# Projective Low-rank Subspace Clustering via Learning Deep Encoder

**Jun Li**[1], **Hongfu Liu**[1], **Handong Zhao**[1] and **Yun Fu**[1,2]
[1]Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, 02115, USA.
[2]College of Computer and Information Science, Northeastern University, Boston, MA, 02115, USA.
junl.mldl@gmail.com, {liu.hongf,hdzhao,yunfu}@ece.neu.edu

## Abstract

Low-rank subspace clustering (LRSC) has been considered as the state-of-the-art method on small datasets. LRSC constructs a desired similarity graph by low-rank representation (LRR), and employs a spectral clustering to segment the data samples. However, effectively applying LRSC into clustering big data becomes a challenge because both LRR and spectral clustering suffer from high computational cost. To address this challenge, we create a projective low-rank subspace clustering (PLrSC) scheme for large scale clustering problem. First, a small dataset is randomly sampled from big dataset. Second, our proposed predictive low-rank decomposition (PLD) is applied to train a deep encoder by using the small dataset, and the deep encoder is used to fast compute the low-rank representations of all data samples. Third, fast spectral clustering is employed to segment the representations. As a non-trivial contribution, we theoretically prove the deep encoder can universally approximate to the exact (or bounded) recovery of the row space. Experiments verify that our scheme outperforms the related methods on large scale datasets in a small amount of time. We achieve the state-of-art clustering accuracy by 95.8% on MNIST using scattering convolution features.

## 1 Introduction

Low-rank subspace clustering (LRSC) has become an important topic because of its impressive performance in many machine learning and computer vision applications, for example, image clustering [Zhang *et al.*, 2015], motion segmentation [Wang *et al.*, 2015; Zhao and Fu, 2015], and dictionary learning [Ding *et al.*, 2016]. The underlying assumption is that data points sampled from multiple high-dimensional subspaces can be well represented by a union of low-dimensional subspaces. It leads to a useful *self-expressiveness* property that each data point can be efficiently reconstructed by a combination of other data points [Elhamifar and Vidal, 2013; Ding *et al.*, 2015]. According to this property, LRSC is to learn low-rank representations for dividing the data samples into their respective subspaces. Hearteningly, many LRSC methods [Li and Fu, 2015; Shen *et al.*, 2016] have been used to achieve the state-of-the-art results on small datasets. While as the data size grows, a critical problem comes, i.e. how to effectively apply LRSC to cluster big data.

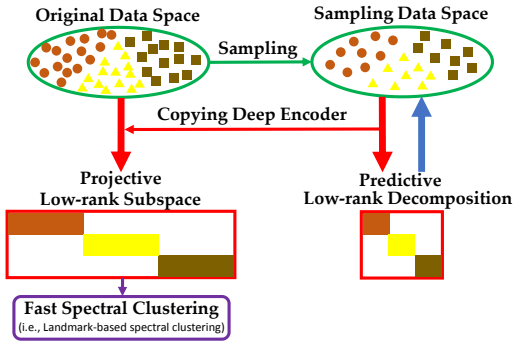



Figure 1: Projective low-rank subspace clustering by predictive low-rank decomposition.

Recall that the classical paradigm, LRSC consists of two steps. In the first step, the desired similarity graph is constructed by learning the lowest-rank representations of all data. The representations are obtained by using the low-rank representation (LRR) methods, such as robust principal component analysis (RPCA) [Candés *et al.*, 2011], latent LRR (latLRR) [Liu and Yan, 2011], and distributed LRR [Talwalkar *et al.*, 2013]. Based on the self-expressiveness property, LRR and its variants usually select all data matrix as a dictionary. *When facing a big dataset, this leads that LRR fails to work on a single machine with limited resource because of the high time and space complexity of the pseudoinverse and singular value decomposition (SVD).* In order to reduce the number size of the dictionary, a small dataset sampled from the big dataset is regarded as a small dictionary [Wang *et al.*, 2014; Peng *et al.*, 2016]. However, it still takes more time to compute the codes (or representations) [Lin *et al.*, 2011].

In the second step, spectral clustering (such as Normalized Cuts (NCut) [Shi and Malik, 2000]) is applied to the similarity graph for segmenting the data samples. *However, it is a difficult problem to apply spectral clustering to big data tasks because of its computational complexity of* $\mathcal{O}(n^3)$*, where* $n$ *is the number of samples* [Cai and Chen, 2015]. Fortunately, fast spectral clustering (FSC) [Cai and Chen, 2015] can carry out the spectral clustering in *linear* time by sparsifying the similarity matrix. Scalable LRR (SLRR) [Peng *et al.*, 2016]

and selection+LRR (selLRR) [Wang *et al.*, 2014] select a small dataset to preform clustering to build a collaborative representation based classifier (CRC) or a linear classifier, respectively. Although these methods can deal with the big data clustering problem, they directly use the original data as the representations to lead to poor clustering results.

In light of the above arguments, we provide an efficient strategy to solve the big data clustering problem. To reduce the expensive computing time, we first select a small dataset from the large dataset for building a small LRR model, and then use the small dataset to learn a deep encoder to quickly approximate the low-rank representations of this small LRR model. After learning, the deep encoder is easily used to fast compute the low-rank representations of large dataset in the first step. Next, fast spectral clustering is employed to cluster the low-rank representations for avoiding the classification problem in the second step. The whole process is shown in Fig. 1. **Our contributions** are summarized as:

- We propose a predictive low-rank decomposition (PLD). PLD replaces the costly and highly non-linear SVD operations by a non-iterative deep encoder for quickly calculating the low-rank coding subspaces. We theoretically prove the deep encoder can universally approximate to the exact (or bounded) recovery of the row space.
- We create a projective low-rank subspace clustering (PLrSC) scheme to large scale clustering problem by a manner of "sampling, PLD, fast spectral clustering".
- Experimental results verify that our scheme outperforms the state-of-art baseline methods on large scale image datasets. It is important to note that we reach the best clustering accuracy 95.8% on MNIST-SC.

**Notations:** For a matrix $\mathbf{A} \in \mathbb{R}^{d\times n}$, the nuclear norm, $\ell_1$ norm, $\ell_{2,1}$ norm, F-norm, and square of F-norm are denoted by $\|\mathbf{A}\|_* = \sum_i \sigma_i(\mathbf{A})$ (the sum of the singular values of $\mathbf{A}$), $\|\mathbf{A}\|_1 = \sum_{ij} |\mathbf{A}_{ij}|$, $\|\mathbf{A}\|_{2,1} = \sum_j \|\mathbf{A}_{:j}\|_2$, $\|\mathbf{A}\|_{\mathrm{F}} = \sqrt{\sum_{ij}(\mathbf{A}_{ij})^2}$, and $\|\mathbf{A}\|_{\mathrm{F}}^2 = \sum_{ij}(\mathbf{A}_{ij})^2$.

