Cascade Deep Networks for Sparse Linear Inverse Problems

Huan Zhang*, Hong Shi*, Wenwu Wang†
*School of Computer Sience and Thechonology, Tianjin University, Tianjin, China
†Centre for Vision Speech and Signal Processing, University of Surrey, Guildford, UK

***Abstract*—Sparse deep networks have been widely used in many linear inverse problems, such as image super-resolution and signal recovery. Its performance is as good as deep learning at the same time its parameters are much less than deep learning. However, when the linear inverse problems involve several linear transformations or the ratio of input dimension to output dimension is large, the performance of a single sparse deep network is poor. In this paper, we propose a cascade sparse deep network to address the above problem. In our model, we trained two cascade sparse networks based on Gregor and LeCun's "learned ISTA" and "learned CoD". The cascade structure can effectively improve the performance as compared to the non-cascade model. We use the proposed methods in image sparse code prediction and signal recovery. The experimental results show that both algorithms perform favorably against a single sparse network.**

I. Introduction

Linear inverse problem which has arisen throughout both engineering and mathematical sciences aims at recovering an original signal from a noisy linear measurement. In this problem, given the output signal $\boldsymbol{y}$ and linear transformation matrix $\boldsymbol{A}$, the input signal $\boldsymbol{x}$ is estimated,

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w} \in \mathbb{R}^M \qquad (1)$$

where $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ and $\boldsymbol{x} \in \mathbb{R}^N$, in many cases, $M \ll N$, we will refer to the problem as a sparse linear inverse problem (see Fig. 1(a) for an illustration).

Examples of sparse linear inverse problems include compressive sensing [1] and sparse coding [2], which have been used in many applications such as signal, image and sound processing.

There are a number of methods proposed to solve this problem. We can roughly divide these methods into two categories based on sparsity constraints, including nonconvex optimization and convex relaxation. The nonconvex optimization method uses an l_0 norm constraint on $\boldsymbol{x}$ (e.g. [3] and [4]), and the convex relaxation method uses an l_1 norm constraint on $\boldsymbol{x}$ (e.g. [5] and [6]).

With the success of deep learning, LeCun and Gregor proposed two algorithms named Learned Iterative Shrinkage-Thresholding Algorithm (LISTA) and Learned Coordinate Descent algorithm (LCoD) [7]. The two algorithms are end to end feed-forward neural network which uses back-propagation and gradient descent to learn parameters. Recently, sparse neural network has been widely used in sparse linear inverse problems [8]–[10] and the accuracy was significantly improved over the traditional methods.

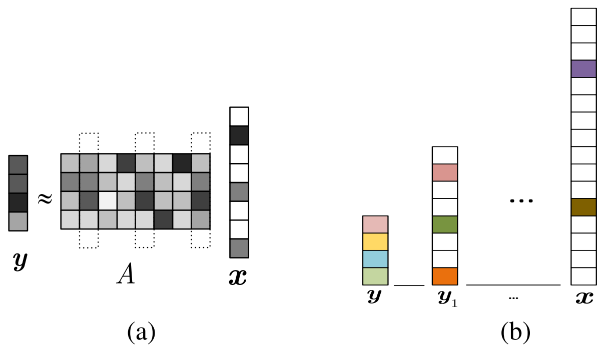


Fig. 1. (a) Illustration of the linear inverse problem. (b) Illustration of the two times transformation of the original signal. When we have the output signal $\boldsymbol{y}$ and we want recovery original signal $\boldsymbol{x}$, we have two ways: 1) $\boldsymbol{y} \rightarrow \boldsymbol{x}$, 2)$\boldsymbol{y} \rightarrow \boldsymbol{y}_1 \rightarrow \cdots \rightarrow \boldsymbol{x}$.

However, the above approaches have two major limitations: 1) for the linear inverse problem, the above methods do not make full use of intermediate information when the original signal is transformed several times; 2) for the image super-resolution problem, the above methods have a limitation caused by a scaling factor, i.e., for every scale we have to retrain the network. To address these limitations, cascade network have been proposed for the image super-resolution [11] and image reconstruction [12] problems. Schlemper et al [12] used cascaded non-linear convolutional neural networks for image super-resolution. Inspired by this, we propose to cascade the sparse networks to solve the sparse linear inverse problem which involves more than one transformations shown in Fig. 1(b). We perform experiments both on image data and randomly generated signals, and compare it with the non-cascade state-of-the-art algorithms. In summary, we focus on the sparse linear inverse problem, which involves more than one transformations, and solve it with efficient and accurate algorithms. The main contributions of this paper are as follows:

1) A cascade sparse coding network is proposed which includes cascade LISTA and cascade LCoD;

2) The proposed algorithm is applied to linear inverse problem to improve signal reconstruction performance.

The remainder of the paper is organized as follows: we discuss related work in Section II. Section III introduces our cascade sparse coding deep networks, and in Section IV we perform extensive experimental comparison with the state-of-the-art none-cascade deep networks. The conclusion is drawn in Section V.

