it a *dual self-supervision* mechanism.[3]

3.4. Training S^{2}ConvSCN

To obtain an end-to-end trainable framework, we design the total cost function of S^{2}ConvSCN by putting together the costs in (2), (3), (5), and (6) as follows:

$$\mathcal{L} = \mathcal{L}_0 + \gamma_1\mathcal{L}_1 + \gamma_2\mathcal{L}_2 + \gamma_3\mathcal{L}_3 + \gamma_4\mathcal{L}_4, \quad (7)$$

where $\mathcal{L}_1 = \|C\|_\ell$, $\mathcal{L}_2 = \frac{1}{2}\|Z - ZC\|_F^2$, $\mathcal{L}_3 = \|C\|_Q$, and γ_1, γ_2, γ_3 and γ_4 are four tradeoff parameters. The tradeoff parameters are set roughly to be inversely proportional to the value of each cost in order to obtain a balance amongst them.

To train S^{2}ConvSCN, we propose a two-stage strategy as follows: a) pre-train the stacked convolutional layers to provide an initialization of S^{2}ConvSCN; b) train the whole network with the assistance of the self-supervision information provided by spectral clustering.

Stage I: Pre-Training Stacked Convolutional Module. The pre-training stage uses the cost $\mathcal{L}_0$. In this stage, we set the weights in the two FC layers as zeros, which yield zeros output. Meanwhile, we also set the output of spectral clustering as zero vectors, i.e., $\mathbf{q}_j = 0$ for $j = 1, \cdots, N$. By doing so, the two FC layers are "sleeping" during this pre-training stage. Moreover, we set the coefficient matrix C as an identity matrix, which is equivalent to training S^{2}ConvSCN without the self-expression layer. As an optional pre-training, we can also use the pre-trained stacked

[3] While it is also sensible to term our approach with "self-training", we prefer to use the term "self-supervision" in order to emphasizes on the mechanism of guiding the training of the whole framework, that is to make each component as consistent as possible (i.e., be separable, self-expressive, and block diagonal).

Algorithm 1 Procedure for training S^{2}ConvSCN

Require: Input data, tradeoff parameters, maximum iteration $\mathtt{T}_{\max}$, $\mathtt{T}_0$, and t=1.
1: Pre-train the stacked convolutional module via stacked CAE.
2: (Optional) Pre-train the stacked convolutional module with the self-expressive layer.
3: Initialize the FC layers.
4: Run self-expressive layer.
5: Run spectral clustering layer to get the segmentation Q.
6: **while** $\mathtt{t} \leq \mathtt{T}_{\max}$ **do**
7: Fixed Q, update the other parts $\mathtt{T}_0$ epoches.
8: Run spectral clustering once to update Q and set t $\leftarrow$ t+1.
9: **end while**
Ensure: trained S^{2}ConvSCN and Q.

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

Table 1. Network settings for Extended Yale B and ORL.

CAE to train the stacked CAE with the self-expression layer.

Stage II: Training the Whole S^{2}ConvSCN. In this stage, we use the total cost $\mathcal{L}$ to train the whole S^{2}ConvSCN as a stacked CAE assisted with the self-expression module and dual self-supervision. To be more specific, given the spectral clustering result Q, we update the other parameters in S^{2}ConvSCN for T_0 epoches, and then perform spectral clustering to update Q. For clarity, we provide the detailed procedure to train S^{2}ConvSCN in Algorithm 1.

Remark 2. In the total cost function as (7), if we set $\gamma_3 = \gamma_4 = 0$, then the two self-supervision blocks will disappear and our S^{2}ConvSCN reduces to DSCNet [14]. Thus, it would be interesting to add an extra pre-training stage, i.e., using the cost function $\mathcal{L}_0 + \gamma_1\mathcal{L}_1 + \gamma_2\mathcal{L}_2$ to train the stacked convolutional module and the self-expressive layer together before evoking the FC layers and the spectral clustering layer. This is effectively a DSCNet [14]. In experiments, as used in [14], we stop the training by setting a maximum number of epoches $T_{\max}$.