## 2 Related Works

We will review fast coding models, and fast spectral clustering.

**Fast Coding** were presented to fast infer the codes by learning a (linear) encoder, such as projective low-rank representation (PLR) [Li *et al.*, 2016; Li *et al.*, 2014; Ding *et al.*, 2016], RPCA encoder (RPCAec) [Sprechmann *et al.*, 2015], and latLRR [Liu and Yan, 2011]. However, PLR was a supervised model for classification tasks, not subspace clustering. RPCAec assumed that the underlying data structure was a single low-rank subspace, while our model (PLD) could better handle the multiple subspaces data. Moreover, although latLRR could quickly calculate the codes by the linear encoder, it essentially resolved the insufficient sampling data [Liu and Yan, 2011], and the linear encoder was difficult to capture the complex data structure [Bengio *et al.*, 2013; Li *et al.*, 2017b]. Fortunately, PLD was to learn a deep encoder, which had power to draw the high-level features.

**Fast Spectral Clustering (FSC)** tried to speed up the spectral clustering algorithms due to the high computational complexity of the traditional spectral clustering, for example, Nyström [Chen *et al.*, 2011] and landmark-based spectral clustering (LSC) [Cai and Chen, 2015]. Nyström was to reduce the computational costs of eigen-decomposition over the whole similarity matrix [Fowlkes *et al.*, 2004]. Moreover, it was parallelized both memory use and computation on distributed systems [Chen *et al.*, 2011]. LSC [Cai and Chen, 2015] selected small data points to sparsely represent other data points for computing the spectral embedding of the data. Unfortunately, LSC and Nyström result in a poor result with the original input data. Our method is to learn "good" features by deep encoder, and employ LSC to cluster the features.

## 3 Predictive Low-rank Decomposition (PLD)

Before building our projective low-rank subspace clustering scheme, we firstly propose a predictive low-rank decomposition (PLD) model to fast calculate the projective low-rank representations in this section. Given a dataset $\mathbf{Y} = [\mathbf{Y}^1, \cdots, \mathbf{Y}^i, \cdots, \mathbf{Y}^k] \in \mathbb{R}^{d\times n}$, where each column is a data point, $k$ is the number of subspaces, $d$ is the number of dimensions, and $n$ is the number of data points, we construct a formulation of PLD, and solve PLD by combining an alternating direction method (ADM) [Liu *et al.*, 2013] and a gradient descent algorithm [Li *et al.*, 2015].

### 3.1 Problem Formulation.

Large scale LRSC is restrained by solving LRR due to the high computational complexity of Singular Value Thresholding (SVT) operator [Cai *et al.*, 2010]. Actually, sparse coding also faces the expensive inference problem [Elhamifar and Vidal, 2013]. In order to avoid the expensive inference, predictive sparse decomposition (PSD) [Gregor and LeCun, 2010; Li *et al.*, 2017a] is to fast approximate the codes or representations learned from the sparse coding model by training a feedforward neural network. Inspired by PSD, we learn a non-iterative deep encoder $f_{de}(\mathbf{Y};\theta)$ ($\theta$ is the learning parameter) to approximate the low-rank representations. The deep encoder is used to quickly calculating the low-rank codes for replacing many costly non-linear SVT operations. Therefore, we propose a predictive low-rank decomposition (PLD), which is written as a following nonconvex optimization:

$$\begin{aligned} &\min_{\mathbf{E},\theta} \|\mathbf{Z}\|_* + \lambda\|\mathbf{E}\|_{2,1} + \gamma\|\mathbf{Z} - f_{de}(\mathbf{Y};\theta)\|_{\mathrm{F}}^2 \qquad (1)\\ &s.t.\ \mathbf{Y} = \mathbf{Y}\mathbf{Z} + \mathbf{E}, \end{aligned}$$

where $\mathbf{Y} \in \mathbb{R}^{d\times n}$ is the input data, $\mathbf{Y}$ is chosen as the dictionary, $\mathbf{E} \in \mathbb{R}^{d\times n}$ is the sparse noise matrix, $\lambda$ is the regularization parameter of the sparse noise term, $\|\mathbf{Z} - f_{de}(\mathbf{Y};\theta)\|_{\mathrm{F}}^2$ is an approximation term, $\gamma$ is a regularization parameter of the approximation term (usually, $\gamma = 1$), $f_{de}(\mathbf{Y};\theta) = g(\mathbf{W}^L \cdots g(\mathbf{W}^i \cdots g(\mathbf{W}^2\mathbf{Y})))$ is a deep encoder with $L$ layers, $g$ is an activation function (such as sigmoid, tanh and ReLU), $\theta = \{\mathbf{W}^2, \cdots, \mathbf{W}^L\} \in \widetilde{\mathbb{R}} = \{\mathbb{R}^{\ell_2\times\ell_1}, \cdots, \mathbb{R}^{\ell_L\times\ell_{L-1}}\}$ is a learning parameter set, and $\ell_i$ is the number of units in the $i$-th layer ($\ell_1 = d$ and $\ell_L = n$). For example, a deep encoder with $L = 3$ layers is $f_{de}(\mathbf{Y};\theta) = g(\mathbf{W}^3 g(\mathbf{W}^2\mathbf{Y}))$.

### 3.2 Optimization.

The problem (1) is non-convex because of the non-convex deep encoder. Clearly, it is an important challenge. We employ an

alternating direction method (ADM) [Liu *et al.*, 2013] and a gradient descent algorithm [Li *et al.*, 2015] to solve (1). For efficiency, the problem (1) is first transformed into the following equivalent problem:

$$\min_{\mathbf{E},\mathbf{J},\mathbf{Z},\theta} \|\mathbf{J}\|_* + \lambda\|\mathbf{E}\|_{2,1} + \gamma\|\mathbf{Z} - f_{de}(\mathbf{Y};\theta)\|_{\mathrm{F}}^2 \quad (2)$$
$$s.t.\ \mathbf{Y} = \mathbf{YZ} + \mathbf{E},\ \ \mathbf{Z} = \mathbf{J}.$$

Then the problem (2) can be solved by minimizing the following augmented Lagrange function:

$$\begin{aligned}\mathcal{L} =& \|\mathbf{J}\|_* + \lambda\|\mathbf{E}\|_{2,1} + \gamma\|\mathbf{Z} - f_{de}(\mathbf{Y};\theta)\|_{\mathrm{F}}^2 + \\ & \mu(\|\mathbf{Y} - \mathbf{YZ} - \mathbf{E} + \mathbf{Q}_1/\mu\|_{\mathrm{F}}^2 + \|\mathbf{Z} - \mathbf{J} + \mathbf{Q}_2/\mu\|_{\mathrm{F}}^2)/2 \\ & - (\|\mathbf{Q}_1\|_{\mathrm{F}}^2 + \|\mathbf{Q}_2\|_{\mathrm{F}}^2)/(2\mu), \end{aligned} \quad (3)$$

where $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are the Lagrange multipliers, and $\mu$ is a penalty parameter. To tackle this unconstrained problem, it can be minimized by ADM, which is to iteratively update the variables $\{\mathbf{E}, \mathbf{J}, \mathbf{Z}, \theta\}$, the Lagrange multipliers $\{\mathbf{Q}_1, \mathbf{Q}_2\}$ and the penalty parameter $\mu$ until convergence. The iteratively key steps are as following:

