The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Predictive Coding Machine for Compressed Sensing and Image Denoising

Jun Li,[1] **Hongfu Liu,**[1] **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@husky.neu.edu, yunfu@ece.neu.edu

Abstract

Sparse and low rank coding has widely received much attention in machine learning, multimedia and computer vision. Unfortunately, expensive inference restricts the power of coding models in real-world applications, e.g., compressed sensing and image deblurring. In order to avoid the expensive inference, we propose a predictive coding machine (PCM) which aims to train a deep neural network (DNN) encoder to approximate the codes. By this means, a test sample can be fast approximated by the well-trained DNN. However, DNN leads PCM to be a non-convex and non-smooth optimization problem, which is extremely hard to solve. To address this challenge, we extend accelerated proximal gradient for PCM by steering gradient descent of DNN. To the best of our knowledge, we are the first to propose a gradient descent algorithm guided by accelerated proximal gradient for solving the PCM problem. Besides, a sufficient condition is provided to ensure the convergence to a critical point. Moreover, when the coding models are convex in PCM, the convergence rate $O(1/(m^2\sqrt{t}))$ can be held in which m is the iteration number of accelerated proximal gradient, and t is the epoch of training DNN. Numerical results verify the promising advantages of PCM in terms of effectiveness, efficiency and robustness.

Introduction

Many sparse and low rank coding models have attracted much attention over the last decade with numerous applications in different domains, such as multimedia (Tian et al. 2016), statistics (Cai, Zhang, and Xu 2014), machine learning (Bruckstein, Donoho, and Elad 2009) and computer vision (Li, Li, and Fu 2014; Luo et al. 2017; Liu et al. 2013). In the coding methods, l_0-norm and rank are usually relaxed by l_1-norm and nuclear norm, respectively, as they lead to the convex optimization problem in many common practices (Yun 2014). Some coding algorithms approach the sparse and low rank solutions by successively alternating one or more components, such as iterative shrinkage and thresholding algorithm (ISTA) (Daubechies, Defrise, and Mol 2004), gradient projection for sparse reconstruction (GPSR) (Figueiredo, Nowak, and Wright 2007), ℓ_1_ls (Kim et al. 2007), and adaptively iterative thresholding (AIT) (Wang et al. 2015a). Moreover, some faster version algorithms, such as accelerated proximal gradient method (APG) (Beck and Teboulle 2009), and fast low rank representation (Xiao et al. 2015) are presented to accelerate the performance of these algorithms. However, the iterations lead to the expensive inference because the inference requires some sort of iterative minimization algorithms. Thus, these sparse and low rank coding models are difficult to fast compute the codes.

In order to overcome this shortcoming, some predictive models, such as, predictive sparse decomposition (PSD) (Kavukcuoglu, Ranzato, and LeCun 2008; Wang, Ling, and Huang 2016; Li, Kong, and Fu 2017) and predictive non-negative matrix factorization (PNMF) (Sprechmann, Bronstein, and Sapiro 2015), are presented by learning a deep neural network (DNN) encoder from the observed data to their representations. Unfortunately, these models have the following limitations in both theory and application.

First, PSD and PNMF are two special cases in the predictive models as they learn DNN encoders to approximate to the ℓ_1 codes and the non-negative matrix factorization, respectively. In fact, many other sparse or low rank codes, which are implemented by ℓ_0, ℓ_1_ls (Kim et al. 2007), or nuclear norms (Liu et al. 2013), still need to learn DNN encoders for fast inference. Thus, the new predictive models are in want of a normal model to unify them.

Second, there is no strict theoretical foundation to guarantee the convergence of the predictive models. They are empirically implemented by alternately using ISTA (Daubechies, Defrise, and Mol 2004) to optimize the sparse or low-rank codes, and employing the gradient descent (GD) algorithm (Haykin 2009) to learn a DNN encoder to approximate the codes. However, these works do not provide the theoretical guarantee. Therefore, we will develop theoretical conditions to ensure that the models are convergence to a critical point.

Third, the existing predictive models are usually applied in the classification tasks, including object recognition (Kavukcuoglu et al. 2009), medical image classification (Chang et al. 2013) and video classification (Farabet et al. 2011). Recently, the predictive sparse model, named sparse coding based network (SCN), also has been applied into image resolution (Wang et al. 2015b). The network parameters of these models are trained by using the learned ISTA (LISTA) (Gregor and LeCun 2010). Unfortunately, LISTA is a slower algorithm since APG has faster convergence rate than ISTA (Beck and Teboulle 2009). Therefore, these appli-

cations drive us to extend APG to accelerate the convergence of the predictive models.

In this paper, we focus on accelerating the convergence of the predictive models, and providing their theoretical convergence guarantee. To achieve this, we propose a general model named predictive coding machine (PCM), which learns a DNN encoder to approximate the representations (such as ℓ_1 norm and nuclear norm). A representation of a test sample can be fast approximated by the well-trained DNN. To obtain high convergence rate of PCM, we extend APG for the PCM optimization problem to guide a gradient descent algorithm of DNN. Moreover, a sufficient condition is provided to ensure the convergence to a critical point. In addition, PCM is more efficiently applied into compressed sensing and image denoising. Numerical results verify the promising advantages of PCM in terms of effectiveness, efficiency and robustness. We highlight our contributions as follows.

- We develop a predictive coding machine to quickly calculate the sparse and low-rank codes because it only computes an efficient DNN encoder instead of many optimization iterations. We also propose a gradient descent algorithm guided by accelerated proximal gradient (GDgAPG) for the non-convex and non-smooth PCM model.
- A sufficient condition is provided to ensure the convergence to a critical point. When the coding models are convex, GDgAPG maintains the convergence rate $O(1/T^2) \leq O(1/(m^2\sqrt{t})) \leq O(1/\sqrt{T})$, in which $T = mt$ is the number of iterations, m is the number of updating APG, and t is the epoch of training DNN.
- We empirically confirm that PCM is more robust than other coding methods in compressed sensing and image denoising. A plausible reason is that the "good" weights (nonlinear filters) of DNN can filter out the heavy noises of the input images.

Notation. For a vector $\mathbf{x} \in \mathbb{R}^p$, the ℓ_1-norm, ℓ_2-norm, and square of ℓ_2-norm are denoted by $\|\mathbf{x}\|_1$, $\|\mathbf{x}\|_2$, and $\|\mathbf{x}\|_2^2$. For a matrix $\mathbf{X} \in \mathbb{R}^{p\times n}$, nuclear norm, F-norm, and square of F-norm are denoted by $\|\mathbf{X}\|_*$, $\|\mathbf{X}\|_{\mathrm{F}}$, $\|\mathbf{X}\|_{\mathrm{F}}^2$.