4. Experimental Evaluations

To evaluate the performance of our proposed S^{2}ConvSCN, we conduct experiments on four benchmark data sets: two face image data sets, the Extended Yale B [6] and ORL [39], and two object image data sets, COIL20 and COIL100 [30]. We compare our proposed S^{2}ConvSCN with the following baseline algorithms, including Low Rank Representation (LRR) [23], Low Rank Subspace Clustering (LRSC) [43], Sparse Subspace Clustering (SSC) [3], Kernel Sparse Subspace Clustering (KSSC) [34], SSC by Orthogonal Matching Pursuit (SSC-OMP) [52], Efficient Dense Subspace Clustering (EDSC) [13], Structured SSC (S^{3}C) [18], SSC with the pre-trained convolutional auto-encoder features (AE+SSC), EDSC with the pre-trained convolutional auto-encoder features (AE+EDSC), Deep Subspace Clustering Networks (DSCNet) [14] and Deep Adversarial Subspace Clustering (DASC) [54]. For EDSC, AE+EDSC, DSCNet and DASC, we directly cite the best results reported in [14] and [54]. For S^{3}C, we use *soft* S^{3}C with a fixed parameter $\alpha = 1$.

The architecture specification of S^{2}ConvSCN used in our experiments for each dataset are listed in Table 1 and Table 4. In the stacked convolutional layers, we set the kernel stride as 2 in both horizontal and vertical directions, and use Rectified Linear Unit (ReLU) [15] as the activation function $\sigma(\cdot)$. In addition, the learning rate is set to 1.0×10^{-3} in all our experiments. The whole data set is used as one batch input. For the FC layers, we set $N_1 = \frac{N}{2}$ and $N_2 = n$.

To find informative affinity matrix, we adopt the vector ℓ_1 norm and the vector ℓ_2 norm to define $\|C\|_\ell$ and denote as S^{2}ConvSCN-ℓ_1 and S^{2}ConvSCN-ℓ_2, respectively. In the second training stage, we update the stacked convolutional layers, the self-expression model, and the FC layers for T_0 epochs and then update the spectral clustering module once, where T_0 is set to $5 \sim 16$ in our experiments.

4.1. Experiments on Extended Yale B

The Extended Yale B database [6] consists of face images of 38 subjects, 2432 images in total, with approximately 64 frontal face images per subject taken under different illumination conditions, where the face images of each subject correspond to a low-dimensional subspace. In our experiments, we follow the protocol used in [14]: a) each image is down-sampled from 192×168 to 48×42 pixels; b) experiments are conducted using all choices of $n \in \{10, 15, 20, 25, 30, 35, 38\}$.

To make a fair comparison, we use the same setting as that used in DSCNet [14], in which a three-layer stacked convolutional encoders is used with $\{10, 20, 30\}$ channels, respectively. The detailed settings for the stacked convolutional network used on Extended Yale B are shown Table 1. The common parameters γ_1 and γ_2 are set the same as that in DSCNet, where $\gamma_1 = 1$ (for the term $\|C\|_\ell$) and

Methods	LRR	LRSC	SSC	AE+ SSC	KSSC	SSC-OMP	soft S^{3}C†	EDSC	AE+ EDSC	DSC-ℓ_1	DSC-ℓ_2	**Ours** (ℓ_2)	**Ours** (ℓ_1)
10 subjects													
Mean	19.76	30.95	8.80	17.06	14.49	12.08	6.34	5.64	5.46	2.23	1.59	**1.18**	**1.18**
Median	18.91	29.38	9.06	17.75	15.78	8.28	3.75	5.47	6.09	2.03	1.25	**1.09**	**1.09**
15 subjects													
Mean	25.82	31.47	12.89	18.65	16.22	14.05	11.01	7.63	6.70	2.17	1.69	1.14	**1.12**
Median	26.30	31.64	13.23	17.76	17.34	14.69	10.89	6.41	5.52	2.03	1.72	**1.14**	**1.14**
20 subjects													
Mean	31.45	28.76	20.11	18.23	16.55	15.16	14.07	9.30	7.67	2.17	1.73	1.31	**1.30**
Median	32.11	28.91	21.41	16.80	17.34	15.23	13.98	10.31	6.56	2.11	1.80	1.32	**1.25**
25 subjects													
Mean	28.14	27.81	26.30	18.72	18.56	18.89	16.79	10.67	10.27	2.53	1.75	1.32	**1.29**
Median	28.22	26.81	26.56	17.88	18.03	18.53	17.13	10.84	10.22	2.19	1.81	1.34	**1.28**
30 subjects													
Mean	38.59	30.64	27.52	19.99	20.49	20.75	20.46	11.24	11.56	2.63	2.07	1.71	**1.67**
Median	36.98	30.31	27.97	20.00	20.94	20.52	21.15	11.09	10.36	2.81	2.19	1.77	**1.72**
35 subjects													
Mean	40.61	31.35	29.19	22.13	26.07	20.29	20.38	13.10	13.28	3.09	2.65	1.67	**1.62**
Median	40.71	31.74	29.51	21.74	25.92	20.18	20.47	13.10	13.21	3.10	2.64	1.69	**1.60**
38 subjects													
Mean	35.12	29.89	29.36	25.33	27.75	23.52	19.45	11.64	12.66	3.33	2.67	1.56	**1.52**
Median	35.12	29.89	29.36	25.33	27.75	23.52	19.45	11.64	12.66	3.33	2.67	1.56	**1.52**