**Updating $\theta$:** $\theta$ is calculated by minimizing $\mathcal{L}$ with respect to the weight parameters $\theta$, while others are fixed. The optimal solution is to solve a subproblem[1]: $\mathcal{L}_\theta = \|\mathbf{Z} - f_{de}(\mathbf{Y};\theta)\|_{\mathrm{F}}^2$. By using a gradient descent (GD) algorithm [Li *et al.*, 2015] to minimize $\mathcal{L}_\theta$, the updating form is

$$\theta = \theta - \zeta\partial\mathcal{L}_\theta/\partial\theta, \quad (4)$$

where $\zeta$ is a learning rate, and $\partial\mathcal{L}_\theta/\partial\theta$ is the gradient of $\mathcal{L}_\theta$.

**Updating J:** $\mathbf{J}$ is calculated by minimizing $\mathcal{L}$ with respect to $\mathbf{J}$, while others are fixed. The optimal solution is to solve a subproblem: $\mathcal{L}_\mathbf{J} = \arg\min_\mathbf{J} \frac{1}{\mu}\|\mathbf{J}\|_* + \frac{1}{2}\|\mathbf{J} - (\mathbf{Z} + \mathbf{Q}_2/\mu)\|_{\mathrm{F}}^2$. This subproblem $\mathcal{L}_\mathbf{J}$ has a closed-form solution given by

$$\mathbf{J} = \Upsilon_{\frac{1}{\mu}}(\mathbf{Z} + \mathbf{Q}_2/\mu), \quad (5)$$

where $\Upsilon_\alpha(\mathbf{C}) = \mathbf{U}\mathcal{S}_\alpha[\Sigma]\mathbf{V}^T$ is the SVT operator [Cai *et al.*, 2010], $\mathbf{U}\Sigma\mathbf{V}^T$ is the singular value decomposition (SVD) of $\mathbf{C}$, and $\mathcal{S}_\epsilon[\cdot]$ is the shrinkage-thresholding operator defined as $\mathcal{S}_\epsilon[x] = \max\{|x| - \epsilon, 0\}\mathrm{sgn}(x)$.

**Updating Z:** $\mathbf{Z}$ is calculated by minimizing $\mathcal{L}$ with respect to $\mathbf{Z}$, while others are fixed. The optimal solution is to solve a subproblem: $\mathcal{L}_\mathbf{Z} = \arg\min_\mathbf{Z} \gamma\|\mathbf{Z} - f_{de}(\mathbf{Y};\theta)\|_{\mathrm{F}}^2 + \mu(\|\mathbf{Y} - \mathbf{YZ} - \mathbf{E} + \mathbf{Q}_1/\mu\|_{\mathrm{F}}^2 + \|\mathbf{Z} - \mathbf{J} + \mathbf{Q}_2/\mu\|_{\mathrm{F}}^2)/2$. This subproblem $\mathcal{L}_\mathbf{Z}$ has a closed-form solution

$$\begin{aligned}\mathbf{Z} = & \left(2\gamma\mathbf{I}/\mu + \mathbf{Y}^T\mathbf{Y}\right)^{-1}\left(\mathbf{Y}^T(\mathbf{Y} - \mathbf{E}) + \mathbf{J} + \right. \\ & \left. + (2\gamma f_{de}(\mathbf{Y};\theta) + \mathbf{Y}^T\mathbf{Q}_1 - \mathbf{Q}_2)/\mu\right). \end{aligned} \quad (6)$$

Due to the nonlinear deep encoder, the term $\gamma\|\mathbf{Z} - f_{de}(\mathbf{Y};\theta)\|_{\mathrm{F}}^2$ will increase the number of iterations as $\mathcal{L}_\mathbf{Z}$ is difficult to guarantee convergence. For fast obtaining local convergence, we can remove this term, and $\mathcal{L}_\mathbf{Z}$ also has another closed-form solution $\mathbf{Z}$, which is as follow:

$$\begin{aligned}\mathbf{Z} = & \left(\mathbf{I} + \mathbf{Y}^T\mathbf{Y}\right)^{-1} \\ & \left(\mathbf{Y}^T(\mathbf{Y} - \mathbf{E}) + \mathbf{J} + (\mathbf{Y}^T\mathbf{Q}_1 - \mathbf{Q}_2)/\mu\right). \end{aligned} \quad (7)$$

[1]Here, we do not show the regularization technique in deep encoder. Generally, the Frobenius norm or $\ell_2$ norm can be also used to prevent the over-fitting of weights.

**Algorithm 1** PLD via ADM and gradient descent.

**Input:** A small data matrix $\mathbf{Y}$.
**Initialize:** $\theta$ is randomly initialized, $\mathbf{Z} = 0, \mathbf{E} = 0, \mathbf{Q}_1 = 0, \mathbf{Q}_2 = 0, \zeta = 0.01, \mu = 10^{-5}, \mu_{max} = 10^6, \rho = 1.2, \epsilon = 10^{-4}$.
**While** not converged **do**
  repeat
    update $\theta$ by Eq. (4),
  until $\|\mathbf{Z} - f_{de}(\mathbf{Y};\theta)\|_{\mathrm{F}}^2 < \epsilon$,
  update $\mathbf{J}$ by Eq. (5),
  update $\mathbf{Z}$ by Eq. (6) or Eq. (7),
  update $\mathbf{E}$ by Eq. (9),
  update $\mathbf{Q}_1$ and $\mathbf{Q}_2$ by Eq. (10) and Eq. (11).
  update the parameter $\mu = \min\{\rho\mu, \mu_{max}\}$,
  check: $\|\mathbf{Y} - \mathbf{YZ} - \mathbf{E}\|_{\mathrm{F}}^2 < \epsilon$ and $\|\mathbf{Z} - \mathbf{J}\|_{\mathrm{F}}^2 < \epsilon$.
**End**
**Return** solutions $\theta$, $\mathbf{Z}$ and $\mathbf{E}$.

