SUBSPACE CLUSTERING: FROM SHALLOW TO DEEP METHODS - A SURVEY

Abdallah Aaraba
Department of Computer Science
University of Sherbrooke
Sherbrooke, QC, Canada
abdallah.aaraba@usherbrooke.ca

Shengrui Wang
Department of Computer Science
University of Sherbrooke
Sherbrooke, QC, Canada
shengrui.wang@usherbrooke.ca

November 16, 2023

ABSTRACT

Self-expressiveness property has become the a key idea for many subspace clustering methods. Indeed, in the past decade, the majority of works base their subspace clustering algorithms on the fact that when there's enough data, one could express a data point with a linear combination of other points lying in the same subspace. Earlier methods took full advantage of this property in order to learn pairwise affinities between data points. However, when the data points are drawn from non-linear manifolds, this property alone is not sufficient for a correct clustering. Motivated with the huge success of deep learning, recent methods have been focusing on the implementation of auto-encoders to non-linearly map raw data into a latent space, in which we hope the subspaces will be linearly clustered through the self-expression property. Experiments on four benchmark datasets show the outperformance of deep subspace clustering methods over shallow ones.

***K*eywords** Subspace Clustering · Unsupervised Learning · Deep Learning

1 Introduction

Numerous image and video processing tasks require dealing with a huge amount of high-dimensional data. In such datasets, there are very high chances to have irrelevant dimensions that do not help in the clustering task, therefore applying basic clustering algorithms on this kind of data will very likely lead to false clusters. To this end, many techniques for finding low-dimensional representations of high-dimensional datasets have been developed. However, the application of conventional dimensionality reduction techniques like Principal Component Analysis (PCA) is doomed to lead to a false clustering when the data is drawn from multiple possibly overlapping subspace. In this case, such data can be approximated by a set of multiple subspaces with low-dimensions, where each cluster corresponds to a subspace. For instance, face images of a subject taken under varying illumination approximately lie in a linear subspace of dimensions up to nine [2], feature trajectories of a rigidly moving object in a video lie in an affine subspace of dimensions up to three [22], and multiple handwritten digit images of a single digit also span a low-dimensional subspace [9]. Thus the need to algorithms that can consider the multi-subspace structure of the data, and cluster the data accordingly. This problem is known as *subspace clustering* [23], which is formally defined as follows.

Problem: Subspace clustering. *Let $X \in \mathbb{R}^{D\times N}$ be the data matrix whose columns represent data points that are drawn from a union of k subspaces of $\mathbb{R}^D$, $\cup_{i=1}^{k}\{S_i\}$, of unknown dimensions $d_i = dim(S_i)$, $d_i \ll D$, for i=1,...,k. The objective of subspace clustering is to segment the columns of the data matrix X into their corresponding subspaces*.

In this paper we will present numerous methods of subspace clustering which we have classified into Three classes: shallow methods [6][16], methods using the kernel trick [18][25], and deep ones [11][30][28] . The difference between the two classes is that methods in the second category use the power of deep neural networks that have showed their potential of handling many tasks such as image classification [13], and regression [14]. Shallow methods use the

original data space to perform clustering; whereas deep ones perform the clustering in a feature space where we hope that the data spaces are clearly distinguishable.

Paper Outline. The remainder of this paper is organized as follows. Section 2 reviews the subspace clustering framework. Section 3 covers shallow subspace clustering methods. Section 4 explores kernel based methods. Section 5 presents deep methods. Section 6 shows experimental results with discussions, and Section 7 concludes the paper.

2 The subspace clustering framework

2.1 Self-expressiveness

Given $\{x_j \in \mathbb{R}^D\}_{j=1}^N$ a collection of points drawn from multiple independent[1] linear subspaces $\{S_i\}_{i=1}^k$. Let $X = [x_1, x_2, ..., x_N]$ be the dataset, and $X_i \in \mathbb{R}^{D \times N_i}$ be the set of the N_i data points drawn from subspace S_i. If we make the assumption that each subspace contains enough data, i.e., $N_i > d_i$ with $\{d_i << D\}_{i=1}^k$ being the unknown dimensions of the k subspaces, and that these data points are in general positions, which means that no d_i points from subspace S_i can be found in a $(d_i - 1)$ dimensional subspace, then our data holds self-expressiveness property. Therefore, one can express a data point drawn from subspace S_i by a linear combination of d_i points of the same subspace. Thus, if x_i is a data point drawn from subspace S_n, then it can be represented by a d_n-sparse vector[2] $c_i \in \mathbb{R}^N$, such that $x_i = Xc_i$, and every entry c_{ij} (where $j \in \{1, 2, ..., N\}$) is set to zero if x_j is not in the same subspace as x_i. So when considering the hole data matrix, one can represent the self-expressiveness property by one single equation, i.e., $X = XC$, where C is the self-representation coefficient matrix.

2.2 The subspace clustering framework

The most popular methods among subspace clustering techniques are the ones that are based on spectral clustering. These methods follow the same framework to achieve the task of subspace clustering. This framework divided into two stages.

- The first stage is finding a good self-representation matrix C that will be used to form an affinity matrix $W = (w_{ij})$ by $W = \frac{1}{2}(|C| + |C^T|)$. W is believed to be a valid representation of the similarity between data points for two reasons. First, if two data points x_i and x_j are not in the same subspace, then C_{ij} has zero as value. Second, if x_i can be written as a linear combination of some points in the same subspace including x_j, then x_j can also be written as a linear combination of some points including x_i (all in the same subspace).
- The second stage consists of applying spectral clustering [24] with the formed affinity matrix W. In this stage, We use W to produce an undirected graph $G = (V, E)$. The set of vertices V contains the N data points of the data matrix X, and there is an edge $(v_i, v_j) \in E$ if $W_{ij} \neq 0$. Thus, W is the adjacency matrix of G. With this setting, the Laplacian matrix of G is defined as $L = D - W$, with $D \in \mathbb{R}^{N \times N}$ being the *degree matrix* of G, which is a diagonal matrix with diagonal terms $D_{ii} = \sum_n W_{in}$. Let k be the number of clusters, we compute the first k eigenvectors $u_1, u_2, ..., u_k$ of the Laplacian matrix L, that will be regrouped as columns of the matrix $U \in \mathbb{R}^{N \times k}$. With $y_i \in \mathbb{R}^k$ being the vector corresponding to the i-th row of U, we apply the k-means clustering algorithm on the set of points $\{y_i\}_{i=1}^N$ to form clusters $C_1, C_2, ..., C_k$, so that in the end we can have a clustering of the original data as $\tilde{C}_i = \{x_j | y_j \in C_i\}$. This methodology of spectral clustering that clusters data by applying k-means on row vectors of the first k eigenvectors of the Laplacian matrix is argued to be valid. And that is because the matrix U is the solution of the relaxed minimization problem