Table 2. Clustering Error (%) on Extended Yale B. The best results are in bold and the second best results are underlined.

$\gamma_2 = 1.0 \times 10^{\frac{n}{10}-3}$. For the specific parameters used in S^{2}ConvSCN, we set $\gamma_3 = 16$ for the term $\|C\|_Q$ and $\gamma_4 = 72$ for the cross-entropy term, respectively. We set $T_0 = 5$ and $T_{\max} = 10 + 40n$.

The experimental results are presented in Table 2. We observe that our proposed S^{2}ConvSCN-ℓ_1 and S^{2}ConvSCN-ℓ_2 remarkably reduced the clustering errors and yield the lowest clustering errors with $n \in \{10, 15, 20, 25, 30, 35, 38\}$ than all the listed baseline methods. We note that DASC [54] reported a clustering error of 1.44% on Extended Yale B with $n = 38$, which is slightly better than our results.

To gain further understanding of the proposed dual self-supervision, we use S^{2}ConvSCN-ℓ_1 as an example and evaluate the effect of using the dual self-supervision modules via an ablation study. Due to space limitation, we only list the experimental results of using a single self-supervision via $\mathcal{L}_3$, using a single self-supervision via $\mathcal{L}_4$, and using dual self-supervision of $\mathcal{L}_3$ plus $\mathcal{L}_4$ on datasets Extended Yale B in Table 3. As a baseline, we show the experimental results of DSCNet [14], which uses the loss $\mathcal{L}_0 + \mathcal{L}_1 + \mathcal{L}_2$. As could be read from Table 3 that, using only a single self-supervision module, i.e., $\mathcal{L}_0 + \mathcal{L}_1 + \mathcal{L}_2$ plus $\mathcal{L}_3$, or $\mathcal{L}_0 + \mathcal{L}_1 + \mathcal{L}_2$ plus $\mathcal{L}_4$, the clustering errors are reduced. Compared to using the self-supervision via a spectral clustering loss $\mathcal{L}_3$ in the self-expression module, using the self-supervision via the classification loss $\mathcal{L}_4$ in FC block is more effective. Nonetheless, using the dual supervision modules further reduces the clustering errors.

4.2. Experiments on ORL

The ORL dataset [39] consists of face images of 40 distinct subjects, each subjects having 10 face images under varying lighting conditions, with different facial expressions (open/closed eyes, smiling/not smiling) and facial details (glasses / no glasses) [39]. As the images were took under variations of facial expressions, this data set is more challenging for subspace clustering due to the nonlinearity and small sample size per subject.

In our experiments, each image is down-sampled from 112×92 to 32×32. We reduce the kernel size in convolution module to 3×3 due to small image size and set the number of channels to $\{3, 3, 5\}$. The specification of the network structure is shown in Table 1. For the tradeoff parameters, we set $\gamma_1 = 0.1, \gamma_2 = 0.01, \gamma_3 = 8$, and $\gamma_4 = 1.2$ for our S^{2}ConvSCN. For the fine-tuning stage, we set $T_0 = 5$ and $T_{\max} = 940$. Experimental results are shown in Table 5. Again, our proposed approaches yield the best results.