**Updating E:** $\mathbf{E}$ is calculated by minimizing $\mathcal{L}$ with respect to $\mathbf{E}$, while others are fixed. The optimal solution is to solve the following subproblem $\mathcal{L}_\mathbf{E}$, which is as follow:

$$\arg\min_\mathbf{E} \frac{\lambda}{\mu}\|\mathbf{E}\|_{2,1} + \frac{1}{2}\|\mathbf{E} - (\mathbf{Y} - \mathbf{YZ} + \mathbf{Q}_1/\mu)\|_{\mathrm{F}}^2. \quad (8)$$

According to the Lemma 4.1 [Liu *et al.*, 2013; Yang *et al.*, 2009], the solution of $\mathcal{L}_\mathbf{E}$ is $\mathbf{E} = \Psi_{\frac{\lambda}{\mu}}(\mathbf{Y} - \mathbf{YZ} + \mathbf{Q}_1/\mu)$, where $\Psi_\alpha(\mathbf{C})$ as follow:

$$\mathbf{E}_{:j} = \begin{cases} \frac{\|\mathbf{C}_{:j}\|_2 - \alpha}{\|\mathbf{C}_{:j}\|_2}\mathbf{C}_{:j}, & \text{if } \|\mathbf{C}_{:j}\|_2 > \alpha, \\ 0, & \text{otherwise.} \end{cases}. \quad (9)$$

**Updating $\mathbf{Q}_1$ and $\mathbf{Q}_2$:** Lagrange multipliers are updated by performing a gradient ascent as follow:

$$\mathbf{Q}_1 = \mathbf{Q}_1 + \mu(\mathbf{Y} - \mathbf{YZ} - \mathbf{E}), \quad (10)$$
$$\mathbf{Q}_2 = \mathbf{Q}_2 + \mu(\mathbf{Z} - \mathbf{J}). \quad (11)$$

The procedure iterates five steps and the penalty parameter until a stopping criteria is met. The whole optimization procedure is summarized in **Algorithm 1**. After training the deep encoder, the low-rank representations can be computed by this deep encoder in *linear* time to $n$.

### 3.3 Complexity and Convergence.

The major computation of Algorithm 1 are Steps 4 and 6. The complexity of Step 4 is $\mathcal{O}(n^3)$ as we use the GD algorithm [Li *et al.*, 2015]. Based on LRR [Liu *et al.*, 2013], the complexity of Step 6 is $\mathcal{O}(nr_\mathbf{Y}^2 + r_\mathbf{Y}^3)$, where $r_\mathbf{Y}$ is the rank of $\mathbf{Y}$. The whole complexity of Algorithm 1 is $\mathcal{O}(Tn^3 + nr_\mathbf{Y}^2 + r_\mathbf{Y}^3)$, where $T$ is the training epoch of training deep encoder, and $n$ is the number of sampling data points. $T$ is less than 5 epochs.

Due to the non-smooth function and the nonlinear deep encoder, it is difficult to theoretically prove the convergence. Fortunately, there actually exist three sufficient conditions for ensuring the convergence of Algorithm 1. The first two conditions are that the dictionary matrix $\mathbf{Y}$ is of full column rank, and the optimality gap produced in each iteration step is monotonically decreasing [Liu *et al.*, 2013]. The third

condition is that any continuous activation function can be approximated uniformly on compact by an over-three layers deep encoder with enough hidden units [Ripley, 1996]. This condition results in that the deep encoder can approximate the low-rank representation by using the gradient descent algorithm [Haykin, 2009]. Notation that while using (7) to solve $\mathcal{L}_{\mathbf{Z}}$, we can move the steps 3-5 to end in Algorithm 1 for saving the training iteration time.

## 4 Projective Low-rank Subspace Clustering

Suppose a big dataset $\mathbf{X} = [\mathbf{X}^1, \cdots, \mathbf{X}^i, \cdots, \mathbf{X}^k] \in \mathbb{R}^{d\times m}$ is over-sufficiently drawn from a union of $k$ subspaces $\mathcal{S}_i$, where $r \ll m$, $m = \sum_{i=1}^k m_i$, $r = \sum_{i=1}^k r_i = \sum_{i=1}^k rank(\mathbf{X}^i)$, $m_i$ is the number of data points $\mathbf{X}^i$ of $\mathcal{S}_i$, and $r_i$ is the number of the bases of $\mathcal{S}_i$. In this section, we create a **P**rojective **L**ow-**r**ank **S**ubspace **C**lustering (**PLrSC**) scheme for clustering $\mathbf{X}$ by using PLD, and give an universal approximation theorem to recover the row space with bounded errors. The whole scheme of clustering the big dataset $\mathbf{X}$ shown in Fig. 1 consists of *sampling*, *PLD*, and *fast spectral clustering (FSC)*.

**Sampling** is used to reduce the number of data points. We assume that a small dataset $\mathbf{Y} = [\mathbf{Y}^1, \cdots, \mathbf{Y}^i, \cdots, \mathbf{Y}^k] \in \mathbb{R}^{d\times n}$ sampled from $\mathbf{X}$ is still sufficient, that is, $r_i \ll n_i < m_i$, where $r_i = rank(\mathbf{Y}^i) = rank(\mathbf{X}^i)$ and $n_i$ is the number of data points $\mathbf{Y}^i$. It leads to $r \ll n < m$, where $n = \sum_{i=1}^k n_i$. In many cases, $\mathbf{X}$ repeatedly drawn from a union of subspaces. Thus, by using some sampling or clustering techniques, we can always obtain the sampled dataset $\mathbf{Y}$, which satisfies that, for any $\mathbf{x} \in \mathbf{X}$ and any small number $\eta$, there is a $\mathbf{y} \in \mathbf{Y}$ such that $\|\mathbf{y} - \mathbf{x}\|_2 \leq \eta$. When $n = m$, $\eta = 0$. For simplicity, we use the function *randperm* in matlab to implement two sampling ways. In the first way, a small dataset $\mathbf{Y}$ is randomly sampled from the whole dataset $\mathbf{X}$ if each subspace has the similar number of data points. The second way is to randomly choose $\mathbf{Y}^i$ from the $i$-th subspace $\mathbf{X}^i$ for ensuring that every subspace has the similar number in $\mathbf{Y}$. Note that the sampling method is not studied in this paper since we focus on PLD.

**PLD** can learn a non-iterative deep encoder $f_{de}(\cdot;\theta)$ from $\mathbf{Y}$ in the above section 3. Projective low-rank representations of the big dataset $\mathbf{X}$ can be quickly computed by the deep encoder $f_{de}(\mathbf{X};\theta)$. Clearly, the complexity of non-iterations deep encoder is *linear* in data point number $m$, that is, $\mathcal{O}(m)$.

**FSC** can fast perform the spectral clustering. In this paper we employ the landmark-based spectral clustering (LSC) [Cai and Chen, 2015] to cluster the projective low-rank representations. The complexity of LSC is $\mathcal{O}(m)$. **Algorithm 2** summarizes the whole procedure of performing our PLrSC scheme. The complexity of Algorithm 2 is $\mathcal{O}(m)$.