$$\min_{A_1,...,A_k} RatioCut(A_1, A_2, ..., A_k), \tag{1}$$

where $A_1, A_2, ..., A_k$ is a partition of G, $RatioCut(A_1, A_2, ..., A_k) := \frac{1}{2}\sum_{i=1}^k \frac{W(A_i, \bar{A}_i)}{card(A_i)}$, and $W(A_i, \bar{A}_i) = \sum_{i \in A, j \in B} w_{ij}$. We can see that the quantity $RatioCut(A_1, A_2, ..., A_k)$ is an objective function that encodes the fact that we want to find a partition $A_1, A_2, ..., A_k$ of our graph G with reasonably large sets, such that points in different clusters are dissimilar from each other, and points within the same cluster are similar to each other. Subsequently, applying the k-means clustering algorithm on row vectors of the matrix U gives us this wanted partition.

[1] A set of linear subspaces $\{S_i\}_{i=1}^n$ are independent if and only if $\sum_{i=1}^n S_i = \oplus_{i=1}^n S_i$, with $\oplus$ designate the direct sum

[2] $v \in \mathbb{R}^N$ is a d-sparse vector, with d $\in \mathbb{N}$ and $d < N$, if it has at most d non-zero entries.

3 Shallow methods

In this section we will present two of the most popular methods of clustering data in their original space, while taking into consideration the multi-subspace structure of the data. Both of these methods follow the framework described above. The difference between these two techniques is the way of learning the data affinity matrix from the original data space.

3.1 Sparse Subspace Clustering (SSC)

Sparse Subspace Clustering algorithm [6] takes full advantage of the self-expressiveness property of data in order to perform the clustering. This is done by solving a sparse optimization program that is used to produce the self-representation coefficient matrix C, so that we can apply spectral clustering to find the wanted clusters.

3.1.1 Noise free data

Consider the collection of data in hand $X = [x_1, x_2, ..., x_N]$ to be noise free. From the self-expressiveness property, each data point $x_i \in S_l$ can be written as $x_i = Xc_i$. The SSC algorithm add the constraint that the entry c_{ii} should be null so that we don't consider the trivial solution of writing a data point as a linear combination of itself. Hence, the self-representation equation becomes

$$x_i = Xc_i, \quad c_{ii} = 0. \tag{2}$$

As aforementioned in subsection 2.1, there exists a sparse solution for equation (2) that ideally expresses x_i as a linear combination of points that lie in the same subspace as x_i. This is formulated as the following non-convex optimization problem

$$\min ||c_i||_0 \quad subject\ to \quad x_i = Xc_i,\ c_{ii} = 0, \tag{3}$$

as the ℓ_0 norm counts the number of nonzero elements of the solution. Such a problem is known to be a NP-hard problem to solve [1].
To relax this problem, we use instead the ℓ_1 norm to turn the minimization problem into a convex one. It is shown in [6] that when the subspaces are independent, the following ℓ_1 minimization problem

$$\min ||c_i||_1 \quad subject\ to \quad x_i = Xc_i,\ c_{ii} = 0, \tag{4}$$

produces a sparse coefficient vector with non-zero entries corresponding to points in the same subspace as x_i as a solution. Rewriting the sparse minimization problem (4) for all data points in X gives

$$\min ||C||_1 \quad subject\ to \quad X = XC,\ diag(C) = 0, \tag{5}$$

where $diag(C) \in \mathbb{R}^N$ represents the diagonal terms of the matrix C. The above minimization problem can be solved using convex programming tools that promote sparse solutions [5], [4], [21].

3.1.2 The case of noisy data

In the case where data is contaminated by some arbitrary noise Z, i.e., $X = XC + Z$, this can be handled by setting the optimization problem as

$$\min ||C||_1 + \lambda ||X - XC||_F^2 \quad subject\ to \quad diag(C) = 0, \tag{6}$$

where $||.||_F$ designate the Frobenius norm.
The above optimization problem can be solved efficiently by using standard alternating direction method of multipliers (ADMM) [6].

3.2 Low Rank Representation (LRR)

Low Rank Representation [16] is an algorithm that seeks a better capture of the global structure of data in order to get a robust subspace clustering. It does so by looking for the lowest-rank representation of all the data points jointly, unlike the SSC algorithm that finds the sparsest representation of each data point individually.
A fundamental assumption of the LRR algorithm is that the subspaces from which the data is drawn should be independent. Such an assumption is believed to be mild, since the high dimensionality of data usually assert the satisfaction of this assumption.

3.2.1 Low-Rank Representation Problem

Given a set of data points $X = [x_1, x_2, ..., x_N]$ where $x_i \in \mathbb{R}^D$. Let $A = [a_1, a_2, ..., a_m]$ be a "dictionary", such that each point x_i can be represented by a linear combination of its columns

$$X = AC, \tag{7}$$

with $C = [c_1, c_2, ..., c_N]$ being the coefficient matrix, where each c_i is the representation of x_i in the dictionary A. Often, the dictionary is overcomplete which leads to multiple possible solutions for equation (7). As we said, this method seeks the lowest representation of data jointly, thus, we want a coefficient matrix C that solves the following problem

$$\min_C \ rank(C) \quad subject\ to \ \ X = AC. \tag{8}$$

Given the discrete property of the above problem, this minimization is difficult to solve. Therefore, we relax the above optimization problem following the suggestion of [3] and [12] by turning it into the following convex minimization

$$\min_C ||C||_* \quad subject\ to \ \ X = AC, \tag{9}$$

where $||.||_*$ designate the nuclear norm [7].

