

Learning Deep Analysis Dictionaries for Image Super-Resolution

Jun-Jie Huang, *Member, IEEE*, and Pier Luigi Dragotti, *Fellow, IEEE*

Abstract—Inspired by the recent success of deep neural networks and the recent efforts to develop multi-layer dictionary models, we propose a Deep Analysis dictionary Model (DeepAM) which is optimized to address a specific regression task known as single image super-resolution. Contrary to other multi-layer dictionary models, our architecture contains L layers of analysis dictionary and soft-thresholding operators to gradually extract high-level features and a layer of synthesis dictionary which is designed to optimize the regression task at hand. In our approach, each analysis dictionary is partitioned into two sub-dictionaries: an Information Preserving Analysis Dictionary (IPAD) and a Clustering Analysis Dictionary (CAD). The IPAD together with the corresponding soft-thresholds is designed to pass the key information from the previous layer to the next layer, while the CAD together with the corresponding soft-thresholding operator is designed to produce a sparse feature representation of its input data that facilitates discrimination of key features. DeepAM uses both supervised and unsupervised setup. Simulation results show that the proposed deep analysis dictionary model achieves better performance compared to a deep neural network that has the same structure and is optimized using back-propagation when training datasets are small. On noisy image super-resolution, DeepAM can be well adapted to unseen testing noise levels by rescaling the IPAD and CAD thresholds of the first layer.

Index Terms—Dictionary Learning, Analysis Dictionary, Deep Neural Networks, Deep Model.

I. INTRODUCTION

DEEP Neural Networks (DNNs) [1] are complex architectures composed of a cascade of multiple linear and non-linear layers. Back-propagation algorithm [2] is usually applied to optimize the parameters of the linear transforms and the non-linearities within this highly non-linear and non-convex system. With the help of massive labeled training data and powerful Graphics Processing Units (GPU), DNNs have achieved outstanding performance in many signal processing and computer vision tasks. However, the working of DNNs is still not completely clear. The optimized DNNs are usually treated as black-box systems. Some natural questions are what are the functions of the linear transform and the non-linearities and what is the role of the “cascade” in DNNs.

Some recent works have tried to provide insights into the workings of DNNs. Bruna and Mallat [3], [4] proposed a Scattering Convolutional Network (SCN) by replacing the learned filters with wavelet-like transforms. SCN provides feature representations which are translation and rotation invariant. Zeiler and Fergus [5] proposed a deconvolution technique to visualize the intermediate feature layers of a Convolutional

Neural Network (CNN) trained for image classification. The filters in the first layer are Gabor like, and the deeper layer feature maps tend to be active only for certain objects. In [6], the authors suggested that an auto-encoder partitions the low-dimensional data manifold into cells and approximates each piece by a hyper-plane. The encoding capability limit of a DNN is described by the upper bound of the number of cells. Montufar *et al.* [7] have shown that the number of cells is exponentially larger than the degrees of freedom. Giryes *et al.* [8] theoretically analyzed the fully connected DNN with i.i.d. random Gaussian weights. They prove that a DNN with random Gaussian weights performs a distance-preserving embedding of the data.

The sparse representation framework [9] can generate sparse features for input signals with respect to a given dictionary and sparse representation over an over-complete dictionary can serve as an effective and robust tool in both classification and regression problems. Depending on the way the signal is modelled, the sparse representation model can be divided into synthesis or analysis model [10].

A synthesis model [10] represents a signal $x = D\gamma$ as a linear combination of a small number of column atoms from a redundant synthesis dictionary $D \in \mathbb{R}^{n \times m}$ with $n < m$ and $\|\gamma\|_0 = k \ll m$. Sparse pursuit is to seek the sparsest representation γ given the input signal x and the dictionary D . Sparse pursuit algorithms include greedy algorithms [11]–[13] and convex relaxation based algorithms [14]–[17]. The greedy algorithms, like Orthogonal Matching Pursuit (OMP) [11], find at each iteration a sparse coefficient with the aim of reducing the approximation error. Convex relaxation algorithms relax the non-convex l_0 norm to a convex l_1 norm. The sparse representation problem can then be solved using basis pursuit (BP) [15] or iterative algorithms, like Iterative Soft-Thresholding Algorithm (ISTA) [16], [17]. The synthesis dictionary learning algorithms [18], [19] mainly take an alternating minimization strategy and iterate between a sparse-coding stage and a dictionary update stage. There also exists one-stage synthesis dictionary learning algorithm e.g. [20]. With well established theories and algorithms, sparse representation over redundant synthesis model has been extensively used in signal and image processing.

The analysis model [10], [21] has also attracted increasing research interests. A redundant analysis dictionary $\Omega \in \mathbb{R}^{m \times n}$ is a tall matrix with $m > n$ where each row of Ω is an atom of the dictionary. The expectation is that the analysis coefficients Ωx are sparse. This means that the analysis dictionary should be able to sparsify the input signal, whilst preserving its essential information. The analysis dictionary usually serves as

The authors are with the Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, U.K. (e-mail: j.huang15@imperial.ac.uk).

a regularization term $\lambda \|\Omega x\|_1$ in the optimization formulation and models the co-sparse prior which can be considered as an extension of the Total Variation (TV) prior. Alternating minimization strategy can also be applied for learning analysis dictionaries [21]–[25]. Analysis K-SVD [21] iterates between an analysis pursuit operation and a K-SVD dictionary update stage. Yaghoobi *et al.* [22] proposed a uniformly-normalized tight frame constraint for learning analysis operators. Analysis Simultaneous Codeword Optimization (ASimCO) algorithm [23] enforces a K -sparse constraint on the sparse-coding stage and updates multiple dictionary atoms simultaneously in the dictionary update stage. Sparsifying transform learning [24], [25] proposed to constrain the analysis operator to be full rank and well-conditioned. The Geometric Analysis Operator Learning (GOAL) algorithm [26], [27] learns the analysis dictionary by employing an alternative optimization strategy. It performs dictionary learning on manifolds by minimizing an objective function which promotes sparse representation and also imposes full rank constraint.