II. Related Works

This section reviews the related approaches for the sparse linear inverse problem. We consider two recent approaches that are most relevant to our proposed algorithm.

A. Iteration Algorithms

The convex constrained objective function [8] for solving the linear inversion problem is

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \tfrac{1}{2} \parallel \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} \parallel_2^2 + \alpha \parallel \boldsymbol{x} \parallel_1 \qquad (2)$$

where $\alpha > 0$ is a coefficient that controls the sparsity level.

1) ISTA and FISTA: One of the popular and simple algorithm for linear inverse problem is the Iterative Shrinkage and Thresholding Algorithm (ISTA) [5]. It iterates the steps ($\hat{\boldsymbol{x}}_0 = \boldsymbol{0}$ and for $t = 0, 1, 2, ...$)

$$\hat{\boldsymbol{x}}_{t+1} = \boldsymbol{\eta}_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_t - \beta \boldsymbol{A}^T(\boldsymbol{A}\hat{\boldsymbol{x}}_t - \boldsymbol{y})) \qquad (3)$$

where β is a step size, and $\boldsymbol{\eta}_{\boldsymbol{\theta}}$ is a "soft thresholding" shrinkage function, defined as

$$[\boldsymbol{\eta}_{\boldsymbol{\theta}}(\boldsymbol{r})]_i = sgn(\boldsymbol{r}_i)(|\boldsymbol{r}_i| - \boldsymbol{\theta}_i)_+ \qquad (4)$$

in which $\boldsymbol{\theta} = \boldsymbol{1}_M \beta\alpha$ is a threshold, $\boldsymbol{1}_M$ is a column vector with M ones. Fast ISTA (FISTA) comes from ISTA, but its convergence speed is much faster than ISTA, because the iterative shrinkage operator of FISTA not only employs the previous $\hat{\boldsymbol{x}}_t$ but also uses a very specific linear combination of two previous points $\{\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{x}}_{t-1}\}$. However, ISTA and FISTA are not monotone algorithms. Beck and Teboulle proposed a "monotone FISTA", see [13] for details.

2) CoD: Coordinate Descent algorithm [14] updates a code component which will be the largest modification of the objective function value, while (F)ISTA updates all the code components simultaneously. Therefore, its time complexity is lower. When we incorporate CoD in a network, it also runs fast. In Section III, we will analyze its time complexity.

B. Deep Learning

In Eqn. (3), let $\boldsymbol{B} = \beta \boldsymbol{A}^T$, $\boldsymbol{S} = \boldsymbol{I} - \beta \boldsymbol{A}^T \boldsymbol{A}$, we have

$$\hat{\boldsymbol{x}}_{t+1} = \boldsymbol{\eta}_{\boldsymbol{\theta}}(\boldsymbol{B}\boldsymbol{y} + \boldsymbol{S}\hat{\boldsymbol{x}}_t) \qquad (5)$$

and this creates a T-layer feed-forward neural network, shown in Fig. 2(a). In this LISTA deep network, the parameter space is $\boldsymbol{\Theta} = [\boldsymbol{B}, \boldsymbol{S}, \boldsymbol{\theta}]$, the training data is $(\boldsymbol{y}^{(d)}, \boldsymbol{x}^{(d)})_{d=1}^{D}$ and the loss function is

$$\mathcal{L}_T(\boldsymbol{\Theta}) = \frac{1}{D} \sum_{d=0}^{(D-1)} \frac{1}{2} \parallel \boldsymbol{x}^{(d)} - f(\boldsymbol{y}^{(d)}, \boldsymbol{\Theta}) \parallel_2^2 \qquad (6)$$

where $\boldsymbol{x} = f(\boldsymbol{y}^{(d)}, \boldsymbol{\Theta})$. We learn the parameters by minimizing the loss function through the training samples. In this operation, we use back-propagation to calculate the gradient of the objective function, and then use the chain rule and gradient descent to update the parameters, see details in [7].

The operating mechanism in LCoD is the same as that in LISTA, with the only difference in the number of parameters that need to be updated each time. The runtime of LCoD is much less than LISTA.

III. Cascade Structure

A. Problem Formulation

We consider that the signals we observed are obtained by multiple linear transforms of the original signal, expressed as

$$\boldsymbol{y} = \boldsymbol{A}_1\boldsymbol{A}_2 \cdots \boldsymbol{A}_{S-1}\boldsymbol{A}_S\boldsymbol{x} + \boldsymbol{w} \in \mathbb{R}^M \qquad (7)$$