3.2.2 Subspace Clustering using LRR

Since we need an affinity matrix that presents the pairwise similarities between data points, we use the data matrix X itself as the dictionary. Therefore, the problem becomes

$$\min_C ||C||_* \quad subject\ to \ \ X = XC. \tag{10}$$

Unlike the SSC method which prohibits data points from representing themselves in order to eliminate trivial solutions, a data point can represent itself in LRR. This guaranties the existence of feasible solutions for (10) even when the data sampling is insufficient ($N_i < d_i$).
It is shown in [16] that under the assumption of the sufficiency of the sampling of data, as well as the independence of the subspaces, there exists an optimal solution C^* to (10) in which only points in the same subspaces have non-zero entries in the coefficient matrix, i.e., $C^*_{ij} \neq 0 \quad iff \quad x_i$ and x_j are drawn from the same subspace.

3.2.3 Robustness to Noise and Outliers

Usually, in real world applications data is often contaminated with noise. Thus, in the case where a fraction of the data is grossly corrupted, a reasonable way to handle this would be setting the following objective

$$\min_{C,E} ||C||_* + \lambda||E||_{2,1} \quad subject\ to \ \ X = XC + E, \tag{11}$$

where $||E||_{2,1} = \sum_{j=1}^{N} \sqrt{\sum_{i=1}^{N}(E_{ij})^2}$ is referred to as the $\ell_{2,1}$-norm, and $\lambda > 0$ is a balancing parameter that can be chosen empirically. The $\ell_{2,1}$-norm is used in order to promote the fact that some data points are contaminated, and some others are clean.

4 Methods using the Kernel trick

While standard SSC and LRR methods have done a good job clustering data drawn from multiple subspaces in a linear way (the affinity matrix is found from the original space of the data), in practice non-linear manifolds represent a better modeling for many datasets. The kernel trick has shown good theoretical and empirical results in handling non-linear manifolds. Therefore, kernel Sparse Subspace Clustering (KSSC) [18], and Robust Kernel Low-Rank Representation (RKLRR) [25], are two methods that combine the ability of the SSC and LRR algorithms in extracting good affinity matrices while taking into consideration the multi-subspace structure of the data, with the capacity of the non-linear mapping of the kernel trick in grouping the data with the same distribution and making them linearly separable.

4.1 Kernel SSC (KSSC)

Let $\Phi : \mathbb{R}^D \rightarrow \mathcal{H}$ be a non-linear mapping from the input space to the reproducing kernel Hilbert space $\mathcal{H}$, and $\Phi(X) = [\Phi(x_1), \Phi(x_2), ..., \Phi(x_N)]$. The non-linear version of the optimization problem in (5) can be formulated as

$$\min ||C||_1 \quad subject\ to \quad \Phi(X) = \Phi(X)\ C,\ diag(C) = 0. \tag{12}$$

In the case of noisy data, the non-linear version of (6) is

$$\min ||C||_1 + \lambda||\Phi(X) - \Phi(X)\ C||_F^2 \quad subject\ to \quad diag(C) = 0. \tag{13}$$

The above problem can be solved efficiently using the ADDM method as shown in [6].

4.2 Robust Kernel LRR (RKLRR)

Following the same steps and notations used in the above subsection, we can formulate the non-linear version of LRR, by turning the equation (10) into

$$\min_C ||C||_* \quad subject\ to\ \ \Phi(X) = \Phi(X)C. \tag{14}$$

When the data is noisy, we will instead work with the non-linear version of (11) as

$$\min_{C,E} ||C||_* + \lambda||E||_{2,1} \quad subject\ to\ \ \Phi(X) = \Phi(X)C + E, \tag{15}$$

The solution for the above program is found using the ADM method [15][27] as shown in [25].

5 Deep Methods

Although Kernel extensions of SSC and LRR seem optimistic, there is no clear reason why the data in the latent feature space produced by the non-linear kernel mapping should lie in low-dimensional subspaces.
Auto-Encoders (AEs) however have shown a great capacity to explicitly transform data into a latent space. Usually the latent space is set to have lower dimensions than the original one, which help reducing the dimensionality of data [10]. More specifically, the convolutional version of AEs has been proven to be a very powerful tool when the task is to find latent space for images, and it is applied to find embedded features and to initialize convolutional neural networks (CNNs) [26].
In this section, we will present some popular works, in which deep AEs are implemented in order to better capture the multi-subspace structure of the data.

5.1 Deep Subspace Clustering Networks (DSC-Nets)

In DSC-Nets [11], deep auto-encoders, are used to non-linearly map the input data into a latent space in which we hope data will be easily clustered. The key idea of this method is that the self-expression coefficient matrix is extracted in a simple but effective way through back-propagation mechanism. This is done by introducing a self-expression layer as part of the neural network between the encoder and decoder to imitate the self-expressiveness property of data, which has been proven effective when data points lie perfectly in multiple independent subspaces.

5.1.1 Self-Expression Layer in Deep AEs

The problem of subspace clustering in traditional methods can be formulated as the following optimization problem which takes into account data corruptions

$$\min ||C||_\ell + \frac{\lambda}{2}||X - XC||_F^2 \quad s.t. \quad (diag(C) = 0), \tag{16}$$

where $||.||_\ell$ is a properly chosen regularization matrix norm, and the constraint $diag(C) = 0$ is optional to prevent trivial solutions when sparsity inducing norms are used, such as the ℓ_1 norm, and λ is a balancing parameter.
However, the self-expressiveness property alone is not enough when the data is drawn from non-linear manifolds. Therefore, we need to explicitly learn a non-linear mapping that eases the separation of subspaces. To this end this method proposes to build a network upon auto-encoders with a self-expressive layer as shown in Figure 1. As a direct consequence of the introduction of such a self-expression layer between the encoder and decoder layers, the auto-encoder is trained to capture the multi-space structure of the data.