Predictive Coding Machine (PCM)

In this section, a general formulation of PCM is firstly built. Secondly, we propose a GD guided by APG (GDgAPG) algorithm for the nonconvex and nonsmooth PCM, Thirdly, we establish the convergence guarantee and the $O(1/(m^2\sqrt{t}))$ convergence rate. Finally, a matrix form of sparse PCM is introduced for compressed sensing and image denoising.

Problem formulation

The sparse coding models (Beck and Teboulle 2009) is popularly written as the following convex optimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^p} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1, \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{d\times p}$ is the dictionary, $\mathbf{y} \in \mathbb{R}^d$ ($\mathbf{Y} \in \mathbb{R}^{d\times n}$) is the input data (data matrix), and λ is the regularization parameter. The problem (1) is classically trained by ISTA (Daubechies, Defrise, and Mol 2004) and APG (Beck and Teboulle 2009) in both theory and practice. To avoid the expensive inference, PSD (Kavukcuoglu, Ranzato, and LeCun 2008; Wang, Ling, and Huang 2016) is presented by learning a deep neural network to approximate the sparse representations. Similar to PSD, predictive non-negative matrix factorization (PNMF) is also proposed for classification tasks (Sprechmann, Bronstein, and Sapiro 2015). Although some alternating minimization algorithms work well in practice, the convergence guarantee of PCM is not provided in the related literature (Gregor and LeCun 2010; Sprechmann, Bronstein, and Sapiro 2015).

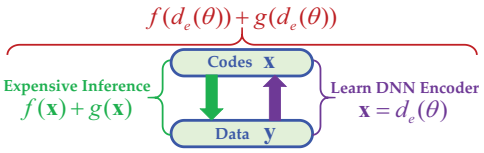

Figure 1: Framework of PCM.

In order to build a general form, the problem (1) is naturally generalized to the following problem (Li and Lin 2015):

$$\min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x}) + g(\mathbf{x}), \tag{2}$$

where f is smooth convex function, and g can be nonsmooth convex function. For example, $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x}-\mathbf{y}\|_2^2$ and $g(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$ for sparse coding. When considering data matrix $\mathbf{Y}$, $f(\mathbf{X}) = \|\mathbf{A}\mathbf{X}-\mathbf{Y}\|_2^2$ and $g(\mathbf{X}) = \lambda\|\mathbf{X}\|_*$ for low-rank coding. Unfortunately, the problem (2) shown in the green arrow of Fig. 1 is usually solved by ISTA and APG, which lead to the expensive inference.

We aim to extend the problem (2) to PCM by learning a DNN encoder shown in the purple arrow of Fig. 1 to approximate the sparse (or low-rank) codes $\mathbf{x}$. The DNN encoder $d_e(\theta)$ is formalized as:

$$\mathbf{x} = d_e(\theta) = s(\mathbf{W}_L \cdots s(\mathbf{W}_2\mathbf{y})), \tag{3}$$

where $\theta = \{\mathbf{W}_2, \cdots, \mathbf{W}_L\} \in \widehat{\mathbb{R}} = \{\mathbb{R}^{h_2\times h_1}, \cdots, \mathbb{R}^{h_L\times h_{L-1}}\}$ collectively designates all the trainable parameters in the DNN encoder, h_i is the number of units in the ith layer ($h_1 = d$ and $h_L = p$), L is the number of layers, and s is the activation function (such as sigmoid, tanh and ReLU). For example, a four-layer neural network is $d_e(\theta) = s(\mathbf{W}_4 s(\mathbf{W}_3 s(\mathbf{W}_2\mathbf{y})))$. By using $d_e(\theta)$ to replace $\mathbf{x}$ in the problem (2), PCM is considered as the following general problem:

$$\min_{\theta\in\widehat{\mathbb{R}}} \{\mathcal{F}(\theta) =: f(d_e(\theta)) + g(d_e(\theta))\}. \tag{4}$$

Moreover, the whole framework of PCM is shown in Fig. 1.

GD guided by APG (GDgAPG)

The problem (4) is a big challenge because the DNN encoder leads to be a non-convex and non-smooth problem. An effective strategy is to alternate the optimizing codes and the training DNN. Fortunately, the accelerated proximal gradient (APG) method (Beck and Teboulle 2009; Li and Lin 2015) and the gradient descent (GD) method (Gregor and LeCun 2010; Li, Chang, and Yang 2015;

Li et al. 2016) are the excellent algorithms for optimizing codes and training the DNN encoder, respectively. To extend APG to steer GD, we propose a GDgAPG algorithm to solve the problem (4). The alternative update procedure is shown in Fig. 2. More specially, our algorithm at m-th iteration consists of the following steps:

- **Step 1:** Given θ_m, $\mathbf{x}_m$ is easily calculated by
$$\mathbf{x}_m = d_e(\theta_m). \tag{5}$$
- **Step 2:** The objective function $\mathcal{F}(\theta_m)$ can be written as a classical problem:
$$\mathcal{F}(\theta_m) = F(\mathbf{x}; \mathbf{x}_m) =: f(\mathbf{x}) + g(\mathbf{x}). \tag{6}$$
$F(\mathbf{x}; \mathbf{x}_m)$ is easily solved by the APG method. The key steps are as follow:
$$\mathbf{z}_{m+1} = \text{prox}_{\alpha_m g}\left(\mathbf{x} - \alpha_m \nabla f(\mathbf{x})\right), \tag{7}$$
$$\beta_{m+1} = \frac{\sqrt{4\beta_m^2 + 1} + 1}{2}, \tag{8}$$
$$\widetilde{\mathbf{x}}_{m+1} = \mathbf{z}_{m+1} + \frac{\beta_m - 1}{\beta_{m+1}}(\mathbf{z}_{m+1} - \mathbf{z}_m), \tag{9}$$
where α_m can be fixed constants satisfying $\alpha_m < \frac{1}{L_f}$, L_f is the Lipschitz constant of f, and proximal mapping is defined as $\text{prox}_{\alpha g}(\mathbf{z}) = \arg\min_{\mathbf{u}} g(\mathbf{u}) + \frac{1}{2\alpha}\|\mathbf{x} - \mathbf{u}\|_2^2$.
- **Step 3:** θ_{m+1} can be obtained by using the least squares error $\|\mathbf{x}_m - \widetilde{\mathbf{x}}_{m+1}\|_2^2$ to train the DNN encoder (5). By employing a gradient descent (GD) algorithm to minimize this error, $\theta_m^1 = \theta_m$, and θ_m^1 is updated as:
repeat $i = 1, 2, 3, \cdots, t_m = \frac{1}{(\omega\|\mathbf{z}_{m+1} - \widetilde{\mathbf{x}}_m\|_2^2)^2}$
$$\theta_m^{i+1} = \theta_m^i - \eta \frac{\partial \|\mathbf{x}_m - \widetilde{\mathbf{x}}_{m+1}\|_2^2}{\partial \theta_m^i}, \tag{10}$$
$$\theta_{m+1} = \theta_m^{i+1},$$
until $\|\mathbf{x}_{m+1} - \widetilde{\mathbf{x}}_{m+1}\|_2^2 < \omega\|\mathbf{z}_{m+1} - \widetilde{\mathbf{x}}_m\|_2^2$, where the step-size $\eta_m = \sqrt{\frac{2|\mathbf{x}_m - \widetilde{\mathbf{x}}_{m+1}|}{L_d \rho^2 t_m}}$, a ρ-bounded gradient is $\|\nabla \mathbf{x}_m\|_2^2 = \|\nabla d_e(\theta)\|_2^2 < \rho^2$, $\omega < \frac{\sqrt{1-\alpha_m L_f}}{\sqrt{\alpha_m L_F} + \sqrt{1-\alpha_m L_f}}$, L_d and L_F are the Lipschitz constants of d_e and F.