where $\boldsymbol{A}_1 \in \mathbb{R}^{M\times K_1}$, $\boldsymbol{A}_2 \in \mathbb{R}^{K_1\times K_2} \cdots \boldsymbol{A}_S \in \mathbb{R}^{K_{S-1}\times N}$ and $M < K_1 < \cdots < K_{S-1} < N$. When $S = 1$, the problem degenerates to what we have introduced at the beginning. We can solve this problem by considering $\boldsymbol{A} = \boldsymbol{A}_1\boldsymbol{A}_2 \cdots \boldsymbol{A}_{S-1}\boldsymbol{A}_S$, and using the traditional methods. In doing so we may lose some intermediate useful information. Empirically, we can get a better solution when the ratio of the column to the row of linear transformation matrix $\boldsymbol{A}_i$ is not very large which will be analyzed in Section III. The intermediate information can be defined as

$$\begin{cases} \boldsymbol{y}_0 = \boldsymbol{A}_1\boldsymbol{y}_1 \in \mathbb{R}^M \\ \boldsymbol{y}_1 = \boldsymbol{A}_2\boldsymbol{y}_2 \in \mathbb{R}^{K_1} \\ \cdots \\ \boldsymbol{y}_{S-1} = \boldsymbol{A}_S\boldsymbol{y}_S \in \mathbb{R}^{K_{S-1}} \\ \boldsymbol{y}_S = \boldsymbol{x} + \boldsymbol{w} \in \mathbb{R}^N \end{cases}$$

where $\boldsymbol{y}_j (j = 1, 2, ..., S)$ is the jth intermediate observation, and the noise $\boldsymbol{w}$ can be placed in any one of the equation alone. The above formula can be simplified as

$$\boldsymbol{y}_{j-1} = \boldsymbol{A}_j\boldsymbol{y}_j, \quad j = 1, 2, ..., S \qquad (8)$$

where $\boldsymbol{y} = \boldsymbol{y}_0$ and $\boldsymbol{x} = \boldsymbol{y}_S + \boldsymbol{w}$.

B. Cascade Algorithm

In order to make full use of the intermediate information and improve the final result, we propose a cascade algorithm.

According to the solution of conventional sparse linear inverse problem, we design the objective function as

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \tfrac{1}{2} \parallel \boldsymbol{y} - \boldsymbol{A}_1\boldsymbol{A}_2 \cdots \boldsymbol{A}_{S-1}\boldsymbol{A}_S\boldsymbol{x} \parallel_2^2 + \alpha \parallel \boldsymbol{x} \parallel_1 \qquad (9)$$

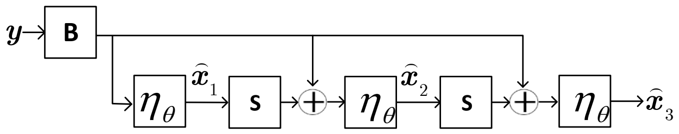
The above formula can be split through a greedy way as

$$\hat{\boldsymbol{y}}_j = \arg\min_{\boldsymbol{y}_j} \tfrac{1}{2} \parallel \boldsymbol{y}_{j-1} - \boldsymbol{A}_j\boldsymbol{y}_j \parallel_2^2 + \alpha_i \parallel \boldsymbol{y}_j \parallel_1 \qquad (10)$$

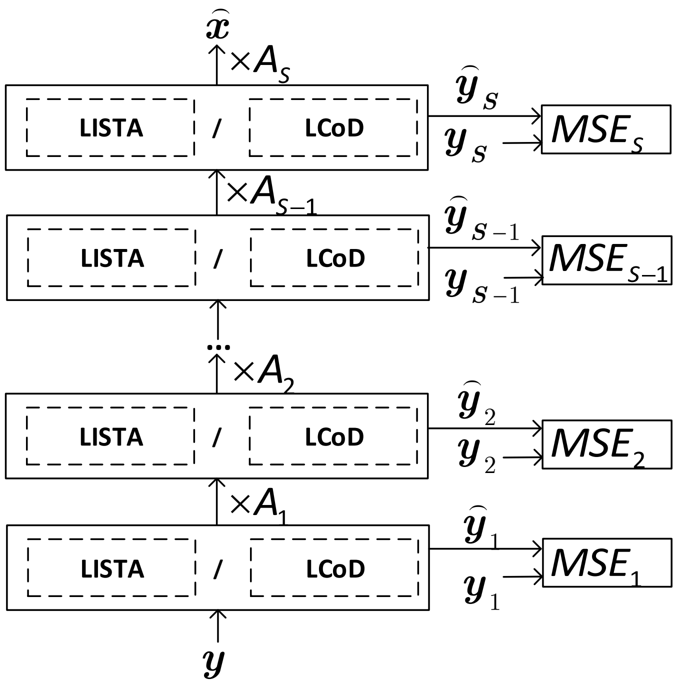
then we can get a multi-scale objective function as

$$\min_{\{\boldsymbol{\Theta}_j\}} \sum_j \sum_d \parallel \boldsymbol{y}_j^{(d)} - f(\boldsymbol{y}_{j-1}^{(d)}; \boldsymbol{\Theta}_j) \parallel \qquad (11)$$

where $j \in (1, 2...S)$ is the number of linear transformations and $d \in (1, 2...D)$ is the number of training samples. For each j, we train a single sparse deep network by minimizing the mean square error and then cascade the networks from end to end. The cascade algorithm is also a deep network, where the output of each lower layer deep network is the input of the



