Adaptive Deep Cascade Broad Learning System and Its Application in Image Denoising

Hailiang Ye, Hong Li, *Member, IEEE*, and C. L. Philip Chen, *Fellow, IEEE*

***Abstract*—This article proposes a novel regularization deep cascade broad learning system (DCBLS) architecture, which includes one cascaded feature mapping nodes layer and one cascaded enhancement nodes layer. Then, the transformation feature representation is easily obtained by incorporating the enhancement nodes and the feature mapping nodes. Once such a representation is established, a final output layer is constructed by implementing a simple convex optimization model. Furthermore, a parallelization framework on the new method is designed to make it compatible with large-scale data. Simultaneously, an adaptive regularization parameter criterion is adopted under some conditions. Moreover, the stability and error estimate of this method are discussed and proved mathematically. The proposed method could extract sufficient available information from the raw data compared with the standard broad learning system and could achieve compellent successes in image denoising. The experiments results on benchmark datasets, including natural images as well as hyperspectral images, verify the effectiveness and superiority of the proposed method in comparison with the state-of-the-art approaches for image denoising.**

***Index Terms*—Broad learning system (BLS), deep neural networks, hyperspectral image (HSI), image denoising, stability.**

I. Introduction

In THE past decade, deep learning (also referred to as deep neural networks) has attracted considerable attention due to its great potential in data representation [1]. Among them, the popular deep networks comprise of the multilayer perceptron (MLP) [2], deep belief networks [3], stacked autoencoders [4], and convolutional neural networks (CNNs) [5], [6], etc. These deep-learning models have exhibited remarkable performance and been successfully applied to pattern recognition, image processing, computer vision, and many others [7]–[9].

Manuscript received September 26, 2019; revised January 7, 2020 and February 29, 2020; accepted March 3, 2020. Date of publication March 23, 2020; date of current version September 8, 2021. This work was supported by the National Natural Science Foundation of China under Grant 61877021, Grant 61751202, Grant U1813203, Grant U1801262, and Grant 61751205. This article was recommended by Associate Editor Q. M. J. Wu. *(Corresponding author: Hong Li.)*

Hailiang Ye and Hong Li are with the School of Mathematics and Statistics, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: yhl575@163.com; hongli@hust.edu.cn).

C. L. Philip Chen is with the Faculty of Science and Technology, University of Macau, Macau, China, and also with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: philip.chen@ieee.org).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2020.2978500.

Digital Object Identifier 10.1109/TCYB.2020.2978500

Due to the promising representation capability of deep learning, some different types of deep-learning structures have been proposed and tried in the field of image denoising in the latest years, where these approaches take noisy images and ground-truth images as training image pairs to learn an underlying model. Burger *et al.* adopted the MLP model [10] to deal with noisy images. Nam *et al.* [11] proposed a data-driven noise estimation algorithm using MLP. Chen and Pock [12] built a trainable nonlinear reaction diffusion (TNRD) model and it could be represented as a feedforward deep network by unfolding a fixed number of gradient descent steps. Vincent *et al.* [13] and Xie *et al.* [14] presented the stacked autoencoders for image denoising. Schmidt and Roth [15] exploited a cascade of shrinkage fields method unifying the random field-based model and the half-quadratic optimization into a learning architecture. Jain and Seung [16] put forward CNNs for disposing of contaminated images. Later, Mao *et al.* [17] presented a deep fully convolutional encoding–decoding framework for image restoration. Zhang *et al.* [18] proposed a denoising CNN (DnCNN) by integrating batch normalization and residual learning in the training phase to boost the denoising performance. Subsequently, Zhang *et al.* [19] designed a fast and flexible DnCNN (FFDNet). In [20], block matching as a preprocessing step was cascaded with CNN for image denoising. Generative adversarial networks based on CNN and autoencoder were introduced and used to image denoising in [21] and [22]. Yuan *et al.* [23] established the deep residual CNN to hyperspectral image (HSI) denoising. In [24], a persistent memory recurrent neural network was proposed for handling the noisy images. Besides, more advanced approaches based on deep learning for image denoising were found in [25]–[28].

Despite the powerful learning ability of deep structure, most deep-learning models could be subjected to a time-consuming training process due to a great number of parameters and complicated structures. It becomes rather hard to analyze the deep structure theoretically since it involves a highly nonconvex optimization problem, and most works spend in tuning the parameters or stacking more layers for better accuracy.

Recently, a broad learning system (BLS) was proposed by Chen and Liu [29] to provide an effective and efficient learning framework for machine learning and pattern recognition. BLS expands the neurons containing feature mapping and enhancement nodes in a broad manner without stacking layers in deep, and then compute the output weights by pseudoinverse quickly. In contrast to some popular deep-learning algorithms suffering from time consuming to train excessive

parameters in the filters and layers, BLS could provide an alternative way of dealing with large-scale data. The training process in BLS can be extended to an incremental learning model without retraining process when comes to new nodes or inputs. Moreover, the universal approximation capability of BLS was presented in [30] and this new method could achieve competitive results with the state-of-the-art methods on various applications [31]–[34]. The fast learning property of BLS motivates us to further explore denoising capability in noisy images. Nevertheless, BLS may encounter a slight loss of accuracy compared with deep-learning methods [29]. Again, the stability of BLS is unclear and there is no definite criterion for the choice of the regularization parameter.

To overcome the above-mentioned limitations and further utilize the advantages of deep-learning methods and BLS, this article is aimed at building a deep architecture based on BLS in an attempt to obtain a more satisfactory and stable model. To this end, we propose a regularization deep cascade BLS (DCBLS) architecture, comprising of one cascaded feature mapping nodes layer with a depth n and one cascaded enhancement nodes layer with a depth m. The idea of adopting cascaded operators could extract more available information from the raw data and improve the performance for practical applications. This has been verified in different fields [35], [36]. Then, the transformation feature representation is easily attained by the enhancement nodes layer together with the feature mapping layer. Once such a representation is built, a final output layer is constructed by solving a simple convex optimization model. The main contributions are summarized as follows.

1) We devise a new DCBLS framework, which could extract more sufficient available information from the raw data. This new method could simplify the complex deep model to only solve a convex optimization problem. In addition, the DCBLS method can be easily and rapidly implemented by pseudoinverse and thereby retain the fast computational nature of BLS.
2) A parallelization framework of the proposed DCBLS approach is presented in order to be compatible with large-scale data. Meanwhile, an adaptive regularization parameter selection criterion is introduced in terms of some certain conditions to replace the conventional fixed one. Besides, the theoretical stability and error estimate of the DCBLS model are deduced to guarantee the feasibility.
3) The proposed DCBLS method can be successfully applied to image denoising. Substantial experiments results on benchmark datasets, such as natural images as well as HSIs, indicate that DCBLS outperforms the existing methods on effectiveness and efficiency.

The remainder of this article is organized as follows. Section II reviews the standard BLS. Furthermore, a new regularization deep cascade architecture based on BLS is proposed and then a parallelization framework has schemed to large-scale data. Later, the adaptive regularization parameter strategy is introduced. Section III discusses the stability and error estimate of the proposed approach mathematically. The experimental results on benchmark datasets are presented in Section IV. Conclusions are drawn in Section V followed by some mathematical proofs in the Appendices.

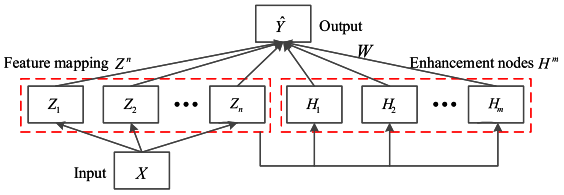


Fig. 1. Structure of BLS. The red-dashed boxes on the left and right are feature mapping nodes $\boldsymbol{Z^n}$ and enhancement nodes $\boldsymbol{H^m}$, respectively; and $\boldsymbol{W}$ represents the output weight connecting $\boldsymbol{Z^n}$ and $\boldsymbol{H^m}$ to the output layer $\hat{\boldsymbol{Y}}$.

II. Adaptive Regularization Deep Cascade Broad Learning System

This section mainly presents a new deep cascade architecture based on BLS and provides the corresponding regularization model, which could extract more useful information from the original input data. Then, we establish a feasible parallelism scheme for large-scale data problems. Furthermore, the adaptive regularization parameter rule is also given.

A. Revisit of Broad Learning System

The basic structure of BLS is illustrated in Fig. 1. Generally, given a training sample set $\{(\boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{X} \in \mathbf{R}^{N\times d}, \boldsymbol{Y} \in \mathbf{R}^{N\times C}\}$ and n feature mappings ϕ_i, the ith feature mapping matrix is

$$\boldsymbol{Z}_i = \phi_i\big(\boldsymbol{X}\boldsymbol{W}_{e_i} + \boldsymbol{b}_{e_i}\big), \quad i = 1, 2, \ldots, n \tag{1}$$

where N is the number of input samples, d is the dimension of each sample, C is the dimension of the corresponding outputs, and $\mathbf{R}^{m\times n}$ represents the set of real numbers with the size of $m \times n$. The ith row of $\boldsymbol{X}$ and $\boldsymbol{Y}$ denotes the data point $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$ and the target vector $\boldsymbol{y}_i = [y_{i1}, y_{i2}, \ldots, y_{iC}]$, respectively, and the weights $\boldsymbol{W}_{e_i}$ and the bias term $\boldsymbol{b}_{e_i}$ are randomly determined with proper dimensions. The functions ϕ_i have no explicit restrictions, which means that the common choices, such as linear mappings, kernel mappings, or nonlinear transformations, are acceptable. For more information, readers can refer to [29]. Denote $\boldsymbol{Z}^n = [\boldsymbol{Z}_1, \boldsymbol{Z}_2, \ldots, \boldsymbol{Z}_n]$ as the collection of n groups featuring mapping nodes. It should be noted that the randomly initialized weights $\boldsymbol{W}_{e_i}$ and the bias $\boldsymbol{b}_{e_i}$ are fine-tuned by a sparse autoencoder to produce better feature $\boldsymbol{Z}^n$ [29].

Subsequently, $\boldsymbol{Z}^n$ is connected to the layer of enhancement nodes by activation functions. Assume that there are m enhancement nodes and the jth group of enhancement nodes is defined as follows:

$$\boldsymbol{H}_j = \xi_j\big(\boldsymbol{Z}^n\boldsymbol{W}_{h_j} + \boldsymbol{b}_{h_j}\big), \quad j = 1, 2, \ldots, m \tag{2}$$

where ξ_j is the activation function, and $\boldsymbol{W}_{h_j}$ and $\boldsymbol{b}_{h_j}$ are randomly generated weights and bias terms connecting the feature nodes to the enhancement nodes. The output matrix of the enhancement nodes is denoted by $\boldsymbol{H}^m = [\boldsymbol{H}_1, \boldsymbol{H}_2, \ldots, \boldsymbol{H}_m]$.

Consolidated by the feature mapping node layer and enhancement nodes layer, the output of BLS could be represented as

$$\hat{\boldsymbol{Y}} = \boldsymbol{A}\boldsymbol{W} \tag{3}$$

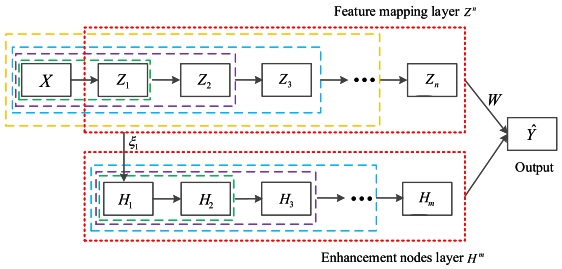


Fig. 2. Structure of DCBLS. The red dashed boxes on the top and bottom are feature mapping layer $\boldsymbol{Z^n}$ and enhancement nodes $\boldsymbol{H^m}$, respectively, and other dashed boxes represent the cascaded operators; and $\boldsymbol{W}$ denotes the output weight connecting $\boldsymbol{Z^n}$ and $\boldsymbol{H^m}$ to the output layer $\hat{\boldsymbol{Y}}$.

where $\boldsymbol{A} = [\boldsymbol{Z}^n, \boldsymbol{H}^m]$, and $\boldsymbol{W}$ is the output weight connecting feature mapping nodes and enhancement nodes to the output layer. The model (3) can be computed rapidly by pseudoinverse of $\boldsymbol{A}$ with actual ouputs $\boldsymbol{Y}$, that is, $\boldsymbol{W} = \boldsymbol{A}^+\boldsymbol{Y}$.