The whole procedure iterates three steps until a stopping criteria criterion is met. The complete optimization procedure of GDgAPG is summarized in **Algorithm 1**. We can get the weights $\widehat{\theta}$ after Algorithm 1. A code $\widehat{\mathbf{x}}$ can be quickly calculated by the DNN encoder $\widehat{\mathbf{x}} = d_e(\widehat{\theta})$ given a test sample $\mathbf{y}_t$. Compared to the traditional coding models, the inference time is still very fast due to just one calculation of DNN.

Here we focus on sparse PCM problem (1) since sparse coding has been successfully applied in compressed sensing and image denoising. In the DNN encoder (3), we consider the liner activation function in the top layer because it leads to a closed-form solution for the weights $\mathbf{W}_L$ given other weights $\{\mathbf{W}_2, \cdots, \mathbf{W}_{L-1}\}$ (Li, Chang, and Yang 2015; Li et al. 2016). Thus, sparse PCM can be formulated as the following problem:
$$\min_{\theta \in \widehat{\mathbb{R}}} \left\{ \mathcal{F}(\theta) =: \|\mathbf{A} d_e(\theta) - \mathbf{Y}\|_{\text{F}}^2 + \lambda \|d_e(\theta)\|_1 \right\}, \tag{11}$$

Algorithm 1 PCM via GD guided by APG.

Initialize: θ_1 is initialized with small random values, $\mathbf{x}_1 = \mathbf{x}_0 = d_e(\theta_1)$, $\alpha_m < \frac{1}{L_f}$, $\lambda = 0.1$, $\omega < \frac{\sqrt{1-\alpha_m L_f}}{\sqrt{\alpha_m L_F} + \sqrt{1-\alpha_m L_f}}$, $\eta_m = \sqrt{\frac{2|\mathbf{x}_m - \widetilde{\mathbf{x}}_{m+1}|}{L_d \rho^2 T_m}}$, $\|\nabla d_e(\theta)\|_2^2 < \rho^2$, and $t_m = \frac{1}{(\omega\|\mathbf{z}_{m+1} - \widetilde{\mathbf{x}}_m\|_2^2)^2}$,
for $m = 1, 2, 3, \cdots$ **do**
  update $\mathbf{x}_m$ by $\mathbf{x}_m = d_e(\theta_m)$,
  update $\mathbf{z}_{m+1}, \beta_{m+1}, \widetilde{\mathbf{x}}_{m+1}$ by Eqs. (7), (8) and (9),
  repeat $i = 1, 2, 3, \cdots, t_m$
    update θ_m^{i+1} by Eq. (10),
    update $\theta_{m+1} = \theta_m^{i+1}$,
  until $\|\widetilde{\mathbf{x}}_{m+1} - \mathbf{x}_{m+1}\|_2^2 < \omega\|\mathbf{z}_{m+1} - \widetilde{\mathbf{x}}_m\|_2^2$.
end for
Return solutions $\widehat{\theta} = \theta_{m+1}$.

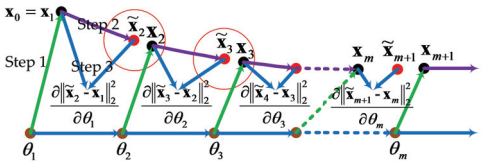

Figure 2: Procedure of updating $\mathbf{x}_m$, $\widetilde{\mathbf{x}}_m$ and θ_m.

where $\mathbf{A}$ is the dictionary, λ is the regularization parameter. The problem (11) can be easily solved by Algorithm 1. In practice, the weights θ are trained by implementing the gradient descent algorithm, and ℓ_1 norm is solved by the shrinkage-thresholding operator, where $\mathcal{S}_\epsilon[\cdot]$ is defined as:
$$\mathcal{S}_\epsilon[x] = \begin{cases} x - \epsilon, & \text{if } x > \epsilon \\ x + \epsilon, & \text{if } x < \epsilon \\ 0, & \text{otherwise} \end{cases}. \tag{12}$$

Remark 1. When considering low-rank coding (Zhang et al. 2015; Liu et al. 2013), low-rank PCM also combines it with a DNN encoder. Solution of low-rank PCM is similar to sparse PCM. The shrinkage-thresholding operator (12) is only changed as a Singular Value Thresholding (SVT) operator (Cai, Candés, and Shen 2010).

Convergence Guarantee

We study the convergence guarantee of Algorithm 1. We provide a sufficient condition to ensure the convergence to a critical point, and prove that PCM has a convergence rate $O(1/(m^2\sqrt{t}))$, where m is the iteration number of APG, and $t = \max\{t_m\}$ is the epoch of training DNN.

After Step 3 in Algorithm 1, there is an error $\mathbf{e}_{m+1}$ between the true $\widetilde{\mathbf{x}}_{m+1}$ and the approximation $\mathbf{x}_{m+1} = d_e(\theta_{m+1})$. For convenience, the error are denoted as $\mathbf{e}_{m+1} = \mathbf{x}_{m+1} - \widetilde{\mathbf{x}}_{m+1}$. Keeping the next iteration, $m = m + 1$, we have $\mathbf{x}_m = \widetilde{\mathbf{x}}_m + \mathbf{e}_m$ after Step 1. So, even if there exists an error, the key point is to make sure sufficient descent, which means
$$F(\mathbf{x}_{m+1}) \leq F(\mathbf{x}_m) - \kappa\|\mathbf{z}_{m+1} - \mathbf{x}_m\|_2^2, \tag{13}$$
where $\kappa > 0$ is a small constant. For ensuring the convergence to a critical point, we present our theorem as follow.