(a) Learned ISTA with iterations $T = 3$



(b) Cascade deep networks

Fig. 2. Illustration of the sparse deep networks. (a) is a single LISTA with 3 iterations, the parameters $\boldsymbol{\Theta} = [\boldsymbol{B}, \boldsymbol{S}, \boldsymbol{\theta}]$ are learned from training data. (b) is the cascade deep network, when train the cascade network, several single deep networks are in parallel, and when test the cascade network, several single deep networks are in series. This is the first time to use LCoD in sparse linear inverse problem, and it can run faster than other similar sparse deep networks. The jth deep network's parameters $\boldsymbol{\Theta}_j = [\boldsymbol{B}_j, \boldsymbol{S}_j, \boldsymbol{\theta}_j]$ are learned in parallel.

next higher layer deep network shown in Fig. 2(b). With the increase in S, we can gradually reconstruct the original signal.

However, the cascade deep networks are not very popular in deep learning, because the error will accumulate from the lower to higher and then lead to a large deviation from the result in some cases. In order to reduce the effect, we trained every network independently, and tested it through cascade network. With the training of individual networks, the error will not be propagated between two adjacent networks and the parameters we obtained will be more accurate. In the test, due to the fact that the ratio of original signal dimension to output signal dimension is not large any more, every single learned deep network can achieve more accurate results. Finally we could gain a satisfactory result.

C. Learning and Testing

We have discussed the problem that we want to address and the structure of cascade deep networks in the above two sections. In this section, we present the algorithm for parameters learning in Algorithm 1 and testing in Algorithm 2.

Algorithm 1: parameters learning of cascade LISTA

```
Input: Original signal $\boldsymbol{x}$, intermediate signal $\boldsymbol{y}_j$,
    observation signal $\boldsymbol{y}$ and linear transformation matrix
    $\boldsymbol{A}_j$;
Output: learned parameters $\boldsymbol{\Theta} = [\boldsymbol{B}, \boldsymbol{S}, \boldsymbol{\theta}]$;
 1: Set $j = 1$;
 2: repeat
 3:     Initialize $\boldsymbol{B}_j = \beta \boldsymbol{A}_j^T$, $\boldsymbol{S}_j = \boldsymbol{I}_j - \beta \boldsymbol{A}_j^T \boldsymbol{A}$,
        $\boldsymbol{\theta}_j = \boldsymbol{1}_{K_j} \beta\alpha$ and the single deep network layer $T$;
 4:     repeat
 5:         Take a set of training data $(\boldsymbol{y}_{j-1}^{(d)}, \boldsymbol{y}_j^{(d)})$;
 6:         Calculate forward network for $T$ times by using
            Eqn. (5);
 7:         Update parameters $\boldsymbol{\Theta}_j = [\boldsymbol{B}_j, \boldsymbol{S}_j, \boldsymbol{\theta}_j]$ through
            back-propagation;
 8:     until $MSE_j$ convergence
 9: until $j > S$
10: return $\boldsymbol{\Theta}$;
```

Algorithm 2: testing of cascade LISTA

```
Input: Observation signal $\boldsymbol{y}$ and learned parameters
    $\boldsymbol{\Theta} = [\boldsymbol{B}, \boldsymbol{S}, \boldsymbol{\theta}]$;
Output: Original signal $\boldsymbol{x}$
 1: Set $j = 1$, Initialize $\boldsymbol{x} = 0$;
 2: repeat
 3:     Using learned parameter $\boldsymbol{\Theta}_j$ in Eqn.(5), calculate $\boldsymbol{y}_j$
        until convergence;
 4: until $j > S$
 5: return $\boldsymbol{x}$;
```

D. Analysis and Discussion

The effectiveness and complexity of the proposed algorithms are discussed below.

1) Effectiveness: As shown in Fig. 3, with the decrease in the ratio of N/M, the convergence rate and the final accuracy achieved can be improved in single deep networks, and this suggests that reducing the ratio is effective for dealing with multiple transforms and large ratio problems. This observation forms the basis of our proposed algorithm. For a linear inverse problem with a large N/M, we convert it to a problem with $N/K_{S-1}, K_{S-1}/K_{S-2}, ..., K_1/M$, and the ratio can be on average decreased to $\frac{N/M}{S}$ in every single deep network, so that we can get a better result in every deep network and in the end a better result in the cascade structure. Since the error

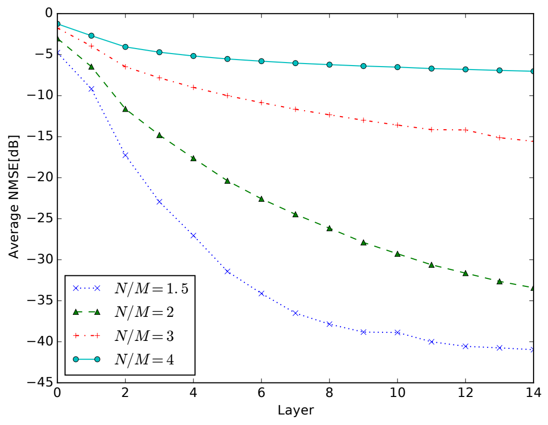


Fig. 3. Illustration of the average NMSE achieved by a single LISTA with different ratios of N/M.

will be propagated between two adjacent networks, the value of $\frac{N/M}{S}$ should not be too small.