However, despite the great success of BLS in many areas, information extraction may not be sufficient in some cases. Furthermore, the stability issue is a challenge.

B. Deep Cascade Broad Learning System

As mentioned earlier, BLS could provide an alternative method for large-scale data that is much faster with a comparable or a slight loss of accuracy. In order to obtain more valuable information from the raw data, we will present a deep cascade architecture in view of BLS, which includes one cascaded feature mapping nodes layer with a depth n and one cascaded enhancement nodes layer with a depth m, thereby referred to as "DCBLS." The complete network structure of DCBLS is displayed in Fig. 2. Mathematically, this structure is defined as follows.

Denote $\boldsymbol{S}_{-1} = \varnothing$ for description convenience. For the given input data $\boldsymbol{X} \in \mathbf{R}^{N\times d}$, if we define $\boldsymbol{Z}_0 = \boldsymbol{X}$ and $\boldsymbol{S}_0 = [\boldsymbol{S}_{-1}, \boldsymbol{Z}_0]$, then the first group of feature mapping nodes is formulated as

$$\boldsymbol{Z}_1 = \phi_1\big(\boldsymbol{S}_0\boldsymbol{W}_{e_1} + \boldsymbol{b}_{e_1}\big) \tag{4}$$

where $\boldsymbol{W}_{e_1}$ and $\boldsymbol{b}_{e_1}$ are weight and bias, respectively. As for the second group, the feature mapping nodes $\boldsymbol{Z}_2$ are established using the combination of the input nodes $\boldsymbol{X}$ and the first group nodes $\boldsymbol{Z}_1$, that is

$$\boldsymbol{Z}_2 = \phi_2\big(\boldsymbol{S}_1\boldsymbol{W}_{e_2} + \boldsymbol{b}_{e_2}\big) \tag{5}$$

where $\boldsymbol{S}_1 = [\boldsymbol{S}_0, \boldsymbol{Z}_1]$ is a concatenated matrix, and $\boldsymbol{W}_{e_2}$ and $\boldsymbol{b}_{e_2}$ are weight and bias, respectively.

Using the similar process continuously, all the n groups of feature mapping nodes are expressed as

$$\boldsymbol{Z}_i = \phi_i\big(\boldsymbol{S}_{i-1}\boldsymbol{W}_{e_i} + \boldsymbol{b}_{e_i}\big), \quad i = 1, 2, \ldots, n \tag{6}$$

where $\boldsymbol{W}_{e_i}$ and $\boldsymbol{b}_{e_i}$ are randomly generated and fine-tuned weight matrix and bias by adopting the approach of reference [29] and $\boldsymbol{S}_{i-1} = [\boldsymbol{S}_{i-2}, \boldsymbol{Z}_{i-1}]$ means that the input of the ith group of feature mapping nodes consists of the input data $\boldsymbol{X}$ and the first $i-1$ groups of feature mapping nodes. Such a feature extraction method can obtain more favorable information that is beneficial to the subsequent process. We define the entire feature mapping nodes as follows:

$$\boldsymbol{Z}^n \triangleq [\boldsymbol{Z}_1, \boldsymbol{Z}_2, \ldots, \boldsymbol{Z}_n]. \tag{7}$$

Next, these feature nodes are sent into the enhancement nodes via nonlinear transformation, which is likely to make up the nonlinear capabilities of the mapped feature nodes. Denote $\boldsymbol{T}_0 = \boldsymbol{Z}^n$, and we have the first group of enhancement nodes is

$$\boldsymbol{H}_1 = \xi_1\big(\boldsymbol{T}_0\boldsymbol{W}_{h_1} + \boldsymbol{b}_{h_1}\big) \tag{8}$$

where the associated weights $\boldsymbol{W}_{h_1}$ and $\boldsymbol{b}_{h_1}$ are randomly sampled. The second group of enhancement nodes $\boldsymbol{H}_2$ is represented as

$$\boldsymbol{H}_2 = \xi_2\big(\boldsymbol{T}_1\boldsymbol{W}_{h_2} + \boldsymbol{b}_{h_2}\big) \tag{9}$$

where $\boldsymbol{T}_1 = \boldsymbol{H}_1$; and $\boldsymbol{W}_{h_2}$ and $\boldsymbol{b}_{h_2}$ are randomly generated. Similar to the previous feature mapping nodes generation, the jth ($j > 2$) group of enhancement nodes are described as

$$\boldsymbol{H}_j = \xi_j\big(\boldsymbol{T}_{j-1}\boldsymbol{W}_{h_j} + \boldsymbol{b}_{h_j}\big), \quad j = 3, 4, \ldots, m \tag{10}$$

where $\boldsymbol{W}_{h_j}$ and $\boldsymbol{b}_{h_j}$ are random weight matrix and bias; and $\boldsymbol{T}_{j-1} = [\boldsymbol{T}_{j-2}, \boldsymbol{H}_{j-1}]$ are the input of the jth group of enhancement nodes, which are connected with the first $j-1$ groups of enhancement nodes. We represent the output of enhancement nodes layer by

$$\boldsymbol{H}^m \triangleq [\boldsymbol{H}_1, \boldsymbol{H}_2, \ldots, \boldsymbol{H}_m]. \tag{11}$$

Finally, the combination of $\boldsymbol{Z}^n$ and $\boldsymbol{H}^m$ is connected directly with the top output layer and these matrices have been fixed according to the preceding generated processes. To maintain stability, the output weight $\boldsymbol{W}$ could be implemented by solving the following convex optimization model:

$$\min_{\boldsymbol{W}} \|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{W}\|_F^2 + \lambda\|\boldsymbol{W}\|_F^2 \tag{12}$$

where $\boldsymbol{A} = [\boldsymbol{Z}^n, \boldsymbol{H}^m]$, $\boldsymbol{Y}$ is the given target data, and $\boldsymbol{W}$ is the output weight connecting feature mapping nodes and enhancement nodes to the top output layer, $\|\cdot\|_F$ represents the Frobenius norm, and $\lambda > 0$ is a regularization parameter to adjust the best tradeoff between the error term and the penalty term. Through a simple derivative operation on $\boldsymbol{W}$, we can obtain

$$\boldsymbol{W} = \Big(\boldsymbol{A}^\top\boldsymbol{A} + \lambda\boldsymbol{I}\Big)^{-1}\boldsymbol{A}^\top\boldsymbol{Y} \tag{13}$$

where $\boldsymbol{I}$ is an identity matrix.

Based on the above-mentioned description , the full procedure is listed in Algorithm 1. Once completing the training process, the weights $\boldsymbol{W}_{e_i}$, $\boldsymbol{W}_{h_j}$, and $\boldsymbol{W}$ should be fixed. Then, the regularization deep cascade BLS responses to a testing sample $\boldsymbol{x}_p$ as

$$\boldsymbol{y}_p = \hat{\boldsymbol{A}}_p\boldsymbol{W} \tag{14}$$

where $\hat{\boldsymbol{A}}_p$ is the transformation feature of $\boldsymbol{x}_p$.

C. Parallelism Scheme of DCBLS

It has been recognized that the output weights in the DCBLS method are computed is time consuming as the size of the

Algorithm 1 DCBLS Algorithm

Input: Training data $(\boldsymbol{X}, \boldsymbol{Y})$, feature mapping function $\phi_i(\cdot)$ $(i = 1, 2, \ldots, n)$, enhancement nodes activation function $\xi_j(\cdot)$ $(j = 1, 2, \ldots, m)$, the number of feature mapping groups n, feature units on each group n_f, the number of enhancement nodes groups m, enhancement units on each group m_e, and regularization parameter $\lambda > 0$;

Output: Return the output weight $\boldsymbol{W}$;

Initialize: Let $\boldsymbol{S}_{-1} = \varnothing$ and $\boldsymbol{Z}_0 = \boldsymbol{X}$;

Step 1:

1) Random $\boldsymbol{W}_{e_i}$, $\boldsymbol{b}_{e_i}$, $i = 1, 2, \ldots, n$;
2) Compute

$$\boldsymbol{Z}_i = \phi_i(\boldsymbol{S}_{i-1}\boldsymbol{W}_{e_i} + \boldsymbol{b}_{e_i}),$$

where $\boldsymbol{S}_{i-1} = [\boldsymbol{S}_{i-2}, \boldsymbol{Z}_{i-1}]$, $i = 1, 2, \ldots, n$;
3) Set the feature mapping nodes as

$$\boldsymbol{Z}^n \triangleq [\boldsymbol{Z}_1, \boldsymbol{Z}_2, \ldots, \boldsymbol{Z}_n];$$

Step 2:

1) Let $\boldsymbol{T}_0 = \boldsymbol{Z}^n$ and random $\boldsymbol{W}_{h_j}$, $\boldsymbol{b}_{h_j}$, $j = 1, 2, \ldots, m$;
2) Calculate

$$\boldsymbol{H}_1 = \xi_1(\boldsymbol{T}_0\boldsymbol{W}_{h_1} + \boldsymbol{b}_{h_1}),$$

$$\boldsymbol{H}_2 = \xi_1(\boldsymbol{T}_1\boldsymbol{W}_{h_2} + \boldsymbol{b}_{h_2}),$$

where $\boldsymbol{T}_1 = \boldsymbol{H}_1$;
3) Compute

$$\boldsymbol{H}_j = \xi_j(\boldsymbol{T}_{j-1}\boldsymbol{W}_{h_j} + \boldsymbol{b}_{h_j}),$$

where $\boldsymbol{T}_{j-1} = [\boldsymbol{T}_{j-2}, \boldsymbol{H}_{j-1}]$, $j = 3, 4, \ldots, m$;
4) Set the enhancement nodes as

$$\boldsymbol{H}^m \triangleq [\boldsymbol{H}_1, \boldsymbol{H}_2, \ldots, \boldsymbol{H}_m];$$

Step 3:

1) Set the transformation feature

$$\boldsymbol{A} = [\boldsymbol{Z}^n, \boldsymbol{H}^m];$$

2) Solve the output weight matrix $\boldsymbol{W}$ by

$$\boldsymbol{W} = (\boldsymbol{A}^\top\boldsymbol{A} + \lambda\boldsymbol{I})^{-1}\boldsymbol{A}^\top\boldsymbol{Y}.$$

data samples becomes extremely large. To address this issue, a data parallelization strategy will be used to handle the case of large-scale datasets.

Parallelization is a computational method in computer systems that can work on different subsample sets of the same algorithm simultaneously. Its major purpose is to save running time for large-scale data problems. For the proposed DCBLS framework, we can decompose them into k $(k \leq N)$ subsample sets for the given data $(\boldsymbol{X}, \boldsymbol{Y})$, that is

$$\boldsymbol{X} = \left[\boldsymbol{X}_1^\top, \boldsymbol{X}_2^\top, \ldots, \boldsymbol{X}_k^\top\right]^\top \tag{15}$$

and

$$\boldsymbol{Y} = \left[\boldsymbol{Y}_1^\top, \boldsymbol{Y}_2^\top, \ldots, \boldsymbol{Y}_k^\top\right]^\top \tag{16}$$

where $\boldsymbol{X}_t \in \mathbf{R}^{N_t \times d}$ and $\boldsymbol{Y}_t \in \mathbf{R}^{N_t \times C}$, $t = 1, 2, \ldots, k$, N_t is the number of samples meeting the equality $\sum_{t=1}^{k} N_t = N$. Accordingly, we could simplify the computational process of DCBLS by parallelization.

For each subsample set $\boldsymbol{X}_t$, the feature mapping nodes and the enhancement nodes are calculated separately by the use of (4)–(11), denoted by $\boldsymbol{Z}_t^n$ and $\boldsymbol{H}_t^m$, respectively. Let $\boldsymbol{A}_t = [\boldsymbol{Z}_t^n, \boldsymbol{H}_t^m]$, and then the output weights of DCBLS could be written by

$$\boldsymbol{W} = \left(\sum_{t=1}^{k}\boldsymbol{A}_t^\top\boldsymbol{A}_t + \lambda\boldsymbol{I}\right)^{-1}\left(\sum_{t=1}^{k}\boldsymbol{A}_t^\top\boldsymbol{Y}_t\right). \tag{17}$$

The individual products $\boldsymbol{A}_t^\top\boldsymbol{A}_t$ and $\boldsymbol{A}_t^\top\boldsymbol{Y}_t$ are computed independently and simultaneously by parallelization, and then summed across all the training subsample sets using a single operation. When the number of features (the number of columns in $\boldsymbol{A}_t$) is less than the number of training data (rows of $\boldsymbol{A}_t$), we can exploit transpose reduction when forming these products—since the product $\boldsymbol{A}_t^\top\boldsymbol{A}_t$ is much smaller than the matrix $\boldsymbol{A}_t$ alone. This dramatically reduces the quantity of data transmitted during the reduce operation. Once these products have been formed, we can compute the pseudoinverse of $\boldsymbol{A}^\top\boldsymbol{A}$ in (13) easily. In addition, if the computing resources are sufficient, then we can set a large k as much as possible, which would reduce the running time greatly. In this way, the DCBLS framework could be done effectively using the transpose reduction strategies that reduce the dimensionality of matrices during the data transmitted process and thereby dispose of large-scale datasets efficiently.