4.3. Experiments on COIL20 and COIL100

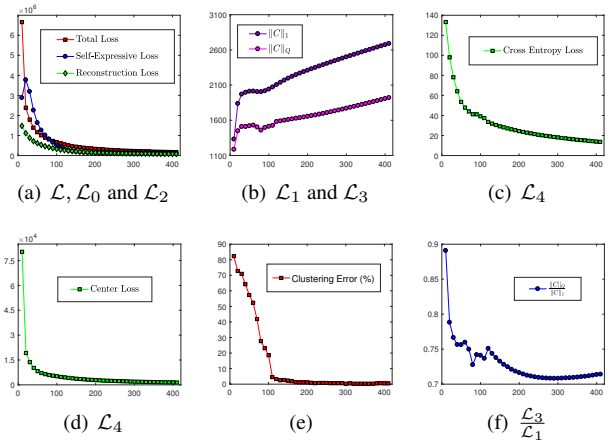
To further verify the effectiveness of our proposed S^{2}ConvSCN, we conduct experiments on dataset COIL20 and COIL100 [30]. COIL20 contains 1440 gray-scale images of 20 objects; whereas COIL100 contains 7200 images of 100 objects. Each image was down-sampled to 32×32. The settings of the stacked convolutional networks used for COIL20 and COIL100 are listed in Table 4.

For the tradeoff parameters on COIL20, we set $\gamma_1 = 1$, $\gamma_2 = 30$ as same as used in DSC-Net [14], and $\gamma_3 = 8$, $\gamma_4 = 6$, $T_0 = 4$, and $T_{\max} = 80$ in our S^{2}ConvSCN. For the tradeoff parameters on COIL100, we set $\gamma_1 = 1$, $\gamma_2 = 30$ as same as used in DSC-Net [14], and $\gamma_3 = 8$, $\gamma_4 = 7$, $T_0 = 16$, and $T_{\max} = 110$ in our S^{2}ConvSCN.

For experiments on COIL20 and COIL100, we initialize the convolutional module with stacked CAE at first, and then train a stacked CAE assisted with a self-expressive model. This is effectively DSCNet [14]. And then, we train the whole S^{2}ConvSCN. Experimental results are listed in Table 5. As could be read, our S^{2}ConvSCN-ℓ_1 and S^{2}ConvSCN-ℓ_2 reduce the clustering errors significantly. This result confirms the effectiveness of the designed dual

Losses \ No. Subjects	10 subjects		15 subjects		20 subjects		25 subjects		30 subjects		35 subjects		38 subjects	
	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median
$\mathcal{L}_0+\mathcal{L}_1+\mathcal{L}_2$(DSC-$\ell_1$ [14])	2.23	2.03	2.17	2.03	2.17	2.11	2.53	2.19	2.63	2.81	3.09	3.10	3.33	3.33
$\mathcal{L}_0+\mathcal{L}_1+\mathcal{L}_2+\mathcal{L}_3$	1.58	1.25	1.63	1.55	1.67	1.57	1.61	1.63	2.74	1.82	2.64	2.65	2.75	2.75
$\mathcal{L}_0+\mathcal{L}_1+\mathcal{L}_2+\mathcal{L}_4$	1.32	**1.09**	1.31	1.30	1.54	1.48	1.48	1.98	1.87	**1.61**	1.82	1.84	1.92	1.92
$\mathcal{L}_0+\mathcal{L}_1+\mathcal{L}_2+\mathcal{L}_3+\mathcal{L}_4$	**1.18**	**1.09**	**1.12**	**1.14**	**1.30**	**1.25**	**1.29**	**1.28**	**1.67**	1.72	**1.62**	**1.60**	**1.52**	**1.52**

Table 3. Ablation Study on S^{2}ConvSCN-ℓ_1 on Extended Yale B.



(a) $\mathcal{L}$, $\mathcal{L}_0$ and $\mathcal{L}_2$ (b) $\mathcal{L}_1$ and $\mathcal{L}_3$ (c) $\mathcal{L}_4$

(d) $\mathcal{L}_4$ (e) (f) $\frac{\mathcal{L}_3}{\mathcal{L}_1}$

Figure 2. The cost functions and clustering error of S^{2}ConvSCN-ℓ_1 during training period on Extended Yale B ($n = 10$).

self-supervision components for the proper use of the useful information from the output of spectral clustering.