2) Time complexity: The time complexity of the cascade algorithm is polynomial. The complexity of LISTA per iteration is $O(N^2)$ while for LCoD this is $O(N)$ [7]. Since the cascade structure is trained in parallel, the time complexity is the same as that for a single network. As for the test part, the time complexity is $\underbrace{O(K_1^2)+O(K_2^2)+\cdots+O(N^2)}_{S} \leqslant O(SN^2)$ for CLISTA and $\underbrace{O(K_1)+O(K_2)+\cdots+O(N)}_{S} \leqslant O(SN)$ for CLCoD per iteration, where S is the number of the cascade networks. When N is large, CLCoD is significantly faster than CLISTA.

IV. Experiments

In this section, we compare our proposed cascade structure with single deep networks on two different types of data, including randomly generated signal [8] and image data. For fair evaluation, we use the original source codes [9] [7] in which the parameters of each method are tuned for best performance. The evaluated algorithms are: FISTA, CoD [14], LISTA, LCoD [7], LAMP [9], LVAMP [9], CLISTA and CLCoD.

A. Experiment on Randomly Generated Signal

We randomly generate the dataset in a way similar to [9]. In this dataset, $\boldsymbol{Y} \in \mathbb{R}^{M\times 1000}$, $\boldsymbol{A}_1 \in \mathbb{R}^{M\times K_1}$ and $\boldsymbol{Y}_1 \in \mathbb{R}^{K_1\times 1000}$ are the output signal, linear transformation matrix and intermediate signal, respectively. In order to compare the performance of the cascade algorithm with the single network algorithm, we sparsely decompose $\boldsymbol{Y}_1$ to obtain the intermediate variables $\boldsymbol{A}_2 \in \mathbb{R}^{K_1\times N}$ and original signals $\boldsymbol{X} \in \mathbb{R}^{N\times 1000}$. It can be thought of as a linear inverse model for $S = 2$, and the model is

$$\boldsymbol{Y} = \boldsymbol{A}_1\boldsymbol{A}_2\boldsymbol{X} + \boldsymbol{W} \qquad and \qquad \boldsymbol{Y}_1 = \boldsymbol{A}_2\boldsymbol{X} \tag{12}$$

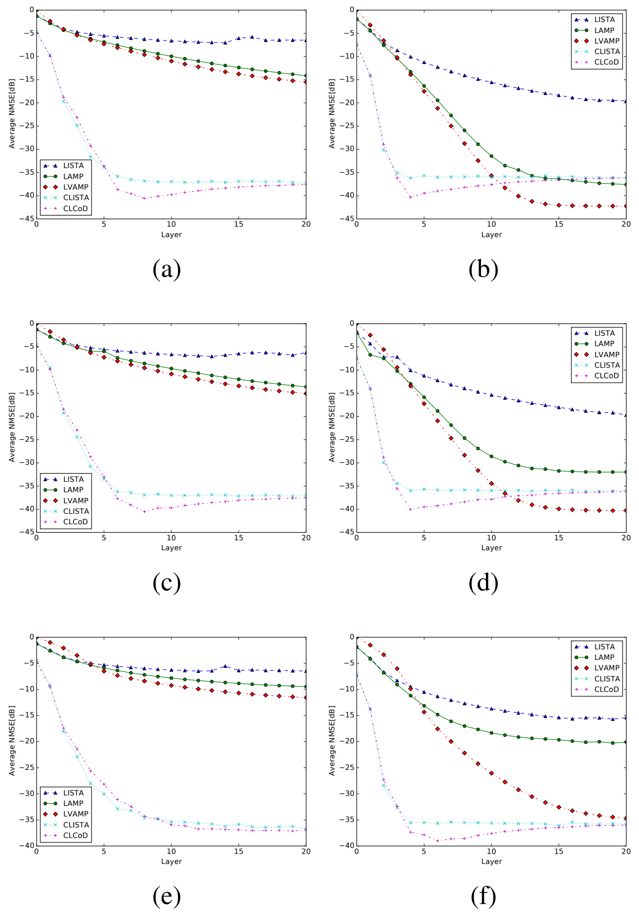


Fig. 4. Illustration of the result of different algorithms on linear inverse problem. (a) For an i.i.d. Gaussian $\boldsymbol{A}_1$ and the dimension is from 250 to 1000. (b) For an i.i.d. Gaussian $\boldsymbol{A}_1$ and the dimension is from 250 to 700. (c) For $\boldsymbol{A}_1$ with the condition number 15 and the dimension is from 250 to 1000. (d) For $\boldsymbol{A}_1$ with the condition number 15 and the dimension is from 250 to 700. (e) For $\boldsymbol{A}_1$ with the condition number 100 and the dimension is from 250 to 1000. (f) For $\boldsymbol{A}_1$ with the condition number 100 and the dimension is from 250 to 700.