Theorem 1[1]**.** *Let f and d_e be proper functions with Lipschitz continuous gradients, and g be proper and lower semi-continuous function. For convex f and g, and nonconvex d_e, assume that $\mathcal{F}(\theta)$ is coercive. Then $\{\theta_m\}$ and $\{\boldsymbol{x}_m\}$ generated by Algorithm 1 are bounded. Let $\widehat{\theta}$ be any accumulation point of $\{\theta_m\}$, we have $0 \in \partial\mathcal{F}(\widehat{\theta})$, i.e., $\widehat{\theta}$ is a critical point.*

Actually, Theorem 1 can still be held even if f and g are noncovex. Next, we study the convergence rate of our GDgAPG algorithm. It is well-known that APG (Beck and Teboulle 2009) and GD (Reddi et al. 2016) have respectively the convergence rate $O(1/m^2)$ and $O(1/\sqrt{t})$. Because GDgAPG employs APG to guide GD to train DNN, by combining APG with GD, the proposition is shown as follow:

Proposition 1. *For convex functions f and g, and nonconvex function d_e, we assume that ∇f and d_e are Lipschitz continuous. Let $\{\boldsymbol{x}_m\}$, $\{\boldsymbol{z}_m\}$, $\{\widetilde{\boldsymbol{x}}_m\}$, and $\{\theta_m^t\}$ be generated by Algorithm 1. Let $\widehat{\theta}$ be any local optimum, and $t = \max\{t_m\}$ be the maximum epoch of the training DNN, where $t_m = \frac{1}{(\omega\|\boldsymbol{z}_{m+1}-\widetilde{\boldsymbol{x}}_m\|_2^2)^2}$. Then for any $m \geq 1$,*

$$F(\theta_{m+1}^t) - F(\widehat{\theta}) \leq \frac{(1-\omega)\rho L_d\|\mathbf{y}\|_2^2}{\alpha_m(m+1)^2\sqrt{t}}\|\theta_1^1 - \widehat{\theta}\|_2^2, \quad (14)$$

where ω is defined in Eq. (10), a ρ-bounded gradient is $\|\nabla d_e(\theta)\|_2^2 < \rho^2$, α_m can be fixed constants satisfying $\alpha_m < \frac{1}{L_f}$, L_f and L_d are Lipschitz constants of f and d_e.

Remark 2. Based on Proposition 1, GDgAPG has a convergence rate $O(1/(m^2\sqrt{t}))$. Moreover, it is improved as $O(1/(\sqrt{2}m^2))$ since we only trained the DNN encoder 1 or 2 times in every APG iteration. Clearly, GDgAPG is better than the convergence rate $O(1/(m\sqrt{t}))$ of LISTA (Gregor and LeCun 2010) as LISTA is a non-accelerating case. In addition, there is no convergence guarantee for LISTA.

Remark 3. In essence, our GDgAPG is a whole gradient descent of $\mathcal{F}(\theta)$ to solve the problem (4) since both GD and APG are the gradient methods. Similarly, LISTA (Gregor and LeCun 2010) is also a whole gradient descent method of $\mathcal{F}(\theta)$. However, LISTA uses ISTA (Daubechies, Defrise, and Mol 2004) to train the DNN encoder, which has a lower convergence rate $O(1/(m\sqrt{t}))$ in Remark 2. Our GDgAPG employs APG to guide GD for learn the DNN encoder and has a fast convergence as APG has a faster convergence rate $O(1/m^2)$ than ISTA $O(1/m)$.

Remark 4. Although d_e with ReLU does not satisfy the Lipschitz continuous gradients assumption, it is very popular activation function in practice (Glorot, Bordes, and Bengio 2011; Li et al. 2017). In fact, ReLU has some advantages for the deep networks. For example, 1) it can easily obtain sparse representations; 2) ReLU does not hurt the optimization of the deep networks; 3) the objection function with ReLU is empirically convergent (Glorot, Bordes, and Bengio 2011; Li et al. 2017).

Remark 5. In addition, some deep models are also used to improve the performance of the coding models. There are some works to integrate the sparse or low-rank coding models with the deep feathers or deep models (Ding, Nasrabadi, and Fu 2016). However, these integrated models still lack theories to guarantee the convergence. Fortunately, our convergence theory also fits for the integrated models.

[1]The proofs of Theorem 1 and Proposition 1 are provided in supplementary materials.

Numerical Experiments

In this section, we apply sparse PCM (11) on compressed sensing and image denoising by using our GDgAPG algorithm. The experiments verify that PCM is faster and more robust than the traditional sparse coding methods. All algorithms were run on Matlab 2015b and Windows 7 with an Intel Core i5 2.40 GHz CPU and 24GB memory.

Note that the predictive models are difficult to recover the sparse signals of the any observed measurements (data points) in compressed sensing and image deblurring since they train a DNN encoder to approximate to the sparse signals of the training data points, not any data points. Fortunately, the predictive models can quickly recover the sparse signals of the training data points by using the DNN encoder. In fact, we have two insights. First, we assume the the training data are enough to sample from the observed data so that the neural networks learned by the training data can capture the distribution of all the observed data. Second, we train on the observed data with low noises, and test on the data with high noises. This leads to better performance of our proposed method when facing the data with high noises. A plausible reason is that the "good" weights (filters) of the DNN encoder can filter out the heavy noises of the inputs.

Compressed Sensing (CS)

Similar to (Candes and Romberg 2005), we evaluate the performance of Algorithm 1 through some numerical simulations. We consider the target k-sparse time-domain signal $\mathbf{x} \in \mathbb{R}^p$ (i.e., $p = 256$), where k varies in the set $\{2, 3, \cdots, 10\}$. We construct 20 signals $\mathbf{x}$ with k peaks chosen k locations at random, and created a $d \times p = 64 \times 256$ measurement matrix $\mathbf{A}$ drawn independently from $\mathcal{N}(0, 1)$. The noise vector $\mathbf{z}$ is also Gaussian with independent $\mathcal{N}(0, \sigma)$. The 20 signals $\mathbf{x}$ are used to construct 500 training examples, which are created by $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{z}$ with $\sigma = 0.001$. We also use the 20 signals to create 11 test datasets with σ varying in the set $\{0.001, 0.1, 0.2, \cdots, 1\}$, and each dataset contains 500 examples. $ReLU$ is voted as the activation functions. We measure the relative reconstruction error (RRE) $\frac{\|\mathbf{A}\widehat{\mathbf{x}}-\mathbf{y}\|_2}{\|\mathbf{y}\|_2}$, and the relative estimation error (REE) $\frac{\|\widehat{\mathbf{x}}-\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$, where $\mathbf{x}$ is the true sparse signal, $\widehat{\mathbf{x}}$ is the estimation sparse signal recovered by (3) with the convergence point $\widehat{\theta}$ by using GDgAPG.

