Knowledge-Based Systems 225 (2021) 107115

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

L_1 model-driven recursive multi-scale denoising network for image super-resolution

Zhongfan Sun [a], Jianwei Zhao [a,b,*], Zhenghua Zhou [a], Qingqing Gao [a]

[a] Department of Information Sciences and Mathematics, College of Sciences, China Jiliang University, Hangzhou 310018, PR China
[b] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, PR China

ARTICLE INFO

Article history:
Received 11 November 2020
Received in revised form 25 April 2021
Accepted 30 April 2021
Available online 5 May 2021

Keywords:
Super-resolution
Deep learning
ℓ_1 model-driven
Iteration algorithm
Denoising network
Majorization–minimization algorithm
Soft thresholding operator

ABSTRACT

Most existing deep learning based single-image super-resolution (SISR) methods mainly improve the reconstruction performances from the perspective of data-driven, i.e., widening or deepening the networks according to the huge scale of the training data. However, it will bring a huge amount of weights and biases, and cost the expensive computations. Recently, some people have proposed a new frame for designing the deep networks according to the algorithms deduced from the ℓ_2-optimization problem. But they did not consider the case with outliers. Since ℓ_1-norm can describe the sparsity of the outliers better than ℓ_2-norm, we propose an effective deep network designed according to the new algorithm deduced from the ℓ_1-optimization problem. In our proposed method, an effective iterative algorithm for the ℓ_1 reconstructed optimization problem is deduced based on the split Bregman algorithm, majorization–minimization algorithm, and soft thresholding operator. Then according to the deduced iterative algorithm, an effective deep network, named ℓ_1 Model-Driven Recursive Multi-Scale Denoising Network (ℓ_1-MRMDN), is designed. Due to the iteration form of the deduced algorithm, the proposed ℓ_1-MRMDN contains an inner recursion and an outer recursion. Therefore, our proposed method can not only relieve its sensitiveness to the outliers because of the ℓ_1 data fidelity term, but also avoid designing the deep network blindly via the guidance of prior knowledge. Extensive experimental results illustrate that our proposed method is superior to some related popular SISR methods.

1. Introduction

Single-image super-resolution (SISR) has attracted many attentions in computer vision recently because of its wide applications in real life, such as security monitoring, medical diagnosis [1], and aerial imaging [2]. It aims to reconstruct a high-resolution (HR) image $\boldsymbol{x}$ from a low-resolution (LR) one $\boldsymbol{y}$, that is,

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\Delta}, \tag{1.1}$$

where $\boldsymbol{A}$ is a subsampling matrix/operation, and $\boldsymbol{\Delta}$ is an additive noise. Obviously, it is an ill-posed inverse problem. Although a lot of literatures have been proposed for SISR based on interpolation-based methods [3–5], reconstruction-based methods [6–10], and learning-based methods [11–29], it is still a challenging task to propose a robust and effective reconstruction method.

Since Dong et al. [13] first introduced the convolutional neural networks (CNNs) in SISR to propose a super-resolution convolutional neural network (SRCNN), many deep-learning based SISR methods have emerged quickly because of their powerful ability of extracting high-level features. In order to accelerate SRCNN, Dong et al. [14] proposed a fast super-resolution convolutional neural network (FSRCNN) method by learning the mapping from the original LR image (without interpolation) to the HR one directly via a deconvolution layer at the end of the network. In the meanwhile, considering speeding up the reconstruction, Shi et al. [15] proposed an efficient sub-pixel convolutional neural network (ESPCN) method by taking LR image as input and using the pixel-shuffle at the end of network to enlarge the feature map. In order to improve the reconstruction accuracy, Kim et al. [16] proposed a very deep super-resolution (VDSR) method by deepening SRCNN from 3 to 20 layers via the residual connection. Lai et al. [17] proposed a Laplacian pyramid super-resolution network (LapSRN) method by reconstructing the HR image through doubling the size of LR image at each pyramid level.

Recently, more and more complex deep networks have been proposed for SISR. Tai et al. [18] proposed a very deep persistent

* Corresponding author at: Department of Information Sciences and Mathematics, College of Sciences, China Jiliang University, Hangzhou 310018, PR China.
E-mail address: zhaojw@amss.ac.cn (J. Zhao).

https://doi.org/10.1016/j.knosys.2021.107115
0950-7051/

memory network (MemNet) method that introduces a memory block to mine the persistent memory adaptively. Li et al. [19] proposed a convolutional anchored regression network (CARN) for SISR by constructing some regression blocks to learn the features, anchors, and regressors jointly. Hui et al. [20] proposed a lightweight information multi-distillation network (IMDN) by extracting the hierarchical features and selecting them for reconstructing the HR images. Haris et al. [21] proposed a deep back-projection network (DBPN) method by using the proposed up-projection unit and down-projection unit alternately to make use of the mutual dependencies between LR image and HR image. Li et al. [22] proposed a multi-scale residual network (MSRN) method by constructing a multi-scale residual block (MSRB) containing some convolutional kernels with different sizes to extract the features under different scales. Zhang et al. [23] proposed a residual channel attention network (RCAN) method that designs a residual-in-residual (RIR) structure to deepen the network. Zhang et al. [24] proposed a residual dense network (RDN) method by applying dense connections to fuse the local features and the global features.

All above deep-learning based SISR methods mainly improve the reconstruction performances from the perspective of data-driven, i.e., widening or deepening the networks according to the huge scale of the training data. Although they can get better reconstruction performances, they generate a huge amount of weights and biases. In order to simplify the neural networks and alleviate the model storage/transmission burden, Li et al. [25] proposed a differentiable pruning method via the hypernetworks for automatic network pruning based on ℓ_1 sparsity regularization and the proximal gradient solver. Some people introduce a recursive structure into the deep networks. Kim et al. [26] proposed a deeply-recursive convolutional network (DRCN) method by sharing the weights and biases in a convolutional layer based on the idea of recursion to decrease the amount of weights and biases while still keep the reconstruction performances. Tai et al. [27] proposed a deep recursive residual network (DRRN) method by combining the ideas of recursive learning and residual learning to design a deeper network for obtaining better performances. Although these recursive networks can reduce the amount of weights and biases greatly, they cannot reduce the computational complexity.

Since it is difficult to further improve the performances of the networks just relying on the idea of data-driven , people begin to explore a new way by fusing the data-driven and the model-driven. For example, Ren et al. proposed a profile enhancement and denoising statistics priors (PEP-DSP) network [28], and Dong et al. proposed a denoising prior driven deep neural network (DPDNN) [29] by taking the prior knowledge as the theoretical guidance for designing the effective deep networks. These two methods both transform the super-resolution inverse problem (1.1) as the following optimization problem from the Bayesian perspective:

$$\boldsymbol{x} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda J(\boldsymbol{x}), \tag{1.2}$$

where $\|\cdot\|_2$ is the ℓ_2-norm, $J(\boldsymbol{x})$ represents the regularization term with respect to $\boldsymbol{x}$, and λ is a regularization parameter. Then PEP-DSP method proposes a split Bregman iteration algorithm to divide (1.2) into two subproblems (i.e., inversion subproblem and denoising subproblem), and designs a profile enhancement prior network (PENet) for the inversion subproblem, a denoising statistics prior (DSP) network for the denoising subproblem, respectively. While for the DPDNN method, it adopts the half-quadratic splitting method to divide (1.2) into two subproblems, and deduces an denoising-based iteration algorithm for designing the deep network.

Table 1
An example of the reconstruction performances for a deep network (MSRN) trained with ℓ_2, $\ell_{2,1}$, and ℓ_1 loss, respectively for the noise images.

Model	Set5		Set14		B100		Urban100	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
ℓ_2-norm	30.44	0.8368	27.66	0.7367	26.82	0.6903	25.26	0.7338
$\ell_{2,1}$-norm	30.56	0.8420	27.69	0.7414	26.86	0.6949	25.35	0.7402
ℓ_1-norm	30.58	0.8422	27.71	0.7414	26.83	0.6932	25.33	0.7398

Both PEP-DSP and DPDNN study the super-resolution from a novel framework that combines the advantages of reconstruction-based methods and the learning-based methods. However, they may be sensitive to the outliers because they are deduced from the optimization problem (1.2) with the ℓ_2-norm data fidelity term $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$. As we know, it is a common phenomenon that the collected LR images may contain the noises in real life. So it is instructive to discuss the reconstruction problem with outliers.

With the development of compressed sensing theory [30, 31], some optimization forms for dealing with the noises have been proposed by using different norms, e.g. ℓ_1-norm and $\ell_{2,1}$-norm [32]. Generally, the ℓ_2-norm data fidelity term penalizes larger errors but is more tolerant to small errors [33], so ℓ_2 optimization model is more sensitive to the outliers than the sparse norm. For example, Table 1 shows the reconstruction performances ($\times 4$) of MSRN [19] trained with ℓ_2, $\ell_{2,1}$, and ℓ_1 loss, respectively on the testing datasets full of Gaussian noise with the standard deviation 3. Observed from the results, the MSRN with ℓ_1-norm or $\ell_{2,1}$ can produce better reconstruction performances than the MSRN with ℓ_2-norm. And the MSRN with ℓ_1-norm shows the best performances on Set5 and Set14, while the MSRN with $\ell_{2,1}$-norm shows the best performances on B100 and Urban100.

Since ℓ_1-norm describes the sparsity of the outliers better than ℓ_2-norm and its form is simpler than $\ell_{2,1}$-norm, we study the SISR problem with outliers by designing a new deep network deduced from the ℓ_1-optimization model. That is, considering the outliers in the training data, we transform the inverse problem (1.1) as the following ℓ_1-optimization problem

$$\boldsymbol{x} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_1 + \lambda J(\boldsymbol{x}), \tag{1.3}$$

where $\|\cdot\|_1$ is the ℓ_1-norm. Different from the easy solvability of the inversion subproblem in DPDNN method by the conjugate gradient because of its differentiability of the ℓ_2-norm, the optimization problem (1.3) is more difficult to be solved because of its non-differentiability of the ℓ_1-norm.

In this paper, we first deduce an effective iterative algorithm for above ℓ_1 optimization problem (1.3) based on the split Bregman method, the majorization–minimization (MM) algorithm, and the soft thresholding operator. Then, according to above deduced iterative algorithm, we design an effective deep network for super-resolution. We name the network ℓ_1 model-driven recursive multi-scale denoising network (ℓ_1-MRMDN). Due to the iteration form of the deduced algorithm, the proposed ℓ_1-MRMDN contains an inner recursion and an outer recursion. Obviously, it is different from the classical recursive networks that are designed without theoretical guidance. Therefore, our proposed ℓ_1-MRMDN method can give a guidance for designing a compact deep network by fusing expert priors and deep learning, which avoids designing the deep network blindly. Furthermore, our method can also relieve its sensitiveness to outliers with ℓ_1 data fidelity term based on the sparse theory. Concretely, the contributions of this paper are as follows:

- For the SISR problem with the outliers, this paper designs an effective deep network, ℓ_1-MRMDN, according to the proposed iteration algorithm deduced from the ℓ_1-optimization model (1.3).

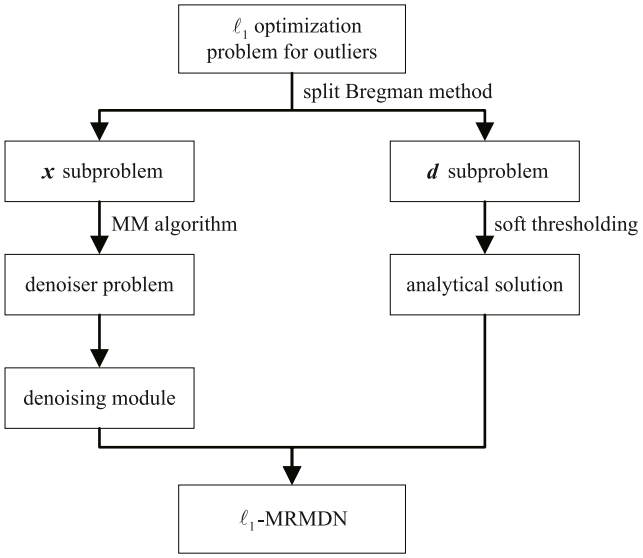

Fig. 1. The flow chart of our proposed ℓ_1-MRMDN method.

- For the non-differentiable ℓ_1-optimization model (1.3), we deduce an effective iterative algorithm based on the split Bregman method, the majorization–minimization algorithm, and the soft thresholding operator.
- This paper gives a way for designing the deep network under the guidance of expert priors, which avoids constructing deep network blindly. Experimental results illustrate that this method is superior to some deep-learning based methods without the guidance of expert priors.