**Theoretical Analysis.** In order to ensure the effectiveness of PLD in Algorithm 2, we give a theorem to show that the deep encoder can recover the row spaces of $\mathbf{Y}$ and $\mathbf{X}$. For better understanding the effectiveness, the big dataset $\mathbf{X}$ is divided into $m/n$ datasets $\overline{\mathbf{X}} \in \mathbb{R}^{d\times n}$. In fact, we can repeatedly select some data points in $\mathbf{X}$ to make $m/n$ an integer.

**Theorem 1.** *Let the size of clear data $\boldsymbol{Y}_0$, noise data $\boldsymbol{Y}$, new clear data $\overline{\boldsymbol{X}}_0$ and new noise data $\overline{\boldsymbol{X}}$ be $d \times n$, and the rank of $\boldsymbol{Y}_0$ be $r_0$. Suppose the skinny SVDs of $\boldsymbol{Y}_0$ and $\overline{\boldsymbol{X}}_0$ are respectively $\boldsymbol{U}_0\Sigma_0\boldsymbol{V}_0^T$ and $\overline{\boldsymbol{U}}_0\overline{\Sigma}_0\overline{\boldsymbol{V}}_0^T$, $\|\overline{\boldsymbol{X}}_0 - \boldsymbol{Y}_0\|_F \leq n\eta$ and $\|\overline{\boldsymbol{X}} - \boldsymbol{Y}\|_F \leq n\eta$. Given any continuous function $g$ (sigmoid, tanh or ReLU), $\lambda > 0$, and $\epsilon > 0$, there exist parameters $\theta = \{\boldsymbol{W}^2, \boldsymbol{W}^3\}$ such that*

$$\|f_{de}(\mathbf{Y}_0;\theta) - \mathbf{V}_0\mathbf{V}_0^T\|_{\mathrm{F}} < \epsilon, \tag{12}$$

$$\|f_{de}(\mathbf{Y};\theta) - \mathbf{V}_0\mathbf{V}_0^T\|_{\mathrm{F}} < \epsilon + \min\{d,n\} + r_0, \tag{13}$$

$$\|f_{de}(\overline{\mathbf{X}}_0;\theta) - \overline{\mathbf{V}}_0\overline{\mathbf{V}}_0^T\|_{\mathrm{F}} < \epsilon + n\alpha_0\eta + O(\eta), \tag{14}$$

$$\|f_{de}(\overline{\mathbf{X}};\theta) - \overline{\mathbf{V}}_0\overline{\mathbf{V}}_0^T\|_{\mathrm{F}} < \epsilon + \min\{d,n\} + r_0 + n\alpha\eta + O(\eta), \tag{15}$$

*where $\alpha_0 = \|\boldsymbol{Y}_0^\dagger\boldsymbol{V}\|_F + \|\frac{\partial f_{de}(\boldsymbol{Y}_0;\theta)}{\partial \boldsymbol{Y}_0}\|_F$, $\alpha = \|\boldsymbol{Y}^\dagger\boldsymbol{V}\|_F + \|\frac{\partial f_{de}(\boldsymbol{Y}_0;\theta)}{\partial \boldsymbol{Y}_0}\|_F$, $\boldsymbol{Y}_0^\dagger$ and $\boldsymbol{Y}^\dagger$ are the pseudoinverses of $\boldsymbol{Y}_0$ and $\boldsymbol{Y}$, $\boldsymbol{V} = \boldsymbol{I} - \overline{\boldsymbol{V}}_0\overline{\boldsymbol{V}}_0^T$, and $O(\eta)$ is a higher-order infinitesimal.*

For $\mathbf{Y}_0$ (or $\mathbf{Y}$) as the input data in (1), (12) (or (13)) naturally implies that $f_{de}(\mathbf{Y}_0;\theta)$ (or $f_{de}(\mathbf{Y};\theta)$) universally approximates to the exact (or bounded) recovery of the row space identified by $\mathbf{V}_0\mathbf{V}_0^T$. Moreover, when $\overline{\mathbf{X}}_0$ (or $\overline{\mathbf{X}}$) is close to $\mathbf{Y}_0$ (or $\mathbf{Y}$), (14) (or (15)) also implies that $f_{de}(\overline{\mathbf{X}}_0;\theta)$ (or $f_{de}(\overline{\mathbf{X}};\theta)$) universally approximates to the bounded recovery of the row space identified by $\overline{\mathbf{V}}_0\overline{\mathbf{V}}_0^T$. Moreover, an experiment verifies the deep encoder effectively approximates to the low-rank recovery of row space in the right picture of Fig. 2.

**Algorithm 2** PLrSC via PLD

1: **Initialize:** Large data matrix $\mathbf{X}$, number $m$ of subspaces.
2: randomly select $\mathbf{Y}$ from $\mathbf{X}$,
3: learn a deep encoder $f_{de}(\cdot;\theta)$ by PLD,
4: fast calculate $\mathbf{Z}_{\mathbf{X}}$ by $f_{de}(\mathbf{X};\theta)$,
5: perform LSC on $\mathbf{Z}_{\mathbf{X}}$ to segment $\mathbf{X}$ into $m$ clusters.

Table 1: Datasets.

| Data set | # data points | Dimension | # classes |
|---|---|---|---|
| MNIST | 70,000 | 784 | 10 |
| MNIST-SC | 70,000 | 3,472 | 10 |
| NORB | 48,600 | 2,048 | 5 |

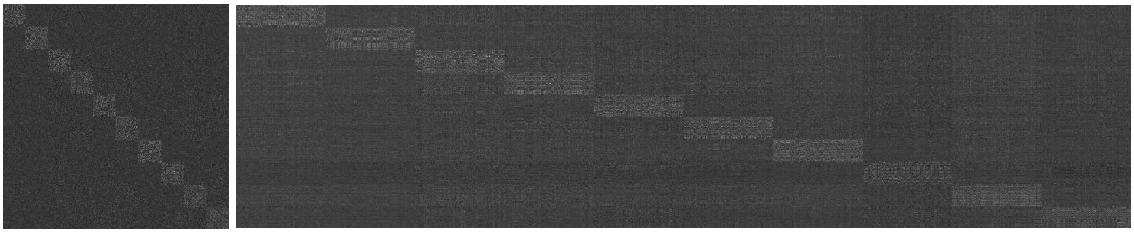
## 5 Experiments

In this section, several experiments were conducted to verify the best results of our PLrSC. First, we described the three real-world datasets. Second, many state-of-art methods and evaluation metric were introduced. Finally, we verified the effectiveness of PLD and the clustering results.