Building a deep model using sparse representation over redundant synthesis dictionaries has facilitated interpretations of DNNs. Rubinstein and Elad [28] proposed an Analysis-Synthesis Thresholding (AST) model for image deblurring which consists of an analysis dictionary, element-wise hard-thresholding operators and a synthesis dictionary. The AST model can be interpreted as a fully connected DNN which uses hard-thresholding as non-linearity and has one hidden layer. The Double-Sparsity model [29] proposes to learn a two-layer synthesis dictionary. The first layer is a dictionary that models a fixed basis, while the second one is a sparse dictionary with sparse atoms. The effective dictionary is therefore the multiplication between the two dictionaries. The Double-Sparsity model provides a more efficient and adaptive modeling and enables learning large dictionary from high-dimensional data (see also [30]). A Multi-Layer Convolutional Sparse Coding (ML-CSC) model [31], [32] is an extension of convolutional sparse coding model [33]–[35] and gives a new interpretation on the working of Convolutional Neural Networks (CNNs). The linear models in CNNs are interpreted as synthesis dictionaries with convolutional structure and the function of the non-linearities is interpreted as a simplified sparse pursuit procedure. The ML-CSC model has multiple layers of synthesis dictionaries where the first dictionary is non-sparse while the following dictionaries are sparse. In [34], [35], the relationship between convolutional analysis operator learning and CNNs is clarified. Tariyal *et al.* [36] proposed a greedy layer-wise deep dictionary learning method which performs synthesis dictionary learning layer-by-layer. A parametric approach is proposed in [37] to learn a deep dictionary for image classification tasks. The proposed dictionary learning method contains a forward pass which performs sparse coding with the given synthesis dictionaries and a backward pass which updates the dictionaries by gradient descent.

The contribution of this paper is two-fold:

- We propose a Deep Analysis dictionary Model (DeepAM) which is composed of multiple layers of analysis dictionaries with associated soft-thresholding operators and a layer of synthesis dictionary. We propose to characterize each analysis dictionary as a combination of two sub-dictionaries: an Information Preserving Analysis Dictionary (IPAD) and a Clustering Analysis Dictionary (CAD). The IPAD together with the soft-thresholding operator preserves the key information of the input data, and the thresholds are set essentially to denoise the data. The CAD with the associated soft-thresholding operator generates a discriminative representation, and the thresholds are set to facilitate such discrimination.
- We propose learning algorithms for DeepAM. To achieve the two different goals of IPAD and CAD, different learning criteria are introduced. The IPAD learning uses an unsupervised setup, while the CAD learning uses a supervised setup. We validate our proposed DeepAM on the single image super-resolution task. Simulation results show that the proposed deep dictionary model is more robust and flexible than a DNN which has the same structure but is optimized using back-propagation. In the self-example image super-resolution task which has limited number of training samples, our proposed DeepAM method achieves significant better performance. Moreover, on noisy image super-resolution, the learned DeepAM can be well adapted to unseen noise levels by rescaling the IPAD and CAD thresholds of the first layer and outperforms DNNs when training and testing noise levels are mismatched.

Parts of this work were presented in [38], [39].

The rest of the paper is organized as follows. Section II gives an overview of the proposed deep analysis dictionary model. Section III analyzes the proposed model and explains the rationale behind splitting each analysis dictionary into an information preserving and a clustering sub-dictionary. Section IV introduces the learning algorithm for the deep analysis dictionary model. Section V presents simulation results on single image super-resolution task and Section VI concludes the paper.

II. OVERVIEW

We begin this section by briefly introducing the single image super-resolution problem and some notations. We then outline the structure of our deep dictionary model.

A. Image Super-Resolution

The task of single image super-resolution (SISR) is to estimate a (high-resolution) HR image $\hat{Y}$ from an observed (low-resolution) LR image X .

We use patch-based single image super-resolution as the sample application to validate our proposed architecture. Instead of estimating the HR image as a whole, the patch-based approaches [40]–[46] divide the LR image into overlapping patches and infer a HR patch for each LR patch. The HR image can then be reconstructed using the estimated HR patches by patch overlapping. Learning-based approaches [40]–[42], [44]–[49] learn the inference model from an external training dataset which contains LR and HR image pairs. The patch-based methods use LR-HR patch pairs $\{(x_i^0, y_i)\}_{i=1}^N$

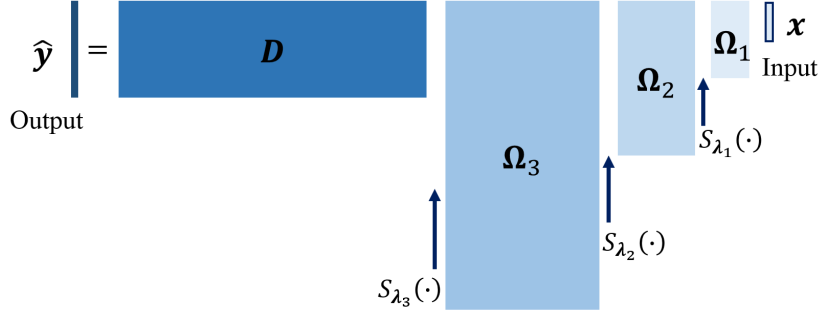


Fig. 1. A 3-layer deep analysis dictionary model. There are 3 layers of analysis dictionaries $\{\Omega_i\}_{i=1}^3$ with element-wise soft-thresholding operators $\{S_{\lambda_i}(\cdot)\}_{i=1}^3$ and a layer of synthesis dictionary D . The output signal $\hat{y}$ is obtained through a cascade of matrix multiplications and soft-thresholding operations with input signal x .

extracted from the training dataset. The size of the LR patches and the HR patches is assumed to be $p \times p$ and $(s \times p) \times (s \times p)$, respectively. The variable s represents the up-scaling factor. To gain illumination invariance property, the mean value of each patch is normally removed. For simplicity, we denote $d_0 = p^2$ and $d_{L+1} = (s \times p)^2$. By vectorizing the image patches and grouping the training vectors into a matrix, we obtain the input LR training data matrix $\mathcal{X}^0 = [\mathbf{x}_1^0, \dots, \mathbf{x}_N^0] \in \mathbb{R}^{d_0 \times N}$ and the corresponding ground-truth HR training data matrix $\mathcal{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_N] \in \mathbb{R}^{d_{L+1} \times N}$.

B. Deep Analysis Dictionary Model

To address the SISR problem, we propose a Deep Analysis dictionary Model (DeepAM) which consists of multiple layers of analysis dictionary interleaved with soft-thresholding operations and a single synthesis dictionary. In an L -layer DeepAM, there are $L + 1$ dictionaries and L layers that correspond to the non-linear operations. The first L dictionaries are analysis dictionaries and are denoted as $\{\Omega_i \in \mathbb{R}^{d_i \times d_{i-1}}\}_{i=1}^L$ with $d_i \geq d_{i-1}$. The row atoms $\{\omega_{i,j}^T\}_{j=1}^{d_i}$ of the analysis dictionary Ω_i are of unit norm. The non-linear operator used in DeepAM is the element-wise soft-threshold¹ $\{S_{\lambda_i}(\cdot)\}_{i=1}^L$ where $\lambda_i \in \mathbb{R}^{d_i}$ denotes the threshold vector at layer i . The dictionary D in the last layer is a synthesis dictionary and is designed to optimize the regression task at hand. Fig. 1 shows an example of a 3-layer deep analysis dictionary model for the image super-resolution task.