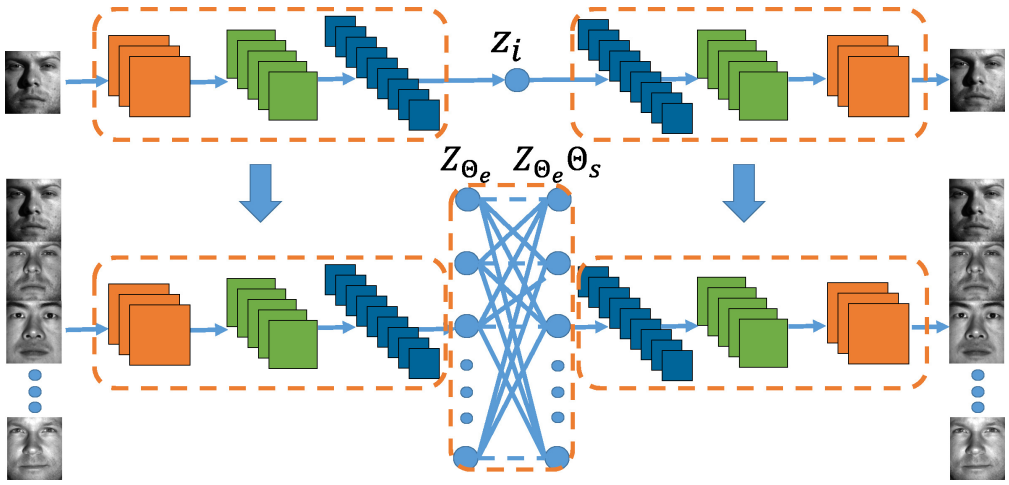

Figure 1: DSC-Nets: we show an example of a deep subspace clustering network with convolutional-deconvolutional in the auto-encoder layers, separated with one self-expressive layer.

To formulate the optimization objective of our network, let Θ be the auto-encoder parameters, which we can divide into encoder parameters Θ_e and decoder parameters Θ_d. Additionally, the output of the encoder denoted Z_{Θ_e} represents the projection of the data matrix X into the latent space. The following objective function is set to encode the self-expressiveness within the network

$$L(\tilde{\Theta}) = \frac{1}{2}||X - \hat{X}_{\tilde{\Theta}}||_F^2 + \lambda_1||\Theta_s||_\ell + \frac{\lambda_2}{2}||X - X\Theta_s||_F^2 \quad s.t. \quad (diag(C) = 0), \tag{17}$$

where $\hat{X}_{\tilde{\Theta}}$ denotes the reconstruction of the input data by the auto-encoder, and Θ_s is the parameters of the self-expressive layer that play the role of the self-representation coefficient matrix (C). $\tilde{\Theta}$ designates all the network parameters, which consists of encoder parameters Θ_e, self-expressive layer parameters Θ_s, as well as the decoder parameters Θ_d. As it is shown in Figure 1, the self-representation matrix is represented with a fully connected layer with no non-linear activation.
With this setting, the minimization of the loss function $L(\tilde{\Theta})$ in (17) is easily done through back propagation mechanism.

5.1.2 Training Strategy

As illustrated in Figure 1, the training of the network is done in two steps:

1. In the first step, we initialize the network by pre-training the deep auto-encoder and discarding the self-expression layer.
2. In the second step, we fine-tune the hole network parameters with a gradient descent algorithm.

After the network is trained, we use the parameters of the self-expression layer Θ_s to form the affinity matrix for spectral clustering.

5.2 Deep Adversarial Subspace Clustering(DASC)

Although DSC-Nets implement a deep auto-encoder to avoid the linear subspace assumption of data, it is not able to fine-tune the learned self-representation coefficient matrix once the clustering is done. In contrast, the DASC [30] method takes advantage of the current clustering in order to enhance the clustering results by learning better representations through an adversarial learning [GAN]. It consists of a Generator that performs the clustering, and a subspace wise discriminator. This method is inspired by the key property of a linear subspace, which states that linearly combining samples of a valid linear subspace would not produce samples lying outside this latter. Subsequently, after the generator determines the clustering and generates real data (data that is strongly considered to lie in the cluster's intrinsic subspace) as well as fake data (linear combination of all the data in a single cluster), the discriminator evaluates the generator's clustering quality by looking whether the produced clusters are homogeneous. That is, if the clustering made by the generator is correct, the samples within the same cluster would span fake samples that lie in the same subspace as the original ones. Thus, the discriminator would not be able to distinguish between real and fake data.

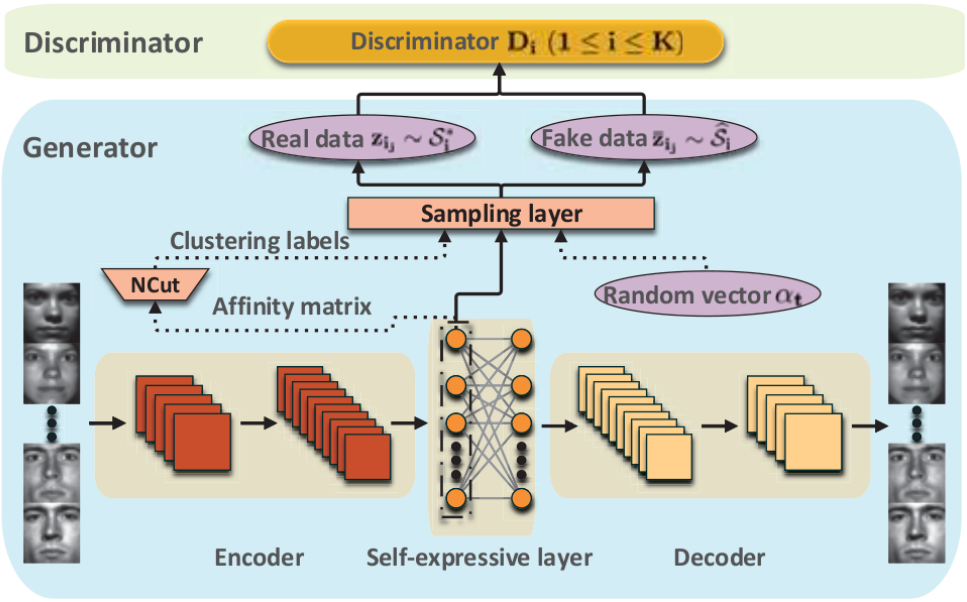

Figure 2: Illustration of the global structure of DASC.