**Datasets:** We conducted experiments on three large scale datasets shown in Table 1. A brief description of the datasets is listed below. **MNIST**[2] contains 70,000 training and testing examples with $28 \times 28$ pixel greyscale images of handwritten digits 0-9. **MNIST-SC** is a variant of MNIST. We follow the settings [You *et al.*, 2016], and compute the feature vectors of each image by using a scattering convolution network [Bruna and Mallat, 2013]. Each feature vector is of size 3,472, which is a concatenation of coefficients in each layer of the network and is dropped to 500 dimensions by PCA. **NORB**[3] contains

[2]http://yann.lecun.com/exdb/mnist/

[3]http://www.cs.nyu.edu/ ylclab/data/norb-v1.0-small/



Low-rank Representation  Predictive Low-rank Decomposition (Ours)

Figure 2: Illustrations of projective low-rank representations on MNIST-SC with 10 classes. Left: Representations produced by LRR on 500 samples. Right: Representations computed by deep encoder learned from PLD on 2,000 samples.

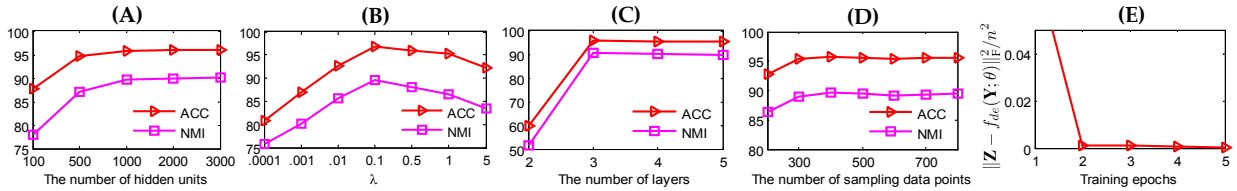



Figure 3: (A), (B), (C) and (D) show ACC (%) and NMI (%) with different hidden units, parameter $\lambda$, layers and sampling data points on MNIST-SC. (E) plots the convergence of approximated accuracy between low-rank representation and deep encoder.

images of 3D object toys belonging to 5 generic categories: four-legged animals, human figures, airplanes, trucks, and cars. There are 48,600 images combined the training samples with test samples. The original $2 \times 96 \times 96$ images were subsampled to $2 \times 32 \times 32$, and reduced to 500 by PCA.

## 5.1 Baseline Methods and Evaluation Metric.

We used the following algorithms as baseline methods.

**LRR** [Liu *et al.*, 2013] is to construct the similarity graph by learning low-rank representations and segment this graph by Normalized Cuts (NCut) [Shi and Malik, 2000]).

**FSC** is to fast build an similarity matrix for segmenting the data points, such as Nyström [Chen *et al.*, 2011], and landmark-based spectral clustering (LSC) [Cai and Chen, 2015]. Nyström was used to learn an approximate eigen-decomposition of the whole similarity matrix, and NyströmO [Chen *et al.*, 2011] was constrained in the orthogonal eigenvectors. LSC [Cai and Chen, 2015] selected a few data points to build the similarity matrix. LSC-K used k-means to select the landmarks, while LSC-R was random.

**Sampling-Clustering-Classification (S-C-C)** is a scalable clustering strategy. Based on LRR, sparse subspace clustering (SSC) and least square regression (LSR), select+LRR (selLRR), select+SSC (selSSC) [Wang *et al.*, 2014], scalable LRR (SLRR), scalable SSC (SSSC) and scalable LSR (SLSR) [Peng *et al.*, 2016] are to sample a few data points to cluster the sampling data points, and then learn a linear classifier or CRC to segment the rest data.

**PLrSC** is our scalable clustering scheme in a pattern of "Sampling-PLD-FSC". We also sample a few data points to train a deep encoder by PLD for fast computing the low-rank representations, and employ LSC-K to segment the data points. Similarly, predictive sparse decomposition (PSD) [Gregor and LeCun, 2010], denoising autoencoder (DAE) [Vincent *et al.*, 2010], robust principal component analysis encoder (RPCAec) [Sprechmann *et al.*, 2015] and latent LRR (latLRR) [Liu and Yan, 2011] are used to learn sparse and low-rank codes.

Clustering results are measured by using Clustering Accuracy (ACC) [Cai and Chen, 2015] and Normalized Mutual Information (NMI) [Cai and Chen, 2015] between the clustering label and the ground truth label. ACC and NMI range from 0 to 1. A higher value reveals a better clustering result. All experiments were implemented in MATLAB R2015a and run on a Linux machine with 2.7 GHz CPU, 24GB memory. For each data set, the final results were reported as the average and variance of 10 times repetition.

Table 2: Effectiveness of PLD compared with LRR on MNIST-SC (randomly selected 1200 samples).

| | ACC | NMI | coding time (s) |
|---|---|---|---|
| PLrSC (ours) | 88.6±3.96 | 81.4±3.50 | **0.025** |
| LRR [Liu *et al.*, 2013] | 86.1±5.49 | 81.0±3.08 | 16.36 |

## 5.2 Effectiveness and Parameter Analysis.

Before reporting clustering results, clustering time and parameter analysis, we first verify the effectiveness of our proposed PLD as PLrSC is built by PLD. PLD is to approximate low-rank codes learned from LRR by training a deep encoder. To verify this approximation, the first experiment is to compare our algorithm with LRR. Due to the limited space, we only select 1,200 training samples on MNIST-SC to conduct this experiment as LRR is nearly impossible to preform clustering on whole MNIST-SC. The results in Table 2 show that PLD has better ACC and NMI than LRR by $2.5\%$ and $0.4\%$ improvement. Furthermore, Fig. 3 (E) plots the convergence of approximated accuracy, and Fig. 2 illustrated the predictive low-rank representations calculated by the deep encoder.

The regularization parameter $\gamma$ and the learning rate $\eta$ are easy to set $\gamma = 1$ and $\eta = 0.001$ or $0.0001$ in this paper. Particularly, we focus on the effects of $\lambda$, hidden units and layers, which are important to the proposed method, because $\lambda$ controls the sparse term, and hidden units and layers restrict the power of deep encoder. Fig. 3 (A) shows PLrSC gets the best results when the number of hidden units was 2000 in the three-layer deep encoder in PLD. Fig. 3 (B) reveals that when $\lambda = 0.1$ PLrSC reaches the best ACC and NMI. Fig. 3 (C) indicates that PLrSC achieves the good results when the

Table 3: ACC (%) and NMI (%) on large scale datasets. The randomly selected number of samples in MNIST-SC, MNIST and NORB are 500, 2000, 3000. The final results were reported as the average and variance of 10 times repetition.