The rest of this paper is organized as follows. Section 2 describes our proposed ℓ_1-MRMDN method in detail. In Section 3, extensive experiments are carried out on some databases. Conclusion is given in Section 4.

2. Proposed ℓ_1-MRMDN method

In this section, we will describe our proposed ℓ_1-MRMDN method in detail. Firstly, we propose an optimization model (1.3) from the inverse problem (1.1) to address the SISR problem with outliers. Secondly, we convert the optimization problem (1.3) to two subproblems: $\boldsymbol{x}$-subproblem and $\boldsymbol{d}$-subproblem by split Bregman method. Thirdly, we deduce an effective iterative algorithm for $\boldsymbol{x}$-subproblem based on the majorization–minimization (MM) algorithm and the soft thresholding operator. For the $\boldsymbol{d}$-subproblem, we give an analytical solution by the soft thresholding operator. Finally, according to the iteration solutions of $\boldsymbol{x}$-subproblem and $\boldsymbol{d}$-subproblem, we design an effective deep network for SISR. The flow chart of our proposed ℓ_1-MRMDN method is shown in Fig. 1.

2.1. Iterative algorithm deduced from ℓ_1 model-driven

Many model-based methods attack the inverse problem (1.1) for SISR by solving an optimization problem, which is often constructed from a Bayesian perspective [34]. In the Bayesian setting, the solution is obtained by maximizing the posterior $p(\boldsymbol{x} \mid \boldsymbol{y})$, which can be described as

$$\begin{aligned}\boldsymbol{x} &= \arg\max_{\boldsymbol{x}} \; p(\boldsymbol{x} \mid \boldsymbol{y}) = \arg\max_{\boldsymbol{x}} \; p(\boldsymbol{y} \mid \boldsymbol{x})p(\boldsymbol{x}) \\ &= \arg\min_{\boldsymbol{x}} \; -\log p(\boldsymbol{y} \mid \boldsymbol{x}) - \log p(\boldsymbol{x}),\end{aligned} \tag{2.1}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectorizations of the images, $\log p(\boldsymbol{y} \mid \boldsymbol{x})$ is the log-likelihood related to the forward imaging model, and $\log p(\boldsymbol{x})$ is the log-prior related to HR images $\boldsymbol{x}$. It is easy to see that the optimization problem (2.1) is equivalent to the following unconstrained problem

$$\boldsymbol{x} = \arg\min_{\boldsymbol{x}} \phi(\boldsymbol{x}) + \lambda J(\boldsymbol{x}), \tag{2.2}$$

where $\phi(\boldsymbol{x})$ is a data fidelity term, and $J(\boldsymbol{x})$ is a regularized term. This generic unconstrained optimization problem can be solved via many optimization algorithms.

PEP-DSP method and DPDNN method replace the data fidelity term $\phi(\boldsymbol{x})$ in (2.2) with an ℓ_2-norm $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2$. As we know, it is a common phenomenon that the collected LR images may contain the outliers in real life. So it is instructive to discuss the reconstruction problem with outliers. Since ℓ_1-norm describes the sparsity of the outliers better than ℓ_2 (see Table 1), in this paper we adopt the ℓ_1 data fidelity term for $\phi(\boldsymbol{x})$ to address the outlier problem. That is,

$$\boldsymbol{x} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_1 + \lambda J(\boldsymbol{x}),$$

where $\|\cdot\|_1$ denotes the ℓ_1-norm of a vector. Different from the easy solvability of the inversion subproblem in DPDNN method by the conjugate gradient because of its differentiability of the ℓ_2-norm, above ℓ_1 optimization problem is more difficult to be solved because of its non-differentiability of the ℓ_1-norm. Hence, we propose a novel method for solving above ℓ_1 optimization problem.

In order to solve above ℓ_1 optimization problem, we take a new variable splitting strategy with $\boldsymbol{d} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}$ instead of $\boldsymbol{d} = \boldsymbol{x}$ in PEP-DSP and DPDNN. The reason is that if we use $\boldsymbol{d} = \boldsymbol{x}$ as PEP-DSP and DPDNN, the subproblem with the ℓ_1 term cannot be directly solved by the soft thresholding operator [35]. Therefore, we transform above ℓ_1 optimization problem with a new variable splitting strategy as follows:

$$(\boldsymbol{x}, \boldsymbol{d}) = \arg\min_{\boldsymbol{x},\boldsymbol{d}} \|\boldsymbol{d}\|_1 + \lambda J(\boldsymbol{x}) \quad \text{s.t.} \;\; \boldsymbol{d} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}, \tag{2.3}$$

where $\boldsymbol{d}$ is an auxiliary variable. Using the split Bregman iteration algorithm, the constrained optimization problem (2.3) can be transformed into an unconstrained optimization form:

$$(\boldsymbol{x}, \boldsymbol{d}) = \arg\min_{\boldsymbol{x},\boldsymbol{d}} \|\boldsymbol{d}\|_1 + \lambda J(\boldsymbol{x}) + \frac{\eta}{2}\|\boldsymbol{d} - (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})\|_2^2, \tag{2.4}$$

where η is a penalty parameter.

Define $E(\boldsymbol{x}, \boldsymbol{d}) = \|\boldsymbol{d}\|_1 + \lambda J(\boldsymbol{x})$, then (2.4) can be rewritten as

$$(\boldsymbol{x}, \boldsymbol{d}) = \arg\min_{\boldsymbol{x},\boldsymbol{d}} E(\boldsymbol{x}, \boldsymbol{d}) + \frac{\eta}{2}\|\boldsymbol{d} - (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})\|_2^2. \tag{2.5}$$

For above optimization problem (2.5), we solve it recursively as follows:

$$(\boldsymbol{x}^{k+1}, \boldsymbol{d}^{k+1}) = \arg\min_{\boldsymbol{x},\boldsymbol{d}} D_E^{\boldsymbol{p}}(\boldsymbol{x}, \boldsymbol{d}, \boldsymbol{x}^k, \boldsymbol{d}^k) + \frac{\eta}{2}\|\boldsymbol{d} - (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})\|_2^2, \tag{2.6}$$

where $D_E^{\boldsymbol{p}}(\boldsymbol{x}, \boldsymbol{d}, \boldsymbol{x}^k, \boldsymbol{d}^k) = E(\boldsymbol{x}, \boldsymbol{d}) - \langle \boldsymbol{p}_{\boldsymbol{x}}^k, \boldsymbol{x} - \boldsymbol{x}^k \rangle - \langle \boldsymbol{p}_{\boldsymbol{d}}^k, \boldsymbol{d} - \boldsymbol{d}^k \rangle$, is the Bregman distance of the convex function E, and $\boldsymbol{p}_{\boldsymbol{x}}^k$, $\boldsymbol{p}_{\boldsymbol{d}}^k$ are the subgradient of the function E with respect to the variables $\boldsymbol{x}$ and $\boldsymbol{d}$.

With the explicit formulas of $\boldsymbol{p}_{\boldsymbol{x}}^k$ and $\boldsymbol{p}_{\boldsymbol{d}}^k$ in the paper [36], the (2.6) can be simplified as

$$\begin{aligned}(\boldsymbol{x}^{k+1}, \boldsymbol{d}^{k+1}) &= \arg\min_{\boldsymbol{x},\boldsymbol{d}} \|\boldsymbol{d}\|_1 + \lambda J(\boldsymbol{x}) + \frac{\lambda}{2}\|\boldsymbol{d} - (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}) - \boldsymbol{b}^k\|^2, \\ \boldsymbol{b}^{k+1} &= \boldsymbol{b}^k + \left((\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}) - \boldsymbol{d}^k\right),\end{aligned} \tag{2.7}$$

where $\boldsymbol{b}$ is a variable similar to add the error back. Now above iteration can be decomposed into following subproblems:

$$\boldsymbol{x}^{k+1} = \arg\min_{\boldsymbol{x}} \lambda J(\boldsymbol{x}) + \frac{\eta}{2}\left\|\boldsymbol{A}\boldsymbol{x} - \left(\boldsymbol{y} - \boldsymbol{b}^k + \boldsymbol{d}^k\right)\right\|_2^2, \tag{2.8}$$

$$\boldsymbol{d}^{k+1} = \arg\min_{\boldsymbol{d}} \|\boldsymbol{d}\|_1 + \frac{\eta}{2}\left\|\boldsymbol{d} - \left(\boldsymbol{A}\boldsymbol{x}^{k+1} - \boldsymbol{y} + \boldsymbol{b}^k\right)\right\|_2^2, \tag{2.9}$$

$$\boldsymbol{b}^{k+1} = \boldsymbol{b}^k + \boldsymbol{A}\boldsymbol{x}^{k+1} - \boldsymbol{y} - \boldsymbol{d}^{k+1}. \tag{2.10}$$

It is noted that the decomposed subproblems (2.8)–(2.10) are different with the subproblems in PEP-DSP and DPDNN. The type of $\boldsymbol{x}$-subproblem (2.8) in our method is similar to the $\boldsymbol{v}$-subproblem in PEP-DSP and DPDNN. Furthermore, different from the $\boldsymbol{x}$-subproblem with ℓ_2-norm in PEP-DSP and DPDNN, the $\boldsymbol{d}$-subproblem (2.9) is with ℓ_1-norm, which cannot be solved directly by the classic conjugate gradient as DPDNN.

Now we give the convergence theorem about above iterative framework.

Theorem 2.1. *Let $\boldsymbol{x}^*$ be a solution of the optimization problem (1.3), $J(\boldsymbol{x})$ be a smooth convex function, and $\eta > 0$, then for the iterations (2.8)–(2.10),*

$$\lim_{k\to+\infty} \|\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{y}\|_1 + \lambda J(\boldsymbol{x}^k) = \|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{y}\|_1 + \lambda J(\boldsymbol{x}^*). \tag{2.11}$$

Furthermore, when (1.3) has a unique solution,

$$\lim_{k\to+\infty} \|\boldsymbol{x}^k - \boldsymbol{x}^*\|_2 = 0. \tag{2.12}$$

Proof. See Appendix. □

Next we give the solutions of $\boldsymbol{x}$-subproblem (2.8) and $\boldsymbol{d}$-subproblem (2.9), respectively.

(1) Solution of $\boldsymbol{x}$-subproblem. It is noted that the $\boldsymbol{x}$-subproblem (2.8) in our method returns to the form of ℓ_2-norm optimization problem as (1.2) in DPDNN. Although we can solve it with the same iteration algorithm in DPDNN, it will make the solution have more iterations and extra hyperparameters. Hence, in our method we use the majorization–minimization algorithm [37] to solve the $\boldsymbol{x}$-subproblem.