We experiment on i.i.d. Gaussian $\boldsymbol{A}_1$ and $\boldsymbol{A}_1$ whose condition number is 15 and 100 respectively. A matrix which possesses a low condition number is said to be well-conditioned, while a high condition number is said to be ill-conditioned. We consider two different sets of dimensions, one is $M = 250$, $K_1 = 500$ and $N = 1000$, the other is $M = 250$, $K_1 = 375$ and $N = 700$.

The parameters are set as follows. The sparsity penalty coefficient is set to $\alpha = 0.1$. The learning rate is set to 10^{-4}.

The evaluation metric is the average normalized MSE (NMSE), where NMSE$\triangleq\| \hat{\boldsymbol{x}} - \boldsymbol{x} \|_2^2 \,/\, \| \boldsymbol{x} \|_2^2$. In order to facilitate the observation, we show the results in decibel as $10\log_{10}$(NMSE).

Fig. 4 shows that for either i.i.d. Gaussian $\boldsymbol{A}_1$ or $\boldsymbol{A}_1$ with the condition number 15 or 100, the cascade sparse deep networks have a much better performance both on convergence speed and final accuracy. Especially for dimension from 250 to 1000, the CLISTA is 30 dB better than other single networks.

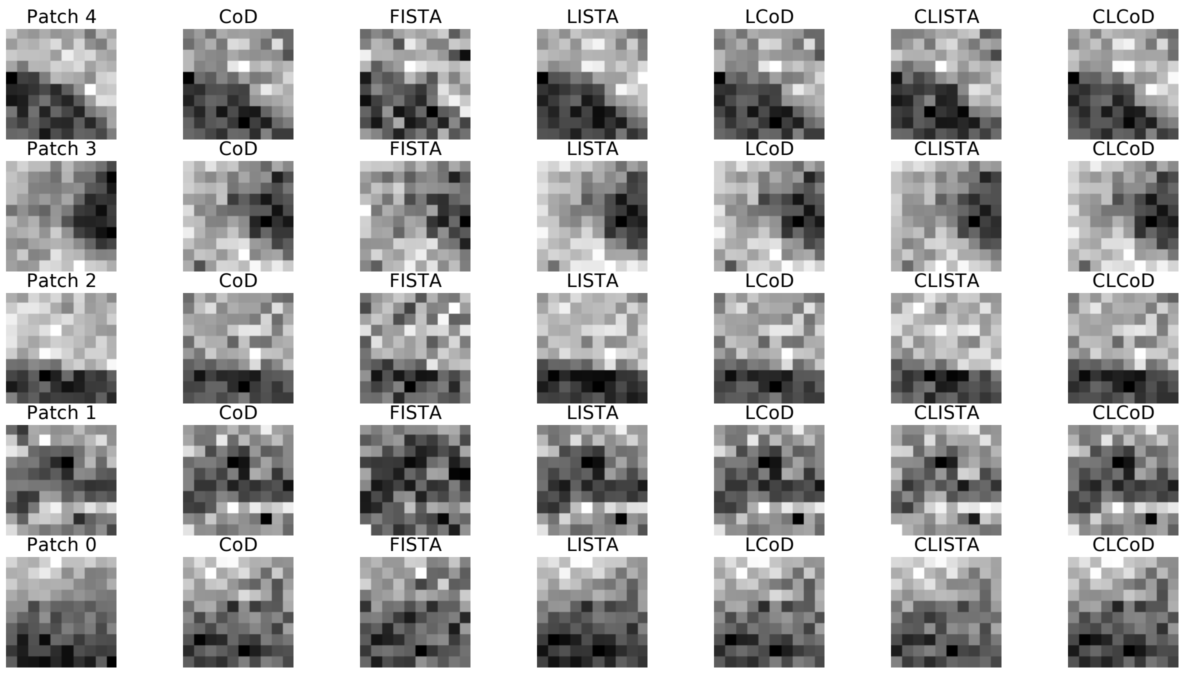


Fig. 5. Screenshots from five image patches and the reconstruction results of six algorithms.

TABLE I
MSE COMPARISONS OF DIFFERENT ALGORITHMS (ROWS), DIFFERENT DATA SET (COLUMNS). **BOLD** FONTS INDICATE THE BEST PERFORMANCE.