The L -layer DeepAM can therefore be expressed mathematically as:

$$\hat{y} = D S_{\lambda_L} (\Omega_L S_{\lambda_{L-1}} (\dots \Omega_2 S_{\lambda_1} (\Omega_1 x^0) \dots)), \quad (1)$$

where x^0 and $\hat{y}$ is the input signal and the estimated output signal, respectively.

Let us denote with $x^i = S_{\lambda_i}(\Omega_i x^{i-1})$ the output of the i -th layer. This means that the input signal x^{i-1} is multiplied with the analysis dictionary Ω_i and then passed through the element-wise soft-thresholding operator $S_{\lambda_i}(\cdot)$. The thresholded output signal x^i will be a sparse signal and is expected to be better at predicting the ground-truth signal y than x^{i-1} . Compared to synthesis sparse coding, analysis dictionary with

soft-thresholding has much lower computational complexity, since the sparse coding step, which is time consuming, can be omitted during both testing and training. After L layers, x^L is transformed to the HR signal domain via the synthesis dictionary. Note that the input LR signal lives in a lower dimensional space when compared to the target HR signal. It is therefore essential for the inference model to be able to non-linearly transform the input data to a higher dimensional space. This is to be achieved by combined use of the analysis dictionaries and the associated soft-thresholding operators.

The proposed DeepAM framework is closely related to Deep Neural Networks (DNNs) with Rectified Linear Unit (ReLU) non-linearity². As ReLU can be considered as the one-sided version of soft-thresholding [31], [50], a layer of analysis dictionary with soft-thresholding can be transferred to a layer of Neural Networks with ReLU operator:

$$\begin{aligned} \Omega_{i+1} S_{\lambda_i} (\Omega_i x^{i-1}) \\ = [\Omega_{i+1} \quad -\Omega_{i+1}] \text{ReLU} \left(\begin{bmatrix} \Omega_i \\ -\Omega_i \end{bmatrix} x - \begin{bmatrix} \lambda_i \\ \lambda_i \end{bmatrix} \right). \end{aligned} \quad (2)$$

From (2), we realize that a layer of analysis dictionary and soft-thresholding operation with n atoms can be represented with a layer of neural networks with ReLU and $2n$ neurons. For data which is symmetrically distributed around the origin, DeepAM with soft-thresholding can be more efficient than DNNs with ReLU. This observation will be validated numerically in Section V-D.

The learning problem for DeepAM can be formulated as learning the parameters that minimize the mean squared error between the ground-truth data and the estimations:

$$\min_{\theta} \|\mathcal{Y} - D S_{\lambda_L} (\dots \Omega_2 S_{\lambda_1} (\Omega_1 \mathcal{X}^0) \dots)\|_F^2, \quad (3)$$

where $\theta = \{D, \{\Omega_i\}_{i=1}^L, \{\lambda_i\}_{i=1}^L\}$ denotes all the parameters of the L -layer DeepAM.

Optimizing (3) directly can be very difficult as it involves non-convex matrix factorization and the learning of the parameters of the non-linear operators. Standard tools for optimizing DNNs can be utilized, for example back-propagation algorithm

¹Soft-thresholding is defined as $S_{\lambda}(a) = \text{sgn}(a) \max(|a| - \lambda, 0)$.

²ReLU is defined as $\text{ReLU}(a) = \max(a, 0)$.

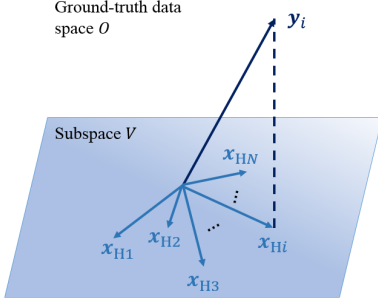


Fig. 2. In image super-resolution, the input data spans a low-dimensional subspace V within the HR data space O . The objective is to estimate the unknown HR signal y_i based on the input LR signal x_{Hi} . The dashed line represents the residual signal $r_i = y_i - x_{Hi}$ which is orthogonal to the subspace spanned by the input data.

[2]. However this would lead to effective but difficult to interpret results.

Our goal instead is to build a deep dictionary model with a higher interpretability through imposing specific objectives to different components in the model. The analysis dictionary and threshold pairs play a key role in DeepAM as they determine the way effective features are generated. The synthesis dictionary instead can be learned using least squares once all the analysis dictionaries and thresholds have been determined. We propose a layer-wise learning strategy to learn the pair of analysis dictionary and soft-thresholding operators. In this way, we can obtain a system where the purpose of every component is easier to interpret.

III. ANALYZING THE DEEP ANALYSIS DICTIONARY MODEL

To justify our approach, we begin by considering a 1-layer DeepAM system:

$$\hat{y}_i = D\mathcal{S}_{\lambda_1}(\Omega_1 x_i^0), \quad (4)$$

where $x_i^0 \in \mathbb{R}^{d_0}$ is one of the elements of $\mathcal{X}$, $\hat{y}_i \in \mathbb{R}^{d_2}$ with $d_2 > d_0$, $D \in \mathbb{R}^{d_2 \times d_1}$, and $\Omega_1 \in \mathbb{R}^{d_1 \times d_0}$.

A 1-layer DeepAM is similar to the analysis-thresholding synthesis model [28] and the single hidden-layer CNN model in iterative neural networks [51]–[53]. We assume, for the sake of argument, that the degradation model is linear. That is, there exists a degradation matrix $H \in \mathbb{R}^{d_0 \times d_2}$ such that $x_i^0 = Hy_i$. Denote $x_{Hi} = H^\dagger x_i^0 \in \mathbb{R}^{d_2}$ as the projection of the LR signal x_i^0 onto the HR signal space with the pseudo-inverse matrix $H^\dagger$.

As shown in Fig. 2, the signal x_{Hi} lies in a low-dimensional subspace $V \subset O$ of the ground-truth HR data space O . A linear operation will not be able to recover the components that are orthogonal to V (i.e. the dashed line in Fig. 2). It is therefore imperative to design the analysis dictionary Ω_1 and the non-linear soft-thresholding operation $\mathcal{S}_{\lambda_1}(\cdot)$ in a way that facilitates the recovery of the information of y in $V^\perp$.