Let

$$U(\boldsymbol{x}) = \lambda J(\boldsymbol{x}) + \frac{\eta}{2}\left\|\boldsymbol{A}\boldsymbol{x} - \left(\boldsymbol{y} - \boldsymbol{b}^k + \boldsymbol{d}^k\right)\right\|_2^2, \tag{2.13}$$

then the $\boldsymbol{x}$-subproblem becomes to the following minimization problem:

$$\boldsymbol{x}^{k+1} = \arg\min_{\boldsymbol{x}} U(\boldsymbol{x}). \tag{2.14}$$

According to the majorization–minimization algorithm, we take

$$\boldsymbol{x}^{k+1,l+1} = \arg\min_{\boldsymbol{x}} \tilde{U}\left(\boldsymbol{x}; \boldsymbol{x}^{k+1,l}\right), \tag{2.15}$$

where $\boldsymbol{x}^{k+1,l+1}$ denotes the $(l+1)$-th iteration solution for the $\boldsymbol{x}^{k+1}$, and the optimizer $\tilde{U}\left(\boldsymbol{x}; \boldsymbol{x}^{k+1,l}\right)$ must satisfy the following two conditions:

$$\begin{aligned} &\tilde{U}\left(\boldsymbol{x}; \boldsymbol{x}^{k+1,l}\right) > U(\boldsymbol{x}) \text{ for all } \boldsymbol{x} \neq \boldsymbol{x}^{k+1,l}, \text{ and} \\ &\tilde{U}\left(\boldsymbol{x}^{k+1,l}; \boldsymbol{x}^{k+1,l}\right) = U\left(\boldsymbol{x}^{k+1,l}\right). \end{aligned} \tag{2.16}$$

Here we take

$$\begin{aligned} \tilde{U}\left(\boldsymbol{x}, \boldsymbol{x}^{k+1,0}\right) = &\arg\min_{\boldsymbol{x}} \lambda J(\boldsymbol{x}) + \frac{\eta}{2}\left\|\boldsymbol{A}\boldsymbol{x} - \left(\boldsymbol{y} - \boldsymbol{b}^k + \boldsymbol{d}^k\right)\right\|_2^2 \\ &+ \frac{\alpha\eta}{2}\left\|\boldsymbol{x} - \boldsymbol{x}^{k+1,0}\right\|_2^2 - \frac{\eta}{2}\left\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}\boldsymbol{x}^{k+1,0}\right\|_2^2, \end{aligned} \tag{2.17}$$

where $\alpha > \|\boldsymbol{A}\|_2^2$. It is checked that $\tilde{U}\left(\boldsymbol{x}, \boldsymbol{x}^{k+1,0}\right)$ satisfy the conditions (2.16). Using the property of inner product, we can simplify above formula (2.17) as following form:

$$\tilde{U}\left(\boldsymbol{x}, \boldsymbol{x}^{k+1,0}\right) = \arg\min_{\boldsymbol{x}} \lambda J(\boldsymbol{x}) + \frac{\alpha\eta}{2}\|\boldsymbol{x} - \boldsymbol{z}\|_2^2 + \boldsymbol{c}, \tag{2.18}$$

where $\boldsymbol{z} = \frac{1}{\alpha}\boldsymbol{A}^\top\left(\boldsymbol{y} - \boldsymbol{b}^k + \boldsymbol{d}^k - \boldsymbol{A}\boldsymbol{x}^{k+1,0}\right) + \boldsymbol{x}^{k+1,0}$, and $\boldsymbol{c} = \frac{\eta}{2}\left(\left\|\boldsymbol{y} - \boldsymbol{b}^k + \boldsymbol{d}^k\right\|_2^2 + \alpha\left\|\boldsymbol{x}^{k+1,0}\right\|_2^2 - \left\|\boldsymbol{A}\boldsymbol{x}^{k+1,0}\right\|_2^2 - \alpha\|\boldsymbol{z}\|_2^2\right)$ that does not depend on $\boldsymbol{x}$.

It is noted that the optimization problem (2.18) is a denoising problem, so it can be solved by some denoising algorithms. That is, we can obtain the approximate solution of $\boldsymbol{x}^{k+1}$ by the following iteration:

$$\begin{aligned} &\boldsymbol{x}^{k+1,l+1} = D\left(\frac{1}{\alpha}\boldsymbol{A}^\top(\boldsymbol{y} - \boldsymbol{b}^k + \boldsymbol{d}^k - \boldsymbol{A}\boldsymbol{x}^{k+1,l}) + \boldsymbol{x}^{k+1,l}\right), \\ &\qquad l = 0, 1, 2, \ldots, \end{aligned} \tag{2.19}$$

where $\boldsymbol{x}^{k+1,0} = \boldsymbol{x}^k$, and D is a denoiser that will be designed based on deep module in the part 2.2.2. Considering the computational complexity and the depth of the network for the denoiser D, we take the result after L iterations as the approximate solution, i.e., $\boldsymbol{x}^{k+1} = \boldsymbol{x}^{k+1,L}$.

(2) Solution of $\boldsymbol{d}$-subproblem. It is noted that the $\boldsymbol{d}$-subproblem (2.9) contains a ℓ_1 term, so it cannot be solved directly by the classic conjugate gradient as PEP-DSP and DPDNN did. Because we use the new variable splitting strategy with $\boldsymbol{d} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}$ in (2.3), the $\boldsymbol{d}$-subproblem (2.9) is fit to the optimization form for iterative shrinkage-thresholding algorithm [35]. With the soft thresholding operator, the $\boldsymbol{d}$-subproblem (2.9) can be solved by its analytical solution as follows:

$$\boldsymbol{d}^{k+1} = soft_{\frac{1}{\eta}}\left(\boldsymbol{A}\boldsymbol{x}^{k+1} - \boldsymbol{y} + \boldsymbol{b}^k\right), \tag{2.20}$$

where

$$soft_{\frac{1}{\eta}}(t) = \begin{cases} t + \frac{1}{\eta}, & t < -\frac{1}{\eta}; \\ 0, & |\,t\,| < \frac{1}{\eta}; \\ t - \frac{1}{\eta}, & t > \frac{1}{\eta}. \end{cases} \tag{2.21}$$

Now we summarize above process of solving as the following Algorithm 1 for guiding the construction of the deep network.

Algorithm 1 The iterative algorithm for guiding the construction of the deep network.

Initialize: Set observation matrix $\boldsymbol{A}$, iterations L, K, $\eta > 0$, $k = 0, l = 0$, $\boldsymbol{d}^0 = \boldsymbol{b}^0 = 0$, and initialize $\boldsymbol{x}$ as $\boldsymbol{x}^0 = \boldsymbol{A}^\top\boldsymbol{y}$.

While $k < K$ **do**

- **While** $l < L$ **do**
  - Using the denoiser D, we obtain the solution $\boldsymbol{x}^{k+1} = \boldsymbol{x}^{k+1,L}$ of the $\boldsymbol{x}$-subproblem by the following formula
    $$\boldsymbol{x}^{k+1,l+1} = D\left(\frac{1}{\alpha}\boldsymbol{A}^\top(\boldsymbol{y} - \boldsymbol{b}^k + \boldsymbol{d}^k - \boldsymbol{A}\boldsymbol{x}^{k+1,l}) + \boldsymbol{x}^{k+1,l}\right),$$
    where $\boldsymbol{x}^{k+1,0} = \boldsymbol{x}^k$;
  - $l = l + 1$;

  End while
- Using soft thresholding operator, we obtain the solution of the $\boldsymbol{d}$-subproblem by the following formula
  $$\boldsymbol{d}^{k+1} = soft_{\frac{1}{\eta}}\left(\boldsymbol{A}\boldsymbol{x}^{k+1} - \boldsymbol{y} + \boldsymbol{b}^k\right);$$
- Update the auxiliary variable $\boldsymbol{b}$ with $\boldsymbol{b}^{k+1} = \boldsymbol{b}^k + \boldsymbol{A}\boldsymbol{x}^{k+1} - \boldsymbol{y} - \boldsymbol{d}^{k+1}$;
- $k = k + 1$.

End while

Output: $\boldsymbol{x}^K$

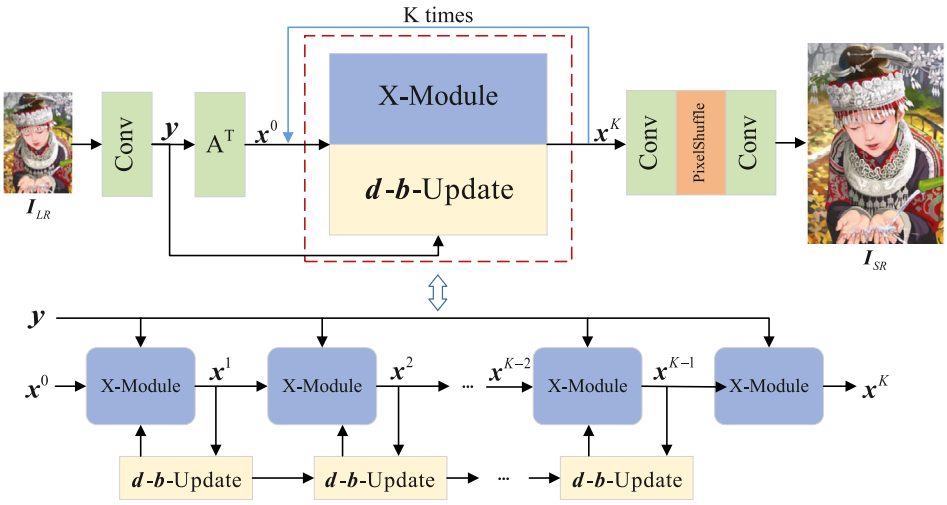

Fig. 2. The structure of our proposed ℓ_1-MRMDN.

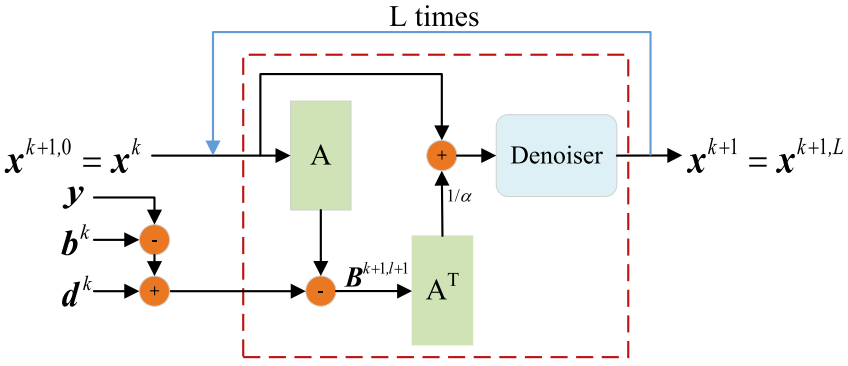

Fig. 3. The structure of X-Module.

2.2. Designing the proposed ℓ_1-MRMDN according to iterative algorithm 1

In Section 2.1, we have deduced the iterative algorithm 1 from the ℓ_1 optimization problem (1.3) for SISR. If we reconstruct the HR image completely according to the iterations, it may need to execute many iterations for attaining the convergence. In order to avoid many iterations and still take full advantage of the expert priors, we will reconstruct the HR image by designing a deep network, ℓ_1-MRMDN, according to the deduced iterative Algorithm 1. Guided by Algorithm 1, our proposed ℓ_1-MRMDN use the recursions of X-Module and $\boldsymbol{d}$-$\boldsymbol{b}$-Update to approximate the solutions of $\boldsymbol{x}$-subproblem and $\boldsymbol{d}$-subproblem, respectively. Therefore, our proposed deep network fuses the expert priors and deep learning, which avoids constructing the network blindly. The structure of our proposed ℓ_1-MRMDN is shown in Fig. 2.

2.2.1. Overall network framework

In this part, we will describe the overall framework of our proposed ℓ_1-MRMDN. For a given LR image $\boldsymbol{I}_{LR} \in \mathbb{R}^{m_1 \times n_1 \times 3}$, we take a convolutional operation on it using a 3×3 convolutional kernel with 64 channels to get the feature map as $\boldsymbol{y}$ in Algorithm 1. The reason for not taking $\boldsymbol{y} = \boldsymbol{I}_{LR}$ directly as DPDNN is that the convolutions in the subsequent X-Module need more channels to avoid converting between the compressing channels and the expanding channels frequently.

For the initial value $\boldsymbol{x}^0 = \boldsymbol{A}^\top \boldsymbol{y}$ in Algorithm 1, we parameterize the degrading matrix $\boldsymbol{A}^\top$ in (1.3) using a 3×3 convolutional kernel with 64 channels. With the initial value $\boldsymbol{x}^0$, we use a recursion operation of X-Module with K times to approximate the solution of $\boldsymbol{x}$-subproblem. In the meanwhile, we update the auxiliary variables $\boldsymbol{d}$ and $\boldsymbol{b}$ for the X-Module according to Algorithm 1. Their mathematical expression can be described as

$$\boldsymbol{x}^{k+1} = \mathcal{F}\left(\boldsymbol{x}^k, \boldsymbol{b}^k, \boldsymbol{d}^k, \boldsymbol{y}\right), k = 0, 1, 2, \ldots, K-1, \tag{2.22}$$

where $\mathcal{F}$ denotes the operation of X-Module, and $\boldsymbol{d}^k, \boldsymbol{b}^k$ are the kth update.

After K iterations of X-Module, we get the feature map $\boldsymbol{x}^K \in \mathbb{R}^{m_1 \times n_1 \times 64}$. At last, the HR image $\boldsymbol{I}_{HR} \in \mathbb{R}^{rm_1 \times rn_1 \times 3}$ can be obtained by magnifying $\boldsymbol{x}^K$ with r times using PixelShuffle method [15].

2.2.2. Structures of modules in network

(1) Structure of X-Module. In this part, we will describe X-Module in detail. The structure of X-Module is shown in Fig. 3.