Otherwise, when the clusters are inaccurate, the original and re-sampled data would lie in different subspaces. Then the discriminator can distinguish between the two types of samples, and feeds back this information to the generator to improve the sample clustering and subspace estimation.

5.2.1 Formulation and Network Architecture

1. GENERATOR: As shown in Figure 2, a deep auto-encoder is used in the generator to project each sample x_i into its latent representation z_i. Following the same architecture of DSC-Nets, DASC introduces a self-expression layer between the encoder and the decoder to generate the self-representation coefficient matrix Θ_c. Then the affinity matrix $A = \frac{1}{2}(|\Theta_c| + |\Theta_c|^T)$ is used for spectral clustering.
   In addition to the above, the generator has another task which is the generation of "real" and "fake" data according to each cluster C_i $(i = 1, ..., k)$, which is done by the sampling layer. The NCut component in the generator is responsible for the application of the NCut algorithm [20] of spectral clustering on the formed affinity matrix A to cluster the representations in the latent space z_i into k clusters C_i. Meanwhile, the discriminator is charged to learn a linear subspace S_i that fits the inherent ground-truth subspace S_i^* of cluster C_i. This estimated subspace S_i is then used by the discriminator to distinguish between real and fake samples through their projection residuals on S_i. Subsequently, the generator selects $\bar{m}_i^*$ "real" samples with small residuals through its sampling layer. The data within the cluster $C_i = \{z_{i_j}\}_{j=1}^{m_i}$ is used to span a subspace $\hat{S}_i = span(C_i)$ in order to generate $\bar{m}_i^*$ fake samples. To do so, the sampling layer first samples from the uniform distribution within $(0, 1]$, $\bar{m}_i^*$ random vectors $\alpha_t \in \mathbb{R}^{m_i}$, and then produces $\bar{m}_i^*$ fake samples as $\bar{z}_t = \sum_{j=1}^{m_i} \alpha_t z_{i_j}$ $(t = 1, ..., \bar{m}_i^*)$.
2. DISCRIMINATOR: To check for a cluster C_i whether its real z_i and fake $\bar{z}_i$ data lie in the same inherent subspace, the discriminator learns the basis $U_i \in \mathbb{R}^{d \times r_i}$ of the subspace S_i that is close enough to the intrinsic one S_i^*, where d is the dimension of the latent space (i.e., $z_i \in \mathbb{R}^d$), and r_i designate the subspace dimension. Hence, the discriminator does the distinguishing between real and fake data by their projection residuals onto S_i as
$$\mathcal{L}_r(z_{i_j}) = ||z_{i_j} - U_i U_i^T z_{i_j}||_2^2, \tag{18}$$
since fake samples will be farther from the subspace than real ones. These above projections are used to express the output of the discriminator as $D_i(z_{i_j}) = \mathbb{P}(z_{i_j} \in S_i) = exp(-\mathcal{L}_r(z_{i_j}))$, where D_i is the discriminator related to the cluster C_i. With this expression, we can formulate the objective function of the discriminator D_i to train on as:
$$\min_{U_i} \mathcal{L}_{D_i} := \frac{1}{\bar{m}_i^*} \sum_{j=1}^{\bar{m}_i^*} \mathcal{L}_r(z_{i_j}) + max(0, \epsilon - L_r(\bar{z}_{i_j})), \tag{19}$$

where $\epsilon > 0$ is a small margin parameter, as done in [29].
From the above objective function of D_i, we can formulate the objective function of the hole discriminator as the following:

$$\min_{U_1,\dots,U_k} \mathcal{L}_D := \frac{1}{k}\sum_{i=1}^{k} \mathcal{L}_{D_i}, \tag{20}$$

where the columns of every basis U_i obey $||U_{i,j}||2^2 = 1$. The following regularization terms $R_1 = \beta_1 \sum_{i\neq j} ||U_i^T U_j||_F^2$ and $R_2 = \beta_2 \sum_{i=1}^{k} ||U_i^T U_i - I||_F^2$ are used in order to enforce subspaces to be separable, and to reduce the redundancy in the bases U_i respectively (β_1 and β_2 are balancing constants). Thus, the final training objective for the discriminator is

$$\min_{U_1,\dots,U_k} \mathcal{L}_D + R_1 + R_2. \tag{21}$$

We can implement the discriminator in DASC by k neural networks of two fully connected layers that share their parameters U_i. Thus each discriminator D_i takes as input z_{ij}, the output of the first layer is $U_i z_{ij}$, and for of the second layer is $U_i U_i^T z_{ij}$.

5.2.2 Training Strategy

The training objective for the generator is as follows:

$$\min_{\Theta} \mathcal{L}_a + \lambda_1 ||X - \hat{X}_\Theta||_F^2 + \lambda_2 ||Z - Z\Theta_c||_F^2 + \lambda_3 ||\Theta_c||_1, \tag{22}$$

where λ_1, λ_2, and λ_3 are balancing parameters, Θ denotes the parameters of the generator, $\hat{X}_\Theta$ is the reconstruction of the data matrix X, and $\mathcal{L}_a = \frac{1}{k}\sum_{i=1}^{k} \frac{1}{\bar{m}_i^*} \sum_{j=1}^{\bar{m}_i^*} \mathcal{L}_r(\bar{z}_{i_j})$. Minimizing the adversarial term $\mathcal{L}_a$ promotes the fact that the produced fake data should be closer to the learned subspace.
The training of this network is done in two phases:

1. In the first phase, we pre-train only the generator while discarding the discriminator, and without considering the adversarial loss $\mathcal{L}_a$.
2. In the second phase, and before training the hole network, we initialize the parameters of the discriminator $U_i \in \mathbb{R}^{d\times r_i}$ by applying the QR decomposition on the representations z_i of the cluster C_i. The number of real and fake samples $\bar{m}_i^*$ is set for every cluster C_i as $\bar{m}_i^* = \alpha m_i$, where $card(C_i) = m_i$,

5.3 Self-Supervised Convolutional Subspace Clustering Network (S2ConvSCN)

Following the same path as DASC, S2ConvSCN [28] takes advantage of current clustering results to improve the feature learning as well as the self-expression coefficient matrix. However, one of the major drawbacks of DASC, is that it requires the knowledge of subspaces' dimensions in order for the discriminator to learn the basis of each subspace, which is usually unknown.