When we multiply x^0 with Ω_1 , the analysis dictionary atoms $\{\omega_{1,j}^T\}_{j=1}^{d_1}$ project the input LR data onto specific directions. After soft-thresholding, the resulting signal $x^1 =$

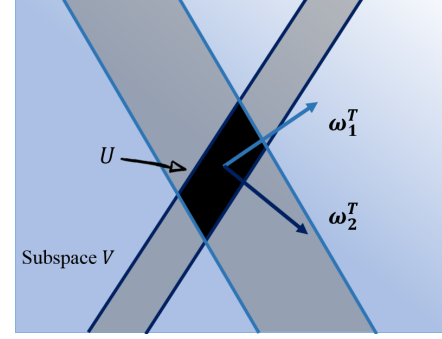


Fig. 3. The analysis and soft-thresholding operations partition the input data subspace V . There are two pairs of analysis atom and soft-thresholding operator. After soft-thresholding, the data in the gray region is with 1 zero coefficient and the data in the convex polyhedron U (i.e. the black region) is with all zero coefficients.

$\mathcal{S}_{\lambda_1}(\Omega_1 x^0)$ is sparse and the end result is a partitioning of V as shown in Fig. 3. The input data within each piece is then linearly transformed for prediction, and all the linear transforms are jointly described by the synthesis dictionary D .

We note that if we assume that all thresholds are large, there is a convex polyhedron U in which all input data will be set to zero by the analysis and the thresholding operations, that is, $\mathcal{S}_{\lambda_1}(\Omega_1 x^0) = 0, \forall x^0 \in U$ (see the central black region in Fig. 3). Therefore, the corresponding estimation $\hat{y}$ will be zero, and the information of the data within the convex polyhedron U will then be completely lost. This may lead to a large mean squared error for prediction.

This suggests that not all thresholds should be too large. The problem can be solved if there is a set of analysis dictionary atoms with small thresholds. If we assume that the signal subspace V has dimension K , then in order not to lose essential information there should be at least K pairs of analysis dictionary atoms and soft-thresholds for *information preservation*. These K atoms are associated with K small soft-thresholds and are able to fully represent the input data space. Therefore the K pairs of analysis atoms and soft-thresholds together with the corresponding synthesis atoms provide a baseline estimation for the input data. The remaining analysis dictionary atoms can instead be associated with large thresholds. The outcome of these analysis and thresholding operations is a *clustering* of the input data. That is, the data with the same sparsity pattern can be grouped into a cluster and shares the same set of atoms in the synthesis dictionary for prediction. The corresponding synthesis atoms then help recover the signal components within the orthogonal subspace $V^\perp$.

Based on the above discussion, we propose to divide an analysis dictionary Ω into two sub-dictionaries $\Omega = [\Omega_I; \Omega_C]$, and similarly divide each threshold vector into two parts $\lambda = [\lambda_I; \lambda_C]$. The Information Preserving Analysis Dictionary (IPAD) Ω_I with its thresholds λ_I aims at passing key information of the input signal to the next layer. The Clustering Analysis Dictionary (CAD) Ω_C with its threshold λ_C is to facilitate the separation of key feature in the signal. The CAD and thresholding operators provide a highly non-linear

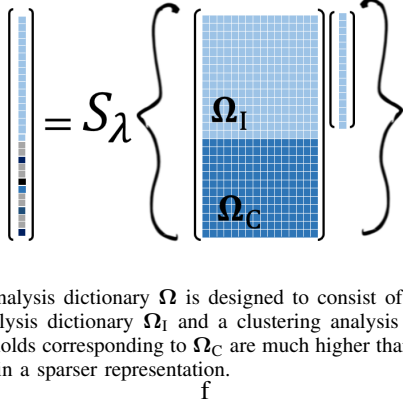


Fig. 4. The analysis dictionary Ω is designed to consist of an information preserving analysis dictionary Ω_I and a clustering analysis dictionary Ω_C . The soft-thresholds corresponding to Ω_C are much higher than those used for Ω_I and result in a sparser representation.

prediction. Fig. 4 shows an analysis dictionary and the soft-thresholding operation with the partition of the IPAD part and the CAD part.

In a multi-layer DeepAM, we adopt the same information preserving and clustering strategy. As depicted in Fig. 5, the analysis dictionary at each layer is composed of an IPAD part and a CAD part. The IPADs and thresholds $\{(\Omega_{Ii}, \lambda_{Ii})\}_{i=1}^L$ create a channel which transmits the information of the input signal to the CAD in each intermediate layer and to the final estimation. The feature representation x_1^L generated by the last layer of IPAD and its thresholds should be able to well represent signal components of the HR signal which are within the input data subspace. The CADs and thresholds $\{(\Omega_{Ci}, \lambda_{Ci})\}_{i=1}^L$ are the main source of non-linearity in DeepAM. The feature representation x_C^L generated by the last layer of CAD should be able to well represent the signal components of y which are orthogonal to the input data subspace. Therefore, the function of CAD and its thresholds can be interpreted as identifying the data with large energy in the subspace orthogonal to the input data subspace. A deep layer of CAD takes the feature representation generated by the IPAD and CAD of the previous layer as input and can generate a non-linear feature representation that cannot be attained by a single layer soft-thresholding with the same number of atoms. Therefore a DeepAM with deeper layers is expected to outperform a shallower one.

IV. LEARNING A DEEP ANALYSIS DICTIONARY MODEL

In this section, we introduce the proposed learning algorithm for DeepAM. In view of the distinctive goals of the two pairs of sub-dictionary and thresholds, different learning criteria have been proposed for learning the IPAD and its thresholds and the CAD and its thresholds.

A. Basic Analysis Dictionary Learning Algorithm

The IPAD and the CAD have different functions but also share some common learning objectives. There are three basic objectives for learning an analysis dictionary: (1) its atoms span a subspace of the input data space; (2) it is able to sparsify the input data; (3) the row atoms are of unit norm.

Our proposed learning algorithm is an extension of the Geometric Analysis Operator Learning (GOAL) algorithm

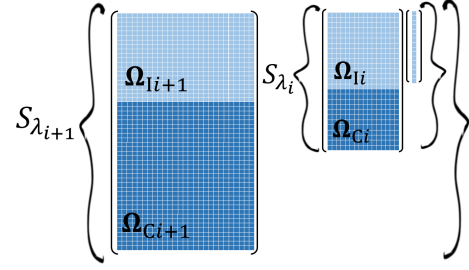


Fig. 5. Two consecutive layers in DeepAM. The IPAD and threshold pairs create an information flow channel from input to output, and the CAD and threshold pairs combine information from the previous layer and generate a feature representation that can well represent the residual part.

[26], [27] and we denote it as GOAL+. The four learning objectives can be attained by using corresponding constraints. The IPAD and the CAD are learned using modified versions of GOAL+ algorithm.

For simplicity, let us denote the analysis dictionary to be learned as $\Omega \in \mathbb{R}^{m \times n}$, the j -th atom of Ω as ω_j^T and the training data as $\mathcal{X} = [x_1, \dots, x_N] \in \mathbb{R}^{n \times N}$. We assume that the data $\mathcal{X}$ span a K dimensional subspace $V \in \mathbb{R}^n$. Let us denote with $W \in \mathbb{R}^{n \times K}$ an orthogonal basis of V , and with $U \in \mathbb{R}^{n \times (n-K)}$ the orthogonal basis of the orthogonal complement $V^\perp$. W is obtained by concatenating the leading K left singular vectors of $\mathcal{X}$, and U is given by the remaining $n - K$ left singular vectors.