In compressed sensing our GDgAPG compares with the baseline methods, including ISTA (Daubechies, Defrise, and Mol 2004), APG (Beck and Teboulle 2009), and LISTA (Gregor and LeCun 2010; Sprechmann, Bronstein, and Sapiro 2015). Also, we compare with the traditional sparse coding algorithm, such as ℓ_1-magic[2], ℓ_1_ls[3] and gradient projection for sparse reconstruction (GPSR)[4].

[2]http://users.ece.gatech.edu/justin/l1magic/

[3]https://stanford.edu/~boyd/l1_ls/

[4]http://www.lx.it.pt/~mtf/GPSR/

Table 1: Computing time in compressed sensing by second.

test time							training time	
ℓ_1-magic	ℓ_1_ls	GPSR	ISTA	APG	LISTA	GDgAPG	LISTA	GDgAPG
1.5212	7.1873	1.1388	0.4658	0.1164	**0.00603**	**0.00599**	5.8782	**5.7698**

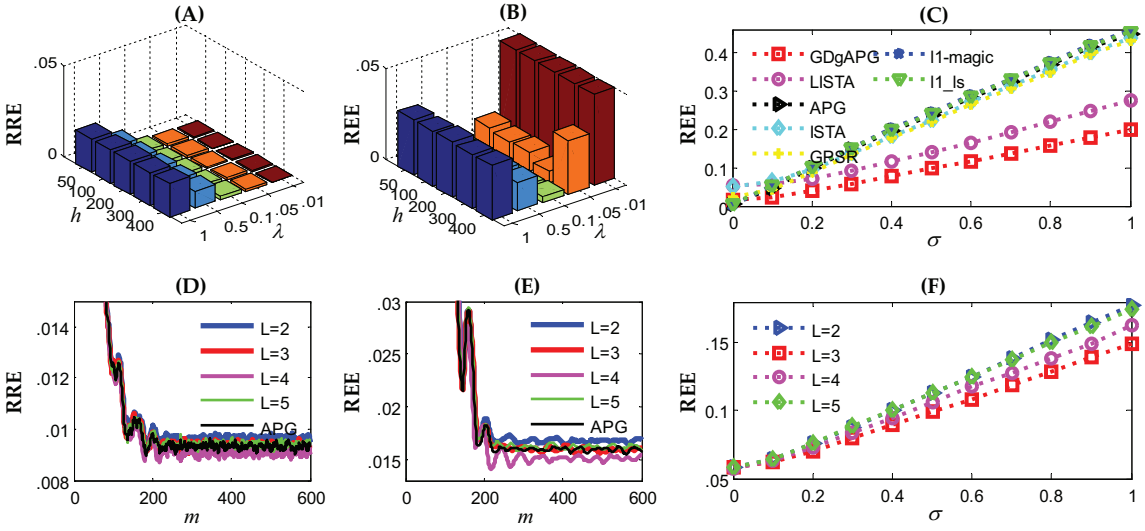

Figure 3: (A) Relative reconstruction error (RRE) with different numbers of hidden units h and regularization parameters λ; (B) Relative estimation error (REE) with h and λ; (C) REE with different methods and Gaussian noise σ; (D) RRE with iteration m; (E) REE with iteration m; (F) REE with different layers L and Gaussian noise σ.

In the first experiment we study the parameter analysis. Actually, ℓ_1 regularization parameter λ, learning rate η and the number of hidden units h are the typical parameters in PCM. η is definitely set as 0.1. We consider $\lambda \in \{1, 0.5, 0.1, 0.05, 0.01\}$ and the three layers DNN encoder: 64-h-256 ($h \in \{400, 300, 200, 100, 50\}$) for the parameter analysis. Fig. 3 (A) and (B) show RRE and REE with different λ and h, respectively. We observe that λ is sensitive to the results, while h is smaller than λ. The good performance is achieved at $\lambda = 0.1$ and $h = 300$, which are selected in the next experiments.

In the second experiment, we verify the convergence and sparse recovery with the different layers by using our GDgAPG algorithm. We consider $L = \{2, 3, 4, 5\}$ layers DNN encoder, and the number of hidden units is 300 in each layer. Fig. 3 (D) and (E) illustrate RRE and REE with the iterations m, respectively. Compared with APG, we observe that the signal is recovered to fairly accuracy 0.0156 by using GDgAPG. This reveals that our GDgAPG in Algorithm 1 successfully converges to a critical point. Moreover, Fig. 3 (E) shows the REE results on the 11 noise datasets. We observe that good results are obtained with $L = 3$ (64-300-256).

In the third experiment we compare the robust performance of Algorithm 1 to the six ℓ_1 optimization methods: ISTA, APG, LISTA, ℓ_1-magic, ℓ_1_ls, and GPSR. The deep neural network is robust to the measurement data with high noises due to the nonlinear transformation. Fig. 3 (C) shows the relative errors of different methods and σ. We observe that the average REE of ℓ_1-magic, ℓ_1_ls, GPSR, ISTA, APG, LISTA, and GDgAPG are 0.2390, 0.2369, 0.2209, 0.2282, 0.2335 0.1465, and 0.1004, respectively. Moreover, the recovered signals by ISTA, APG, LISTA, and GDgAPG are plotted in Fig. S1 in the supplementary materials, and this shows that GDgAPG is more robust than the other algorithms.

More importantly, LISTA and GDgAPG are always faster than ℓ_1 optimization methods in inference time because they only compute an efficient DNN encoder once instead of many optimization iterations. In the test time we compare with ℓ_1-magic, ℓ_1_ls, GPSR, ISTA, LISTA, and APG in terms of the computation time. As shown in Table 1, LISTA and GDgAPG are many times faster than other algorithms. We also observe that GDgAPG and LISTA have more training time than APG and ISTA because they takes some time to update the weights, and GDgAPG has the lower training time than LISTA as it needs more iterations for convergence. In fact, the training time of LISTA and GDgAPG depend on the structure of DNN, that is, bigger structure, more training time. Fortunately, although a bigger deep structure brings more training time, it is still very fast for the test sample.