5.3.1 Network formulation

As shown in Figure 3, the network is composed of a feature extraction module, a self-expression module, and self-supervision modules for improving the previous two ones.

1. FEATURE EXTRACTION MODULE: This is basically a deep convolutional auto-encoder that non-linearly maps the input data x_i into a latent representation z_i. This is done through the reconstruction loss function:

$$\mathcal{L}_0 = \frac{1}{2N}||X - \hat{X}||_F^2. \tag{23}$$

2. SELF-EXPRESSION MODULE: Once again, the self-expression layer is introduced between the encoder and decoder to mimic the self-expressiveness of data in the latent space as in DSC-Nets:

$$\lambda||C||_\ell + \frac{1}{2}||Z - ZC||_F^2 \quad s.t. \quad diag(C) = 0, \tag{24}$$

where $\lambda > 0$ is a tradeoff parameter.

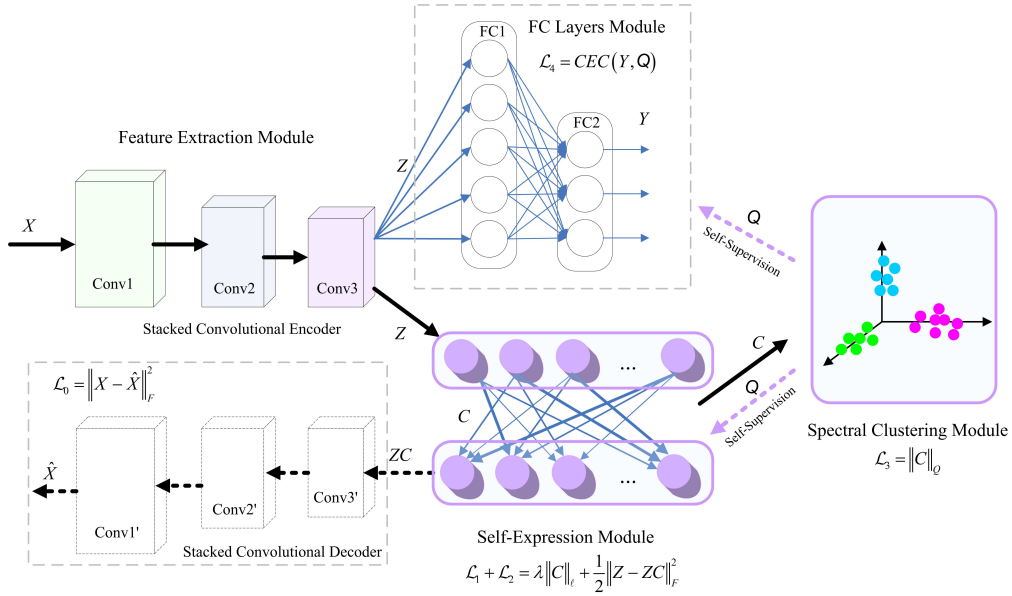

Figure 3: Architecture of S2ConvSCN

3. SELF-SUPERVISION MODULES: After obtaining the self-representation matrix X, the data affinity matrix is calculated as $A = \frac{1}{2}(|C| + |C^T|)$. Thus, we can apply spectral clustering on A to segment the data by solving the following relaxed program:

$$\min_{Q} \sum_{i,j} a_{ij}||q_i - q_j||_2^2, \quad s.t. \quad QQ^T = I. \tag{25}$$

This way, the output of spectral clustering gives a meaningful labeling of the data which represents a good supervision for both the feature extraction and self-expression modules. This supervision is done as follows:

Self-Supervision for self-expression: The labeling produced by spectral clustering is used to supervise the self-expression module as:

$$\frac{1}{2}\sum_{i,j} a_{ij}||q_i - q_j||_2^2 = \sum_{i,j} c_{ij}\frac{||q_i - q_j||_2^2}{2} := ||C||_Q. \tag{26}$$

After the labeling Q is produced, minimizing this objective enforces c_{ij} to be close to zero when data points x_i and x_j have distant labeling. Thus the refinement of the self-expression coefficient matrix C.

Self-Supervision for Feature-extraction: The key idea for the supervision of the feature-extraction module states that, with just the learned features z_i we should be able to predict the label q_i through a two layered Fully Connected (FC) network designed as $d \times M_1 \times M_2 \times k$, where $z_i \in d$, and M_1 and M_2 are the numbers of neurons in the two layers in the FC network respectively.

Let $y_i \in \mathbb{R}^k$ be the output of the FC network. With the results $\{q_j\}_{j=1}^N$ of the spectral clustering module being the target of the FC network, a mixture of *cross-entropy loss* and *center loss* is defined for the self-supervision of this module as

$$\mathcal{L}_4 = \frac{1}{N}\sum_{j=1}^{N}(\ln(1 + e^{-\tilde{y}_j q_j}) + \tau||y_j - \mu_{\pi(y_j)}||_2^2), \tag{27}$$

where $\tilde{y}_j$ is the softmax normalization of y_j, $\mu_{\pi(y_j)}$ designates the center of the cluster corresponding to y_j, $\pi(y_j)$ takes as value the index of y_j from the output of spectral clustering, and $\tau \in [0, 1]$ is a balancing parameter. The second term of$\mathcal{L}_4$ is there to compress variations inside clusters.

5.3.2 The Training of S2ConvSCN

The final objective function of S2ConvSCN is as follows:

$$\mathcal{L} = \mathcal{L}_0 + \gamma_1\mathcal{L}_1 + \gamma_2\mathcal{L}_2 + \gamma_3\mathcal{L}_3 + \gamma_4\mathcal{L}_4, \tag{28}$$

where $\{\gamma_n > 0\}_{n=1}^4$ are tradeoff parameters, $\mathcal{L}_1 = ||C||_1$, $\mathcal{L}_2 = \frac{1}{2}||Z - ZC||_F^2$, $\mathcal{L}_3 = ||C||_Q$. The training is done following a two steps strategy:

(a) COIL 20/100 (b) ORL (c) Extended Yale B

Figure 4: Samples from COIL 20/100, ORL, and Extended Yale B.