4.4. Convergence Behaviors

To show the convergence behavior during training iterations, we conduct experiments on Extended Yale B with $n = 10$. We record the clustering errors and each cost function during training period, and show them as a function of the number of epoches in Fig. 2. As could be observed from Fig. 2(a), (c), (d) and (e), the cost functions $\mathcal{L}$, $\mathcal{L}_0$, $\mathcal{L}_2$, and $\mathcal{L}_4$, and the cluster error decrease rapidly and tend to "flat". To show more details in the iterations, in Fig. 2 (b) and (f), we show the curves of $\|C\|_1$, $\|C\|_Q$ and $\frac{\|C\|_Q}{\|C\|_1}$. Note that $\|C\|_Q$ and $\frac{\|C\|_Q}{\|C\|_1}$ are the cost and the relative cost of spectral clustering, respectively. Compared to $\|C\|_Q$, we argue that $\frac{\|C\|_Q}{\|C\|_1}$ is more indicative to the clustering performance. As could be observed, while $\|C\|_1$ and $\|C\|_Q$ are increasing[4], the curve of $\frac{\|C\|_Q}{\|C\|_1}$ tends to "flat"—which is largely consistent to the curve of the clustering error in Fig. 2 (e).

5. Conclusion

We have proposed an end-to-end trainable framework for simultaneous feature learning and subspace clustering, called Self-Supervised Convolutional Subspace Clustering Network (S^{2}ConvSCN). Specifically, in S^{2}ConvSCN, the feature extraction via stacked convolutional module, the affinity learning via self-expression model, and the data segmentation via spectral clustering are integrated into a joint optimization framework. By exploiting a dual self-supervision mechanism, the output of spectral clustering are effectively used to improve the training of the stacked convolutional module and to refine the self-expression model, leading to superior performance. Experiments on benchmark datasets have validated the effectiveness of our proposed approach.

[4]The observation that the curves of $\mathcal{L}_1$ and $\mathcal{L}_3$ go up is because the entries of the extracted feature Z are slowly shrinking and thus the absolute values of entries of C are slowly increasing, due to the absence of normalization step in feature learning at each epoch.

Layers	COIL20 kernel size	COIL20 channels	COIL100 kernel size	COIL100 channels
encoder-1	3×3	15	5×5	50
decoder-1	3×3	15	5×5	50

Table 4. Network settings for COIL20 and COIL100.

Methods	ORL	COIL20	COIL100
LRR	33.50	30.21	53.18
LRSC	32.50	31.25	50.67
SSC	29.50	14.83	44.90
AE+SSC	26.75	22.08	43.93
KSSC	34.25	24.65	47.18
SSC-OMP	37.05	29.86	67.29
EDSC	27.25	14.86	38.13
AE+EDSC	26.25	14.79	38.88
soft S^{3}C†	26.00	11.87	41.71
DSC-ℓ_1	14.25	5.65	33.62
DSC-ℓ_2	14.00	5.42	30.96
DASC [54]	11.75	3.61	-
S^{2}ConvSCN-ℓ_2	11.25	2.33	27.83
S^{2}ConvSCN-ℓ_1	**10.50**	**2.14**	**26.67**

Table 5. Clustering Error (%) on ORL, COIL20 and COIL100.

Acknowledgment

J. Zhang and C.-G. Li are supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61876022, and the Open Project Fund from Key Laboratory of Machine Perception (MOE), Peking University. H. Zhang is partially supported by NSFC under Grant Nos. 61701032 and 61806184. X. Qi is supported by Shenzhen Fundamental Research Fund under Grants Nos. ZDSYS201707251409055 and 2017ZT07X152. Z. Lin is supported by 973 Program of China under Grant No. 2015CB352502, NSFC under Grant Nos. 61625301 and 61731018, Qualcomm, and Microsoft Research Asia.