Image Denoising

We use the ten 256×256 images shown in the top line of Fig. S2 in supplementary materials to do experiments for imaging denoising. All pixels is normalized to $[0, 1]$. The training images are blurred through a Gaussian blur of size 9 and standard deviation 4 (applied by the MATLAB functions *imfilter*) and noised by adding a white Gaussian noise $\mathcal{N}(0, \sigma)$, where $\sigma = e^{-4}$. We also use the blurred method to create 3

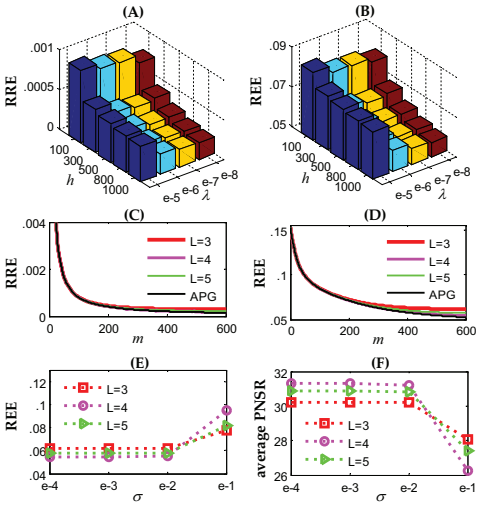

Figure 4: (A) RRE with different numbers of hidden units h and regularization parameters λ; (B) REE with h and λ; (C) RRE with iteration m; (D) REE with iteration m; (E) REE with different layers L and Gaussian noise σ; (F) Average PNSR with L and σ.

test noise images with σ varying in the set $\{e^{-3}, e^{-2}, e^{-1}\}$. The 3 test noise images are shown in three low lines of of Fig. S2. $tanh$ is voted as the activation function. Similar to compressed sensing, RRE and REE are used to evaluate our PCM. We also measure PNSR for image denoising.

Based on the experiment settings (Beck and Teboulle 2009), we then test ISTA, LISTA, APG, and GDgAPG for solving PCM problem (11), where $\mathbf{y}$ represents the 65536 observed image vector, and $\mathbf{A} = \mathbf{RK}$, where $\mathbf{R}$ is the matrix representing the blur operator and $\mathbf{K}$ is the inverse of a three stage Haar wavelet transform. Moreover, we also compare with the state-of-the-art denoising methods (e.g., Weighted Nuclear Norm Minimization (WNNM)[5] (Gu et al. 2017), Estimating Blur Kernel (EBK)[6] (Pan et al. 2016), and sparse coding based network (SCN) (Wang et al. 2015b)). It is worth mentioning that for the sake of fairness, SCN is approximatively represented by LISTA in this paper as SCN is a predictive sparse model which learns the dictionary and sparse codes by LISTA, while we only focus on the fast computing sparse codes by DNN learned by GDgAPG.

First, we analyze the parameters λ, η h, and L in sparse PCM by using our GDgAPG algorithm. η is easily set as $e^{-}5$. We consider $\lambda \in \{e^{-5}, e^{-6}, e^{-7}, e^{-8}\}$ and the three layers DNN encoder: 65536-h-65536 ($h \in \{100, 300, 500, 800, 1000\}$) for the parameter analysis. After 600 iterations, RRE and REE with λ and h are shown in Fig. 4 (A) and (B), respectively. We consider $L = \{3, 4, 5\}$ layers DNN encoder, and the number of hidden units is 500 in each layer. Fig. 4 (C) and (D) illustrates RRE and REE

[5]http://www4.comp.polyu.edu.hk/~cslzhang/code/WNNM_code.zip

[6]http://vllab1.ucmerced.edu/~jinshan/projects/dark-channel-deblur/

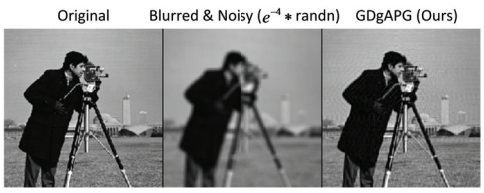

Figure 5: Left: Original cameraman image; Middle: Training image (Blurred and Gaussian noise with standard deviation e^{-4}); Right: Deblurring of the training image.

Table 2: Function values and computing time for the test sample with different noises.

	Straw	Peppers	P.gon	Parrots	Hat	C.man	Boats	Bike	Barbara	Baboon	Avg.
Blurred & Noisy ($\sigma = e^{-3}$)											
Noisy	19.04	22.79	21.30	23.32	25.12	21.31	23.41	18.74	22.91	19.39	21.73
GDgAPG	**26.76**	**33.34**	**30.91**	**33.11**	**33.91**	**31.67**	**35.41**	**28.07**	**33.30**	**25.91**	**31.24**
LISTA	22.35	28.73	26.57	27.97	29.60	26.64	30.65	22.93	27.76	21.90	26.51
APG	23.73	29.03	27.42	28.56	29.46	27.51	30.59	24.19	28.63	22.81	27.19
ISTA	22.35	28.56	26.50	27.82	29.39	26.53	30.39	22.88	27.63	21.88	26.39
EBK	19.81	23.09	20.96	24.23	26.42	21.73	25.02	19.76	23.28	19.30	22.36
WNNM	18.66	21.77	20.83	22.80	24.74	20.77	22.65	18.08	22.34	19.21	21.18
Blurred & Noisy ($\sigma = e^{-2}$)											
Noisy	19.07	22.87	21.36	23.41	25.26	21.37	23.51	18.77	22.99	19.43	21.80
GDgAPG	**26.76**	**33.31**	**30.87**	**33.06**	**33.85**	**31.63**	**35.31**	**28.05**	**33.26**	**25.88**	**31.20**
LISTA	22.35	28.72	26.56	27.97	29.58	26.63	30.62	22.93	27.76	21.91	26.50
APG	20.45	24.09	23.08	23.78	24.51	23.09	24.68	20.81	23.60	20.23	22.83
ISTA	20.66	25.51	23.86	25.12	26.49	23.84	26.34	21.01	24.60	20.50	23.79
EBK	19.50	20.87	19.67	19.47	19.65	20.07	22.76	18.98	20.23	16.28	19.75
WNNM	18.66	21.77	20.83	22.80	24.74	20.77	22.65	18.08	22.34	19.21	21.18
Blurred & Noisy ($\sigma = e^{-1}$)											
Noisy	16.50	18.20	17.61	18.39	18.89	17.63	18.44	16.32	18.25	16.70	17.69
GDgAPG	**24.04**	**26.84**	**26.28**	**27.62**	**27.41**	**27.19**	**28.94**	**24.54**	**26.46**	**23.90**	**26.32**
LISTA	21.42	25.65	24.52	26.37	27.03	25.23	27.39	22.09	25.72	21.12	24.65
APG	16.66	18.39	17.78	18.42	18.85	17.80	18.61	16.65	18.25	16.73	17.81
ISTA	16.66	18.41	17.78	18.44	18.88	17.80	18.62	16.64	18.28	16.74	17.83
EBK	15.12	15.56	15.92	15.94	16.55	15.18	15.79	13.88	16.30	15.33	15.56
WNNM	18.33	21.46	20.38	22.52	24.39	20.58	22.25	17.87	21.98	19.03	20.88