Given the inputs $\boldsymbol{x}^{k+1,l}, \boldsymbol{y}, \boldsymbol{b}^k$ and $\boldsymbol{d}^k$, we first take a convolutional operation on $\boldsymbol{x}^{k+1,l}$ using a 3×3 convolutional kernel with 64 channels to stand for the degrading matrix $\boldsymbol{A}$, then compute it with $\boldsymbol{y}$, $\boldsymbol{b}^k$, and $\boldsymbol{d}^k$ to get $\boldsymbol{B}^{k+1,l}$. Its mathematical expression is as follows:

$$\boldsymbol{B}^{k+1,l+1} = \boldsymbol{y} - \boldsymbol{b}^k + \boldsymbol{d}^k - \boldsymbol{A}\boldsymbol{x}^{k+1,l}, \quad l = 0, 1, 2, \ldots, L-1, \tag{2.23}$$

where $\boldsymbol{x}^{k+1,0} = \boldsymbol{x}^k$, and L denotes the number of iterations.

Pass $\boldsymbol{B}^{k+1,l}$ through the convolutional operation $\boldsymbol{A}^\top$ and an adaptive parameter $\frac{1}{\alpha}$, then take the sum of above output with $\boldsymbol{x}^{k+1,l}$ as the input of the denoising module D. This process can be described as

$$\boldsymbol{x}^{k+1,l+1} = D\left(\frac{1}{\alpha}\boldsymbol{A}^\top\left(\boldsymbol{B}^{k+1,l+1}\right) + \boldsymbol{x}^{k+1,l}\right), \quad l = 0, 1, 2, \ldots, L-1,$$

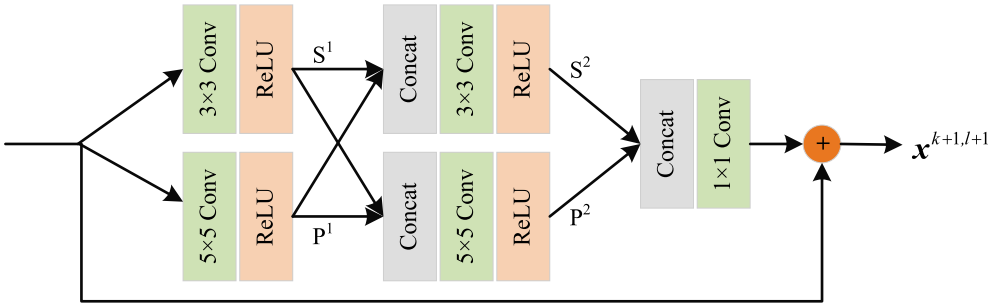

Fig. 4. The structure of MSRB.

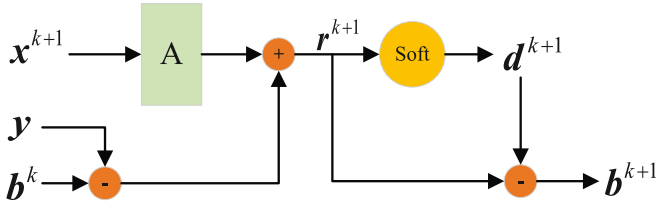

Fig. 5. The process of d-b-update.

(2.24)

where $D(\cdot)$ denotes the denoising module that will be described in the next part. After L recursions, we take $\boldsymbol{x}^{k+1,L}$ as $\boldsymbol{x}^{k+1}$.

(2) Structure of Denoising Module. Here we introduce the structure of denoising module D in X-Module. The role of denoising module D is to help solving the optimization problem (2.19) for $\boldsymbol{x}^{k+1}$. In order to enable the denoiser to be embedded in the network and optimize the remaining parameters of the network simultaneously, we construct the denoiser using deep network. There are many effective deep networks for constructing the denoiser D. In this paper, we apply the effective MSRB [22] as our denoising module. For the convenience of reading, the structure of MSRB is shown in Fig. 4.

(3) Structure of d-b-Update. According to the iteration Algorithm 1, before we update $\boldsymbol{x}$ in each iteration, we need to update the auxiliary variables $\boldsymbol{d}$ and $\boldsymbol{b}$. Correspondingly, $\boldsymbol{d}$ and $\boldsymbol{b}$ need be updated after each recursion for X-Module in our ℓ_1-MRMDN (see Fig. 2). The concrete $\boldsymbol{d}$-$\boldsymbol{b}$-update procedure is shown in Fig. 5.

Given the input variables $\boldsymbol{x}^{k+1}$, $\boldsymbol{y}$ and $\boldsymbol{b}^k$, we first take a convolutional operation on $\boldsymbol{x}^{k+1}$ for the degrading matrix $\boldsymbol{A}$, then compute it with $\boldsymbol{y}$ and $\boldsymbol{b}^k$ to get $\boldsymbol{r}^{k+1}$. Its mathematical expression is as follows:

$$\boldsymbol{r}^{k+1} = \boldsymbol{A}(\boldsymbol{x}^{k+1}) - \boldsymbol{y} + \boldsymbol{b}^k. \tag{2.25}$$

We input $\boldsymbol{r}^{k+1}$ into the soft thresholding operator $soft(\cdot)$ to obtain the updated feature $\boldsymbol{d}^{k+1}$, then use $\boldsymbol{r}^{k+1}$ minus $\boldsymbol{d}^{k+1}$ to get $\boldsymbol{b}^{k+1}$. This process can be expressed as

$$\boldsymbol{d}^{k+1} = Soft_{\frac{1}{\eta}}\left(\boldsymbol{r}^{k+1}\right), \tag{2.26}$$

$$\boldsymbol{b}^{k+1} = \boldsymbol{r}^{k+1} - \boldsymbol{d}^{k+1}. \tag{2.27}$$

Up to now, we have finished the construction of ℓ_1-MRMDN according to the deduced iteration Algorithm 1. The designed network embodies the fusion of expert prior and deep learning well, which avoids designing the deep network blindly. Since the proposed ℓ_1 optimization problem and its iteration algorithm for SISR with the outlier is different from the ℓ_2 optimization problem and its iteration algorithm in DPDNN, the structure of our ℓ_1-MRMDN is of course different with DPDNN. Due to the iteration form of our deduced algorithm, the proposed ℓ_1-MRMDN contains an inner recursion and an outer recursion.

Table 2
Comparison of average PSNR and SSIM of our networks with different L and K on Set5.

Networks	D_1T_{12}	D_2T_6	D_3T_4	D_4T_3
PSNR	32.10	32.19	32.14	32.14
SSIM	0.8933	0.8948	0.8943	0.8940

Table 3
Comparison of average PSNR and SSIM for our networks with different K under $L = 2$ on four datasets.

Dataset criterion	Set5		Set14		B100		Urban100	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
D_2T_2	31.97	0.8919	28.48	0.7789	27.50	0.7333	25.80	0.7764
D_2T_3	32.10	0.8932	28.52	0.7797	27.53	0.7340	25.90	0.7794
D_2T_4	32.09	0.8935	28.54	0.7802	27.54	0.7347	25.95	0.7817
D_2T_5	32.16	0.8942	28.54	0.7801	27.56	0.7350	26.01	0.7832
D_2T_6	32.19	0.8948	28.54	0.7803	27.55	0.7351	26.01	0.7838

Table 4
Comparison of average PSNR and SSIM for our networks with different $1/\eta$ on Set5 dataset.

$1/\eta$	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10
PSNR	32.13	32.12	32.16	32.11	32.07	32.12
SSIM	0.8937	0.8943	0.8942	0.8938	0.8932	0.8935

3. Experiments

In this section, we first carry out some experiments on our proposed ℓ_1-MRMDN to show the significance of our network's construction. Secondly, we carry out some comparison experiments on the reconstruction performances and complexity of our proposed ℓ_1-MRMDN with some other popular SISR methods on non-noise images. Finally, we discuss the effectiveness of our proposed method for the reconstruction images with noises.

3.1. Environment setting

In all the experiments, we take 800 images from DIV2K [38] as samples for training the deep networks. In order to train deep networks sufficiently, we augment the data by rotating or flipping the images randomly. To train the deep networks conveniently, LR images are cropped into image patches of size 48 × 48, and every 16 image patches is taken as a batch. Furthermore, we take Set5 [39], Set14 [40], B100 [41], and Urban100 [42] as our testing datasets.

In the process of training networks, we take Adam [43] as an optimizer for the network's loss, where the concerned parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, and the learning rate is initially taken to be 10^{-4} and decreases into half every 200 epoches. For the parameter α in (2.24), it is initialized to be 1 empirically, and is adjusted adaptively together with the network's weights.

Table 5
The ablation study on our method.

Model	Para#	FLOPs	Set5		Set14		B100		Urban100	
			PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Im-network	565K	5202.9G	31.16	0.8794	27.94	0.7650	27.13	0.7203	25.04	0.7475
Den-network	1343K	448.3G	32.06	0.8930	28.53	0.7798	27.53	0.7340	25.95	0.7809
Ours-ℓ_2	1380K	499.3G	32.01	0.8919	28.53	0.7794	27.54	0.7342	25.96	0.7812
Ours	1380K	499.3G	32.16	0.8942	28.54	0.7801	27.56	0.7350	26.01	0.7832

Table 6
Comparison of average PSNR and SSIM of our proposed ℓ_1-MRMDN with some related popular SISR methods on four datasets under three scales.

Algorithm	Scale	Set5 PSNR/SSIM	Set14 PSNR/SSIM	B100 PSNR/SSIM	Urban100 PSNR/SSIM
Bicubic	x2	33.66/0.9299	30.23/0.8687	29.56/0.8431	26.87/0.8401
A+ [44]	x2	36.54/0.9544	32.28/0.9056	31.21/0.8863	29.20/0.8938
SelfExSR [42]	x2	36.49/0.9537	32.22/0.9034	31.18/0.8855	29.54/0.8967
SRCNN [13]	x2	36.66/0.9542	32.42/0.9063	31.36/0.8879	29.50/0.8946
ESPCN [15]	x2	37.00/0.9559	32.75/0.9098	31.51/0.8939	29.87/0.9065
FSRCNN [14]	x2	37.06/0.9554	32.76/0.9078	31.53/0.8912	29.88/0.9024
VDSR [16]	x2	37.53/0.9587	33.03/0.9124	31.90/0.8960	30.76/0.9140
DRCN [26]	x2	37.63/0.9584	33.06/0.9108	31.85/0.8947	30.76/0.9147
LapSRN [17]	x2	37.52/0.9591	33.08/0.9109	31.80/0.8949	30.41/0.9112
DRRN [27]	x2	37.74/0.9591	33.23/0.9136	32.05/0.8973	31.23/0.9188
MemNet [18]	x2	37.78/0.9597	33.28/0.9142	32.08/0.8978	31.31/0.9195
IDN [45]	x2	37.83/0.9600	33.30/0.9148	32.08/0.8985	31.27/0.9196
DPDNN [29]	x2	37.75/0.960	33.30/0.915	32.09/0.899	31.50/0.922
Ours	x2	37.95/0.9603	33.52/0.9170	32.17/0.8997	32.11/0.9283
Bicubic	x3	30.39/0.8682	27.54/0.7736	27.21/0.7384	24.46/0.7344
A+[44]	x3	32.58/0.9088	29.13/0.8188	28.29/0.7835	26.03/0.7973
SelfExSR [42]	x3	32.58/0.9093	29.16/0.8196	28.29/0.7840	26.44/0.8088
SRCNN [13]	x3	32.75/0.9090	29.30/0.8215	28.41/0.7863	26.24/0.7989
ESPCN [15]	x3	33.02/0.9135	29.49/0.8271	28.50/0.7937	26.41/0.8161
FSRCNN [14]	x3	33.18/0.9140	29.37/0.8240	28.53/0.7910	26.43/0.8080
VDSR [16]	x3	33.66/0.9213	29.77/0.8314	28.82/0.7976	27.14/0.8279
DRCN [26]	x3	33.85/0.9215	29.89/0.8317	28.81/0.7954	27.16/0.8311
LapSRN [17]	x3	33.82/0.9227	29.79/0.8320	28.82/0.7973	27.07/0.8272
DRRN [27]	x3	34.03/0.9244	29.96/0.8349	28.95/0.8004	27.53/0.8378
MemNet [18]	x3	34.09/0.9248	30.00/0.8350	28.96/0.8001	27.56/0.8376
IDN [45]	x3	34.11/0.9253	29.99/0.8354	28.95/0.8013	27.42/0.8359
PEP-DSP [28]	x3	34.16/0.9247	29.98/0.8330	28.51/0.7940	26.64/0.7868
DPDNN [29]	x3	33.93/0.924	30.02/0.836	29.00/0.801	27.61/0.842
Ours	x3	34.35/0.9267	30.31/0.8415	29.07/0.8043	28.05/0.8507
Bicubic	x4	28.42/0.8104	26.00/0.7019	25.96/0.6674	23.14/0.6570
A+ [44]	x4	30.28/0.8603	27.32/0.7491	26.82/0.7087	24.32/0.7183
SelfExSR [42]	x4	30.31/0.8619	27.40/0.7518	26.84/0.7106	24.79/0.7374
SRCNN [13]	x4	30.48/0.8628	27.49/0.7503	26.90/0.7101	24.52/0.7221
ESPCN [15]	x4	30.66/0.8646	27.71/0.7562	26.98/0.7124	24.60/0.7360
FSRCNN [14]	x4	30.72/0.8660	27.61/0.7550	26.98/0.7150	24.62/0.7280
VDSR [16]	x4	31.35/0.8830	28.02/0.7680	27.29/0.0726	25.18/0.7540
DRCN [26]	x4	31.56/0.8810	28.15/0.7627	27.24/0.7150	25.25/0.7530
LapSRN [17]	x4	31.54/0.8855	28.19/0.7720	27.32/0.7280	25.21/0.7553
DRRN [27]	x4	31.68/0.8888	28.21/0.7721	27.38/0.7284	25.44/0.7638
MemNet [18]	x4	31.74/0.8893	28.26/0.7723	27.40/0.7281	25.50/0.7630
IDN [45]	x4	31.82/0.8903	28.25/0.7730	27.41/0.7297	25.41/0.7632
PEP-DSP [28]	x4	31.60/0.8893	28.17/0.7723	26.85/0.7211	24.95/0.7115
DPDNN [29]	x4	31.72/0.889	28.28/0.773	27.44/0.729	25.53/0.768
Ours	x4	32.16/0.8942	28.54/0.7801	27.56/0.7350	26.01/0.7832