References

[1] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005. 1

[2] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pages 2790–2797, 2009. 1, 2, 4

[3] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2765–2781, 2013. 1, 2, 6

[4] P. Favaro, R. Vidal, and A. Ravichandran. A closed form solution to robust subspace estimation and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1801 –1807, 2011. 1, 2

[5] J. Feng, Z. Lin, H. Xu, and S. Yan. Robust subspace segmentation with block-diagonal prior. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3818–3825, 2014. 1, 2

[6] A.-S. Georghiades, P.-N. Belhumeur, and D.-J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660, 2001. 6

[7] X. Guo. Robust subspace segmentation by simultaneously learning data representations and their affinity matrix. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 3547–3553, 2015. 2

[8] T. Hastie and P.-Y. Simard. Metrics and models for handwritten character recognition. *Statistical Science*, pages 54–65, 1998. 1

[9] R. He, L. Wang, Z. Sun, Y. Zhang, and B. Li. Information theoretic subspace clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 27(12):2643–2655, 2016. 2

[10] G. Hinton, L. Deng, D. Yu, G.-E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, and T. N. Sainath. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012. 3

[11] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D.-J. Kriegman. Clustering appearances of objects under varying illumination conditions. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pages 11–18, 2003. 1

[12] P. Ji, I. Reid, R. Garg, H. Li, and M. Salzmann. Adaptive low-rank kernel subspace clustering. *arXiv:1707.04974v4*, 2019. 3

[13] P. Ji, M. Salzmann, and H. Li. Efficient dense subspace clustering. In *IEEE Winter conferance on Applications of Computer Vision*, pages 461–468, 2014. 6

[14] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid. Deep subspace clustering networks. In *Neural Information Processing Systems (NIPS)*, 2017. 2, 3, 4, 6, 7, 8

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems*, pages 1097–1105, 2012. 2, 3, 4, 6

[16] J. Lezama, Q. Qiu, P. Muse, and G. Sapiro. Ole: Orthogonal low-rank embedding - a plug and play geometric loss for deep learning. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pages 8109–8118, 2018. 2

[17] C.-G. Li and R. Vidal. Structured sparse subspace clustering: A unified optimization framework. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pages 277–286, 2015. 1, 2, 5

[18] C.-G. Li, C. You, and R. Vidal. Structured sparse subspace clustering: A joint affinity learning and subspace clustering framework. *IEEE Transactions on Image Processing*, 26(6):2988–3001, 2017. 1, 2, 6

[19] C.-G. Li, C. You, and R. Vidal. On geometric analysis of affine sparse subspace clustering. *IEEE Journal on Selected Topics in Signal Processing*, 12(6), 2018. 2

[20] C.-G. Li, J. Zhang, and J. Guo. Constrained sparse subspace clustering with side information. In *Proceedings of the 24th International Conference on Pattern Recognition (ICPR)*, pages 2093–2099, August 2018. 1

[21] Q. Li, Z. Sun, Z. Lin, R. He, and T. Tan. Transformation invariant subspace clustering. *Pattern Recognition*, pages 142–155, 2016. 1

[22] G. Liu, Z. Lin, S.-C. Yan, J. Sun, Y. Yu, and Y. Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):171–184, 2013. 1, 2

[23] G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *Proceedings of International Conference on Machine Learning*, pages 663–670, 2010. 1, 2, 6

[24] G. Liu and S. Yan. Latent low-rank representation for subspace segmentation and feature extraction. In *IEEE International Conference on Computer Vision*, pages 1615–1622, 2011. 3

[25] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 20:91–110, 2004. 1

[26] C. Lu, Z. Lin, and S. Yan. Correlation adaptive subspace segmentation by trace lasso. In *Proceedings of IEEE International Conference on Computer Vision*, pages 1345–1352, 2014. 1, 2

[27] C.-Y. Lu, H. Min, Z.-Q. Zhao, L. Zhu, D.-S. Huang, and S.-C. Yan. Robust and efficient subspace segmentation via least squares regression. *Proceedings of European Conference on Computer Vision*, pages 347–360, 2012. 1, 2

[28] B. McWilliams and G. Montana. Subspace clustering of high dimensional data: a predictive approach. *Data Mining and Knowledge Discovery*, 28(3):736–772, 2014. 1

[29] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957. 5

[30] S.-A. Nene, S.-K. Nayar, and H. Murase. Columbia object image library. *Columbia University*, 1996. 6, 7

[31] O. M. Parkhi, A. Vedaldi, A. Zisserman, et al. Deep face recognition. In *BMVC*, volume 1, page 6, 2015. 2