with the iterations m, respectively. We observe that GDgAPG with different layers approximatively overlap APG. This reveals that our GDgAPG is successfully guided by APG, and converges to a critical point. The denoising and deblurring image is shown in the right of Fig. 5. Moreover, Fig. 4 (E) and (F) plots REE and average PNSR with different layers on the 4 noise images, respectively . We also observe that the good performances are achieved at $L = 4$ except for $\sigma = e^{-1}$ (Fortunately, it still has better results when $\sigma = e^{-1}$). For simpleness, we select $\lambda = e^{-8}$, $h = 500$, and $L = 4$ (65536-500-500-65536) in the next experiments.

Second, we compare our GDgAPG algorithm to LISTA, APG, ISTA, EBK and WNNM. The PNSR and denoising images are shown in Table 2 and Fig. 6 respectively. In Table 2 we observe that [1] GDgAPG has 4.73, 4.70, and 1.67 improvements on average PSNR when the noises changes

Table 3: All times on ten images in image denoising by second.

test time						training time	
EBK	WNNM	ISTA	APG	LISTA	GDgAPG	LISTA	GDgAPG
1718.6293	783.1552	151.8110	149.9066	**0.2568**	**0.2534**	1152.6231	**1137.1975**

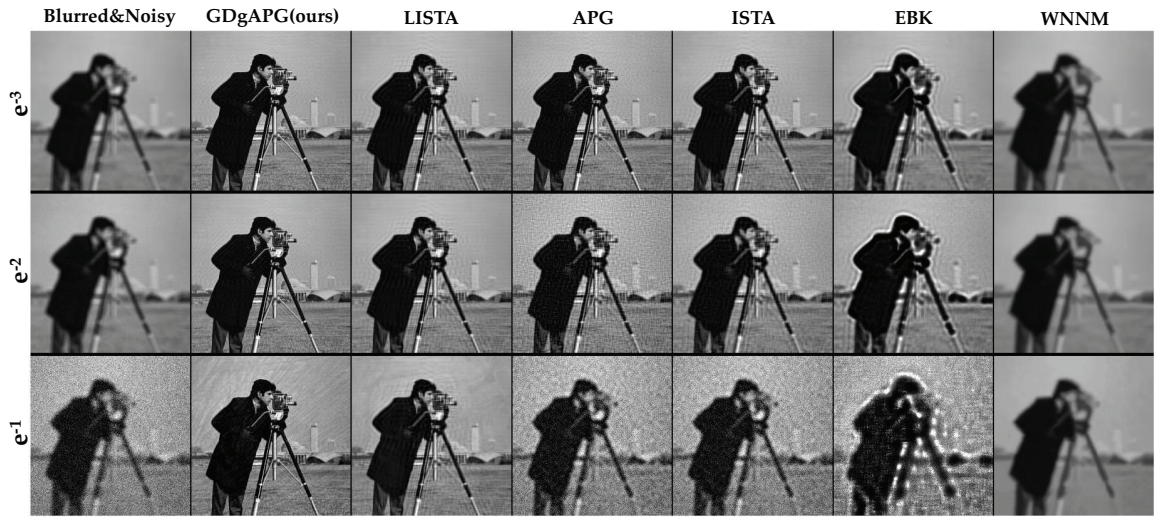

Figure 6: Denoising results on image *cameraman*. From left to right, the methods are GDgAPG (Ours), APG, LISTA, ISTA, EBK and WNNM. From top to down, the standard deviations of Gaussian noises are from e^{-3} to e^{-1}.

from $\sigma = e^{-3}$ to $\sigma = e^{-2}$ and $\sigma = e^{-1}$; [2] average PSNRs of APG, ISTA and EBK are lower than the noised images with $\sigma = e^{-1}$; [3] WNNM cannot clean the blurring images although it has relatively higher PSNR; [4] GDgAPG has higher PSNR than LISTA. Moreover, Fig. 6 also shows that APG, ISTA, EBK and WNNM fail in the high noise images with $\sigma = e^{-1}$, while GDgAPG and LISTA are effective because the weights and nonlinear transformation in DNN keep the useful information on the training images. This demonstrates that the DNN encoders in GDgAPG and LISTA are robust to high noises. In addition, we also observe that GDgAPG is better denoising and deblurring than LISTA since APG is faster convergent than ISTA.

Third, Table 3 shows LISTA and GDgAPG are always faster than APG, ISTA, EBK and WNNM in test computing time due to only once calculating an efficient DNN encoder though they needs 1138 seconds to train the DNN encoder. Ideally, GDgAPG and LISTA are 590 times faster than APG and ISTA, and over 3000 times faster than EBK and WNNM. Moreover, GDgAPG is higher than LISTA in term of average PSNR although they have the similar computing time.

Overall, our GDgAPG is more robust and faster than the related algorithms. In addition, due to the limited space the RRE and REE results are shown in Table S1, and more images are plotted in Fig. S3-S5 in supplementary materials.

Conclusion

We built a predictive coding model (PCM) to approximate the traditional (sparse) coding model by learning a deep neural network (DNN) encoder. We proposed a gradient descent algorithm guided by accelerated proximal gradient (GDgAPG) for solving PCM. We established the convergence condition and the convergence rate $O(1/(m^2\sqrt{t}))$, in which m is the number of updating APG, and t is the epoch of training DNN. We have successfully applied PCM to compressed sensing and image denoising. After training in low noise data, PCM was confidently faster than ℓ_1 optimization methods in inference time because it only computed an efficient DNN encoder instead of many optimization iterations, and PCM was more robust than other coding methods since the high noises of the inputs can be filtered out by the "good" weights (filters) in the DNN encoder. Numerical results verified that our GDgAPG algorithm was convergent, and demonstrated that it was fast and robust than the related methods.

Acknowledgments

This research is supported in part by the NSF IIS award 1651902, ONR Young Investigator Award N00014-14-1-0484, and U.S. Army Research Office Award W911NF-17-1-0367.