	S2ConvSCN		DSC-Nets		DASC	
Layers	kernel size	channels	kernel size	channels	kernel size	channels
encoder-1	5×5	10	5×5	10	5×5	10
encoder-2	3×3	20	3×3	20	3×3	20
encoder-3	3×3	30	3×3	30	3×3	30
decoder-1	3×3	30	3×3	30	3×3	30
decoder-2	3×3	20	3×3	20	3×3	20
decoder-3	5×5	10	5×5	10	5×5	10

Table 1: The settings of deep methods for the Extended Yale B dataset.

1. In the first step, we pretrain the network using only the $\mathcal{L}_0$ term, while discarding all the other terms in the global objective, as well as ignoring the FC network and the spectral clustering module.
2. In the second step, we train the hole network with all the terms in the global loss function.

6 Experiments

The evaluation of the performance of the seven presented methods is done through experiments on four datasets: the ORL and Extended Yale B face image datasets, and two object image datasets COIL 20/100.
For a fair comparison, the settings of each algorithm are identical to its original paper.
The metric that has been used for the evaluation is the percentage of the clustering error:

$$clustering\ error = \frac{number\ of\ misclassified\ data\ points}{number\ of\ total\ points}\ \%. \tag{29}$$

The activation function used in the three deep methods is the Rectified Linear Unit (ReLU) [13]. For the FC layers in S2ConvSCN, M_1 and M_2 are set to $\frac{N}{2}$ and k respectively. For the DASC method, the following parameters are fixed for all the experiments as $\lambda_1 = 0.5$, $\beta_1 = \beta_2 = 0.01$, and $\epsilon = 0.1$. Both DSC-Nets and S2ConvSCN use 1.0×10^{-3} as a value for learning rate. Whereas DASC, uses a learning rate of the value of 1.0×10^{-3} in the pretraining stage, a value of 2.0×10^{-4} in the fine-tuning stage. Empirical Results are taken from [28].

6.1 Extended Yale B Dataset

The Extended Yale B [8] is a well known benchmark dataset for subspace clustering. It includes 38 subjects, with 64 face images per subject obtained under different lightning conditions (see Figure 4(c)). In these experiments, the face images are down-sampled from 92×168 to 42×42 pixels. Experiments are done using all combinations of $k \in \{10, 15, 20, 25, 30, 35, 38\}$.
For this dataset, the settings of S2ConvSCN, DSC-Nets, and the generator in DASC are the same, as shown in Table 1. The balancing parameters for S2ConvSCN have the following values $\gamma_1 = 1$, $\gamma_2 = 1.0 \times 10^{\frac{k}{10}-3}$, $\gamma_3 = 16$, and $\gamma_4 = 72$. For DSC-Nets, the regularization parameters are set to $\lambda_1 = 1$, and $\lambda_2 = 1.0 \times 10^{\frac{k}{10}-3}$. In DASC, the dimensions of U_i used in the discriminator in DASC are set to $r_i = 10$, $\lambda_3 = 1$, $\lambda_2 = 3$.
The clustering performance results are shown in Table 2. From these results, we see that the clustering error of shallow

Method	LRR	SSC	KSSC	DSC-Net-L1	DSC-Net-L2	S2ConvSCN-L2	S2ConvSCN-L2	DASC
10 subjects								
Mean	19.76	8.80	14.49	2.23	1.59	**1.18**	**1.18**	-
Median	18.91	9.06	15.78	2.03	1.25	**1.09**	**1.09**	-
15 subjects								
Mean	25.82	12.89	16.22	2.17	1.69	1.14	**1.12**	-
Median	26.30	13.23	17.34	2.03	1.72	**1.14**	**1.14**	-
20 subjects								
Mean	31.45	20.11	16.55	2.17	1.73	1.31	**1.30**	-
Median	32.11	21.41	17.34	2.11	1.80	1.32	**1.25**	-
25 subjects								
Mean	28.14	26.30	18.56	2.53	1.75	1.32	**1.29**	-
Median	28.22	26.56	18.03	2.19	1.81	1.34	**1.28**	-
30 subjects								
Mean	38.59	27.52	20.49	2.63	2.07	1.71	**1.67**	-
Median	36.98	27.97	20.94	2.81	2.19	1.77	**1.72**	-
35 subjects								
Mean	40.61	29.19	26.07	3.09	2.65	1.67	**1.62**	-
Median	40.71	29.51	25.92	3.10	2.64	1.69	**1.60**	-
38 subjects								
Mean	35.12	29.36	27.75	3.33	2.67	1.56	1.52	**1.44**
Median	35.12	29.36	27.75	3.33	2.67	1.56	1.52	**1.44**

Table 2: Clustering error (in %) on Extended Yale B. The best results are market in bold

	S2ConvSCN		DSC-Nets		DASC	
Layers	kernel size	channels	kernel size	channels	kernel size	channels
encoder-1	3×3	3	5×5	5	5×5	5
encoder-2	3×3	3	3×3	3	3×3	3
encoder-3	3×3	5	3×3	3	3×3	3
decoder-1	3×3	5	3×3	3	3×3	3
decoder-2	3×3	3	3×3	3	3×3	3
decoder-3	3×3	3	5×5	5	5×5	50

Table 3: The settings of deep methods for the ORL dataset.

methods and KSSC dramatically increases as the number of clusters (subjects) increases. On the other hand, deep methods show a stability of performance w.r.t. the number of clusters. From this we can say that deep clustering methods are much better than the shallow and kernel methods on the Extended Yale B dataset.