All the experiments are implemented under the framework of Pytorch 1.1.0, in Intel® Xeon® Silver 4108 processor with the speed of 1.80 GHz, sixteen kernels, memory of 48GB, GTX2080Ti GPU, Windows 10.

3.2. Model analysis

In this subsection, we first discuss the best setting for the network-structure's parameters to make our proposed method attain the best performances. Secondly, we give an ablation study on our method to prove the effectiveness of each part in our proposed deep network. Finally, we show the convergence of our method by plotting the curves of peak signal to noise ratio (PSNR) and structural similarity (SSIM) with respect to epoches. All above experiments are to show the significance of our network's construction. In order to distinguish the constructed deep networks with different number of recursions about X-Module conveniently, we denote the network with L recursions for denoiser D and K recursions for X-Module as $D_L T_K$.

(1) Balances between L and K

Taking the computation and network's performances into account, we discuss the balance of the recursion number L for denoiser D and the recursion number K for X-Module under $LK = 12$. The comparison of average PSNR and SSIM of our networks with different L and K on Set5 are shown in Table 2.

Observed from Table 2, when the recursion number L for denoiser D is taken to be 2 and the recursion number K for X-Module is taken to be 6, the network D_2T_6 attains the best performances. With the increase of L and the decrease of K, the performances of the networks drop. The reason is that the effect of outer recursion on X-Module is larger than that of inner recursion on denoiser in X-Module.

In view of above analysis, we fix the inner recursion number $L = 2$ to discuss the impact of outer recursion number K further. The comparison of average PSNR and SSIM for our networks with different K under $L = 2$ on four datasets is shown in Table 3.

Observed from Table 3, when the recursion number K is less than or equal to 5, the performances of each network increase gradually with the increase of K on almost datasets. When $K = 6$, there is almost no improvement in the performances for the networks. Considering the computation and performances of the network, we take our deep network with $L = 2$ and $K = 5$ for all the left experiments.

(2) Impact of Hyperparameter $1/\eta$ in Soft Thresholding

In this part, we discuss the network hyperparameter $1/\eta$ of soft thresholding in (2.26) for the update of $\boldsymbol{d}$. The comparison of average PSNR and SSIM for our networks with different $1/\eta$ on Set5 dataset is shown in Table 4.

Observed from Table 4, when $1/\eta = 10^{-2}$, our network has the best PSNR and SSIM. When $1/\eta$ is less or larger 10^{-2}, the performances of our network decrease slightly. The reason is that soft thresholding operator needs an appropriate hyperparameter $1/\eta$. When $1/\eta$ is too small, soft thresholding operator has less effect on the network's performances. When $1/\eta$ is too large, it cannot enforce $\boldsymbol{d} - (\boldsymbol{Ax} - \boldsymbol{y}) \approx \boldsymbol{0}$ well in the optimization model (2.4). Hence, we take $1/\eta = 10^{-2}$ for soft thresholding operator in all the left experiments.

(3) Ablation Study

In this part, we carry out an ablation study on our proposed network to show the effectiveness of the construction of our network. In order to illustrate the effect of the operation in our network that does not take $\boldsymbol{y} = \boldsymbol{I}_{LR}$ directly for avoid converting between the compressing channels and the expanding channels frequently, we give a control network, Im-network, that uses our network but removing the first convolution layer and the final PixelShuffle, and takes the Bicubic interpolation of the LR image as the input. Since Bicubic interpolation amplifies the LR image, Im-network does not need the operation of amplification, for example, PixelShuffle. In order to show the significance of designing our network according to Algorithm 1, we first give a control

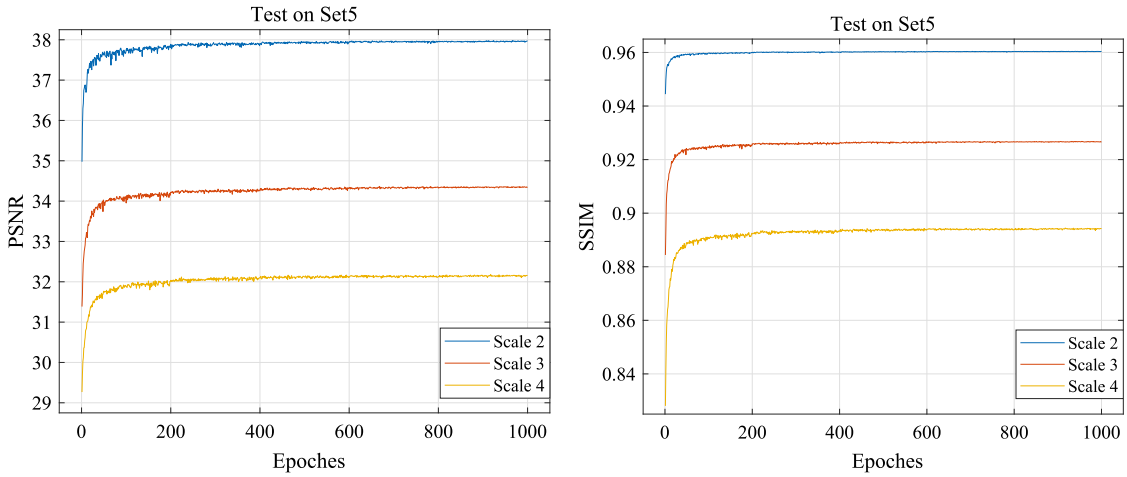

Fig. 6. The curves of average PSNR and SSIM under different scales with respect to epoches.

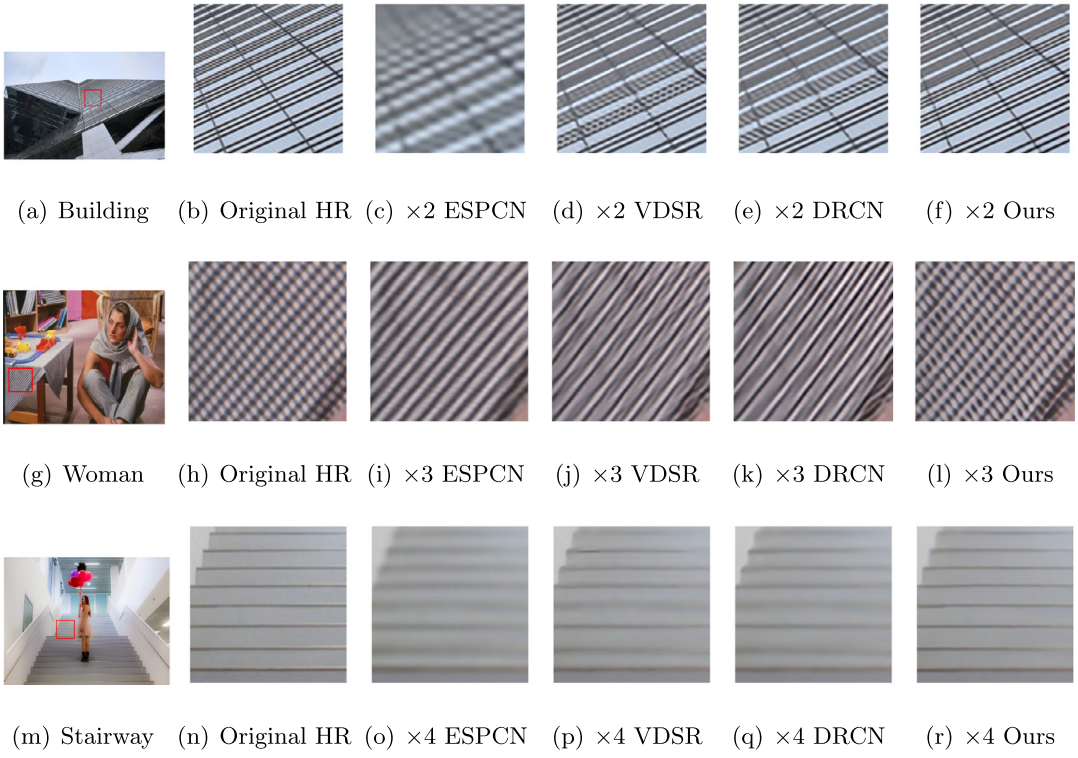

Fig. 7. Comparison of some reconstructed HR images by different SISR methods under three scales.

network, Den-network, that contains *LK* denoising modules *D* in X-Module. In the meanwhile, we give another control network, Ours-ℓ_2, that uses the same network as ours, but takes ℓ_2 loss function for training the network. Their comparisons of average PSNR and SSIM on four datasets are shown in Table 5.

As shown in Table 5, the PSNR and SSIM of our proposed ℓ_1-MRMDN outperforms those of Im-network, Den-network, and Ours-ℓ_2 on four datasets. Although the number of parameters for Im-network is about half of our method, the floating point operations (FLOPs) [46] are ten times of that for our method. The reason for our method over Im-network is that the conversion between compressing and expanding channels frequently in Im-network will reduce its performance greatly because of their restriction on the feature extraction. Although the number of parameters and the computational complexity of Den-network are almost consistent with our method, it has worse performances than our method. The reason is that our network is designed according to the iteration Algorithm. Even if the network of Ours-ℓ_2 is same with our method, but under different loss functions, our method outperforms than Ours-ℓ_2. The reason is that our network is designed and trained according to ℓ_1 optimization model, while Ours-ℓ_2 is trained according to ℓ_2 optimization model. Compared with ℓ_2 loss function, ℓ_1 loss function has better reconstruction performances.

(4) Convergence Analysis

In order to illustrate the convergence of our proposed network well, we give a convergence analysis of our proposed method under three scales. The curves of average PSNR and SSIM with respect to epoches under three scales are given in Fig. 6.