The first learning objective is that the learned analysis dictionary Ω should span only the subspace V . There are two main reasons for this requirement. First, the analysis dictionary Ω which spans V can fully preserve the information of the input data. Second, if an atom ω^T belongs to $V^\perp$, it is an unnecessary atom. This is because the coefficients $\omega^T \mathcal{X}$ will be zero since $V^\perp$ is in the null-space of $\mathcal{X}$. We apply a logarithm determinant (log-det) term $h(\Omega)$ to impose the information preservation constraint:

$$h(\Omega) = -\frac{1}{K \log K} \log \det \left(\frac{1}{m} W^T \Omega^T \Omega W \right). \quad (5)$$

Together with the unit norm constraint, the feasible set of the analysis dictionary Ω is therefore defined as $\Theta = \mathbb{S}_{n-1}^m \cap U^\perp$ with $\mathbb{S}_{n-1}$ being the unit sphere in $\mathbb{R}^n$ and $\mathbb{S}_{n-1}^m$ being the product of m unit spheres $\mathbb{S}_{n-1}$. In other words the feasible set Θ restricts the learned atoms in Ω to be of unit norm and excludes the contributions from U . (5) is a generalization of the log-det constraint term applied in GOAL [26], [27]. This is because in our case, W defines a basis of the input data space whose size K could be much smaller than the dimension of the input signal n in particular when considering dictionaries in deeper layers. In GOAL [26], [27], W defines a much larger subspace and is with $K = n$ or $K = n - 1$.

We achieve the constraint $\Omega^T \in \Theta$ by performing orthogonal projection onto the tangent space $T_\Omega(\Theta)$ of the manifold Θ at location Ω . For a row atom ω^T , the operation of the orthogonal projection onto the tangent space $T_\omega(\Theta)$ can be represented by the projection matrix P_ω [27]:

$$P_\omega = I_n - Q_\omega^\dagger Q_\omega, \quad (6)$$

Algorithm 1: GOAL+ Algorithm

```

1 Input: Training data , row number of the dictionary;
2 Initialize: Initialized  $\Omega^{(0)}$ ,  $t = 0$ ;
3 while halting criterion false do
4    $t \leftarrow t + 1$  ;
5   Compute gradient of the objective function
      $\nabla f(\Omega^{(t)})$ ;
6   Orthogonal project  $\nabla f(\Omega^{(t)})$  onto the tangent
     space of manifold  $\Theta$  at  $\Omega^{(t)}$ :
      $\mathcal{G} \doteq \Pi_{T_{\Omega^{(t)}}(\Theta)}(\nabla f(\Omega^{(t)}))$ ;
7   Find new search direction
      $\mathcal{H}^{(t)} = -\mathcal{G} + \beta^{(t)}\mathcal{T}_{\mathcal{H}^{(t-1)}}$ ;
8   Update  $\Omega^{(t+1)}$  along the search direction  $\mathcal{H}^{(t)}$ 
     using backtracking line search.
9 end
10 Output: Learned analysis dictionary  $\Omega$ .

```

where $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ is the identity matrix, and $\mathbf{Q}_\omega = [2\omega, \mathbf{U}]^T \in \mathbb{R}^{(n-k+1) \times n}$.

Sparsifying ability is essential for both IPAD and CAD. The sparsifying constraint is imposed by using a log-square function which is a good approximation of the l_0 norm:

$$g(\Omega) = \frac{1}{Nm \log(1 + \nu)} \sum_{i=1}^N \sum_{j=1}^m \log(1 + \nu(\omega_j^T \mathbf{x}_i)^2), \quad (7)$$

where ν is a tunable parameter which controls the sparsifying ability of the learned dictionary.

The combination of the information preservation constraint in (5) and sparsifying constraint in (7) leads to the objective function of GOAL+:

$$\Omega = \arg \min_{\Omega^T \in \Theta} f(\Omega), \quad (8)$$

where $f(\Omega) = g(\Omega) + \kappa h(\Omega)$ with κ being the regularization parameter.

The objective function defined in (8) is optimized using a geometric conjugate gradient descent method [26], [54] which has a good convergence property and with low computational complexity. The analysis dictionary learning algorithm GOAL+ is summarized in **Algorithm 1**. At iteration t , the gradient of the objective function $\nabla f(\Omega^{(t)})$ is computed and orthogonal projected on the tangent space of the manifold Θ at location $\Omega^{(t)}$. The orthogonal projection of $\nabla f(\Omega)$ onto the tangent space $T_\Omega(\Theta)$ can be expressed as $\Pi_{T_\Omega(\Theta)}(\nabla f(\Omega)) = [\mathbf{P}_{\omega_1} \nabla f(\omega_1), \dots, \mathbf{P}_{\omega_{d_{li}}} \nabla f(\omega_{d_{li}})]$. Let us denote $\mathcal{G} \doteq \Pi_{T_{\Omega^{(t)}}(\Theta)}(\nabla f(\Omega^{(t)}))$, the search direction can be set as $\mathcal{H}^{(t)} = -\mathcal{G}$. In practice, the search direction is a combination of $\mathcal{G}$ and the previous search direction $\mathcal{T}_{\mathcal{H}^{(t-1)}}$. The updated analysis dictionary $\Omega^{(t+1)}$ is then obtained by gradient descent with backtracking line search along the search direction $\mathcal{H}^{(t)}$. The halting condition is when the analysis dictionary converges or when a pre-defined maximum number of iterations is reached. In summary, our optimization approach is similar to that in GOAL [26] with the exception of the orthogonal projection step as described in (6) which represents the constraint introduced by the feasible set Θ . For a more

detailed treatment of the geometric conjugate gradient descent we refer to [26], [54]. Now that the overall objectives of GOAL+ have been introduced, we can focus on how to tailor the optimization in (8) to achieve the objectives of IPAD and CAD respectively.

B. Learning IPAD and Threshold Pairs

The function of the IPAD and threshold pair $(\Omega_{li}, \lambda_{li})$ is to pass key information of the input data $\mathcal{X}^0$ to deeper layers. The learned IPADs create a channel that enables the information flow from the input signal to the estimated output signal. The IPAD learning uses the unsupervised learning setup.