6.2 ORL Dataset

The ORL dataset [19] contains images of 40 subjects, each of which is represented with 10 facial images taken under varying illuminations (Figure 4(b)). In this dataset, the subspace clustering problem is more challenging because of the small number of samples per subject, as well as the fact that images are taken in different facial expressions which introduces more non-linearity to the subspaces. Images are down-sampled from 112×92 to 32×32.
The settings of S2ConvSCN, DSC-Nets, and the generator in DASC for this dataset can be found in Table 3. For S2ConvSCN, γ_1, γ_2, γ_3, and γ_4 are set to 0.1, 0.01, 8, and 1.2 respectively. For DSC-Nets, the balancing parameters are set to $\lambda_1 = 1$ and $\lambda_2 = 0.2$. Once again, the dimensions of the bases U_i are set to $r_i = 10$ in DASC, and the regularization parameters are set to $\lambda_3 = 1$ and $\lambda_2 = 0.1$.
Clustering results are shown in Table 4. We can see from this Table that the clustering error results for all competing methods are worse than on Extended Yale B due to the small number of samples per subject. Yet again, deep methods outperform significantly shallow ones.

Methods	ORL	COIL20	COIL100
LRR	33.50	30.21	53.18
SSC	29.50	14.83	44.90
KSSC	34.25	24.65	47.18
DSC-Net-L1	14.25	5.65	33.62
DSC-Net-L2	14.00	5.42	30.96
S2ConvSCN-L1	11.25	2.33	27.83
S2ConvSCN-L2	**10.50**	**2.14**	**26.67**
DASC	11.75	3.61	-

Table 4: Clustering error (%) on ORL, COIL20, COIL100

		S2ConvSCN		DSC-Nets		DASC	
	Layers	kernel size	channels	kernel size	channels	kernel size	channels
COIL20	encoder-1	3×3	15	3×3	15	3×3	15
	decoder-1	3×3	15	3×3	15	3×3	15
COIL100	encoder-1	5×5	50	5×5	50		
	decoder-11	5×5	50	5×5	50		

Table 5: The settings of deep methods for COIL datasets.

6.3 COIL20 and COIL100 Datasets

Here, the competing methods are evaluated on a more challenging clustering task, which is object clustering on COIL object image datasets [17]. COIL20 consists of 1440 gray-scale image samples of 20 objects; while COIL100 consists of 7200 images of 100 objects. What makes these two datasets more challenging than the former ones, is that images of the same object are taken from different angles, which makes them unaligned and have less similar structure in contrast with human face datasets. once again, images are down-sampled to 32×32.
The settings of the deep methods are shown in Table 5. For balancing parameters of S2ConvSCN on COIL20, they are set to $\gamma_1 = 1$, $\gamma_2 = 30$, $\gamma_3 = 8$, and $\gamma_4 = 6$. For COIL100, they are set to $\gamma_1 = 1$, $\gamma_2 = 30$, $\gamma_3 = 8$, and $\gamma_4 = 7$. As for DSC-Nets, the regularization parameters are set to $\lambda_1 = 1$ and $\lambda_2 = 30$ for both COIL20 and COIL100 datasets. In DASC, the dimensions of U_i used in the discriminator are set to $r_i = 30$, and the balancing parameter take the values of $\lambda_2 = 15$ and $\lambda_3 = 1$ for both datasets.
Performance results are shown in Table 4. Once again, and specially on the COIL100 dataset, all the compared methods perform in badly, which is due to the challenging task of object clustering when there is not much samples per object for learning. Deep methods prove anew that they outperform shallow ones.

7 Conclusion

We have provided a review of the progress made in the subspace clustering task, in which we presented a number of popular methods. While shallow ones perform the clustering under the assumption that the subspaces are linearly separable, the deep ones take advantage of neural networks' power in order to non-linearly extract the self-expression coefficients to form the affinity matrix for spectral clustering. As shown empirically, deep clustering methods, that implement convolutional auto-encoders to project data into a latent space, outperforms shallow ones. However, among these deep algorithms, the ones that use current clustering results to refine the self-expression matrix and fine-tune the extracted features are taking the lead recently. Finally, in our opinion we will see int the coming years more deep methods that uses current clustering as self-supervision to improve feature extraction and clustering results.

References

[1] E. Amaldi, V. Kann, et al. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1):237–260, 1998.

[2] R. Basri and D. W. Jacobs. Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):218–233, 2003.

[3] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717, 2009.

[4] E. J. Candes and T. Tao. Decoding by linear programming. *IEEE transactions on information theory*, 51(12):4203–4215, 2005.

[5] D. L. Donoho. For most large underdetermined systems of linear equations the minimal 1-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(6):797–829, 2006.

[6] 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.

[7] S. M. Fazel. Matrix rank minimization with applications. 2003.

[8] 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.

[9] T. Hastie and P. Simard. Metrics and models for handwritten character recognition. In *Conference on Statistical Science Honouring the Bicentennial of Stefano Franscini's Birth*, pages 203–219. Springer, 1997.

[10] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

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

[12] R. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries. In *Advances in neural information processing systems*, pages 952–960, 2009.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[14] S. Lathuilière, P. Mesejo, X. Alameda-Pineda, and R. Horaud. A comprehensive analysis of deep regression. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[15] Z. Lin, R. Liu, and Z. Su. Linearized alternating direction method with adaptive penalty for low-rank representation. In *Advances in neural information processing systems*, pages 612–620, 2011.

[16] G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 663–670, 2010.

[17] S. A. Nene, S. K. Nayar, H. Murase, et al. Columbia object image library (coil-100). 1996.

[18] V. M. Patel and R. Vidal. Kernel sparse subspace clustering. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 2849–2853, 2014.

[19] F. S. Samaria and A. C. Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of 1994 IEEE workshop on applications of computer vision*, pages 138–142. IEEE, 1994.

[20] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[21] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[22] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International journal of computer vision*, 9(2):137–154, 1992.

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

[24] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[25] 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, 2015.

[26] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016.

[27] J. Yang and Y. Zhang. Alternating direction algorithms for \ell_1-problems in compressive sensing. *SIAM journal on scientific computing*, 33(1):250–278, 2011.

[28] J. Zhang, C.-G. Li, C. You, X. Qi, H. Zhang, J. Guo, and Z. Lin. Self-supervised convolutional subspace clustering network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5473–5482, 2019.

[29] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

[30] P. Zhou, Y. Hou, and J. Feng. Deep adversarial subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1596–1604, 2018.