Observed from Fig. 6, with the increase of epoches, the curves of PSNR and SSIM become flat gradually and finally converge

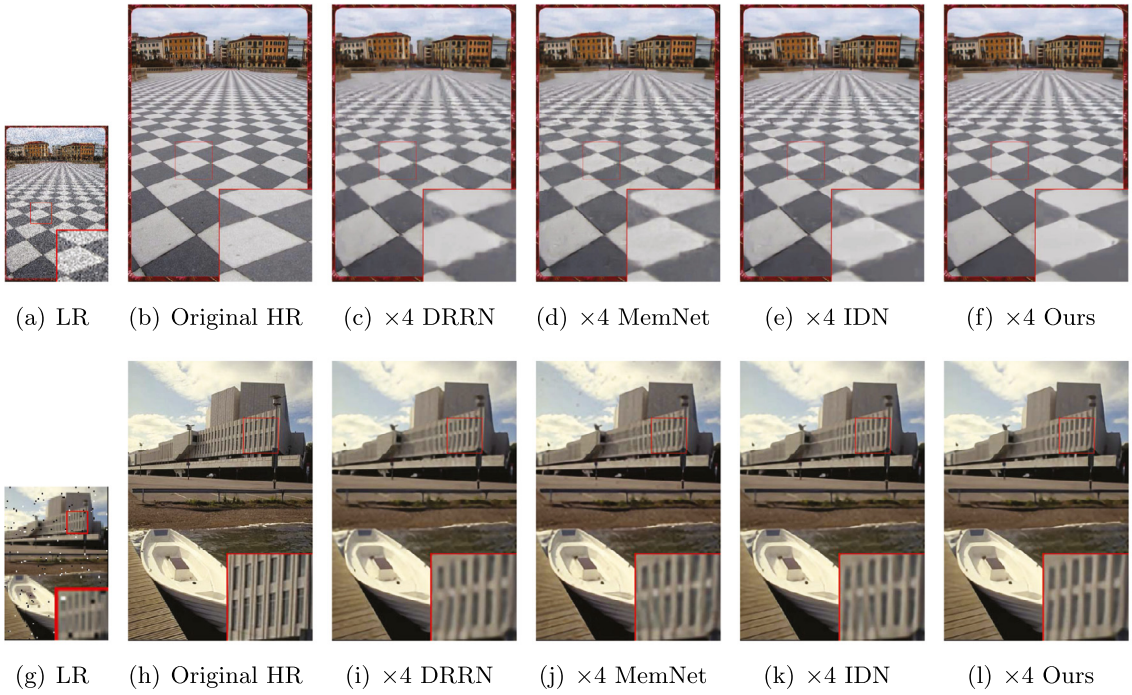

Fig. 8. Comparison of some reconstructed HR images by different SISR methods with Gaussian noises and salt–pepper noises.

to the stable values, which is consistent with our analysis in Theorem 2.1.

3.3. Comparisons with other methods

In this subsection, we compare our proposed ℓ_1-MRMDN with some related popular SISR methods, including Bicubic, A+ [44], SelfExSR [42], SRCNN [13], ESPCN [15], FSRCNN [14], VDSR [16], DRCN [26], LapSRN [17], DRRN [27], MemNet [18], IDN [45], PEPDSP [28], and DPDNN [29]. Their comparison results of average PSNR and SSIM on four datasets under three scales are shown in Table 6.

Observed from Table 6, we can see that our proposed ℓ_1-MRMDN obtains the best PSNR and SSIM among all the SISR methods. Compared with the best results of other SISR methods in Table 6, our method outperforms 0.07 $\sim$ 0.61 and 0.0003 $\sim$ 0.0152 on PSNR and SSIM, respectively. The reason is that our method designs the deep network according to the iteration algorithm deduced from the ℓ_1 optimization model, which considers the expert prior knowledge and deep learning simultaneously.

In Fig. 7, we give some reconstructed HR images by several SISR methods in Table 6. Observed from Figs. 7(c)$\sim$7(f) for "Building", the glasses reconstructed by our method are more distinct than the others. In Figs. 7(i)$\sim$7(l) for "Woman", our method produces more similar textures than the others. In Figs. 7(o)$\sim$7(r) for "Stairway", our method reconstructs clearer stairways than the others. In a summary, our method can reconstruct the textures and details better.

In order to compare our proposed method with some latest SISR methods, we need deepen our network for fairness because these latest methods have more parameters than our network. We deepen our network using four cascade MSRB as the denoising module, denoted as Ours-D. Then the comparison of average PSNR and SSIM of our method and MSRN [22], EDSR [47], and RDN [24] on four datasets under scale ×4 is shown in Table 7.

Observed from Table 7, RDN and EDSR rank the first sometimes on four datasets. Although the PSNR and SSIM of our proposed method are less than those of RDN and EDSR, they are close. The reason is that our proposed network contains two recursion layers according to the iteration algorithm. Hence, our network has far less weights and biases than the other methods.

Table 7
Comparison of average PSNR and SSIM of our method with some latest SISR methods on four datasets under scale ×4.

Model	Set5		Set14		B100		Urban100	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
MSRN	32.07	0.8903	28.60	0.7751	27.52	0.7273	26.04	0.7896
RDN	32.47	0.8990	28.81	0.7871	27.72	0.7419	26.61	0.8028
EDSR	32.46	0.8968	28.80	0.7876	27.71	0.7420	26.64	0.8033
Ours-D	32.37	0.8965	28.71	0.7846	27.64	0.7384	26.35	0.7946

Table 8
Comparison of complexity for our proposed method with some advanced SISR methods on Urban100 dataset.

Networks	MemNet	DRRN	IDN	MSRN	DPDNN	Ours
Run time (s/image)	1.19	1.22	0.62	0.53	0.69	0.64
Network parameters	677K	297K	714K	6366K	1400K	1380K
FLOPs	623.9G	6796.9G	54.9G	367.2G	2301.9G	499.3G
PSNR (db)	25.50	25.44	25.41	26.04	25.53	26.01

3.4. Complexity analysis

In this subsection, we discuss the complexity of our method from the aspects of run time, network parameter, and FLOPs [46]. The comparison results of our proposed method with some advanced SISR methods on Urban100 dataset are shown in Table 8.

Observed from Table 8, although the number of parameters in MemNet and DRRN are less than that of our proposed method, they have less PSNR and get more run time and FLOPs than ours because the network takes the Bicubic interpolated image as input. Although IDN has less run time, parameters, and FLOPs than ours, it produces the lowest PSNR because of the shallow network. The PSNR of our method far exceeds above three methods. Although the FLOPs of our method is higher than MSRN, the number of parameters of ours is less than MSRN because

Table 9
Comparison of average PSNR and SSIM of our method with some other methods on images with Gaussian noise and salt–pepper noise.

Model	Gaussian	Set5		Set14		B100		Urban100	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
DRRN	10	29.17	0.8361	26.74	0.7139	26.10	0.6654	24.27	0.7107
IDN		29.26	0.8383	26.84	0.7193	26.18	0.6712	24.36	0.7176
MemNet		29.36	0.8412	26.92	0.7217	26.21	0.6722	24.54	0.7227
Ours		29.37	0.8412	26.96	0.7226	26.24	0.6738	24.53	0.7254
DRRN	30	26.51	0.7790	24.70	0.6507	24.69	0.6171	22.67	0.6454
IDN		26.67	0.7838	25.00	0.6616	24.90	0.6283	22.98	0.6616
MemNet		26.75	0.7857	25.00	0.6616	24.87	0.6259	23.01	0.6609
Ours		26.84	0.7906	25.08	0.6658	24.97	0.6319	23.01	0.6673
Model	Salt–pepper	Set5		Set14		B100		Urban100	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
MemNet	0.01	31.25	0.8837	28.10	0.7735	27.14	0.7277	25.60	0.7682
DRRN		31.48	0.8851	28.21	0.7714	27.29	0.7248	25.38	0.7587
IDN		31.75	0.8880	28.34	0.7747	27.39	0.7288	25.64	0.7696
Ours		31.98	0.8922	28.46	0.7778	27.48	0.7321	25.92	0.7801
MemNet	0.05	29.78	0.8378	26.97	0.7250	26.30	0.6839	24.45	0.7081
IDN		31.20	0.8816	27.99	0.7655	27.12	0.7188	25.34	0.7606
DRRN		31.41	0.8853	28.17	0.7700	27.24	0.7232	25.35	0.7570
Ours		31.49	0.8856	28.12	0.7688	27.22	0.7221	25.64	0.7713

our method uses a recursive strategy to share the parameters. Our method is almost same as DPDNN in terms of running time and the number of parameters, but it is superior to DPDNN on FLOPs and PSNR. In summary, our method achieves considerable performances with less running time and fewer parameters. The reason is that we utilize the effective prior knowledge to design the deep network.

3.5. Effect on noise images

As ℓ_1 model is robust to outlier, we compare our proposed ℓ_1-MRMDN with DRRN, MemNet, and IDN on noise images. Here we add two types of noises into LR images for training. One is the Gaussian noise with the standard deviation 10 and 30, and the other is the salt–pepper noise with the density 0.01 and 0.05. Their average PSNR and SSIM for LR images with Gaussian noise and salt–pepper noise are shown in Table 9.

Observed from Table 9, under different noises with different scales, the rank of DRRN, MemNet, and IDN have changed, while our proposed ℓ_1-MRMDN still keeps the best reconstruction performances in most cases. It implies that different deep networks have different sensitivity to different type of noise. For example, MemNet is more robust to Gaussian noise than to salt–pepper noise. While our ℓ_1-MRMDN is robust to both Gaussian noise and salt–pepper noise. The reason is that our network deduced by ℓ_1 model is more robust to noises than the deep networks deduced or trained by ℓ_2 model.

In Fig. 8, we give some HR images reconstructed by the SISR methods in Table 9 under Gaussian noise with the standard deviation 30 and salt–pepper noise with the density 0.01.

As shown in Fig. 8, the LR images 8(a) of "Ground",X4SLR "Architecture" are blurring because of the added Gaussian noise and salt–pepper noise, respectively. However, our proposed method and the other SISR methods still show the good reconstruction results. Furthermore, our method gets better clarity than others. For example, as shown in Figs. 8(c)∼8(f) under Gaussian noise, our method reconstructs clearer edges than the other methods. Similarly, in Figs. 8(i)∼8(l) under salt-and-pepper noise, the architecture reconstructed by the other methods is relatively blurry, while the one reconstructed by our method has more distinct textures. In a summary, our method can also reconstruct better HR images even under noises.

4. Conclusions

Different from existing deep-learning based SISR methods that improve the networks' performances by widening or deepening the networks directly, we study the SISR under a new frame by fusing the data-driven and the model-driven as DPDNN. But different with the DPDNN that designs the deep network according to the iteration algorithm deduced from ℓ_2 optimization model, we address the SISR with outliers. Since ℓ_1-norm describes the sparsity of the outliers better than ℓ_2-norm, we design a new deep network, ℓ_1-MRMDN, according to the new iterative algorithm deduced from the ℓ_1 optimization model. Our proposed ℓ_1-MRMDN method first deduces an effective iterative algorithm from the ℓ_1 optimization problem based on the split Bregman method, the majorization–minimization algorithm, and the soft thresholding operator. Then according to above deduced iterative algorithm, we design an effective deep network that contains an inner recursion and an outer recursion. Therefore, our proposed ℓ_1-MRMDN method can embody the fusion of expert prior and deep learning well, which avoids designing the deep network blindly. What is more, our method can also relieve the sensitiveness to outliers because it considers the ℓ_1-constraint in the process of designing the network. Extensive experiments show that our proposed method is superior to some popular SISR methods.

CRediT authorship contribution statement

Zhongfan Sun: Methodology, Software, Validation, Writing - original draft, Investigation. **Jianwei Zhao:** Conceptualization, Writing - review & editing, Supervision, Funding acquisition. **Zhenghua Zhou:** Methodology, Funding acquisition. **Qingqing Gao:** Software, Validation, Visualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was funded by the National Natural Science Foundation of China (61571410) and the Zhejiang Provincial Nature Science Foundation, China (LY18F020018 and LSY19F020001).

Research involving human participants and/or animals

This study did not involve Human Participants and Animals.

Informed consent

The all authors of this paper have consented the submission.

Appendix. Proof of Theorem 2.1

Proof. Let $H(\boldsymbol{x}) = \lambda J(\boldsymbol{x})$. Since the $\boldsymbol{x}$-subproblem and $\boldsymbol{d}$-subproblem are convex, we can get the following system of equations

$$\begin{cases} 0 = \nabla H\left(\boldsymbol{x}^{k+1}\right) + \eta \boldsymbol{A}^{\top}\left(\left(\boldsymbol{A}\boldsymbol{x}^{k+1} - \boldsymbol{y}\right) - \boldsymbol{d}^{k} + \boldsymbol{b}^{k}\right), \\ 0 = \boldsymbol{p}^{k+1} + \eta\left(\boldsymbol{d}^{k+1} - \left(\boldsymbol{A}\boldsymbol{x}^{k+1} - \boldsymbol{y}\right) - \boldsymbol{b}^{k}\right), \\ \boldsymbol{b}^{k+1} = \boldsymbol{b}^{k} + \left(\boldsymbol{A}\boldsymbol{x}^{k+1} - \boldsymbol{y}\right) - \boldsymbol{d}^{k+1} \end{cases} \tag{A.1}$$

by applying the gradients on them, where $\boldsymbol{p}^{k+1} \in \partial \left\|\boldsymbol{d}^{k+1}\right\|_1$.