[32] V. M. Patel, H. V. Nguyen, and R. Vidal. Latent space sparse and low-rank subspace clustering. *IEEE Journal of Selected Topics in Signal Processing*, 9(4):691–701, 2015. 3

[33] V.-M. Patel, H. V. Nguyen, and R. Vidal. Latent space sparse subspace clustering. In *Proceedings of IEEE International Conference on Computer Vision*, pages 225–232, Dev 2013. 3

[34] V.-M. Patel. and R. Vidal. Kernel sparse subspace clustering. In *Proceedings of IEEE International Conference on Image Processing*, pages 2849–2853, 2014. 3, 6

[35] X. Peng, J. Feng, S. Xiao, J. Lu, Z. Yi, and S. Yan. Deep sparse subspace clustering. *arXiv preprint arXiv:1709.08374*, 2017. 1, 2, 3

[36] X. Peng, S. Xiao, J. Feng, W. Y. Yau, and Z. Yi. Deep subspace clustering with sparsity prior. In *International Joint Conference on Artificial Intelligence*, pages 1925–1931, 2016. 1, 2, 3

[37] X. Peng, Z. Yu, Z. Yi, and H. Tang. Constructing the l_2-graph for robust subspace learning and subspace clustering. *IEEE Transactions on Cybernetics*, 47(4):1053–1066, 2017. 1, 2

[38] X. Qi, R. Xiao, C.-G. Li, Y. Qiao, J. Guo, and X. Tang. Pairwise rotation invariant co-occurrence local binary pattern. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2199–2213, 2014. 1

[39] F.-S. Samaria and A.-C. Harter. Harter, a.: Parameterisation of a stochastic model for human face identification. In *Proceedings of the Second IEEE Workshop on Applications of Computer Vision*, pages 138–142, 1994. 6, 7

[40] R. Shankar, T. Roberto, R. Vidal, and Y. Ma. Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1832–1845, 2010. 1

[41] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal on Computer Vision*, 9(2):137–154, 1992. 1

[42] R. Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2011. 1

[43] R. Vidal and P. Favaro. Low rank subspace clustering (lrsc). *Pattern Recognition Letters*, 43:47–61, 2014. 6

[44] R. Vidal, Y. Ma, and S. Sastry. Generalized Principal Component Analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1–15, 2005. 1

[45] R. Vidal, Y. Ma, and S. Sastry. *Generalized Principal Component Analysis*. Springer Verlag, 2016. 1, 2, 4

[46] Y. X. Wang, H. Xu, and C. Leng. Provable subspace clustering: when lrr meets ssc. In *Neural Information Processing Systems (NIPS)*, pages 64–72, 2013. 2

[47] S. Xiao, M. Tan, D. Xu, and Z.-Y. Dong. Robust kernel low-rank representation. *IEEE Transactions on Neural Networks and Learning Systems*, 27(11):2268–2281, 2016. 3

[48] B. Xin, Y. Wang, W. Gao, and D. Wipf. Building invariances into sparse subspace clustering. *IEEE Transactions on Signal Processing*, 66(2):449–462, 2018. 2

[49] H. N. W. Yang, F. Shen, and C. Sun. Kernel low-rank representation for face recognition. *Neurocomputing*, 155:32–42, 2015. 3

[50] C. You, C. Li, D. P. Robinson, and R. Vidal. Scalable exemplar-based subspace clustering on class-imbalanced data. In *Proceedings of European Conference on Computer Vision*, September 2018. 2

[51] C. You, C.-G. Li, D. Robinson, and R. Vidal. Oracle based active set algorithm for scalable elastic net subspace clustering. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pages 3928–3937, 2016. 1, 2

[52] C. You, D. Robinson, and R. Vidal. Scalable sparse subspace clustering by orthogonal matching pursuit. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pages 3918–3927, 2016. 6

[53] J. Zhang, C.-G. Li, H. Zhang, and J. Guo. Low-rank and structured sparse subspace clustering. In *Proceeding of IEEE Visual Communication and Image Processing*, 2016. 1, 2

[54] P. Zhou, Y. Hou, and J. Feng. Deep adversarial subspace clustering. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, June 2018. 2, 3, 6, 7, 8