D. Adaptive Regularization Parameter

Since the regularization parameter λ plays an important role in adjusting the relationship between the data fidelity and the stability of the solution. Previous studies have shown that the computational cost could increase significantly if the fixed parameter λ is chosen too small or too large [37]. Accordingly, a dynamical λ may be required in real applications. With the aid of the way proposed in [37], a regularization functional is introduced to be used in the place of a constant regularization parameter.

From (12), we write

$$J(\lambda(\boldsymbol{W}), \boldsymbol{W}) = \|\boldsymbol{A}\boldsymbol{W} - \boldsymbol{Y}\|_F^2 + \lambda(\boldsymbol{W})\|\boldsymbol{W}\|_F^2 \tag{18}$$

as the functional to be minimized. Select $\lambda(\boldsymbol{W})$ as a linear form on $J(\cdot)$, that is

$$\lambda(\boldsymbol{W}) = \tau J(\lambda(\boldsymbol{W}), \boldsymbol{W}) \tag{19}$$

where τ is a constant that controls the convexity. To ensure that $J(\lambda(\boldsymbol{W}), \boldsymbol{W})$ is convex for any $\boldsymbol{W}$, the parameter τ should satisfy $\tau < 1/\|\boldsymbol{W}\|_F^2$. Usually, we choose $\tau = [1/(2\|\boldsymbol{Y}\|_F^2)]$ which would meet this requirement according to the analysis of [37], and thereby adopting this value in all our experiments. With (18), it follows that:

$$\lambda(\boldsymbol{W}) = \frac{\|\boldsymbol{A}\boldsymbol{W} - \boldsymbol{Y}\|_F^2}{1/\tau - \|\boldsymbol{W}\|_F^2}. \tag{20}$$

With this design, we provide a feasible way to adaptively determine the regularization parameter in a data-driven manner. This could effectively overcome the difficulty of handcrafted hyperparameter.

Remark 1: This adaptive parameter selection can also apply to the case of the parallelism scheme in Section II-C, simply changing (20) to the form of

$$\lambda(\boldsymbol{W}) = \frac{\sum_{t=1}^{k} \|\boldsymbol{A}_t\boldsymbol{W} - \boldsymbol{Y}_t\|_F^2}{1/\tau - \|\boldsymbol{W}\|_F^2}. \tag{21}$$

Remark 2: Although the linear regularization functional form (19) is adopted to determine the regularization parameter in this article, other various choices, such as quadratic regularization functional [37], can also be applied as long as some conditions could be satisfied.

III. Property Analysis of DCBLS

In this section, some properties of the DCBLS method, such as the stability and error estimate, are investigated.

A. Stability Analysis

It is interesting that we wonder if the output weights on the model (12) converge to the ideal solution $\boldsymbol{W}_0^\star$ when $\boldsymbol{A}\boldsymbol{W}_0^\star$ has a sufficiently small perturbation from the original data $\boldsymbol{Y}$. Note that the minimizer $\boldsymbol{W}_\lambda^\star$ of

$$\Phi_{\lambda;\boldsymbol{Y}}(\boldsymbol{W}) \coloneqq \|\boldsymbol{A}\boldsymbol{W} - \boldsymbol{Y}\|_F^2 + \lambda\|\boldsymbol{W}\|_F^2 \tag{22}$$

differs from $\boldsymbol{W}_0^\star$ if $\lambda \neq 0$, we should include the condition of $\lambda \to 0$. Next, the concept of stability is stated as follows, which was established in [38] and [39]. For each $\boldsymbol{W}_0^\star$ in a certain class of functions, there exists a function $\lambda(e)$ of the perturbation level e, such that $\lambda(e) \to 0$ and

$$\sup_{\|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{W}_0^\star\|_F \le e} \left\|\boldsymbol{W}_{\lambda(e);\boldsymbol{Y}}^\star - \boldsymbol{W}_0^\star\right\|_F \to 0 \tag{23}$$

as $e \to 0$, where $\boldsymbol{W}_{\lambda(e);\boldsymbol{Y}}^\star$ is the minimizer of

$$\Phi_{\lambda(e);\boldsymbol{Y}}(\boldsymbol{W}) \coloneqq \|\boldsymbol{A}\boldsymbol{W} - \boldsymbol{Y}\|_F^2 + \lambda(e)\|\boldsymbol{W}\|_F^2. \tag{24}$$

The following theorem characterizes the stability of the proposed DCBLS method. The proof is given in Appendix A.

Theorem 1: Assume that the regularization parameter $\lambda(e) : (0, +\infty) \to (0, +\infty)$ is a function of perturbation level e satisfying

$$\lim_{e\to 0} \lambda(e) = 0 \quad \text{and} \quad \lim_{e\to 0} \frac{e^2}{\lambda(e)} = 0.$$

Then, for $\boldsymbol{W}_0^\star$

$$\lim_{e\to 0}\left(\sup_{\|\boldsymbol{A}\boldsymbol{W}_0^\star - \boldsymbol{Y}\|_F \le e} \left\|\boldsymbol{W}_{\lambda(e);\boldsymbol{Y}}^\star - \boldsymbol{W}^\dagger\right\|_F\right) = 0 \tag{25}$$

where $\boldsymbol{W}_{\lambda(e);\boldsymbol{Y}}^\star$ is a minimizer of $\|\boldsymbol{A}\boldsymbol{W} - \boldsymbol{Y}\|_F^2 + \lambda(e)\|\boldsymbol{W}\|_F^2$ and $\boldsymbol{W}^\dagger$ is the unique element in $\Gamma = \{\boldsymbol{W} | \boldsymbol{A}\boldsymbol{W} = \boldsymbol{A}\boldsymbol{W}_0^\star\}$ minimizing the norm $\|\cdot\|_F$ and the supreme in (25) takes over all $\boldsymbol{Y}$ satisfying $\|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{W}_0^\star\|_F \le e$ for some $e > 0$.

B. Error Estimate

Now, we are in a position to discuss the convergence rate of the proposed approach. In the classical Tikhonov regularization problem [40], the error estimate in the case of vectors has been explored. Similarly, the next theorem shows that the error estimate of DCBLS holds under mild conditions.

Theorem 2: Assume $\boldsymbol{W}^\dagger$ is the same as Theorem 1 and $\|\boldsymbol{A}\boldsymbol{W}^\dagger - \boldsymbol{Y}\|_F \le e$. Let $\boldsymbol{W}_{\lambda(e);\boldsymbol{Y}}^\star$ be a minimizer of $\|\boldsymbol{A}\boldsymbol{W} - \boldsymbol{Y}\|_F^2 + \lambda\|\boldsymbol{W}\|_F^2$. If there exists some $\boldsymbol{\beta}$ such that $\boldsymbol{A}^*\boldsymbol{\beta} = \boldsymbol{W}^\dagger$, where $\boldsymbol{A}^*$ is the adjoint matrix, then the following conclusions hold:

$$\left\|\boldsymbol{A}\boldsymbol{W}_{\lambda(e);\boldsymbol{Y}}^\star - \boldsymbol{Y}\right\|_F \le e + 2\lambda\|\boldsymbol{\beta}\|_F \tag{26}$$

and

$$\left\|\boldsymbol{W}_{\lambda(e);\boldsymbol{Y}}^\star - \boldsymbol{W}^\dagger\right\|_F \le \frac{e}{\sqrt{\lambda}} + \sqrt{\lambda}\|\boldsymbol{\beta}\|_F. \tag{27}$$

The proof is shown in Appendix B. Theorem 2 implies that for the choice $\lambda = \mathcal{O}(e)$, $\boldsymbol{W}_{\lambda(e);\boldsymbol{Y}}^\star$ achieves a convergence rate of $\mathcal{O}(e^{1/2})$.

IV. Experiments

In this section, we will apply the proposed DCBLS method to the image denoising task. A series of experiments is performed to illustrate the performance of DCBLS. The experiments are carried out in a MATLAB R2017a environment running on an Intel, Xeon CPU E5-2650 v3 @2.3 GHz with 128-GB main memory. We use the feature mapping function with a linear function and an enhancement nodes activation function $\xi_j(\cdot) = \tanh(\cdot)$ for simplicity. In the experiments, only the Gaussian noise is used to validate the effectiveness of the proposed method.

A. Natural Image Denoising

1) Experimental Setup: To train the proposed model, 400 images with 180×180 from the Berkeley segmentation dataset (BSD400) [41] are used. We extract small image patches of size 13×13 from the BSD400 images. The noisy images are obtained by adding independent and identically distributed Gaussian noise. Unfold each image patch into the row vector and then form the entire training set. Totally, approximately 1.3 million noisy–clean patch pairs are cropped. In this experiment, we consider three noise level $\sigma = 15, 25, 50$, and set the number of feature mapping groups $n = 6$, feature units on each group $n_f = 55$, the number of enhancement nodes groups $m = 9$, and enhancement units on each group $m_e = 30$. These are determined empirically and discussed in the next section. In addition, the regularization parameter selection rule refers to the method in Section II-D and the initial value is set as 1. Simultaneously, the parallelism scheme is adopted and each subset is the size of 6498 samples with a total of $k = 200$. We compare the proposed method with several state-of-the-art denoising approaches, including the block-matching and 3-D filtering (BM3D) algorithm [42], the weighted nuclear norm minimization (WNNM) approach [43], the multichannel WNNM (MCWNNM) approach [44], the trilateral weighted sparse coding (TWSC) method [45], MLP [10],