Let $L(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_1 + H(\boldsymbol{x})$ and $\boldsymbol{x}^*$ be one solution of (1.3), then $\boldsymbol{x}^*$ must satisfy the following equation

$$0 = \boldsymbol{A}^{\top}\boldsymbol{p}^{*} + \nabla H\left(\boldsymbol{x}^{*}\right) \tag{A.2}$$

by applying the gradient on it, where $\boldsymbol{p}^* \in \partial \left\|\boldsymbol{d}^*\right\|_1$ with $\boldsymbol{d}^* = \boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{y}$.

Solve the limit of (A.1), we can get the following system of equations:

$$\begin{cases} 0 = \nabla H\left(\boldsymbol{x}^{*}\right) + \eta \boldsymbol{A}^{\top}\left(\left(\boldsymbol{A}\boldsymbol{x}^{*} - \boldsymbol{y}\right) - \boldsymbol{d}^{*} + \boldsymbol{b}^{*}\right), \\ 0 = \boldsymbol{p}^{*} + \eta\left(\boldsymbol{d}^{*} - \left(\boldsymbol{A}\boldsymbol{x}^{*} - \boldsymbol{y}\right) - \boldsymbol{b}^{*}\right), \\ \boldsymbol{b}^{*} = \boldsymbol{b}^{*} + \left(\boldsymbol{A}\boldsymbol{x}^{*} - \boldsymbol{y}\right) - \boldsymbol{d}^{*}. \end{cases} \tag{A.3}$$

Take $\boldsymbol{b}^* = \frac{1}{\eta}\boldsymbol{p}^*$, we find that $(\boldsymbol{x}^*, \boldsymbol{d}^*, \boldsymbol{b}^*)$ is a fixed point for the equations in (A.1). Therefore, if the iterations (2.8)–(2.10) are convergent, they will converge to a solution of (1.3) as explained in [36].

Put $\boldsymbol{x}_e^k = \boldsymbol{x}^k - \boldsymbol{x}^*, \boldsymbol{d}_e^k = \boldsymbol{d}^k - \boldsymbol{d}^*, \boldsymbol{b}_e^k = \boldsymbol{b}^k - \boldsymbol{b}^*$. Let the first equation in (A.1) be subtracted by the corresponding one in (A.3) to get

$$\nabla H\left(\boldsymbol{x}^{k+1}\right) - \nabla H\left(\boldsymbol{x}^{*}\right) + \eta \boldsymbol{A}^{\top}\left(\boldsymbol{A}\boldsymbol{x}_e^{k+1} - \boldsymbol{d}_e^{k} + \boldsymbol{b}_e^{k}\right) = 0. \tag{A.4}$$

Then take the inner product for both sides in above equation with $\boldsymbol{x}_e^{k+1}$ to get

$$\begin{aligned} &\left\langle \nabla H\left(\boldsymbol{x}^{k+1}\right) - \nabla H\left(\boldsymbol{x}^{*}\right), \boldsymbol{x}_e^{k+1}\right\rangle + \eta\left\|\boldsymbol{A}\boldsymbol{x}_e^{k+1}\right\|_2^2 \\ &- \eta\left\langle \boldsymbol{A}^{\top}\boldsymbol{d}_e^{k}, \boldsymbol{x}_e^{k+1}\right\rangle + \eta\left\langle \boldsymbol{A}^{\top}\boldsymbol{b}_e^{k}, \boldsymbol{x}_e^{k+1}\right\rangle = 0. \end{aligned} \tag{A.5}$$

For the second equation in (A.1), we take the similar operations for the first equation to get

$$\left\langle \boldsymbol{p}^{k+1} - \boldsymbol{p}^{*}, \boldsymbol{d}_e^{k+1}\right\rangle + \eta\left\|\boldsymbol{d}_e^{k+1}\right\|_2^2 - \eta\left\langle \boldsymbol{A}\boldsymbol{x}_e^{k+1}, \boldsymbol{d}_e^{k+1}\right\rangle - \eta\left\langle \boldsymbol{b}_e^{k}, \boldsymbol{d}_e^{k+1}\right\rangle = 0, \tag{A.6}$$

where $\boldsymbol{p}^{k+1} \in \partial \left\|\boldsymbol{d}^{k+1}\right\|_1$ and $\boldsymbol{p}^* = \eta\boldsymbol{b}^* \in \partial \left\|\boldsymbol{d}^*\right\|_1$.

Add (A.5) with (A.6) to obtain

$$\begin{aligned} &\left\langle \nabla H\left(\boldsymbol{x}^{k+1}\right) - \nabla H\left(\boldsymbol{x}^{*}\right), \boldsymbol{x}_e^{k+1}\right\rangle + \left\langle \boldsymbol{p}^{k+1} - \boldsymbol{p}^{*}, \boldsymbol{d}_e^{k+1}\right\rangle \\ &+\eta\left\|\boldsymbol{A}\boldsymbol{x}_e^{k+1}\right\|_2^2 + \eta\left\|\boldsymbol{d}_e^{k+1}\right\|_2^2 \\ &-\eta\left\langle \boldsymbol{A}^{\top}\boldsymbol{d}_e^{k}, \boldsymbol{x}_e^{k+1}\right\rangle + \eta\left\langle \boldsymbol{A}^{\top}\boldsymbol{b}_e^{k}, \boldsymbol{x}_e^{k+1}\right\rangle \\ &-\eta\left\langle \boldsymbol{A}\boldsymbol{x}_e^{k+1}, \boldsymbol{d}_e^{k+1}\right\rangle - \eta\left\langle \boldsymbol{b}_e^{k}, \boldsymbol{d}_e^{k+1}\right\rangle = 0. \end{aligned} \tag{A.7}$$

Let the third equation in (A.1) be subtracted by the corresponding one in (A.3) to get

$$\boldsymbol{b}_e^{k+1} = \boldsymbol{b}_e^{k} + \boldsymbol{A}\boldsymbol{x}_e^{k+1} - \boldsymbol{d}_e^{k+1}. \tag{A.8}$$

Then solve the ℓ_2-norm for b_e^{k+1} as follows:

$$\left\|\boldsymbol{b}_e^{k+1}\right\|_2^2 = \left\|\boldsymbol{b}_e^{k}\right\|_2^2 + \left\|\boldsymbol{A}\boldsymbol{x}_e^{k+1} - \boldsymbol{d}_e^{k+1}\right\|_2^2 + 2\left\langle \boldsymbol{b}_e^{k}, \boldsymbol{A}\boldsymbol{x}_e^{k+1} - \boldsymbol{d}_e^{k+1}\right\rangle. \tag{A.9}$$

Simplifying above equation, we get

$$\left\langle \boldsymbol{b}_e^{k}, \boldsymbol{A}\boldsymbol{x}_e^{k+1} - \boldsymbol{d}_e^{k+1}\right\rangle = \frac{1}{2}\left(\left\|\boldsymbol{b}_e^{k+1}\right\|_2^2 - \left\|\boldsymbol{b}_e^{k}\right\|_2^2\right) - \frac{1}{2}\left\|\boldsymbol{A}\boldsymbol{x}_e^{k+1} - \boldsymbol{d}_e^{k+1}\right\|_2^2. \tag{A.10}$$

Subtract the Eq. (A.10) from the Eq. (A.7) to get

$$\begin{aligned} &\frac{\eta}{2}\left(\left\|\boldsymbol{b}^{k}\right\|_2^2 - \left\|\boldsymbol{b}_e^{k+1}\right\|_2^2\right) \\ &= \left\langle \nabla H\left(\boldsymbol{x}^{k+1}\right) - \nabla H\left(\boldsymbol{x}^{*}\right), \boldsymbol{x}^{k+1} - \boldsymbol{x}^{*}\right\rangle + \left\langle \boldsymbol{p}^{k+1} - \boldsymbol{p}^{*}, \boldsymbol{d}^{k+1} - \boldsymbol{d}^{*}\right\rangle \\ &+\eta\left(\left\|\boldsymbol{A}\boldsymbol{x}_e^{k+1}\right\|_2^2 + \left\|\boldsymbol{d}_e^{k+1}\right\|_2^2 - \left\langle \boldsymbol{A}\boldsymbol{x}_e^{k+1}, \boldsymbol{d}_e^{k} + \boldsymbol{d}_e^{k+1}\right\rangle\right. \\ &\left. - \tfrac{1}{2}\left\|\boldsymbol{A}\boldsymbol{x}_e^{k+1} - \boldsymbol{d}^{k+1}\right\|_2^2\right). \end{aligned} \tag{A.11}$$

Taking a summation on above equation (A.11) from $k = 0$ to $k = K$, we can get

$$\begin{aligned} &\frac{\eta}{2}\left(\left\|\boldsymbol{b}_e^{0}\right\|_2^2 - \left\|\boldsymbol{b}_e^{K+1}\right\|_2^2\right) \\ &= \sum_{k=0}^{K}\left\langle \nabla H\left(\boldsymbol{x}^{k+1}\right) - \nabla H\left(\boldsymbol{x}^{*}\right), \boldsymbol{x}^{k+1} - \boldsymbol{x}^{*}\right\rangle \\ &+ \sum_{k=0}^{K}\left\langle \boldsymbol{p}^{k+1} - \boldsymbol{p}^{*}, \boldsymbol{d}^{k+1} - \boldsymbol{d}^{*}\right\rangle \\ &+ \frac{\eta}{2}\left(\sum_{k=0}^{K}\left\|\boldsymbol{A}\boldsymbol{x}_e^{k+1} - \boldsymbol{d}_e^{k}\right\|_2^2 + \left\|\boldsymbol{d}_e^{K+1}\right\|_2^2\right) - \frac{\eta}{2}\left\|\boldsymbol{d}_e^{0}\right\|_2^2. \end{aligned} \tag{A.12}$$

Since the two inner products in Eq. (A.12) are non-negative due to the convexity of H and $\|\cdot\|_1$, we can get following inequality:

$$\frac{\eta}{2}\left\|\boldsymbol{b}_e^{0}\right\|_2^2 + \frac{\eta}{2}\left\|\boldsymbol{d}_e^{0}\right\|_2^2 \geq \sum_{k=0}^{K}\left\langle \nabla H\left(\boldsymbol{x}^{k+1}\right) - \nabla H\left(\boldsymbol{x}^{*}\right), \boldsymbol{x}^{k+1} - \boldsymbol{x}^{*}\right\rangle. \tag{A.13}$$

So

$$\sum_{k=0}^{+\infty}\left\langle \nabla H\left(\boldsymbol{x}^{k+1}\right) - \nabla H\left(\boldsymbol{x}^{*}\right), \boldsymbol{x}^{k+1} - \boldsymbol{x}^{*}\right\rangle < +\infty.$$

It implies that

$$\lim_{k\to+\infty}\left\langle \nabla H\left(\boldsymbol{x}^{k}\right) - \nabla H\left(\boldsymbol{x}^{*}\right), \boldsymbol{x}^{k} - \boldsymbol{x}^{*}\right\rangle = 0. \tag{A.14}$$

As we know, for an arbitrary convex function G, the Bregman distance [36] satisfies

$$\boldsymbol{B}_G^{\boldsymbol{p}}(\boldsymbol{u}, \boldsymbol{v}) + \boldsymbol{B}_G^{\boldsymbol{q}}(\boldsymbol{v}, \boldsymbol{u}) = \langle \boldsymbol{q} - \boldsymbol{p}, \boldsymbol{u} - \boldsymbol{v}\rangle, \quad \forall \boldsymbol{p} \in \partial G(\boldsymbol{v}), \boldsymbol{q} \in \partial G(\boldsymbol{u}). \tag{A.15}$$

According to (A.15), (A.14), and the non-negativity of the Bregman distance, we can get

$$\lim_{k\to+\infty} \boldsymbol{B}_H^{\nabla H(\boldsymbol{x}^*)}\left(\boldsymbol{x}^{k}, \boldsymbol{x}^{*}\right) = 0,$$

i.e.,

$$\lim_{k\to+\infty} H\left(\boldsymbol{x}^{k}\right) - H\left(\boldsymbol{x}^{*}\right) - \left\langle \boldsymbol{x}^{k} - \boldsymbol{x}^{*}, \nabla H\left(\boldsymbol{x}^{*}\right)\right\rangle = 0. \tag{A.16}$$