1) *IPAD Learning*: The training data for learning Ω_{li} is the i -th layer input training data $\mathcal{X}^{i-1}$ (the $(i-1)$ -th layer training data is obtained as $\mathcal{X}^i = \mathcal{S}_{\lambda_i}(\Omega_i \mathcal{X}^{i-1})$ for $i \geq 1$). Let us denote the rank of the input training data $\mathcal{X}^0$ at the first layer as $k_0 = \text{rk}(\mathcal{X}^0)$ where $\text{rk}(\cdot)$ outputs the rank of a matrix. The IPAD $\Omega_{li} \in \mathbb{R}^{d_{li} \times d_{i-1}}$ is assumed to have $d_{li} \geq \text{rk}(\mathcal{X}^0)$ atoms to ensure that the learned IPAD can well represent the input data subspace.

By setting the training data as $\mathcal{X}^{i-1}$, the i -th layer analysis dictionary Ω_{li} can be learned using GOAL+. The orthonormal basis $\mathbf{W} \in \mathbb{R}^{d_{i-1} \times k_0}$ is set to be an arbitrary basis of $\mathcal{X}^{i-1}$ that corresponds to the signal subspace of $\mathcal{X}^0$. The orthogonal basis $\mathbf{U} \in \mathbb{R}^{d_{i-1} \times (d_{i-1} - k_0)}$ is set to span the orthogonal complement of the subspace spanned by $\mathbf{W}$.

2) *Learning the Thresholds for IPAD*: Given a learned IPAD Ω_{li} , the analysis coefficients $\alpha^i = \Omega_{li} \mathbf{x}^{i-1}$ contain sufficient information of $\mathbf{x}^{i-1}$. When α^i is a redundant representation or when the input data $\mathbf{x}^{i-1}$ is noisy, applying a proper thresholding operation to α^i can further enhance the robustness of the representation. We propose to apply soft-thresholding with small thresholds to the IPAD analysis coefficients as $\mathbf{z} = \mathcal{S}_{\lambda_{li}}(\Omega_{li} \mathbf{x}^{i-1})$ and interpret the soft-thresholding operation as a form of denoising.

There are related works in the literature about thresholding for redundant representations [55]–[57]. Elad [55] shows that simple shrinkage is effective for redundant representation and interprets the simple shrinkage as the first iteration for solving Basis Pursuit Denoising (BPDN) problems. Raphan and Simoncelli [56] proposed a denoising scheme for redundant representations based on Stein's Unbiased Risk Estimator. Lin and Lee [57] proposed a Bayesian framework for finding the optimal l_1 -norm regularization.

Let us consider a weighed l_1 -norm regularized minimization problem:

$$\min_{\mathbf{z}} \frac{1}{2} \|\mathbf{x} - \Omega^T \mathbf{z}\|_2^2 + \sum_{j=1}^m \lambda_{lj} |z_j|, \quad (9)$$

where z_j is the j -th coefficient of the sparse vector $\mathbf{z}$, and λ_{lj} is the corresponding regularization parameter.

Selecting the soft-threshold λ_{li} is equivalent to finding suitable regularization parameters in (9) as the soft-thresholding operation $\mathcal{S}_\lambda(\Omega \mathbf{x})$ can be interpreted as the first iteration of the Iterative Soft-Thresholding Algorithm (ISTA) [16] for solving (9):

$$\mathbf{z}^{(1)} = \mathcal{S}_\lambda \left(\mathbf{z}^{(0)} + \Omega \left(\mathbf{x} - \Omega^T \mathbf{z}^{(0)} \right) \right), \quad (10)$$

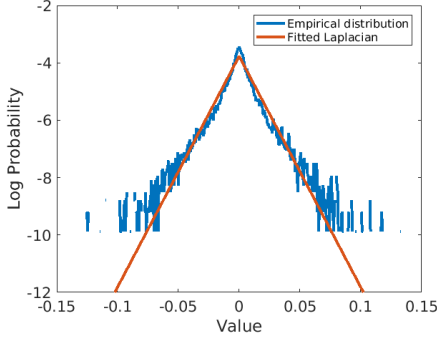


Fig. 6. An example of the empirical distribution of the response of an atom and the fitted Laplacian distribution.

where the initial sparse code $z^{(0)} = \mathbf{0}$.

Lin and Lee [57] proposed a method to choose the optimal regularization parameters based on a Bayesian analysis. They assume that the data $\mathbf{x}$ is with additive i.i.d. zero mean Gaussian noise with variance σ^2 :

$$P(\mathbf{x}|\mathbf{\Omega}^T, \mathbf{z}, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x} - \mathbf{\Omega}^T \mathbf{z}\|_2^2\right) \quad (11)$$

and assume a Laplacian distribution prior for the sparse code $\mathbf{z}$ with parameters $\boldsymbol{\lambda}$:

$$P(\mathbf{z}|\boldsymbol{\lambda}) = \prod_{j=1}^m \frac{\lambda_j}{2} \exp(-\lambda_j |z_j|). \quad (12)$$

Empirically, we have found that the prior distribution $P(\mathbf{z})$ can be well characterized by an i.i.d. zero-mean Laplacian distribution. Fig. 6 shows an example of the empirical distribution of the inner product between an atom and the input data and the fitted Laplacian distribution. Based on the analysis in [57], the optimal regularization parameters for (9) can be set as proportional to the variance of the noise and inversely proportional to the variance of the Laplacian distribution:

$$\boldsymbol{\lambda} \propto \left[\frac{1}{\sigma_1}, \dots, \frac{1}{\sigma_m} \right]^T, \quad (13)$$

where σ_j is the variance of the Laplacian distribution for the j -th sparse code z_j .

From (13), the soft-threshold associated with IPAD $\boldsymbol{\Omega}_{li}$ is:

$$\boldsymbol{\lambda}_{li} = \rho_l \left[\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \dots, \frac{1}{\sigma_{d_{li}}} \right]^T, \quad (14)$$

where ρ_l is a scaling parameter, and the variance σ_j of the j -th coefficient can be estimated using the obtained IPAD $\boldsymbol{\Omega}_{li}$ and its input data³.

There is only a free parameter ρ to be determined. It can be obtained by solving a 1-dimensional search problem. The optimization problem for ρ is therefore formulated as:

$$\hat{\rho}_l = \arg \min_{\rho \in \mathcal{D}} \|\mathbf{y} - \mathbf{G} \mathcal{S}_{\rho \boldsymbol{\lambda}}(\boldsymbol{\Omega}_{li} \mathbf{x}^{i-1})\|_F^2, \quad (15)$$

³We estimate the variance σ_j of the Laplacian distribution using the mean absolute value of the response of the atom: $\sigma_j = \frac{1}{N} \sum_{k=1}^N |\boldsymbol{\omega}_{li,j}^T \mathbf{x}_k^{i-1}|$.

where $\boldsymbol{\lambda} = [1/\sigma_1, 1/\sigma_2, \dots, 1/\sigma_{d_{li}}]^T$, $\mathbf{G} = \mathbf{y} \mathbf{z}^T (\mathbf{z} \mathbf{z}^T)^{-1}$ with $\mathbf{Z} = \mathcal{S}_{\rho \boldsymbol{\lambda}}(\boldsymbol{\Omega}_{li} \mathbf{x}^{i-1})$, and $\mathcal{D}$ is a discrete set of values.