TABLE I
PSNR VALUES OF DIFFERENT METHODS ON SET12 WITH VARIOUS NOISE LEVELS

Image	Cameraman	House	Peppers	Starfish	Monar	Airpl	Parrot	Lena	Barbara	Boat	Man	Couple	Average
Noise level						$\sigma = 15$							
BM3D	31.1495	31.0944	31.1332	31.0825	31.0305	31.1493	31.1375	31.6459	31.6589	31.6771	31.6597	31.6420	31.3384
WNNM	32.1093	35.1377	32.8985	31.7795	32.7580	31.4431	31.7155	34.3800	**33.4902**	32.2401	32.1568	32.1435	32.6877
MCWNNM	31.6299	34.6469	32.4187	31.2616	32.1025	30.8912	31.1866	34.0298	33.2058	31.8295	31.7132	31.6927	31.8430
TWSC	31.9033	34.8272	32.5277	31.3848	32.1638	30.9728	31.4005	34.1981	33.3142	32.1190	31.9986	32.0076	32.4015
TNRD	32.1095	34.5861	33.0748	31.7484	32.7340	31.5045	31.6958	34.2612	32.0844	32.1548	32.2167	32.0730	32.5203
DnCNN	32.6051	34.9711	33.3007	32.1995	33.0941	31.6986	31.8251	34.6203	32.6404	32.4161	32.4589	32.4725	32.8585
FFDNet	32.4158	35.0134	33.0976	32.0221	32.7713	31.5849	31.7686	34.6327	32.5023	32.3510	32.4047	32.4544	32.7516
MWCNN	**32.7618**	35.6322	33.5134	32.4297	33.3583	31.8789	32.0009	34.8933	33.3040	32.6856	32.6089	32.7304	33.1498
FC-AIDE	32.3815	35.1586	33.1754	31.9478	32.9004	31.5051	31.8314	34.543	32.4632	32.3236	32.3503	32.4264	32.7505
BLS	29.8442	35.1637	33.0840	33.0430	33.1837	31.5178	31.4542	36.1587	30.4079	34.1893	35.1117	34.8099	33.1640
DCBLS	31.5247	**37.6154**	**33.8553**	**34.0722**	**33.5191**	**31.9395**	**32.0801**	**37.6754**	30.4105	**34.7935**	**35.5204**	**35.2417**	**34.0206**
Noise level						$\sigma = 25$							
BM3D	28.3923	28.2756	28.3750	28.3742	28.3236	28.3168	28.4025	29.6426	29.6416	29.6756	29.6568	29.6759	28.8961
WNNM	29.5295	32.9515	30.0428	28.8837	29.7772	28.5571	29.1284	32.0627	30.9916	29.9062	29.7362	29.6373	30.1004
MCWNNM	29.1100	32.6123	29.8594	28.5469	29.3132	28.3146	28.7720	31.6050	30.4668	29.4261	29.2314	29.0737	29.6943
TWSC	29.3869	32.7107	30.0351	28.8184	29.4556	28.4048	28.9495	32.0523	**31.0049**	29.8939	29.6471	29.6937	30.0044
MLP	29.5901	32.5827	30.3888	28.8139	29.5982	28.8627	29.2693	32.2468	29.5128	29.9540	29.8120	29.6806	30.0260
TNRD	29.6378	32.6709	30.6401	28.9160	30.0464	28.9012	29.2567	31.9835	29.3574	29.9325	29.8329	29.6915	30.0722
DnCNN	30.1839	33.0593	**30.8664**	29.4138	**30.2816**	29.1294	29.4264	32.4371	29.9969	30.2076	30.1033	30.1238	30.4358
FFDNet	30.0585	33.2684	30.7874	29.3343	30.1444	29.0458	**29.4308**	32.5910	29.9780	30.2260	30.0971	30.1834	30.4288
MWCNN	30.0785	33.0202	30.7915	29.8078	30.0394	**29.189**	29.1327	32.8740	29.8684	30.4256	30.1527	30.2709	30.4709
FC-AIDE	**30.0869**	33.2703	30.813	29.3514	30.2621	29.0547	29.4233	32.463	29.988	30.2335	29.985	30.0849	30.418
BLS	28.1879	33.0036	30.1254	29.2783	29.4489	28.4341	28.4520	34.3027	27.3050	31.1847	31.2101	30.2211	30.0961
DCBLS	28.2356	**34.1098**	30.1894	**30.1437**	29.5883	28.5201	28.5539	**34.3272**	27.3067	**31.4921**	**32.2042**	**31.6041**	**30.5229**
Noise level						$\sigma = 50$							
BM3D	25.2802	25.2359	25.3067	25.2197	25.2536	25.2996	25.2937	26.5613	26.5874	26.5740	26.5696	26.5362	25.8098
WNNM	26.2158	29.4988	26.5577	25.0860	25.9382	25.2874	25.7185	28.8356	27.3116	26.7375	26.7262	26.3976	26.6926
MCWNNM	25.8586	28.6398	26.1312	24.6358	25.4210	24.8971	25.6830	28.1105	26.2444	25.9579	26.0826	25.3273	26.0824
TWSC	26.3731	29.8992	26.6490	25.3597	26.1014	25.2770	26.0188	28.9596	27.4535	26.8178	26.7507	26.4047	26.8387
MLP	26.3708	29.6210	26.8420	25.3449	26.2136	25.6419	26.1115	29.2905	25.2943	26.9703	27.0338	26.6659	26.7834
TNRD	26.8338	29.3912	27.0069	25.4069	26.3082	25.6105	26.1718	28.8497	25.7176	26.8737	26.9919	26.5220	26.8070
DnCNN	27.0298	30.0032	27.3162	25.7015	26.7830	25.8687	26.4780	29.3904	26.2188	27.2043	27.2380	26.9030	27.1779
FFDNet	27.0287	30.4269	27.4262	25.7687	26.8846	25.8975	26.5771	29.6784	26.4766	27.3162	27.2956	27.0730	27.3208
MWCNN	27.3178	31.0868	27.6839	26.3434	27.1616	26.1559	26.8135	30.1002	27.5695	27.6285	27.4770	27.489	27.7356
FC-AIDE	27.0532	**30.5991**	27.4869	25.5805	26.9112	25.7817	26.5641	29.4819	26.4476	27.2917	27.1676	27.0259	27.2826
BLS	27.0217	28.3461	27.1790	26.6914	26.5203	27.0170	26.2754	29.5990	27.2654	28.8021	29.0042	27.9673	27.6407
DCBLS	**27.2160**	29.3522	**27.7845**	**27.6767**	**26.9366**	**27.1691**	**26.7596**	**29.9885**	**27.7578**	**29.0075**	**29.3975**	**28.9611**	**28.1673**

TABLE II
MPSNRS AND SSIMS OF DIFFERENT METHODS ON BSD68 WITH VARIOUS NOISE LEVEL

	Index	BM3D	WNNM	MCWNNM	TWSC	MLP	TNRD	DnCNN	FFDNet	MWCNN	FC-AIDE	BLS	DCBLS
$\sigma = 15$	MPSNR	31.08	31.3688	30.5972	31.2183	/	31.4306	31.6949	31.6325	31.8588	31.6389	31.8717	**32.2768**
	MSSIM	0.8722	0.8806	0.8585	0.8815	/	0.8828	0.8905	0.8902	0.8947	0.8879	0.9183	**0.9396**
$\sigma = 25$	MPSNR	28.57	28.8215	28.1247	28.7309	28.9489	28.9138	29.2219	29.1925	29.4123	29.1814	29.1226	**29.9963**
	MSSIM	0.8017	0.8155	0.7781	0.8119	0.8197	0.8156	0.8282	0.8289	0.8361	0.823	0.8382	**0.879**
$\sigma = 50$	MPSNR	25.62	25.7788	25.0225	25.7455	26.0118	25.9605	26.269	26.2938	26.5357	26.2604	27.146	**28.1411**
	MSSIM	0.6869	0.7035	0.6375	0.6925	0.7082	0.7027	0.7213	0.7245	0.7364	0.711	0.743	**0.7629**

TNRD [12], DnCNN [18], FFDNet [19], multilevel wavelet-CNN (MWCNN) [46], and fully convolutional adaptive image denoiser (FC-AIDE) [47]. Especially, we also list the results of BLS [30] to further verify the fact that DCBLS could extract more available information. For a fair comparison, the parameters settings of BLS are the same as that of DCBLS.

2) Quantitative and Qualitative Evaluation: Three widely used benchmark datasets [18] are tested to assess the denoising performance. One dataset is Set12 containing 12 gray images. The second dataset is referred to as BSD68 that has 68 natural gray images from BSD. The last dataset is Set5 including five color images. We regard each channel of the color image as a gray image and apply it to the DCBLS framework. Note that all these images are not included in the training dataset. In the experiments, the peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) indexes are calculated for quantitative evaluation (see [48]). In the case of color images, we compute the average of PSNRs and SSIMs for three channels as the final results. To demonstrate that DCBLS could enhance the stability over BLS, we run the 15 repeated trials. The stability is measured by the standard deviation (SD) of PSNRs. Smaller SD indicates a more stable result.

Tables I–III list the results of the competing methods on the two gray images and one color image datasets, respectively. The best results are highlighted in bold. The results of MLP with noise level 15 are not shown since Burger *et al.* [10] did not give the training weight matrix for this case and thereby the test images could not be validated. From Table I, the PSNRs of most images for DCBLS in various noise levels are higher than those of the other 11 approaches and DCBLS acquires the best denoising results. It could also be found from Tables II and III that the mean PSNR (MPSNR) and SSIM values of DCBLS outperform those of other approaches. These show the advantage of DCBLS in image denoising. Furthermore, Figs. 3–6 illustrate the visual results of different denoising methods, where two images ("house" and "starfish") from Set12, one

TABLE III
MPSNRS AND SSIMS OF DIFFERENT METHODS ON SET5 WITH VARIOUS NOISE LEVEL

	Index	BM3D	WNNM	MCWNNM	TWSC	MLP	TNRD	DnCNN	FFDNet	MWCNN	FC-AIDE	BLS	DCBLS
$\sigma = 15$	MPSNR	32.6099	32.8854	31.3729	32.7557	/	32.8281	34.2894	34.3068	33.3553	33.0525	34.1791	**35.7628**
	MSSIM	0.8898	0.8934	0.8647	0.8722	/	0.8918	0.9174	0.9175	0.9049	0.9008	0.9293	**0.9483**
$\sigma = 25$	MPSNR	30.3452	30.437	28.9473	29.421	30.535	30.5218	32.1064	32.11	31.1205	30.8555	31.5535	**32.2495**
	MSSIM	0.8459	0.848	0.806	0.8217	0.8499	0.8467	0.8836	0.8828	0.8654	0.8583	0.8869	**0.9036**

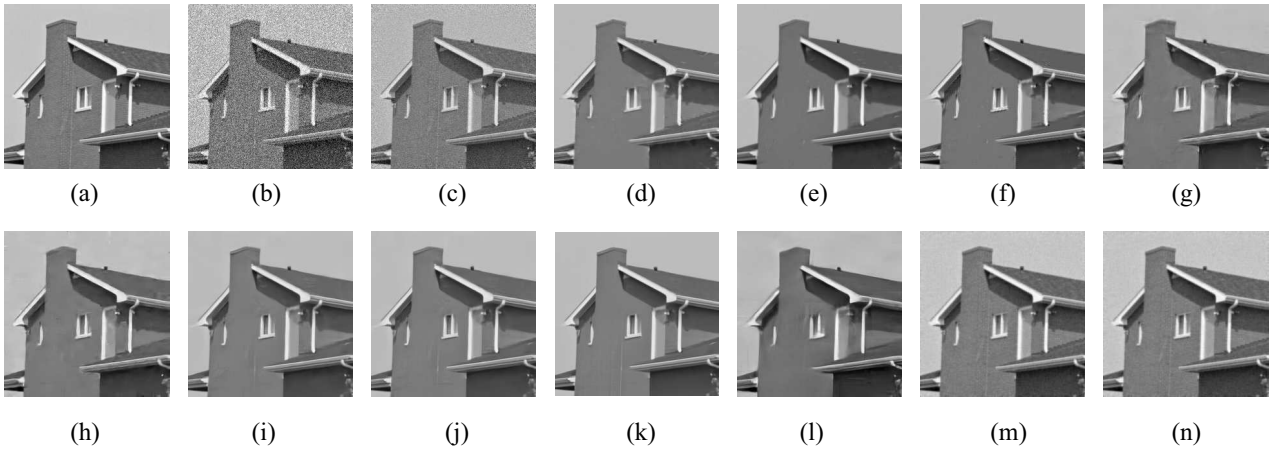


Fig. 3. Comparisons of denoising results for image "house" from Set12. (a) Original. (b) Noisy. (c) BM3D. (d) WNNM. (e) MCWNNM. (f) TWSC. (g) MLP. (h) TNRD. (i) DnCNN. (j) FFDNet. (k) MWCNN. (l) FC-AIDE. (m) BLS. (n) DCBLS.

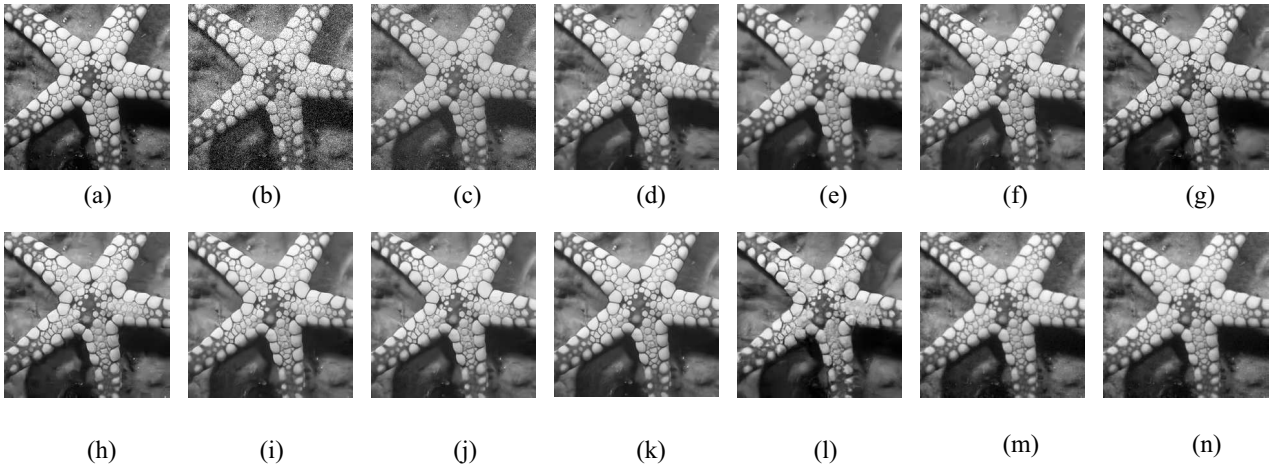


Fig. 4. Comparisons of denoising results for image "starfish" from Set12. (a) Original. (b) Noisy. (c) BM3D. (d) WNNM. (e) MCWNNM. (f) TWSC. (g) MLP. (h) TNRD. (i) DnCNN. (j) FFDNet. (k) MWCNN. (l) FC-AIDE. (m) BLS. (n) DCBLS.

from BSD68 with noise level 25, and the other color image from Set5 are used as examples. As can be seen, BM3D could not remove the noise completely, while WNNM, MCWNNM, TWSC, MLP, TNRD, DnCNN, FFDNet, MWCNN, and FC-AIDE tend to produce over-smooth edges and textures. BLS can remove the noise, but some local details are lost. In contrast, DCBLS could preserve more edges and fine details. This shows that DCBLS could extract more useful information. In brief, these results demonstrate the effectiveness of DCBLS.