	Berkeley	Holidays	VOC2012	Flower	Scene
CoD	0.2320	0.4495	0.1117	0.1176	0.3674
FISTA	0.2838	0.2470	0.3402	0.2493	0.2516
LISTA	0.3126	0.2681	0.3923	0.3042	0.3061
LCoD	0.1227	0.0097	0.0402	0.1907	0.1975
CLISTA	**0.0014**	**0.0012**	**0.0011**	**0.0009**	**0.0011**
CLCoD	0.0024	0.0025	0.0025	0.0018	0.0017

TABLE II
MSE COMPARISONS OF DIFFERENT ALGORITHMS (ROWS), DIFFERENT DATA SET (COLUMNS). **BOLD** FONTS INDICATE THE BEST PERFORMANCE.

	Berkeley	Holidays	VOC2012	Flower	Scene
CoD	0.7792	0.3073	0.0117	0.0292	0.6843
FISTA	0.2798	0.1961	0.2911	0.2689	0.2656
LISTA	0.3319	0.2409	0.3496	0.3109	0.3229
LCoD	0.7371	0.2583	0.0047	0.0298	0.6347
CLISTA	**0.0021**	**0.0037**	**0.0024**	**0.0023**	**0.0018**
CLCoD	0.0050	0.0070	0.0025	0.0049	0.0043

For dimension from 250 to 700, the state-of-the-art single sparse network LAMP or LVAMP needs to use 10 more layers than CLISTA or CLCoD to reach the same accuracy.

B. Experiment on Image Patches

In this set of experiments we compare the performance of different methods on image sparse code prediction. We perform experiments on 5 image datasets, including (1) Berkeley image (2) INRIA Holidays [15] (3) VOC2012 data [16] (4) 102 Category Flower [17] and (5) RGB-NIR Scene [18]. We randomly select small blocks of 10×10 pixels in each image datasets, then we remove the patches whose standard deviation

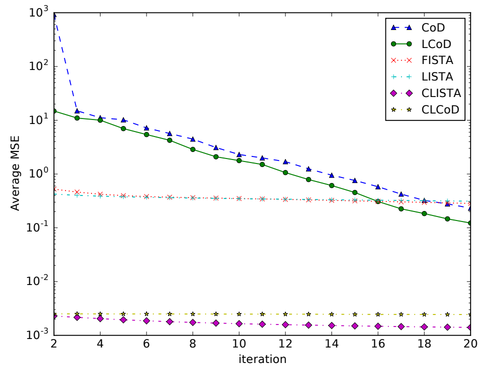


Fig. 6. Average MSE of six algorithms versus iteration numbers.

is smaller than 15 and preprocess each patch to remove its mean and normalize its variance. We set sparse coefficient as $\alpha = 0.5$. We change the dimension of patches from 100 to 200 then to 400 and 100 to 150 then to 300.

1) Train the linear transformation matrix: For every image patch, we can convert it to a vector $\boldsymbol{y} \in \mathbb{R}^{100}$, then we train the dictionary of basis vector $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$. At beginning, we initialize two dictionaries containing random i.i.d. Gaussian elements and next we first train $\boldsymbol{A}_1$ as follows: 1) get a vector from the training set $\boldsymbol{Y}^d$; 2) use $\boldsymbol{A}_1$ to calculate the optimal intermediate code $\boldsymbol{Y}_1^{*d}$ by the FISTA algorithm; 3) update $\boldsymbol{A}_1$ through stochastic gradient $\boldsymbol{A}_1 = \boldsymbol{A}_1 - \frac{1}{t} d\text{Eqn.}(10)/d\boldsymbol{A}_1$ with $j = 1$ in Eqn.(10) and normalize the column of $\boldsymbol{A}_1$; 4) iterate. Then we train $\boldsymbol{A}_2$ in the same way as for $\boldsymbol{A}_1$ except the optimal intermediate code $\boldsymbol{X}^{*d}$ which is different in step

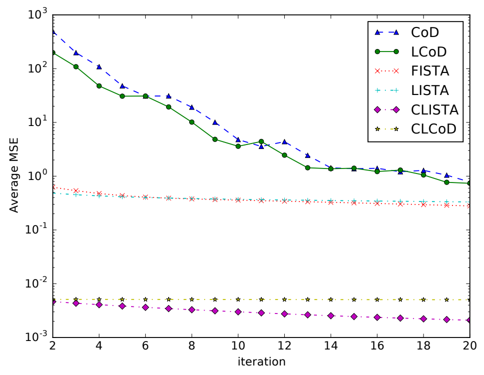


Fig. 7. Average MSE of six algorithms versus iteration numbers.

2) and the value of j is different in step 3).

2) Train the cascade learned algorithm: After gaining $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, we train the CLISTA or CLCoD as follows: 1) get a vector from training set $\boldsymbol{Y}^d$; 2) use $\boldsymbol{A}_1$ to calculate the optimal immediate code $\boldsymbol{Y}_1^{*d}$ by the FISTA algorithm; 3) use back-propagation to train the first LISTA or LCoD network; 4) use $\boldsymbol{Y}_1^{*d}$ and $\boldsymbol{A}_1$ to calculate the optimal immediate code $\boldsymbol{X}^{*d}$ by the FISTA algorithm; 5) use back-propagation to train the second LISTA or LCoD network; 6) iterate.