In the same way, we can get

$$\lim_{k\to+\infty} \boldsymbol{B}_{\|\cdot\|_1}^{\boldsymbol{p}^*}\left(\boldsymbol{d}^{k}, \boldsymbol{d}^{*}\right) = 0,$$

i.e.,

$$\lim_{k\to+\infty} \|\boldsymbol{d}^k\|_1 - \left\|\boldsymbol{d}^*\right\|_1 - \left\langle \boldsymbol{d}^k - \boldsymbol{d}^*, \boldsymbol{p}^*\right\rangle = 0. \tag{A.17}$$

Since the term $\sum_{k=0}^{K} \left\|\boldsymbol{A}\boldsymbol{x}_e^{k+1} - \boldsymbol{d}_e^k\right\|^2$ in Eq. (A.12) is bounded, we can get

$$\lim_{k\to+\infty} \left\|\boldsymbol{A}\boldsymbol{x}_e^{k+1} - \boldsymbol{d}_e^k\right\| = 0.$$

As $\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{y} = \boldsymbol{d}^*$, we can obtain

$$\lim_{k\to+\infty} \left\|\boldsymbol{A}\boldsymbol{x}^{k+1} - \boldsymbol{y} - \boldsymbol{d}^k\right\|_2 = 0. \tag{A.18}$$

Combining (A.17), (A.18), and the continuity of $\|\cdot\|_1$, we get

$$\lim_{k\to+\infty} \|\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{y}\|_1 - \left\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{y}\right\|_1 - \left\langle \boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{A}\boldsymbol{x}^*, \boldsymbol{p}^*\right\rangle = 0. \tag{A.19}$$

Add (A.19) with (A.16) to acquire

$$\begin{aligned}\lim_{k\to+\infty} &\left(\left\|\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{y}\right\|_1 + H\left(\boldsymbol{x}^k\right)\right) - \left(\left\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{y}\right\|_1 + H\left(\boldsymbol{x}^*\right)\right) \\ &- \left\langle \boldsymbol{x}^k - \boldsymbol{x}^*, \nabla H\left(\boldsymbol{x}^*\right) + \boldsymbol{A}^\top \boldsymbol{p}^*\right\rangle = 0.\end{aligned} \tag{A.20}$$

Up to now, we have finished proving (2.11) based on (A.2) and (A.20).

Next we begin to prove (2.12) under the assumption that (1.3) has the unique solution $\boldsymbol{x}^*$. Suppose that (2.12) does not hold, then there exists an $\epsilon_0 > 0$ and a subsequence $\{\boldsymbol{x}^{k_i}\}_{i=1}^{+\infty}$ such that $\left\|\boldsymbol{x}^{k_i} - \boldsymbol{x}^*\right\|_2 > \epsilon_0$. Take $E(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_1 + H(\boldsymbol{x})$, then $E(\boldsymbol{x})$ is a convex and lower semicontinuous function. So $E\left(\boldsymbol{x}^{k_i}\right) > \min\{E(\boldsymbol{x}) : \|\boldsymbol{x} - \boldsymbol{x}^*\|_2 = \epsilon_0\}$. Let $\boldsymbol{q}$ be the intersection of the line segment from $\boldsymbol{x}^*$ to $\boldsymbol{x}^{k_i}$ and the sphere $\{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{x}^*\|_2 = \epsilon_0\}$, then there exists a positive number $t \in (0, 1)$ such that $\boldsymbol{q} = t\boldsymbol{x}^* + (1-t)\boldsymbol{x}^{k_i}$.

According to the definition of $\boldsymbol{x}^*$ and the convexity of E, we can get

$$\begin{aligned}E\left(\boldsymbol{x}^{k_i}\right) &> tE\left(\boldsymbol{x}^*\right) + (1-t)E\left(\boldsymbol{x}^{k_i}\right) \geq E\left(t\boldsymbol{x}^* + (1-t)\boldsymbol{x}^{k_i}\right) = E(\boldsymbol{q}) \\ &\geq \min\left\{E(\boldsymbol{x}) : \left\|\boldsymbol{x} - \boldsymbol{x}^*\right\|_2 = \epsilon_0\right\}.\end{aligned}$$

Put $\tilde{\boldsymbol{x}} = \arg\min\{E(\boldsymbol{x}) : \|\boldsymbol{x} - \boldsymbol{x}^*\|_2 = \epsilon_0\}$ and use the property (2.11), then we can obtain

$$E\left(\boldsymbol{x}^*\right) = \lim_{i\to+\infty} E\left(\boldsymbol{x}^{k_i}\right) \geq E(\tilde{\boldsymbol{x}}) > E\left(\boldsymbol{x}^*\right).$$

Obviously, it is a contradiction. Therefore, property (2.12) holds. □

References

[1] W.Z. Shi, J. Caballero, C. Ledig, X.H. Zhuang, W.J. Bai, K. Bhatia, A.M.S.M. Marvao, T. Dawes, D. Regan, D. Rueckert, Cardiac image super-resolution with global correspondence using multi-atlas patchmatch, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, vol. 8151, 2013, pp. 9-16.

[2] D. Yıldırım, O. Güngör, A novel image fusion method using IKONOS satellite images, J. Geod. Geoinf. 1 (1) (2012) 75–83.

[3] R.G. Keys, Cubic convolution interpolation for digital image processing, IEEE Trans. Acoust. 29 (6) (1981) 1153–1160.

[4] X. Zhang, X. Wu, Image interpolation by adaptive 2-D autoregressive modeling and soft-decision estimation, IEEE Trans. Image Process. 17 (6) (2008) 887–896.

[5] L. Zhang, X. Wu, An edge-guided image interpolation algorithm via directional filtering and data fusion, IEEE Trans. Image Process. 15 (8) (2006) 2226–2238.

[6] A. Marquina, S.J. Osher, Image super-resolution by TV-regularization and bregman iteration, J. Sci. Comput. 37 (3) (2008) 367–382.

[7] J. Sun, J. Zhu, M.F. Tappen, Context-constrained hallucination for image super-resolution, in: IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 231-238.

[8] J. Sun, J. Sun, Z. Xu, H.Y. Shum, Image super-resolution using gradient profile prior, in IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1-8.

[9] L. Wang, H. Wu, C. Pan, Fast image upsampling via the displacement field, IEEE Trans. Image Process. 23 (12) (2014) 5123–5135.

[10] X. Fang, Q. Zhou, J. Shen, C. Jacquemin, L. Shao, Text image deblurring using kernel sparsity prior, IEEE Trans. Cybern. 50 (3) (2020) 997–1008.

[11] W.T. Freeman, T.R. Jones, E.C. Pasztor, Example-based super-resolution, IEEE Comput. Graph. Appl. 22 (2) (2002) 56–65.

[12] K.I. Kim, Y. Kwon, Single-image super-resolution using sparse regression and natural image prior, IEEE Trans. Pattern Anal. Mach. Intell. 32 (6) (2010) 1127–1133.

[13] C. Dong, C.C. Loy, K. He, X. Tang, Learning a deep convolutional network for image super-resolution, in: European Conference on Computer Vision, 2014, pp. 184-199.

[14] C. Dong, C.C. Loy, X. Tang, Accelerating the super-resolution convolutional neural network, in: European Conference on Computer Vision, 2016, pp. 391-407.

[15] W. Shi, J. Caballero, F. Huszar, J. Totz, Z. Wang, Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1874-1883.

[16] J. Kim, J.K. Lee, K.M. Lee, Accurate image super-resolution using very deep convolutional networks, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1646-1654.

[17] W.S. Lai, J.B. Huang, N. Ahuja, M.H. Yang, Deep Laplacian pyramid networks for fast and accurate super-resolution, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5835-5843.

[18] Y. Tai, J. Yang, X. Liu, C. Xu, MemNet: a persistent memory network for image restoration, in: IEEE International Conference on Computer Vision, 2017, pp. 4549-4557.

[19] Y. Li, E. Agustsson, S. Gu, R. Timofte, L. Van, Carn: convolutional anchored regression network for fast and accurate single image super-resolution, in: European Conference on Computer Vision, 2018, pp. 166-181.

[20] Z. Hui, X. Gao, Y. Yang, X. Wang, Lightweight image super-resolution with information multi-distillation network, in: ACM International Conference on Multimedia, 2019, pp. 2024-2032.

[21] M. Haris, G. Shakhnarovich, N. Ukita, Deep back-projection networks for super-resolution, in: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1664-1673.

[22] J. Li, F. Fang, K. Mei, G. Zhang, Multi-scale residual network for image super-resolution, in: European Conference on Computer Vision, 2018, pp. 517-532.

[23] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, Y. Fu, Image super-resolution using very deep residual channel attention networks, in: European Conference on Computer Vision, 2018, pp. 286-301.

[24] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, Y. Fu, Residual dense network for image super-resolution, in: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2472-2481.

[25] Y.W. Li, S.H. Gu, K. Zhang, V.G. Luc, R. Timofte, DHP: differentiable meta pruning via hypernetworks, in: European Conference on Computer Vision, 2020, pp. 608-624.

[26] J. Kim, J.K. Lee, K.M. Lee, Deeply-recursive convolutional network for image super-resolution, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1637-1645.

[27] Y. Tai, J. Yang, X. Liu, Image super-resolution via deep recursive residual network, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2790-2798.

[28] C. Ren, X. He, Y. Pu, T.Q. Nguyen, Learning image profile enhancement and denoising statistics priors for single-image super-resolution, IEEE Trans. Cybern. 99 (2019) 1–14.

[29] W. Dong, P. Wang, W. Yin, G. Shi, F. Wu, X. Lu, Denoising prior driven deep neural network for image restoration, IEEE Trans. Pattern Anal. Mach. Intell. 41 (10) (2019) 2305–2318.

[30] E.J. Candès, Compressive sampling, in: Proc. Int. Congr. Math., ICM 2006, vol. 3, 2006, pp. 1433-1452.

[31] B.K. Natarajan, Sparse approximate solutions to linear systems, SIAM J. Comput. 24 (2) (1995) 227–234.

[32] Y. Yang, H.T. Shen, Z.G. Ma, H. Zi, X.F. Zhou, $L_{2,1}$-norm regularized discriminative feature selection for unsupervised learning, in: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, vol. 3, 2011, pp. 1589-1594.

[33] Z.H. Wang, J. Chen, S.C.H. Hoi, Deep learning for image super-resolution: a survey, IEEE Trans. Pattern Anal. Mach. Intell. 99 (2020) 1–22.

[34] S.H. Chan, X. Wang, O.A. Elgendy, Plug-and-play ADMM for image restoration: fixed-point convergence and applications, IEEE Trans. Comput. Imaging 3 (1) (2016) 84–98.

[35] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM J. Imaging Sci. 2 (1) (2009) 183–202.

[36] T. Goldstein, S. Osher, The split Bregman method for L_1-regularized problems, SIAM J. Imaging Sci. 2 (2) (2009) 323–343.

[37] D.R. Hunter, K. Lange, A tutorial on MM algorithms, Am. Stat. 58 (1) (2004) 30–37.

[38] R. Timofte, S. Gu, L. Van Gool, L. Zhang, M.H. Yang, NTIRE 2018 challenge on single image super-resolution: methods and results, in: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 965-976.
[39] M. Bevilacqua, A. Roumy, C. Guillemot, M.L.A. Morel, Low-complexity single-image super-resolution based on nonnegative neighbor embedding, Br. Mach. Vis. Conf. 135 (2012) 1–10.
[40] R. Zeyde, M. Elad, M. Protter, On single image scale-up using sparse-representations, in: International Conference on Curves and Surfaces, 2010, pp. 711-730.
[41] P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, Contour detection and hierarchical image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 33 (5) (2011) 898–916.
[42] J.B. Huang, A. Singh, N. Ahuja, Single image super-resolution from transformed self-exemplars, in: IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5197-5206.
[43] D.P. Kingma, J.L. Ba, Adam: A method for stochastic optimization, in: the 32nd International Conference on Machine Learning, ICML 2015, 2015, pp. 1-15.
[44] R. Timofte, V. De Smet, L. Van Gool, A+: adjusted anchored neighborhood regression for fast super-resolution, in: Asian Conference on Computer Vision, vol. 9006, 2015, pp. 111-126.
[45] Z. Hui, X. Wang, X. Gao, Fast and accurate single image super-resolution via information distillation network, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2018, pp. 723-731.
[46] K. Zhang, M. Danelljan, Y.W. Li, R. Timofte, AIM 2020 challenge on efficient super-resolution: methods and results, 2020, pp. 1–16.
[47] B. Lim, S. Son, S. Nah, K.M. Lee, Enhanced deep residual networks for single image super-resolution, in: IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 1132-1140.