Also, the SDs are recorded in Table IV. It can be observed that DCBLS obtains the smallest SDs comparing to BLS and the case of $\lambda = 0$. This further validates the stability of DCBLS and is consistent with the theory analysis. It is probably since the regularization term could promote the stability of the proposed method. Furthermore, we compare the performance of adaptive λ and a fixed λ from the set $[0, 0.01, 0.1, 0.5, 1]$ which takes Set12 as an example. The results are listed in Table V. It is easy to see that the dynamic λ obtains better results, indicating the superiority and effectiveness of the proposed adaptive regularization parameter selection criterion.

Again, Table VI records the running time and the number of parameters of different methods with noise level 25. It should be emphasized that the training time of BM3D, WNNM, MCWNNM, and TWSC without the training process is not given any values. MLP, MWCNN, and FC-AIDE are not trained on the BSD400 dataset and we do not give the training time. Moreover, the training time of TNRD, DnCNN, and FFDNet stems from the results in [12], [18], and [19]. The test time of all the methods is computed in

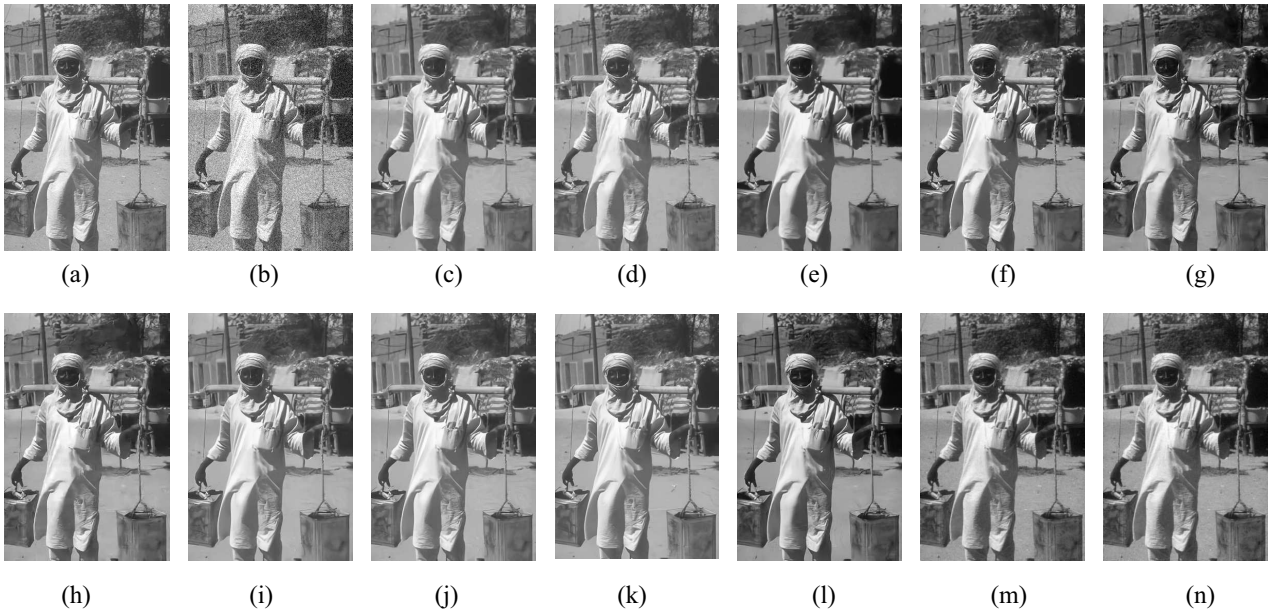


Fig. 5. Denoising results of one image from BSD68 with noise level 25. (a) Original. (b) Noisy. (c) BM3D (28.6296). (d) WNNM (28.8279). (e) MCWNNM (28.1789). (f) TWSC (28.7090). (g) MLP (28.9120). (h) TNRD (28.9944). (i) DnCNN (29.2788). (j) FFDNet (29.2677). (k) MWCNN (29.5204). (l) FC-AIDE (29.2457). (m) BLS (30.0787). (n) DCBLS (30.5143). Here, the values in the brackets are the PSNRs.

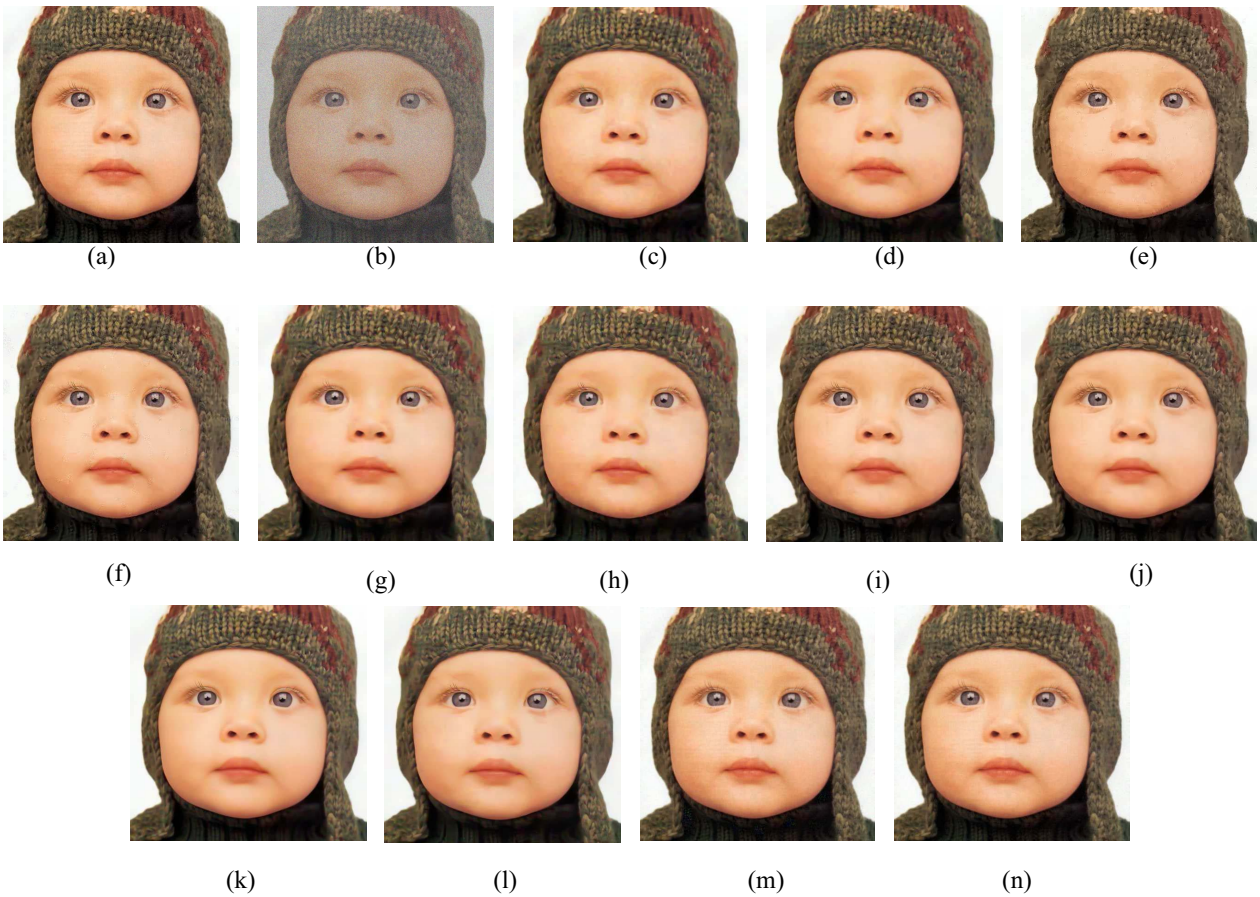


Fig. 6. Denoising results of one image from Set5 with noise level 25. (a) Original. (b) Noisy. (c) BM3D (31.627). (d) WNNM (31.5994). (e) MCWNNM (30.3101). (f) TWSC (30.6572). (g) MLP (31.7982). (h) TNRD (31.6286). (i) DnCNN (33.2757). (j) FFDNet (33.3205). (k) MWCNN (32.0663). (l) FC-AIDE (31.8613). (m) BLS (34.7063). (n) DCBLS (35.4932). Here, the values in the brackets are the PSNRs.

a server with Intel, Xeon E5-2650 v3 CPU 2.3 GHz with 128-GB main memory. It is clear that DCBLS takes 224.53 s for training, which is slightly more than that of BLS and far less than that of other deep-learning methods (TNRD, DnCNN, and FFDNet) according to Table VI. This is reasonable that since DCBLS is more complicated than BLS,

TABLE IV
SDs FOR BLS AND DCBLS ON DIFFERENT DATASETS WITH VARIOUS NOISE LEVEL

Datasets	Set12			BSD68			Set5	
Noise level	$\sigma = 15$	$\sigma = 25$	$\sigma = 50$	$\sigma = 15$	$\sigma = 25$	$\sigma = 50$	$\sigma = 15$	$\sigma = 25$
BLS	0.0445	0.07	0.132	0.4522	0.6102	0.7522	0.2715	0.5072
DCBLS ($\lambda = 0$)	0.014	0.078	0.059	0.13	0.22	0.28	0.275	0.0329
DCBLS	**0.005**	**0.0052**	**0.0086**	**0.0008**	**0.0021**	**0.0023**	**0.0171**	**0.0255**

TABLE V
COMPARISONS OF PSNRS ON FIXED REGULARIZATION PARAMETER AND ADAPTIVE REGULARIZATION PARAMETER FOR SET12

	$\lambda = 0$	$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 0.5$	$\lambda = 1$	adaptive λ
$\sigma = 15$	34.0152	34.0193	34.0174	34.007	33.9712	**34.0206**
$\sigma = 25$	30.4988	30.5204	30.5198	30.5153	30.4932	**30.5229**
$\sigma = 50$	28.1654	28.1658	28.1671	28.1626	28.1612	**28.1673**

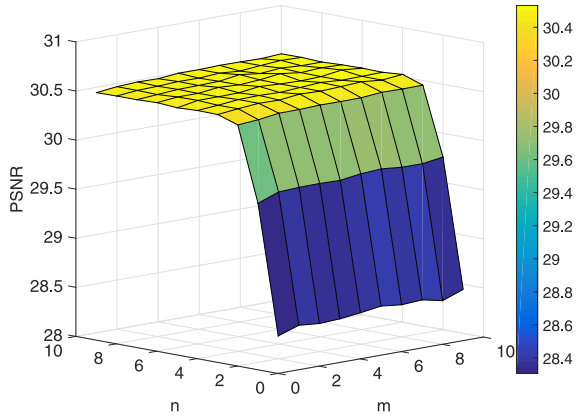


Fig. 7. Effects of different feature mapping nodes and enhancement nodes for DCBLS.

and other deep-learning approaches involve a great number of parameters and complex structures. On the other hand, the test time of DCBLS on Set12, BSD68, and Set5 is roughly at least twice as fast as that of other approaches except FFDNet and BLS. The possible explanation is that DCBLS is a discriminant learning way and is able to get rid of the iterative optimization procedure in the test phase in comparison with the BM3D, WNNM, MCWNNM, and TWSC. Meanwhile, the layer number of deep-learning approaches is deeper than that of DCBLS and thereby increasing the computational time. Besides, in terms of Table VI, DCBLS has fewer parameters compared with other deep-learning methods except for TNRD. Overall, the above-mentioned analyses validate the efficiency of the proposed method in both training and testing.