The obtained pair $(\boldsymbol{\Omega}_{li}, \boldsymbol{\lambda}_{li})$ should be able to preserve the important information within the input signal and give no worse performance when compared to a linear model without any non-linearity.

C. Learning CAD and Threshold Pairs

The function of the clustering analysis dictionary and threshold pair $(\boldsymbol{\Omega}_{Ci}, \boldsymbol{\lambda}_{Ci})$ is to sparsify its input data and identify the data with large residual energy. The CAD and threshold pairs at shallow layers provide low-level feature representations for the CADs at deeper layers.

1) *CAD Learning*: The CAD learning uses the supervised learning setup. Different from IPAD, learning CAD $\boldsymbol{\Omega}_{Ci}$ requires supervision from both the input training data $\mathbf{x}^{i-1}$ and the ground-truth HR training data $\mathbf{y}$.

Let us denote with $\mathbf{y}^i = \mathbf{D}_i \mathbf{x}^{i-1}$ the middle resolution data and with $\boldsymbol{\varepsilon}^i = \mathbf{y} - \mathbf{y}^i$ the residual data where $\mathbf{D}_i \in \mathbb{R}^{d_{L+1} \times d_i}$ is the synthesis dictionary of layer i which minimizes the mean squared reconstruction error and can be obtained by solving:

$$\mathbf{D}_i = \arg \min_{\mathbf{D}} \{\|\mathbf{D} \mathbf{x}^{i-1} - \mathbf{y}\|_F^2 + \|\mathbf{D}\|_2^2\}. \quad (16)$$

It has a closed-form solution given by:

$$\mathbf{D}_i = \mathbf{y} \mathbf{x}^{i-1 T} \left(\mathbf{x}^{i-1} \mathbf{x}^{i-1 T} + \mathbf{I} \right)^{-1}. \quad (17)$$

The learning objective for CAD $\boldsymbol{\Omega}_{Ci}$ is that its atoms should be able to project $\mathbf{x}^{i-1}$ onto directions where the data with large residual error has responses with large magnitude. To achieve that, we propose to first learn an analysis dictionary $\boldsymbol{\Psi}_i \in \mathbb{R}^{d_{Ci} \times d_{L+1}}$ which acts on the data $\mathbf{y}^i$ and is able to jointly sparsify the data $\mathbf{y}^i$ and the residual data $\boldsymbol{\varepsilon}^i$. That is, the atoms in $\boldsymbol{\Psi}_i$ are able to identify the data in $\mathbf{y}^i$ with large residual energy. The i -th layer CAD is then re-parameterized as:

$$\boldsymbol{\Omega}_{Ci} = \boldsymbol{\Psi}_i \mathbf{D}_i. \quad (18)$$

As a result, the learned CAD $\boldsymbol{\Omega}_{Ci}$ will have the same identification ability as $\boldsymbol{\Psi}_i$ since $\boldsymbol{\Omega}_{Ci} \mathbf{x}_j^{i-1} = \boldsymbol{\Psi}_i \mathbf{y}_j^i$.

We propose the following constraint for learning the analysis dictionary $\boldsymbol{\Psi}_i$. Each analysis atom $\boldsymbol{\psi}_l^T$ is enforced to be able to jointly sparsify $\mathbf{y}^i$ and $\boldsymbol{\varepsilon}^i$:

$$p(\boldsymbol{\Psi}) = c \sum_{j=1}^N \sum_{l=1}^{d_{Ci}} \log \left(1 + \nu \left((\boldsymbol{\psi}_l^T \mathbf{y}_j^i)^2 - (\boldsymbol{\psi}_l^T \mathbf{e}_j^i)^2 \right)^2 \right), \quad (19)$$

where $c = 1/N d_{C1} \log(1 + \nu)$, ν is a tunable parameter, and $\mathbf{y}_j^i$ and $\mathbf{e}_j^i$ refers to the j -th column of $\mathbf{y}^i$ and $\boldsymbol{\varepsilon}^i$, respectively.

The objective function for learning the analysis dictionary $\boldsymbol{\Psi}_i$ is then formulated as:

$$\boldsymbol{\Psi}_i = \arg \min_{\boldsymbol{\Psi}^T \in \Theta} f(\boldsymbol{\Psi}), \quad (20)$$

where $f(\boldsymbol{\Psi}) = g(\boldsymbol{\Psi}) + \kappa h(\boldsymbol{\Psi}) + \mu p(\boldsymbol{\Psi})$ with κ , and μ being the regularization parameters. Here, $g(\cdot)$ and $h(\cdot)$ are those defined in (7) and (5), respectively.

The input training data is set to $(\mathcal{X}^{i-1}, \mathcal{Y})$. Let us denote the rank of $\mathcal{Y}$ as $k_L = \text{rk}(\mathcal{Y})$. The orthogonal basis $\mathbf{W} \in \mathbb{R}^{d_{L+1} \times k_L}$ is set to be an arbitrary basis of the signal subspace of $\mathcal{Y}$. The orthonormal basis $\mathbf{U} \in \mathbb{R}^{d_{L+1} \times (d_{L+1} - k_L)}$ is set to be a basis spanning the orthogonal complement to the subspace spanned by $\mathbf{W}$.

The new objective function in (20) is optimized using GOAL+ algorithm. With the learned analysis dictionary Ψ_i , the i -th layer CAD Ω_{Ci} is obtained as in (18).

2) *Learning the Thresholds for CAD*: The thresholds for CAD are crucial to the performance of DeepAM as the CAD and threshold pair is the main source of non-linearity in DeepAM. The atoms of the learned CAD project the input data onto directions where the data with large residual prediction error will have responses with large magnitude. After soft-thresholding, the coefficients should be as sparse as possible to achieve a strong discriminative power.

We propose to set the CAD thresholds as follows:

$$\lambda_{Ci} = \rho_C [\sigma_1, \sigma_2, \dots, \sigma_{d_{Ci}}], \quad (21)$$

where ρ_C is a scaling parameter, and σ_j is the variance of the Laplacian distribution for the j -th atom.

As discussed in the previous section, the analysis sparse codes can be well modelled by Laplacian distributions. By setting the CAD thresholds proportional to the variance of the analysis coefficients, the proportion of data that has been set to zero for each pair of atom and threshold will be the same. When the synthesis dictionary is applied for reconstruction, the synthesis atoms corresponding to the CAD atoms will be activated with a similar frequency.