| | MNIST-SC (500) | | MNIST (2000) | | NORB (3000) | |
|---|---|---|---|---|---|---|
| | ACC | NMI | ACC | NMI | ACC | NMI |
| Fast Spectral Clustering (FSC) | | | | | | |
| Nyström [Chen *et al.*, 2011] | 70.8±7.56 | 64.2±3.58 | 52.7±1.46 | 47.4±0.38 | 37.1±0.84 | 18.7±0.59 |
| NyströmO [Chen *et al.*, 2011] | 71.0±7.43 | 63.9±3.65 | 52.1±3.48 | 46.9±1.49 | 37.3±0.88 | 18.9±0.25 |
| LSC-R [Cai and Chen, 2015] | 81.2±1.06 | 73.9±1.11 | 58.5±4.19 | 55.8±2.65 | 43.3±2.29 | 27.6±2.33 |
| LSC-K [Cai and Chen, 2015] | 85.9±0.34 | 80.2±0.45 | 68.3±4.99 | 67.7±2.29 | 45.5±1.35 | 30.9±0.52 |
| Sampling-Clustering-Classification (S-C-C) | | | | | | |
| selSSC [Wang *et al.*, 2014] | 74.7±5.58 | 69.9±2.93 | 55.2±1.63 | 54.4±2.19 | 36.0±3.36 | 18.9±6.75 |
| selLRR [Wang *et al.*, 2014] | 75.4±4.31 | 69.2±2.27 | 55.4±5.11 | 52.1±2.50 | 40.6±0.18 | 21.5±0.21 |
| SLSR [Peng *et al.*, 2016] | 75.4±5.60 | 70.6±2.91 | 54.1±1.56 | 48.1±0.87 | 39.9±0.31 | 20.6±0.15 |
| SLRR [Peng *et al.*, 2016] | 79.7±0.68 | 76.5±1.44 | 50.0±3.87 | 49.1±2.27 | 40.1±0.74 | 21.6±0.81 |
| SSSC [Peng *et al.*, 2016] | 78.9±4.88 | 76.4±3.25 | 54.9±1.89 | 49.9±1.15 | 39.3±2.00 | 15.1±3.06 |
| Sampling-Projective Coding-Fast Clustering (S-PC-FC) | | | | | | |
| DAE [Vincent *et al.*, 2010] | 88.0±5.90 | 82.4±2.80 | 68.9±3.71 | 65.6±1.67 | 36.8±3.79 | 17.1±4.49 |
| PSD [Gregor and LeCun, 2010] | 79.0±3.06 | 77.3±2.10 | 50.1±1.79 | 43.7±0.38 | 41.3±3.46 | 24.4±2.43 |
| latLRR [Liu and Yan, 2011] | 88.8±5.85 | 83.1±2.47 | 53.2±2.38 | 49.5±1.21 | 39.0±0.45 | 21.6±0.99 |
| RPCAec [Sprechmann *et al.*, 2015] | 71.0±5.44 | 64.6±4.06 | 54.6±5.95 | 52.2±4.16 | 45.3±0.97 | 31.2±2.53 |
| PLrSC (ours) | **95.8±0.18** | **89.6±0.31** | **71.1±2.98** | **68.1±1.51** | **50.3±1.61** | **36.5±3.66** |

number of layers is 3 in PLD. Fig. 3 (D) shows ACC and NMI with different numbers of sampling data points, and there are similar results when the number is over 400.

### 5.3 Clustering Results.

All clustering results on MNIST-SC, MNIST, and JCNORB are shown in Table 3. From Table 3, we observe that our approach (PLrSC) performs best among the existing methods. *Note that our result on MNIST-SC exceeded the state-of-art ACC 93.8% [You* et al.*, 2016] by more than 2%, and 94.3% [Li* et al.*, 2017a] by more than 1%. The former spends 1,680 seconds, while our model only takes 28 seconds. Moreover, the latter only uses 6,000 data points.*

Compared to RPCAed, DAE PSD and latLRR, PLrSC performs best among all the fast coding methods although they have the similar inference times. The important reason is that PLrSC can effectively compute the predictive low-rank representation by deep encoder in Fig. 2. In the experiments, we use the best three-layer deep encoder in RPCAed, DAE and PSD, while latLRR is self-restricted to a linear encoder. In fact, PLrSC has higher ACC than the predictive methods by average 15.4% (RPCAed), 12.1% (latLRR), 15.6% (PSD) and 7.8% (DAE) improvement on the three datasets. PLrSC also is better in NMI than the predictive methods by at least 9.7% improvement on average.

Compared to selLRR, selSSC, SLRR, SSSC and SLSR, PLrSC's performance is much better, since it makes at least 14.7% ACC and 15.7% NMI improvement in Table 3. The important reason is that the ill clustering accuracy on the selected samples leads to poor classification results, while PLrSC quickly computes the low-rank codes and employs LSC-K to segment the data points. Compared to LSC-K, LSC-R, Nyström and NyströmO, fast coding is effective for FSC. Actually, Table 3 shows that PLrSC also registers at leat 5.8% ACC and 5.1% NMI improvement over FSC. This reveals that fast low-rank representations is very important to FSC.

We report the coding and clustering time costs in Tables 2 and 4, respectively. From Table 2, we see that PLD is at least 600 times faster than LRR in the coding time as it only computes a non-iterative deep encoder. Furthermore, Table 4 also shows that PLrSC was at least 6 times faster than S-C-C in the whole clustering time. Although PLrSC has the similar clustering time to the other S-PC-FC methods, it achieves the best results in Table 3. Compared to FCS, PLrSC needs a little more time cost as it uses the deep encoder to fast compute the low-rank codes. In fact, the coding time of PLrSC can be ignored when increasing the number of data points.

Table 4: Overall clustering time (s) compared to S-C-C.

| | MNIST-SC | MNIST | NORB |
|---|---|---|---|
| selSSC [Wang *et al.*, 2014] | 310.62 | 739.58 | 759.32 |
| selLRR [Wang *et al.*, 2014] | 549.21 | 1569.15 | 2078.52 |
| SLSR [Peng *et al.*, 2016] | 289.91 | 601.23 | 620.39 |
| SLRR [Peng *et al.*, 2016] | 283.25 | 638.76 | 667.57 |
| SSSC [Peng *et al.*, 2016] | 291.53 | 613.01 | 651.16 |
| PLrSC (ours) | **27.12** | **93.53** | **43.61** |

## 6 Conclusions

In this paper we have proposed a fast low-rank coding model, named predictive low-rank decomposition (PLD), which trained a deep encoder to approximate the low-rank codes. It was solved by an alternating direction method and a gradient descent algorithm. Based on PLD, we introduced a projective low-rank subspace clustering (PLrSC) scheme to segment big dataset. This scheme firstly sampled a few data points from big datasets, learned a deep encoder by PLD, computed low-rank codes of all data points by deep encoder, and employed fast spectral clustering to cluster the low-rank codes for segmenting big dataset. Experimental results have verified that PLrSC outperforms several state-of-the-art baseline methods.