To further verify the influences of layers in the feature mapping nodes n and enhancement nodes m, the PSNRs with different trials on n and m from an integer set $[1, 2, \ldots, 10]$ are given in Fig. 7. It can be seen that DCBLS can obtain the main information from the feature mapping nodes and further extract the supplementary information from the enhancement nodes. In the experiment, we empirically select the number of layers corresponding to the best PSNR as the final result, that is, $n = 6$ and $m = 9$. However, how to devise a better criterion to determine the number of layers adaptively is still a challenging issue and we will explore it in the future.

B. Hyperspectral Image Denoising

1) Experimental Setup: In this section, HSIs are exploited to verify the denoising performance. The Washington DC Mall dataset [49], including 191 spectral bands and 1208×307 pixels in each band, which was collected by the hyperspectral digital imagery collection experiment (HYDICE). We divide the dataset into two parts of $1080 \times 307 \times 191$ for training and of $200 \times 307 \times 191$ for testing. A zero-mean Gaussian noise is added to all the bands of the dataset, where the noise level is 25. Each noisy spectral vector and original spectral vector are regarded as input and output of DCBLS, respectively. A total of almost 0.33 million training sample pairs and 61 400 testing sample pairs. To facilitate the numerical computation and visualization, the gray values of each spectral band are normalized into [0, 1] and then will be stretched to the original level after denoising. In the experiment, the parameters on the number of feature mapping groups, feature units on each group, the number of enhancement nodes groups, and enhancement units on each group are, respectively, set as 9, 20, 10, and 30 empirically. Moreover, the regularization parameter selection rule is adopted as described in Section II-D and the initial value is set as 1. Furthermore, the parallelism scheme is utilized and each subset is the size of 2763 samples with a total of $k = 120$. We compare the proposed method with several related HSI denoising approaches, containing the block-matching and 4-D filtering (BM4D) algorithm [50], a low-rank matrix recovery (LRMR) method [48], LRMR in a low-energy principal component analysis (called pLRMR) [51], and a weighted Schatten p-norm low-rank matrix approximation (WSNLRMA) approach [52]. Moreover, the results of BLS [30] are also given.

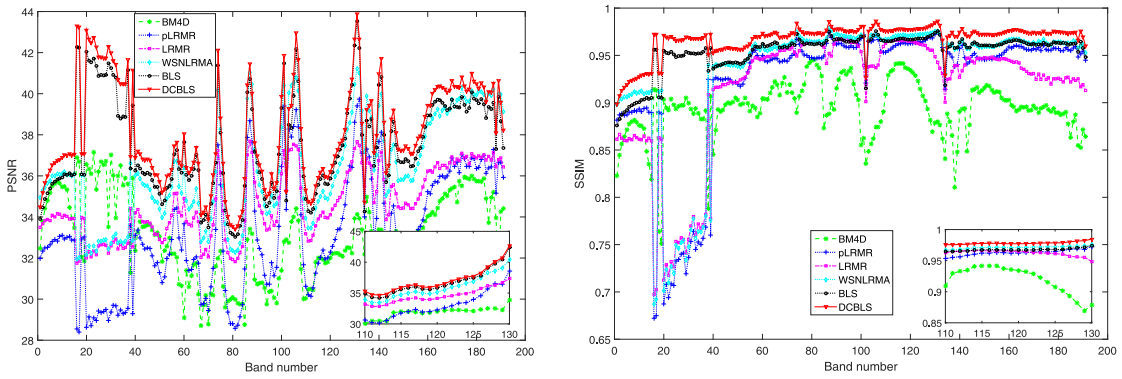
2) Quantitative and Qualitative Evaluation: The two indices (PSNR and SSIM) [48] are still adopted to perform the quantitative evaluation in the HSI denoising. The MPSNR and mean SSIM (MSSIM) are computed by averaging over the PSNR and SSIM of all the spectral bands, respectively. In Fig. 8, the PSNR and SSIM of each spectral band for the Washington DC Mall dataset are calculated. Apparently, it is seen that the PSNR and SSIM of almost all bands obtained by DCBLS are higher than those of BM4D, pLRMR, LRMR, WSNLRMA, and BLS. Furthermore, to make a quantitative comparison from overall, Table VII presents the MPSNR and MSSIM values by different denoising approaches. It is obvious that DCBLS acquires the best denoising results in comparison with other methods. The above-mentioned analysis indicates that DCBLS shows great superiority. Furthermore, we also list the computational test time for six methods in Table VII. The test time of DCBLS is ten or more faster than that of other methods except BLS. Simultaneously, the training time of DCBLS takes only 92.77 s. This verifies the efficiency of the proposed method.

To measure the spectral fidelity, a spectral angle metric (SAM) [23] is used between the original spectrum and the denoising spectrum. The mean SAM (MSAM) is calculated by averaging over the entire spatial domain, and the smaller value indicates better performance. The results are shown in Table VII. One can see that DCBLS produces the smaller MSAM values in comparison with other methods,

TABLE VI
RUNNING TIME (SECONDS) AND THE NUMBER OF PARAMETERS WITH NOISE LEVEL 25 FOR DIFFERENT METHODS

	Dataset	BM3D	WNNM	MCWNNM	TWSC	MLP	TNRD	DnCNN	FFDNet	MWCNN	FC-AIDE	BLS	DCBLS
Training Time	BSD400	/	/	/	/	/	74880 [a]	21600 [a]	172800 [a]	/	/	215.72	224.53
Number of Parameters	-	/	/	/	/	$\approx$ 18000000	26645	627840	485316	$\approx$ 5100000	2586050	101400	202545
Test Time	Set12	40.12	2689	1787	2666.7	93.3	45.95	42.13	7.71	33.88	83.17	19.58	24.82
	BSD68	185.62	20453	7396.1	15051	507.01	175.46	177.68	40.74	157.82	2171.61	78.6	83
	Set5	34.18	3191.4	3432.7	2957.9	50.93	44.22	30.37	17.63	29.3	98.57	11.92	17.8

[a] The training time of TNRD, DnCNN and FFDNet is almost 20.5 hours, 6 hours and 2 days and we convert them into data with seconds, where these results stem from the references [12], [18], and [19], respectively. According to those references, TNRD is based on a server with Intel(R) Xeon E5-2680 CPU 2.80GHz, and DnCNN and FFDNet is running on a PC with Core(TM) i7-5820K CPU 3.30GHz and an Nvidia Titan X GPU. However, the training time of BLS and DCBLS, and the test time of all the methods are computed in a server with Intel(R), Xeon(R) E5-2650 v3 CPU 2.3 GHz with 128 GB main memory.



(a) (b)

Fig. 8. PSNRs and SSIMs of each spectral band by different methods on the Washington DC Mall dataset. (a) PSNR. (b) SSIM. To clearly reflect the performance, insets of (a) and (b) are the amplified curves of different methods ranging from bands 110 to 130.

TABLE VII
COMPARISON OF DIFFERENT METHODS USING THE WASHINGTON DC MALL DATASET

Index	BM4D	pLRMR	LRMR	WSNLRMA	BLS	DCBLS
MPSNR	30.0682	33.2507	34.6447	36.3048	37.4925	**38.1444**
MSSIM	0.8938	0.9217	0.9147	0.9332	0.9542	**0.9675**
MSAM	4.258	3.6135	3.5864	2.8826	2.5073	**2.3136**
Test time	380.57	121.59	146.64	10712	9.81	11.71

clarifying that the proposed method provides the smaller spectral distortion.

In addition, from the view of visual effects, Fig. 9 shows the results of band 10 corrupted by the Gaussian noise. As can be seen, some details of BM4D are over smooth. pLRMR could not remove the noise completely. Some useful local details of LRMR, WSNLRMA, and BLS are lost. Nevertheless, DCBLS could effectively suppress the Gaussian noise and keep the edge and local details of the original image. This advantage may largely attribute to the good learning model. Furthermore, we plot the spectral signatures of the pixel $(43, 144)$ as an example. The result is shown in Fig. 10, in which the horizontal axis and vertical axis represent the spectral band number and digital number (DN) values, respectively. It is noted that the curve shapes of the spectral signatures by DCBLS closely resemble that of noise-free spectral signatures. This can be further determined by the differences between the noise-free spectral signatures and denoising results in Fig. 11, since the proposed method produces the flattest curve in contrast to other approaches.

In a nutshell, the above-mentioned analysis manifests that the proposed DCBLS method is promising for HSI denoising.

V. CONCLUSION

This article proposed a novel adaptive regularization DCBLS method. This new deep structure consists of one cascaded feature mapping nodes layer with a depth n and one cascaded enhancement nodes layer with a depth m, which is likely to obtain more available information from the raw data. Then, a convex regularization model is introduced to enhance the stability and could be addressed easily by closed-form solution. Furthermore, we devise a parallelization framework to large-scale data problems. Moreover, an adaptive regularization parameter selection strategy is presented in terms of some conditions. More important, the stability and error estimate of the model is derived theoretically. Experimental results on the benchmark datasets (including natural images and HSIs) illustrate that the DCBLS method can be successfully applied to image denoising.

There is scope for further research in this area. For instance, other regularization terms, such as $\ell_{2,1}$-norm [53] and ℓ_p-norm [54], are used to replace ℓ_2 regularization in order to keep sparsity of the model and thereby reducing the parameters. Also, investigating the maximum correntropy loss [55] instead of least-squares loss in our framework may learn a more robust model to effectively handle the noisy data.

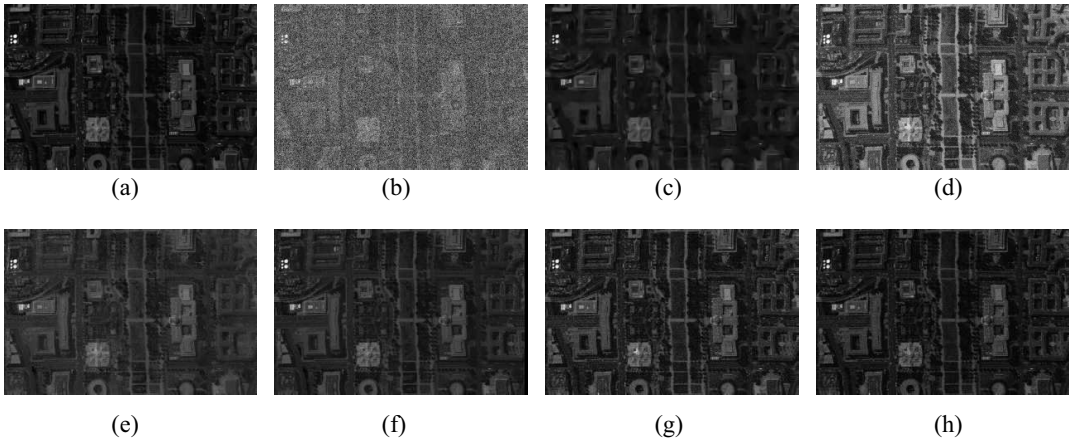


Fig. 9. Denoising results of the Washington DC Mall dataset with noise level 25. (a) Original. (b) Noisy. (c) BM4D. (d) pLRMR. (e) LRMR. (f) WSNLRMA. (g) BLS. (h) DCBLS.

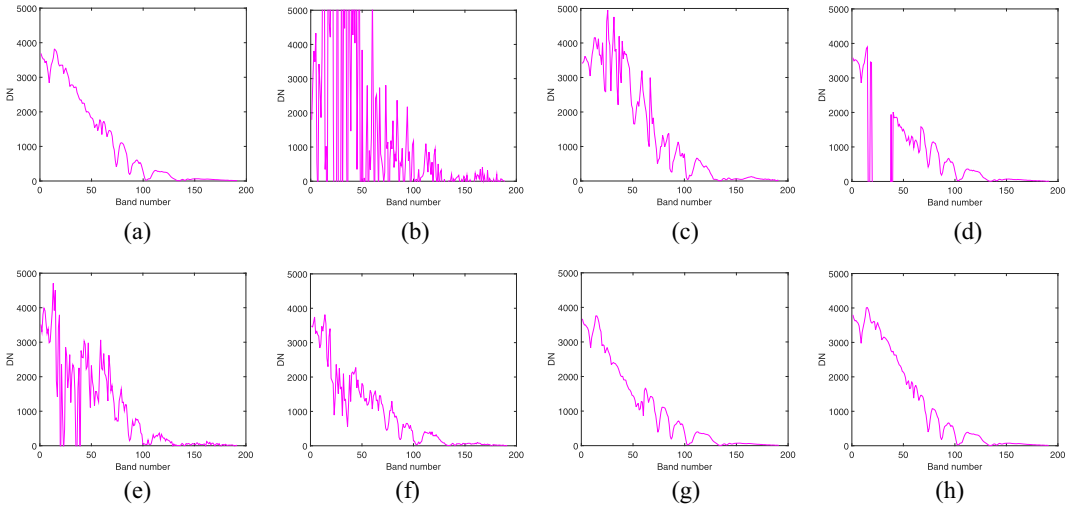


Fig. 10. Spectral signatures of pixel (43, 144) in the denoising results on the Washington DC Mall dataset. (a) Original. (b) Noisy. (c) BM4D. (d) pLRMR. (e) LRMR. (f) WSNLRMA. (g) BLS. (h) DCBLS.

