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^2ConvSCN), 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 PRCOLBP [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 of other data points, i.e., \( \mathbf{x}_j = \sum_{i \neq j} \lambda_{ij} \mathbf{x}_i + \mathbf{e}_j \), 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 regularizations are used. For example, in [2] the \( \ell_1 \) norm is used to find sparse linear combination; in [25] the nuclear norm of the coefficients matrix is used to find low-rank representation; in [46, 51] the mixture of the \( \ell_1 \) norm and the \( \ell_2 \) 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 \( \ell_1 \) 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\footnote{If data points lie in a union of affine subspaces [19], then the linear combination will be modified to affine combination.}.
The architecture of the proposed Self-Supervised Convolutional Subspace Clustering Network (S\textsuperscript{2}ConvSCN) is shown in Figure 1. 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.

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.\footnote{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.}

On the other hand, the latent feature space has also been constructed via matrix decomposition, e.g., [24], [33]. The second type uses explicit feature space, which is designed by manual feature extraction, e.g., [36], or is learned from data, e.g., [14, 54].

Explicit Feature Space. Deep learning has gained a lot of research interest 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. In-
spired 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 con-
volutional feature extraction module into subspace cluster-
ing to form an end-to-end trainable joint optimization frame-
work, called Self-Supervised Convolutional Subspace Clus-
tering Network (S²ConvSCN). In S²ConvSCN, both the s-
tacked convolutional layers based feature extraction and the
self-expression based affinity learning are effectively self-
supervised by exploiting the feedback from spectral clus-
tering.

3. Our Proposal: Self-Supervised Convolution-
al Subspace Clustering Network

In this section, we present our S²ConvSCN for joint fea-
ture learning and subspace clustering. We start with intro-
ducing our network formulation (see Fig. 1), then introduce
the self-supervision modules. Finally, we present an effec-
tive procedure for training the proposed network.

3.1. Network Formulation

As aforementioned, our network is composed of a fea-
ture 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 S²ConvSCN is the feature extraction module,
which is used to extract features from raw data that are a-
menable to subspace clustering. To extract localized fea-
tures while preserving spatial locality, we adopt the con-
volutional neural network which is comprised of multiple
convolutional layers. We denote the input to the network as
\( x^{(0)} \) where \( x \) is the image. A convolutional lay-
er \( \ell \) contains a set of filters \( w^{(\ell)} \) and the associated bias-
es \( b^{(\ell)}_i, i = 1, \ldots, m^{(\ell)} \), and produces \( m^{(\ell)} \) feature maps
from the output of the previous layer. The feature maps
\( \{ h^{(L)}_i \}_{i=1,\ldots,m^{(L)}} \) in the top layer \( L \) of the network are
then used to form a representation of the input data \( x \). Specif-
ically, the \( m^{(L)} \) feature maps \( \{ h^{(L)}_i \}_{i=1,\ldots,m^{(L)}} \) are vectorized and
concatenated to form a representation vector \( z \), i.e.,

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

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

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

\[
L_0 = \frac{1}{2N} \sum_{j=1}^N \| x_j - \hat{x}_j \|_2^2 = \frac{1}{2N} \| X - \hat{X} \|_F^2, \tag{2}
\]
where \( N \) is the number of images in the training set.

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

\[
\lambda \| C \|_\ell + \frac{1}{2} \| Z - ZC \|_F^2 \quad \text{s.t. } \text{diag}(C) = 0, \tag{3}
\]
where \( Z = [z_1, \ldots, z_N] \) is a matrix containing features
from the feature extraction module as its columns, \( \| C \|_\ell \)
is a properly chosen regularization term, the constraint
\( \text{diag}(C) = 0 \) 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 coef-
cicient matrix \( C \) is obtained, we can compute a data affin-
ity matrix as \( A = \frac{1}{2} (|C| + |C^\top|) \). Subsequently, spectral clus-
tering can be applied on \( A \) to obtain a segmentation of the
data by minimizing the following cost:

\[
\min_Q \sum_{i,j} a_{ij} \| q_i - q_j \|_2^2 \quad \text{s.t. } Q \in \mathcal{Q}, \tag{4}
\]
where \( \mathcal{Q} = \{ Q \in \{0, 1\}^{n \times N} : 1^\top Q = 1^\top \text{ and rank}(Q) = n \} \) is a set of all valid segmentation matrices with \( n \) groups,
and \( q_i \) and \( q_j \) 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 infor-
mation about the data. This motivates us to supervise the
training of the feature extraction and self-expression mod-
ules 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 lay-
er on top of the feature extraction module which is expect-
ted to produce labels that aligns with the labels generated
in spectral clustering. Furthermore, the segmentation pro-
duced by spectral clustering can also be used to construct a
binary segmentation matrix, which contains information re-
garding which data points should be used in the expression.
where we have used the fact that $\alpha_j$ is a normalization of $y_j$ via softmax. $\mu_\pi(y_j)$ denotes the cluster center which corresponds to $y_j$. $\pi(y_j)$ is to take the index of $y_j$ from the output of spectral clustering, and $0 \leq \tau \leq 1$ is a tradeoff parameter. The first term of $L_4$ is effectively a cross-entropy loss and the second term of $L_4$ 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 $\{q_j\}_{j=1}^N$ provides merely pseudo labels for the input data. That is, the label 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:

$$L = L_1 + \gamma_1 L_2 + \gamma_2 L_3 + \gamma_3 L_4,$$

where $L_1 = \|C\|_F$, $L_2 = \frac{1}{2}\|Z - ZC\|_F^2$, $L_3 = \|C\|_Q$, and $\gamma_1, \gamma_2, \gamma_3$ and $\gamma_4$ 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 $L_0$. 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., $q_j = 0$ for $j = 1, \ldots, N$. 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 emphasize 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 $T_{\text{max}}$, $T_0$, 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 $t \leq T_{\text{max}}$ do
7: Fixed $Q$, update the other parts $T_0$ epochs.
8: Run spectral clustering once to update $Q$ and set $t \leftarrow t+1$.
9: end while

Ensure: trained $S^2$ConvSCN and $Q$.

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 $T_0$ epochs, 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 epochs $T_{\text{max}}$.

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 (SSC) [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 $1.0 \times 10^{-3}$ 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 $\ell_1$ norm and the vector $\ell_2$ norm to define $\|C\|_\ell$ and denote as $S^2$ConvSCN-$\ell_1$ and $S^2$ConvSCN-$\ell_2$, 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 $192 \times 168$ to $48 \times 42$ 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 $\gamma_1$ and $\gamma_2$ are set the same as that in DSCNet, where $\gamma_1 = 1$ (for the term $\|C\|_\ell$).
\( \gamma_2 = 1.0 \times 10^7 \text{m}^{-3} \). For the specific parameters used in S^2 ConvSCN, we set \( \gamma_3 = 16 \) for the term \( |\mathbf{C}|_{Q} \) and \( \gamma_4 = 72 \) for the cross-entropy term, respectively. We set \( T_0 = 5 \) and \( T_{\text{max}} = 10 + 40n \).

The experimental results are presented in Table 2. We observe that our proposed S^2 ConvSCN-\( \ell_1 \) and S^2 ConvSCN-\( \ell_2 \) 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-\( \ell_1 \) 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 subject having 10 face images under varying lighting conditions, with different facial expression—s (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 \( T_{\text{max}} = 940 \). 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 \( T_{\text{max}} = 80 \) 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 \( T_{\text{max}} = 110 \) 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-\( \ell_1 \) and S^2 ConvSCN-\( \ell_2 \) reduce the clustering errors significantly. This result confirms the effectiveness of the designed dual
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 $L_0$, $L_2$, and $L_4$, 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 $\Vert C \Vert_1$, $\Vert C \Vert_Q$ and $\Vert \frac{\Vert C \Vert_1}{\Vert C \Vert_Q} \Vert$. Note that $\Vert C \Vert_1$ and $\Vert C \Vert_Q$ are the cost and the relative cost of spectral clustering, respectively. Compared to $\Vert C \Vert_Q$, we argue that $\Vert \frac{\Vert C \Vert_1}{\Vert C \Vert_Q} \Vert$ is more indicative to the clustering performance. As could be observed, while $\Vert C \Vert_1$ and $\Vert C \Vert_Q$ are increasing, the curve of $\Vert \frac{\Vert C \Vert_1}{\Vert C \Vert_Q} \Vert$ 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 (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.

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