Fig. 5 shows the optimal image patches and image patches obtained by different algorithms. We can see empirically that the image patches obtained by the cascade algorithm are closer to the original image patches. Table I shows the performance of different algorithms on different datasets when the iteration number is 20 and the original dimension of $\boldsymbol{x}$ is 400. The MSE of the cascade algorithm can reach 10^{-3}, while other non-cascade methods only to 10^{-1}. Table II shows the performance when the original dimension of $\boldsymbol{x}$ is 300. Both tables show that the cascade algorithm can greatly improve the performance. Figs 6 and 7 show as the number of iterations changes, the error results from different algorithms on the Berkeley image dataset. Figs 6 and 7 show the results for $\boldsymbol{x}$ dimension of 300 and 400 respectively. From them we can know that regardless of the number of iterations, the cascading approach works better than the non-cascade networks.

C. Analyze the Number of Deep Networks

As the error will accumulate between the networks, in order to know the suitable number S, we perform an experiment on the Berkeley dataset with CLISTA for the ratio of output dimension and input dimension at 100 and 400 respectively. We found that $S = 6$ is a suitable number for the cascade deep networks to achieve good results.

V. Conclusion

In this paper, we have proposed a cascade deep learning algorithm to deal with the linear inverse problem which involves multiple linear transformations and the ratio of dimension between original signal to output signal is large. Numerous comparisons with state-of-the-art algorithms on both randomly generated signals and image patches have shown that the proposed algorithm achieves favorable performance in terms of accuracy and convergence speed. In future work, we will apply this model to other problems such as compressive sensing and sparse coding.

Acknowledgment

This work is partly supported by the National Nature Science Foundation of China under Grant 61502335.

References

[1] H. Yin, J. Li, Y. Chai, and S. X. Yang, "A survey on distributed compressed sensing: theory and applications," *Frontiers of Computer Science*, vol. 8, no. 6, pp. 893–904, 2014.

[2] A. Xiao, Z. Shao, and Z. Wang, "Sparse coding for super-resolution via k-means classification," in *IEEE International Conference on Multimedia & Expo Workshops, ICME Workshops, Hong Kong, China, July 10-14*, 2017, pp. 363–368.

[3] V. K. Singh, A. K. Rai, and M. Kumar, "Sparse data recovery using optimized orthogonal matching pursuit for wsns," in *The 8th International Conference on Ambient Systems, Networks and Technologies (ANT2017) / The 7th International Conference on Sustainable Energy Information Technology (SEIT2017), Madeira, Portugal,16-19 May*, 2017, pp. 210–216.

[4] R. Chartrand, "Exact reconstruction of sparse signals via nonconvex minimization," *IEEE Signal Process. Lett.*, vol. 14, no. 10, pp. 707–710, 2007.

[5] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[6] D. L. Donoho and Y. Tsaig, "Fast solution of l_1-norm minimization problems when the solution may be sparse," *IEEE Trans. Information Theory*, vol. 54, no. 11, pp. 4789–4812, 2008.

[7] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 399–406.

[8] M. Borgerding and P. Schniter, "Onsager-corrected deep learning for sparse linear inverse problems," in *IEEE Global Conference on Signal and Information Processing, GlobalSIP*, 2016, pp. 227–231.

[9] M. Borgerding, P. Schniter, and S. Rangan, "Amp-inspired deep networks for sparse linear inverse problems," *IEEE Trans. Signal Processing*, vol. 65, no. 16, pp. 4293–4308, 2017.

[10] Z. Wang, D. Liu, J. Yang, W. Han, and T. S. Huang, "Deep networks for image super-resolution with sparse prior," in *2015 IEEE International Conference on Computer Vision, ICCV*, 2015, pp. 370–378.

[11] Z. Cui, H. Chang, S. Shan, B. Zhong, and X. Chen, "Deep network cascade for image super-resolution," in *13th European Conference on Computer Vision, ECCV*, 2014, pp. 49–64.

[12] J. Schlemper, J. Caballero, J. V. Hajnal, A. N. Price, and D. Rueckert, "A deep cascade of convolutional neural networks for MR image reconstruction," in *25th International Conference Information Processing in Medical Imaging, IPMI*, 2017, pp. 647–658.

[13] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," *IEEE Trans. Image Processing*, vol. 18, no. 11, pp. 2419–2434, 2009.

[14] S. J. Wright, "Coordinate descent algorithms," *Math. Program.*, vol. 151, no. 1, pp. 3–34, 2015.

[15] H. Jegou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *10th European Conference on Computer Vision, ECCV*, 2008, pp. 304–317.

[16] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[17] M. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Sixth Indian Conference on Computer Vision, Graphics & Image Processing, ICVGIP*, 2008, pp. 722–729.

[18] M. Brown and S. Süsstrunk, "Multispectral SIFT for scene category recognition," in *Computer Vision and Pattern Recognition (CVPR)*, June 2011, pp. 177–184.