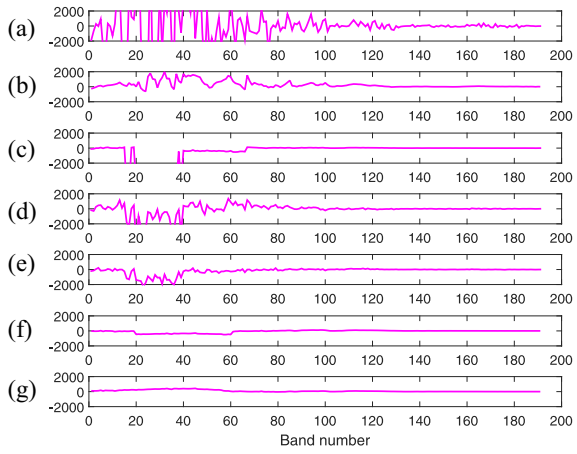


Fig. 11. Differences between the noise-free spectral signatures and the denoising results of pixel (43, 144) on the Washington DC Mall dataset. (a) Noisy. (b) BM4D. (c) pLRMR. (d) LRMR. (e) WSNLRMA. (f) BLS. (g) DCBLS.

From the view of applications, the proposed method could also be applied to other fields such as classification. Furthermore, other types of noises, such as salt noises, could be considered further by using BLS and its variants. Besides, if the noise information is unknown, then the paired training dataset would be unavailable. The proposed approach might not be exploited to deal with such blind image denoising problems directly. As such, further exploration of blind image denoising issues is of great significance. Moreover, some visualization methods, such as the t-distributed stochastic neighbor embedding (t-sne) [56], could be combined with DCBLS in qualitative analysis. Overall, these will be treated in depth in the future.

APPENDIX A
PROOF OF THEOREM 1

We first establish the weak convergence. Let us prove that if $\{\boldsymbol{Y}_l\}_{l=1}^{\infty}$ is a sequence such that $\|\boldsymbol{Y}_l - \boldsymbol{A}\boldsymbol{W}_0^*\|_F \le e_l \to 0$ as $l \to \infty$, then $\boldsymbol{W}^{\star}_{\lambda(e_l);\boldsymbol{Y}}$ weakly converges to $\boldsymbol{W}^{\dagger}$.

In fact, the minimizing property of $\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}$ indicates

$$\begin{aligned}&\left\|\boldsymbol{A}\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}} - \boldsymbol{Y}\right\|_F^2 + \lambda(e)\left\|\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}\right\|_F^2 \\ &\le \left\|\boldsymbol{A}\boldsymbol{W}^{\dagger} - \boldsymbol{Y}\right\|_F^2 + \lambda(e)\left\|\boldsymbol{W}^{\dagger}\right\|_F^2 \\ &\le e^2 + \lambda(e)\left\|\boldsymbol{W}^{\dagger}\right\|_F^2. \end{aligned} \tag{28}$$

By the assumptions on $\lambda(e)$, the sequences $\|\boldsymbol{A}\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{Y}\|_F$ and $\|\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}\|_F$ are uniformly bounded. Accordingly, there exists a subsequence of $\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}$, denoted by $\{\boldsymbol{W}^{\star}_{\lambda(e_l);\boldsymbol{Y}_l}\}_{l=1}^{\infty}$, and some $\widetilde{\boldsymbol{W}}$ such that $\boldsymbol{W}^{\star}_{\lambda_{e_l};\boldsymbol{Y}_l}\to\widetilde{\boldsymbol{W}}$ as $l\to\infty$.

By the weak lower semicontinuity, the triangle inequality and (28), we derive

$$\begin{aligned}&\left\|\boldsymbol{A}\widetilde{\boldsymbol{W}}-\boldsymbol{A}\boldsymbol{W}^{\dagger}\right\|_F^2\\&\quad\le\lim_{e_l\to0}\inf\left\{\left\|\boldsymbol{A}\boldsymbol{W}^{\star}_{\lambda(e_l);\boldsymbol{Y}_l}-\boldsymbol{Y}_l\right\|_F^2+\left\|\boldsymbol{A}\boldsymbol{W}^{\dagger}-\boldsymbol{Y}_l\right\|_F^2\right\}\\&\quad\le\lim_{e_l\to0}\inf\left\{e_l^2+\lambda(e_l)\left\|\boldsymbol{W}^{\dagger}\right\|_F^2+e_l^2\right\}=0\end{aligned}$$

where the second inequality stems from (28). Thereby it follows that $\|\boldsymbol{A}\widetilde{\boldsymbol{W}}-\boldsymbol{A}\boldsymbol{W}^{\dagger}\|_F^2=0$, that is, $\boldsymbol{A}\widetilde{\boldsymbol{W}}=\boldsymbol{A}\boldsymbol{W}^{\dagger}$. This implies $\widetilde{\boldsymbol{W}}\in\Gamma$. Similarly

$$\begin{aligned}\|\widetilde{\boldsymbol{W}}\|_F^2&\le\lim_{e_l\to0}\inf\left\{\left\|\boldsymbol{W}^{\star}_{\lambda(e_l);\boldsymbol{Y}_l}\right\|_F^2\right\}\\&\le\lim_{e_l\to0}\inf\left\{\left\|\boldsymbol{W}^{\dagger}\right\|_F^2+\frac{e_l^2}{\lambda(e_l)}\right\}\\&=\left\|\boldsymbol{W}^{\dagger}\right\|_F^2.\end{aligned}$$

On the other hand, $\boldsymbol{W}^{\dagger}$ is the unique element in $\Gamma=\{\boldsymbol{W}|\boldsymbol{A}\boldsymbol{W}=\boldsymbol{A}\boldsymbol{W}_0^{\star}\}$ minimizing the norm $\|\cdot\|_F$, so we have $\|\boldsymbol{W}^{\dagger}\|_F^2\le\|\widetilde{\boldsymbol{W}}\|_F^2$. To summarize, $\|\widetilde{\boldsymbol{W}}\|_F=\|\boldsymbol{W}^{\dagger}\|_F$. By the uniqueness of $\boldsymbol{W}^{\dagger}$ in Γ, it can be concluded $\widetilde{\boldsymbol{W}}=\boldsymbol{W}^{\dagger}$. Analogously, we can prove that any other weakly convergent subsequence of $\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}$ has a weak limit $\boldsymbol{W}^{\dagger}$, and we know $\lim_{l\to\infty}\|\boldsymbol{W}^{\star}_{\lambda(e_l);\boldsymbol{Y}_l}\|_F=\|\boldsymbol{W}^{\dagger}\|_F$. This gives

$$\begin{aligned}&\lim_{l\to\infty}\left\|\boldsymbol{W}^{\star}_{\lambda(e_l);\boldsymbol{Y}_l}-\boldsymbol{W}^{\dagger}\right\|_F^2\\&\quad=\lim_{l\to\infty}\left(\left\|\boldsymbol{W}^{\star}_{\lambda(e_l);\boldsymbol{Y}_l}\right\|_F^2+\left\|\boldsymbol{W}^{\dagger}\right\|_F^2-2\left\langle\boldsymbol{W}^{\star}_{\lambda(e_l);\boldsymbol{Y}_l},\boldsymbol{W}^{\dagger}\right\rangle\right)\\&\quad=\left\|\boldsymbol{W}^{\dagger}\right\|_F^2+\left\|\boldsymbol{W}^{\dagger}\right\|_F^2-2\left\langle\boldsymbol{W}^{\dagger},\boldsymbol{W}^{\dagger}\right\rangle=0\end{aligned}$$

where $\langle\cdot,\cdot\rangle$ represents the inner product. Consequently, it follows that:

$$\lim_{e\to0}\left(\sup_{\|\boldsymbol{A}\boldsymbol{W}_0^{\star}-\boldsymbol{Y}\|_F\le e}\left\|\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{W}^{\dagger}\right\|_F\right)=0.$$

This completes the proof.

APPENDIX B
PROOF OF THEOREM 2

By the minimizing property of $\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}$, there holds

$$\left\|\boldsymbol{A}\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{Y}\right\|_F^2+\lambda\left\|\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}\right\|_F^2\le\left\|\boldsymbol{A}\boldsymbol{W}^{\dagger}-\boldsymbol{Y}\right\|_F^2+\lambda\left\|\boldsymbol{W}^{\dagger}\right\|_F^2$$

which leads to

$$\begin{aligned}&\left\|\boldsymbol{A}\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{Y}\right\|_F^2+\lambda\left(\left\|\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}\right\|_F^2-\left\|\boldsymbol{W}^{\dagger}\right\|_F^2\right)\\&\quad\le\left\|\boldsymbol{A}\boldsymbol{W}^{\dagger}-\boldsymbol{Y}\right\|_F^2.\end{aligned}$$

Using the identity $\|\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}\|_F^2-\|\boldsymbol{W}^{\dagger}\|_F^2=\|\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{W}^{\dagger}\|_F^2+2\langle\boldsymbol{W}^{\dagger},\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{W}^{\dagger}\rangle$, where $\langle\cdot,\cdot\rangle$ represents the inner product, we can obtain

$$\begin{aligned}&\left\|\boldsymbol{A}\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{Y}\right\|_F^2+2\lambda\left\langle\boldsymbol{W}^{\dagger},\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{W}^{\dagger}\right\rangle\\&\quad+\lambda\left\|\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{W}^{\dagger}\right\|_F^2\le\left\|\boldsymbol{A}\boldsymbol{W}^{\dagger}-\boldsymbol{Y}\right\|_F^2.\end{aligned}$$

According to the assumptions, there exists some $\boldsymbol{\beta}$ such that $\boldsymbol{A}^*\boldsymbol{\beta}=\boldsymbol{W}^{\dagger}$. Therefore

$$\begin{aligned}&\left\|\boldsymbol{A}\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{Y}\right\|_F^2+2\lambda\left\langle\boldsymbol{A}^*\boldsymbol{\beta},\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{W}^{\dagger}\right\rangle\\&\quad+\lambda\left\|\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{W}^{\dagger}\right\|_F^2\le\left\|\boldsymbol{A}\boldsymbol{W}^{\dagger}-\boldsymbol{Y}\right\|_F^2.\end{aligned}$$

That is to say

$$\begin{aligned}&\left\|\boldsymbol{A}\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{Y}\right\|_F^2+2\lambda\left\langle\boldsymbol{\beta},\boldsymbol{A}\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{Y}\right\rangle\\&\quad+\lambda\left\|\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{W}^{\dagger}\right\|_F^2\\&\quad\le\left\|\boldsymbol{A}\boldsymbol{W}^{\dagger}-\boldsymbol{Y}\right\|_F^2+2\lambda\left\langle\boldsymbol{\beta},\boldsymbol{A}\boldsymbol{W}^{\dagger}-\boldsymbol{Y}\right\rangle.\end{aligned}$$

Completing the squares on both sides by adding $\lambda^2\|\boldsymbol{\beta}\|_F^2$ leads to

$$\begin{aligned}&\left\|\boldsymbol{A}\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{Y}+\lambda\boldsymbol{\beta}\right\|_F^2+\lambda\left\|\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{W}^{\dagger}\right\|_F^2\\&\quad\le\left\|\boldsymbol{A}\boldsymbol{W}^{\dagger}-\boldsymbol{Y}+\lambda\boldsymbol{\beta}\right\|_F^2.\end{aligned}$$

Accordingly, we have

$$\left\|\boldsymbol{A}\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{Y}+\lambda\boldsymbol{\beta}\right\|_F\le\left\|\boldsymbol{A}\boldsymbol{W}^{\dagger}-\boldsymbol{Y}+\lambda\boldsymbol{\beta}\right\|_F$$

and

$$\left\|\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{W}^{\dagger}\right\|_F\le\frac{1}{\sqrt{\lambda}}\left\|\boldsymbol{A}\boldsymbol{W}^{\dagger}-\boldsymbol{Y}+\lambda\boldsymbol{\beta}\right\|_F.$$

By the triangle inequality, it follows that:

$$\begin{aligned}\left\|\boldsymbol{A}\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{Y}\right\|_F-\lambda\|\boldsymbol{\beta}\|_F&\le\left\|\boldsymbol{A}\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{Y}+\lambda\boldsymbol{\beta}\right\|_F\\&\le\left\|\boldsymbol{A}\boldsymbol{W}^{\dagger}-\boldsymbol{Y}+\lambda\boldsymbol{\beta}\right\|_F\\&\le\left\|\boldsymbol{A}\boldsymbol{W}^{\dagger}-\boldsymbol{Y}\right\|_F+\lambda\|\boldsymbol{\beta}\|_F\\&\le e+\lambda\|\boldsymbol{\beta}\|_F\end{aligned}$$

and

$$\begin{aligned}\left\|\boldsymbol{W}^{\star}_{\lambda(e);\boldsymbol{Y}}-\boldsymbol{W}^{\dagger}\right\|_F&\le\frac{1}{\sqrt{\lambda}}\left(\left\|\boldsymbol{A}\boldsymbol{W}^{\dagger}-\boldsymbol{Y}\right\|_F+\lambda\|\boldsymbol{\beta}\|_F\right)\\&\le\frac{e}{\sqrt{\lambda}}+\sqrt{\lambda}\|\boldsymbol{\beta}\|_F.\end{aligned}$$

This completes the proof.