References

Beck, A., and Teboulle, M. 2009. A fast iterative shrinkage thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences* 2(1):183–202.

Bruckstein, A.; Donoho, D.; and Elad, M. 2009. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review* 51:34–81.

Cai, J.; Candés, E.; and Shen, Z. 2010. A singular value thresholding algorithm for matrix completion. *SIAM J. Optim.* 20(4):1956–1982.

Cai, T.; Zhang, A.; and Xu, G. 2014. Sparse representation of a polytope and recovery of sparse signals and low-rank matrices. *IEEE Trans. on Info. Theory* 60(1):122–132.

Candes, E., and Romberg, J. 2005. L1-magic: Recovery of sparse signals via convex programming. *http://www.acm.caltech.edu/l1magic/* 1–19.

Chang, H.; Zhou, Y.; Spellman, P.; and Parvin, B. 2013. Stacked predictive sparse coding for classification of distinct regions in tumor histopathology. In *Proc. IEEE Int. Conf. Comput. Vis.*, 169–176.

Daubechies, I.; Defrise, M.; and Mol, C. D. 2004. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. Pure Appl. Math.* 57(11):1413–1457.

Ding, Z.; Nasrabadi, N.; and Fu, Y. 2016. Task-driven deep transfer learning for image classification. In *Proc. of IEEE Conf. on Acoust., Speech, Signal Process.*, 2414–2418.

Farabet, C.; LeCun, Y.; Kavukcuoglu, K.; and Culurciello, E. 2011. Largescale fpga-based convolutional networks. In *In R. Bekkerman, M. Bilenko, and J. Langford, editors, Scaling up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press.

Figueiredo, M.; Nowak, R.; and Wright, S. 2007. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE J. of Selected Topics in Signal Process.* 1(4):586–597.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Deep sparse rectifier neural networks. In *Proc. of the Int. Conf. Artif. Intell. Statist.*, 315–323.

Gregor, K., and LeCun, Y. 2010. Learning fast approximations of sparse coding. In *Proc. 27th Int. Conf. Mach. Learn.*, 807–814.

Gu, S.; Xie, Q.; Meng, D.; Zuo, W.; Feng, X.; and Zhang, L. 2017. Weighted nuclear norm minimization and its applications to low level vision. *Int. J. Comput. Vis.* 121(2):183–208.

Haykin, S. 2009. *Neural Networks and Learning Machines*. Pearson Education Inc., 3 edition.

Kavukcuoglu, K.; Ranzato, M.; Fergus, R.; and LeCun, Y. 2009. Learning invariant features through topographic filter maps. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1605–1612.

Kavukcuoglu, K.; Ranzato, M.; and LeCun, Y. 2008. Fast inference in sparse coding algorithms with applications to object recognition. In *CBLL-TR-2008-12-01, New York University*.

Kim, S.; Koh, K.; Lustig, M.; Boyd, S.; and Gorinevsky, D. 2007. An interior-point method for large-scale l1-regularized least squares. *IEEE J. of Selected Topics in Signal Process.* 1(4):606–617.

Li, H., and Lin, Z. 2015. Accelerated proximal gradient methods for nonconvex programming. In *Proc. Adv. Neural Inf. Process. Syst.*, 379–387.

Li, J.; Kong, Y.; Zhao, H.; Yang, J.; and Fu, Y. 2016. Learning fast low-rank projection for image classification. *IEEE Trans. Image Process.* 25(10):4803–4814.

Li, J.; Zhang, T.; Luo, W.; Yang, J.; Yuan, X.; and Zhang, J. 2017. Sparseness analysis in the pertraining of deep neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 28(6):1425–1438.

Li, J.; Chang, H.; and Yang, J. 2015. Sparse deep stacking network for image classification. In *Proc. of the AAAI Conf. on Artif. Intell.*, 3804–3810.

Li, J.; Kong, Y.; and Fu, Y. 2017. Sparse subspace clustering by learning approximation ℓ_0 codes. In *Proc. of the AAAI Conf. on Artif. Intell.*, 2189–2195.

Li, L.; Li, S.; and Fu, Y. 2014. Learning balanced and unbalanced graphs via low-rank coding. *Image and Vision Computing* 32:814–823.

Liu, G.; Lin, Z.; Yan, S.; Sun, J.; Yu, Y.; and Ma, Y. 2013. Robust recovery of subspace structures by low-rank representation. *IEEE Trans. Pattern Anal. Mach. Intell.* 35(1):171–184.

Luo, L.; Yang, J.; Qian, J.; Tai, Y.; and Lu, G. 2017. Robust image regression based on the extended matrix variate power exponential distribution of dependent noise. *IEEE transactions on neural networks and learning systems* 28(9):2168–2182.

Pan, J.; Sun, D.; Pfister, H.; and Yang, M. 2016. Blind image deblurring using dark channel prior. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1628–1636.

Reddi, S.; Hefny, A.; Sra, S.; Poczos, B.; and Smola, A. 2016. Stochastic variance reduction for nonconvex optimization. In *Proc. 33th Int. Conf. Mach. Learn.*, 314–323.

Sprechmann, P.; Bronstein, A. M.; and Sapiro, G. 2015. Learning efficient sparse and low rank models. *IEEE Trans. Pattern Anal. Mach. Intell.* 37(9):1821–1833.

Tian, Y.; Ruan, Q.; An, G.; and Fu, Y. 2016. Action recognition using local consistent group sparse coding with spatio-temporal structure. In *ACM Multimedia*, 317–321.

Wang, Y.; Zeng, J.; Peng, Z.; Chang, X.; and Xu, Z. 2015a. Linear convergence of adaptively iterative thresholding algorithms for compressed sensing. *IEEE Trans. on Signal Process.* 63(11):2957–2971.

Wang, Z.; Liu, D.; Yang, J.; Han, W.; and Huang, T. 2015b. Deep networks for image super-resolution with sparse prior. In *Proc. IEEE Int. Conf. Comput. Vis.*, 370–378.

Wang, Z.; Ling, Q.; and Huang, T. 2016. Learning deep l_0 encoders. In *Proc. of the AAAI Conf. on Artif. Intell.*, 2194–2200.

Xiao, S.; Li, W.; Xu, D.; and Tao, D. 2015. Falrr: A fast low rank representation solver. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 4612–4620.

Yun, F. 2014. In *Low-rank and sparse modeling for visual analysis*. Springer.

Zhang, C. Q.; Fu, H. Z.; Liu, S.; Liu, G. C.; and Cao, X. C. 2015. Low-rank tensor constrained multiview subspace clustering. In *Proc. IEEE Int. Conf. Comput. Vis.*, 1582–1590.