## Acknowledgments

This work is supported in part by the NSF IIS award 1651902, ONR Young Investigator Award N00014-14-1-0484, and U.S. Army Research Office Young Investigator Award W911NF-14-1-0218.

## References


[Bengio *et al.*, 2013] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, Aug. 2013.

[Bruna and Mallat, 2013] Joan Bruna and Stephane Mallat. Invariant scattering convolution networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1872–1886, Aug. 2013.

[Cai and Chen, 2015] Deng Cai and Xinlei Chen. Large scale spectral clustering via landmark-based sparse representation. *IEEE Trans. on Cybernetics*, 45(8):1669–1680, 2015.

[Cai *et al.*, 2010] Jian-Feng Cai, Emmanuel J. Candés, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM J. Optim.*, 20(4):1956–1982, Mar. 2010.

[Candés *et al.*, 2011] Emmanuel J. Candés, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM*, 58:1–37, 2011.

[Chen *et al.*, 2011] Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y. Chang. Parallel spectral clustering in distributed systems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(3):568–586, 2011.

[Ding *et al.*, 2015] Zhengming Ding, Ming Shao, and Yun Fu. Deep low-rank coding for transfer learning. In *Proc. of of Int. Joint Conf. on Artif. Intell.*, pages 3453–3459, 2015.

[Ding *et al.*, 2016] Zhengming Ding, Ming Shao, and Yun Fu. Deep robust encoder through locality preserving low-rank dictionary. In *Proc. of Euro. Conf. Comput. Vis.*, pages 567–582, 2016.

[Elhamifar and Vidal, 2013] Ehsan Elhamifar and Rene Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35:2765–2781, 2013.

[Fowlkes *et al.*, 2004] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the nystrom method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):214–225, 2004.

[Gregor and LeCun, 2010] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proc. of Int. Conf. Mach. Learn.*, pages 399–406, 2010.

[Haykin, 2009] Simon Haykin. In *Neural Networks and Learning Machines*. Pearson Education Inc., 2009.

[Li and Fu, 2015] Sheng Li and Yun Fu. Learning balanced and unbalanced graphs via low-rank coding. *IEEE Trans. on Knowl. Data Eng.*, 27:1274–1287, 2015.

[Li *et al.*, 2014] Jun Li, Heyou Chang, and Jian Yang. Learning discriminative low-rank representations for image classification. In *Proc. of Int. Joint Conf. Neural Netw.*, pages 313–318, 2014.

[Li *et al.*, 2015] Jun Li, Heyou Chang, and Jian Yang. Sparse deep stacking network for image classification. In *Proc. of the AAAI Conf. on Artif. Intell.*, pages 3804–3810, 2015.

[Li *et al.*, 2016] Jun Li, Yu Kong, Handong Zhao, Jian Yang, and Yun Fu. Learning fast low-rank projection for image classification. *IEEE Trans. Image Process.*, 25(10):4803–4814, 2016.

[Li *et al.*, 2017a] Jun Li, Yu Kong, and Yun Fu. Sparse subspace clustering by learning approximation $\ell_0$ codes. In *Proc. of the AAAI Conf. on Artif. Intell.*, pages 2189–2195, 2017.

[Li *et al.*, 2017b] Jun Li, Tong Zhang, Wei Luo, Jian Yang, Xiaotong Yuan, and Jian Zhang. Sparseness analysis in the pertraining of deep neural networks. *IEEE Trans. Neural Netw. Learn. Syst.*, 28(6):1425–1438, 2017.

[Lin *et al.*, 2011] Zhouchen Lin, Risheng Liu, and Zhixun Su. Linearized alternating direction method with adaptive penality for low rank representation. In *Proc. of Adv. Neural Inf. Process. Syst.*, pages 612–620, 2011.

[Liu and Yan, 2011] Guangcan Liu and Shuicheng Yan. Latent low-rank representation for subspace segmentation and feature extraction. In *Proc. of IEEE Int. Conf. Comput. Vis.*, pages 1615–1622, 2011.

[Liu *et al.*, 2013] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35:171–184, 2013.

[Peng *et al.*, 2016] Xi Peng, Huajin Tang, Lei Zhang, Yi Zhang, and Shijie Xiao. A unified framework for representation-based subspace clustering of out-of-sample and large-scale data. *IEEE Trans. Neural Netw. Learn. Syst.*, 27(12):2499–2512, 2016.

[Ripley, 1996] Brian D. Ripley. In *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[Shen *et al.*, 2016] Jie Shen, Ping Li, and Huan Xu. Online low-rank subspace clustering by basis dictionary pursuit. In *Proc. of Int. Conf. Mach. Learn.*, 2016.

[Shi and Malik, 2000] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[Sprechmann *et al.*, 2015] Pablo Sprechmann, Alex M. Bronstein, and Guillermo Sapiro. Learning efficient sparse and low rank models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(9):1821–1833, 2015.

[Talwalkar *et al.*, 2013] Ameet Talwalkar, Lester Mackey, Yadong Mu, Shih-Fu Chang, and Michael I. Jordan. Distributed low-rank subspace segmentation. In *Proc. of IEEE Int. Conf. Comput. Vis.*, pages 3543–3550, 2013.

[Vincent *et al.*, 2010] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, 2010.

[Wang *et al.*, 2014] Shusen Wang, Bojun Tu, Congfu Xu, and Zhihua Zhang. Exact subspace clustering in linear time. In *Proc. of the AAAI Conf. on Artif. Intell.*, pages 2113–2120, 2014.

[Wang *et al.*, 2015] Yu Wang, David Wipf, Qing Ling, Wei Chen, and Ian Wassell. Multi-task learning for subspace segmentation. In *Proc. of Int. Conf. Mach. Learn.*, 2015.

[Yang *et al.*, 2009] Junfeng Yang, Wotao Yin, Yin Zhang, and Yilun Wang. A fast algorithm for edge-preserving variational multichannel image restoration. *SIAM J. Imaging Sciences*, 2:569–592, 2009.

[You *et al.*, 2016] Chong You, Chun-Guang Li, Daniel Robinson, and Rene Vidal. Oracle based active set algorithm for scalable elastic net subspace clustering. In *Proc. of IEEE Comput. Vis. Pattern Recognit.*, pages 3928–3937, 2016.

[Zhang *et al.*, 2015] Changqing Zhang, Huazhu Fu, Si Liu, Guangcan Liu, and Xiaochun Cao. Low-rank tensor constrained multiview subspace clustering. In *Proc. of IEEE Int. Conf. Comput. Vis.*, pages 1582–1590, 2015.

[Zhao and Fu, 2015] Handong Zhao and Yun Fu. Semantic single video segmentation with robust graph representation. In *Proc. of of Int. Joint Conf. on Artif. Intell.*, pages 2219–2226, 2015.