REFERENCES

[1] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[2] W. Guo, Y.-S. Ong, Y. Zhou, J. R. Hervas, A. Song, and H. Wei, "Fisher information matrix of unipolar activation function-based multilayer perceptrons," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 3088–3098, Aug. 2019.

[3] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[4] K. Zeng, J. Yu, R. Wang, C. Li, and D. Tao, "Coupled deep autoencoder for single image super-resolution," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 27–37, Jan. 2017.

[5] Y. Zhang, M. J. Er, R. Zhao, and M. Pratama, "Multiview convolutional neural networks for multidocument extractive summarization," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3230–3242, Oct. 2017.

[6] L. Wang, X. Qian, Y. Zhang, J. Shen, and X. Cao, "Enhancing sketch-based image retrieval by CNN semantic re-ranking," *IEEE Trans. Cybern.*, early access, doi: 10.1109/TCYB.2019.2894498.

[7] W. Hou, X. Gao, D. Tao, and X. Li, "Blind image quality assessment via deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 6, pp. 1275–1286, Jun. 2015.

[8] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[9] L. Wu, Y. Wang, X. Li, and J. Gao, "Deep attention-based spatially recursive networks for fine-grained visual recognition," *IEEE Trans. Cybern.*, vol. 49, no. 5, pp. 1791–1802, May 2018.

[10] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with BM3D?" in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2392–2399.

[11] S. Nam, Y. Hwang, Y. Matsushita, and S. J. Kim, "A holistic approach to cross-channel image noise modeling and its application to image denoising," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1683–1691.

[12] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1256–1272, Jun. 2017.

[13] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.

[14] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2012, pp. 341–349.

[15] U. Schmidt and S. Roth, "Shrinkage fields for effective image restoration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 2774–2781.

[16] V. Jain and H. S. Seung, "Natural image denoising with convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2009, pp. 769–776.

[17] X.-J. Mao, C. Shen, and Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Proc. Adv. Neural Inf. Process. Syst.*, Barcelona, Spain, Dec. 2016, pp. 2802–2810.

[18] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017.

[19] K. Zhang, W. Zuo, and L. Zhang, "FFDNet: Toward a fast and flexible solution for CNN based image denoising," *IEEE Trans. Image Process.*, vol. 27, no. 9, pp. 4608–4622, Sep. 2018.

[20] S. Lefkimmiatis, "Non-local color image denoising with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 3587–3596.

[21] N. Divakar and R. V. Babu, "Image denoising via CNNs: An adversarial approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1076–1083.

[22] A. Creswell and A. A. Bharath, "Denoising adversarial autoencoders," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 968–984, Apr. 2019.

[23] Q. Yuan, Q. Zhang, J. Li, H. Shen, and L. Zhang, "Hyperspectral image denoising employing a spatial–spectral deep residual convolutional neural network," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 1205–1218, Feb. 2019.

[24] Y. Tai, J. Yang, X. Liu, and C. Xu, "Memet: A persistent memory network for image restoration," in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, Oct. 2017, pp. 4549–4557.

[25] A. Majumdar, "Blind denoising autoencoder," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 1, pp. 312–317, Jan. 2019.

[26] B. Mildenhall, J. T. Barron, J. Chen, D. Sharlet, R. Ng, and R. Carroll, "Burst denoising with kernel prediction networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2502–2510.

[27] B. Du, W. Xiong, J. Wu, L. Zhang, L. Zhang, and D. Tao, "Stacked convolutional denoising auto-encoders for feature representation," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 1017–1027, Apr. 2017.

[28] D. Liu, B. Wen, J. Jiao, X. Liu, Z. Wang, and T. S. Huang, "Connecting image denoising and high-level vision tasks via deep learning," *IEEE Trans. Image Process.*, vol. 29, no. 1, pp. 3695–3706, Jan. 2020.

[29] C. L. P. Chen and Z. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 10–24, Jan. 2018.

[30] C. L. P. Chen, Z. Liu, and S. Feng, "Universal approximation capability of broad learning system and its structural variations," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 1191–1204, Apr. 2019.

[31] M. Han, S. Feng, C. L. P. Chen, M. Xu, and T. Qiu, "Structured manifold broad learning system: A manifold perspective for large-scale chaotic time series analysis and prediction," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 9, pp. 1809–1821, Sep. 2019.

[32] S. Feng and C. L. P. Chen, "Fuzzy broad learning system: A novel neuro-fuzzy model for regression and classification," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 414–424, Feb. 2020.

[33] Y. Kong, X. Wang, Y. Cheng, and C. L. P. Chen, "Hyperspectral imagery classification based on semi-supervised broad learning system," *Remote Sens.*, vol. 10, no. 5, p. 685, May 2018.

[34] T.-L. Zhang, R. Chen, X. Yang, and S. Guo, "Rich feature combination for cost-based broad learning system," *IEEE Access*, vol. 7, pp. 160–172, 2019.

[35] G. Deshpande, P. Wang, D. Rangaprakash, and B. Wilamowski, "Fully connected cascade artificial neural network architecture for attention deficit hyperactivity disorder classification from functional magnetic resonance imaging data," *IEEE Trans. Cybern.*, vol. 45, no. 12, pp. 2668–2679, Dec. 2015.

[36] J. Schlemper, J. Caballero, J. V. Hajnal, A. N. Price, and D. Rueckert, "A deep cascade of convolutional neural networks for dynamic MR image reconstruction," *IEEE Trans Med. Imag.*, vol. 37, no. 2, pp. 491–503, Feb. 2018.

[37] M. G. Kang and A. K. Katsaggelos, "General choice of the regularization functional in regularized image restoration," *IEEE Trans. Image Process.*, vol. 4, no. 5, pp. 594–602, May 1995.

[38] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Commun. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, Aug. 2004.

[39] F. Cao, D. Wang, H. Zhu, and Y. Wang, "An iterative learning algorithm for feedforward neural networks with random weights," *Inf. Sci.*, vol. 328, pp. 546–557, Jan. 2016.

[40] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of Inverse Problems*. Dordrecht, The Netherlands: Kluwer, 1996.

[41] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2, Jul. 2001, pp. 416–423.

[42] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[43] S. Gu, Q. Xie, D. Meng, W. Zuo, X. Feng, and L. Zhang, "Weighted nuclear norm minimization and its applications to low level vision," *Int. J. Comput. Vis.*, vol. 121, no. 2, pp. 183–208, Jan. 2017.

[44] J. Xu, L. Zhang, D. Zhang, and X. Feng, "Multi-channel weighted nuclear norm minimization for real color image denoising," in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, Oct. 2017, pp. 1096–1104.

[45] J. Xu, L. Zhang, and D. Zhang, "A trilateral weighted sparse coding scheme for real-world image denoising," in *Proc. Eur. Conf. Comput. Vis.*, Munich, Germany, Sep. 2018, pp. 20–36.

[46] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo, "Multi-level wavelet-CNN for image restoration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 773–782.

[47] S. Cha and T. Moon, "Fully convolutional pixel adaptive image denoiser," in *Proc. IEEE Int. Conf. Comput. Vis.*, Seoul, South Korea, Oct. 2019, pp. 4160–4169.

[48] H. Zhang, W. He, L. Zhang, H. Shen, and Q. Yuan, "Hyperspectral image restoration using low-rank matrix recovery," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 8, pp. 4729–4743, Aug. 2014.

[49] L. Zhang, Q. Zhang, B. Du, X. Huang, Y. Y. Tang, and D. Tao, "Simultaneous spectral–spatial feature selection and extraction for hyperspectral images," *IEEE Trans. Cybern.*, vol. 48, no. 1, pp. 16–28, Jan. 2018.

[50] M. Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi, "Nonlocal transform-domain filter for volumetric data denoising and reconstruction," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 119–133, Jan. 2013.
[51] H. Song, G. Wang, and K. Zhang, "Hyperspectral image denoising via low-rank matrix recovery," *Remote Sens. Lett.*, vol. 5, no. 10, pp. 872–881, Oct. 2014.
[52] Y. Xie, Y. Qu, D. Tao, W. Wu, Q. Yuan, and W. Zhang, "Hyperspectral image restoration via iteratively regularized weighted Schatten p-norm minimization," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 8, pp. 4642–4659, Apr. 2016.
[53] L. Zhang, L. Zhang, B. Du, J. You, and D. Tao, "Hyperspectral image unsupervised classification by robust manifold matrix factorization," *Inf. Sci.*, vol. 485, pp. 154–169, Jun. 2019.
[54] H. Li and J. Wen, "A new analysis for support recovery with block orthogonal matching pursuit," *IEEE Signal Process. Lett.*, vol. 26, no. 2, pp. 247–251, Feb. 2019.
[55] B. Du, T. Xinyao, Z. Wang, L. Zhang, and D. Tao, "Robust graph-based semisupervised learning for noisy labeled data via maximum correntropy criterion," *IEEE Trans. Cybern.*, vol. 49, no. 4, pp. 1440–1453, Apr. 2019.
[56] L. V. D. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 2579–2605, Nov. 2008.

Hailiang Ye received the B.Sc. and M.Sc. degrees in applied mathematics from China Jiliang University, Hangzhou, China, in 2012 and 2015, respectively, and the Ph.D. degree in computational mathematics from the Huazhong University of Science and Technology, Wuhan, China, in 2019.

He is currently a Lecturer with the College of Sciences, China Jiliang University. His research interests include pattern recognition, image processing, neural networks, and deep learning.

Hong Li (Member, IEEE) received the M.Sc. degree in mathematics and the Ph.D. degree in pattern recognition and intelligence control from the Huazhong University of Science and Technology, Wuhan, China, in 1986 and 1999, respectively.

She is currently a Professor with the School of Mathematics and Statistics, Huazhong University of Science and Technology. Her current research interests include approximation theory, wavelet analysis, learning theory, neural networks, signal processing, and pattern recognition.

C. L. Philip Chen (Fellow, IEEE) received the M.S. degree in electrical engineering from the University of Michigan at Ann Arbor, Ann Arbor, MI, USA, in 1985, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1988.

He is currently a Chair Professor with the Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Macau, China. He is currently a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. His current research interests include systems, cybernetics, and computational intelligence.

Prof. Chen was the Chair of the TC 9.1 Economic and Business Systems of the International Federation of Automatic Control from 2015 to 2017. He is the Editor-in-Chief of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS and the IEEE TRANSACTIONS ON CYBERNETICS, and an Associate Editor of the IEEE TRANSACTIONS ON FUZZY SYSTEMS. From 2012 to 2013, he was the IEEE Systems, Man, and Cybernetics Society President. He is a fellow of the American Association for the Advancement of Science, the international Association for Pattern Recognition, the Chinese Association of Automation, and HKIE.