With this simplification, the CAD thresholds can be learned in an efficient way. The scaling parameter ρ_C can be obtained using the same strategy as in (15) by solving a 1-dimensional search problem. The optimization problem for ρ_C is formulated as:

$$\rho_C = \arg \min_{\rho \in \mathcal{D}} \|\mathcal{Y}_R - \mathbf{G} \mathcal{S}_{\rho\lambda}(\Omega_{Ci} \mathcal{X}^{i-1})\|_F^2, \quad (22)$$

where $\mathcal{Y}_R$ is the estimation residual obtained after using IPAD, $\lambda = [\sigma_1, \sigma_2, \dots, \sigma_{d_{Ci}}]^T$, $\mathbf{G} = \mathcal{Y}_R \mathbf{Z}^T (\mathbf{Z} \mathbf{Z}^T)^{-1}$ with $\mathbf{Z} = \mathcal{S}_{\rho\lambda}(\Omega_{Ci} \mathcal{X}^{i-1})$, and $\mathcal{D}$ is a discrete set of values.

In the simulation results in Section V-D and V-E, we will show that the learned CAD thresholds lead to an effective system.

D. Synthesis Dictionary Learning

The deep analysis dictionary learning can be considered as a layer-wise representation learning process in which the input data $\mathcal{X}^0$ is consistently non-linearly transformed to a high dimensional feature representation $\mathcal{X}^L$ with good descriptive and discriminative properties. The synthesis dictionary $\mathbf{D}$ models the linear transformation from $\mathcal{X}^L$ to the desired HR counterpart $\mathcal{Y}$. Similar to (17), the synthesis dictionary is learned using least squares:

$$\mathbf{D} = \mathcal{Y} \mathcal{X}^{LT} (\mathcal{X}^L \mathcal{X}^{LT} + \mathbf{I})^{-1}. \quad (23)$$

Algorithm 2: DeepAM Learning Algorithm

```

1 Input: Training data pair  $(\mathcal{X}^0, \mathcal{Y})$ , the number of
   layers  $L$ , the structure of DeepAM, the parameters
   for learning IPAD  $(\nu, \kappa)$ , and the parameters for
   learning CAD  $(\nu, \kappa, \mu)$ ;
2 for  $i \leftarrow 1$  to  $L$  do
3   Learning  $\Omega_{Li}$  using GOAL+ with training data
    $\mathcal{X}^{i-1}$  and objective function (8);
4   Learning  $\Omega_{Ci}$  using GOAL+ with training data
    $(\mathcal{X}^{i-1}, \mathcal{Y})$  and objective function (20);
5   Learning thresholds  $\lambda_{Li}$  and  $\lambda_{Ci}$ ;
6    $\Omega_i \leftarrow [\Omega_{Li}; \Omega_{Ci}]$ ,  $\lambda_i \leftarrow [\lambda_{Li}; \lambda_{Ci}]$ ;
7   Update input training data as  $\mathcal{X}^i = \mathcal{S}_{\lambda_i}(\Omega_i \mathcal{X}^{i-1})$ ;
8 end
9 Learning the synthesis dictionary  $\mathbf{D}$  as in (23).
10 Output: Learned DeepAM  $\{\{\Omega_i, \lambda_i\}_{i=1}^L, \mathbf{D}\}$ .
```

E. DeepAM Learning Algorithm

The overall learning algorithm for DeepAM is summarized in **Algorithm 2**. We adopt a layer-wise learning strategy for DeepAM. At each layer, two sub-dictionaries IPAD and CAD are independently learned and then combined to form the analysis dictionary, and the thresholds for IPAD and CAD are obtained with two different strategies. In this way, each pair of analysis dictionary and soft-thresholding operations fulfil two different functionalities. Finally, the synthesis dictionary is learned using least squares.

V. SIMULATION RESULTS

In this section, we report the implementation details and numerical results of our proposed DeepAM method and compare our proposed method with Deep Neural Networks learned using back-propagation.

A. Implementation Details

We use the standard 91 training images [40] as training dataset and use *Set5* [40] and *Set14* [41] as the testing datasets. The Peak Signal-to-Noise Ratio (PSNR)⁴ is used as the evaluation metric. The color images have been converted from RGB color space to YCbCr color space and image super-resolution is only applied to the luminance component. The low-resolution (LR) images are obtained by down-sampling the ground-truth high-resolution (HR) images by a factor $s = 2$ using Matlab function `imresize`. The size of the low-resolution patches is set to $p = 6$ for the purpose of better visualization and easier interpretation. The size of the high-resolution patches is then 12×12 .

During testing, full patch overlapping is applied to reconstruct the HR images. Instead of predicting a 12×12 HR patch for each input LR 6×6 patch, the output of DeepAM is

⁴PSNR=10 log($\frac{255^2}{\sqrt{MSE}}$), where MSE is the mean squared error between the ground-truth HR image and the estimated HR image

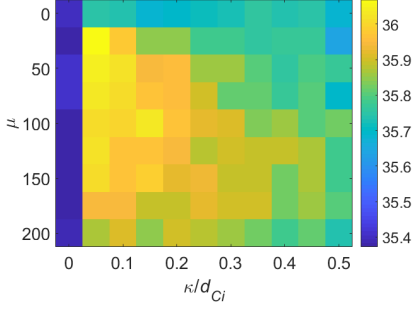


Fig. 7. The performance of 1-layer DeepAM with different combinations of parameters evaluated on *Set5*. All the models are with 1 layer and have 256 atoms.

Parameters	ν	κ	μ
IPAD	$100 \times d_{i-1}$	d_{li}	—
CAD	$100 \times d_{i-1}$	$0.1 \times d_{Ci}$	100

TABLE I

PARAMETERS SETTING OF GOAL+ ALGORITHM FOR LEARNING THE i -TH LAYER IPAD $\Omega_{li} \in \mathbb{R}^{d_{li} \times d_{i-1}}$ AND CAD $\Omega_{Ci} \in \mathbb{R}^{d_{Ci} \times d_{i-1}}$.

the central 8×8 HR patch since the input LR patch does not contain sufficient information to predict the boundary pixels⁵.

B. Parameter Setting

In this section, we discuss how several parameters are determined. For IPAD learning, there are two parameters that need to be set: ν and κ , and for CAD learning, there are three parameters ν , κ , and μ . The sparsifying parameter ν for learning the i -th layer IPAD and CAD is set to $100 \times d_{i-1}$ which can lead to learned atoms with good sparsifying ability.

As introduced in Section IV, the level of the thresholds for soft-thresholding is determined through a grid search. The discrete set $\mathcal{D}$ defines the searching grid. In this paper, the grid $\mathcal{D}$ is set as $[\dots, 1 \times 10^{-2}, 2 \times 10^{-2}, 3 \times 10^{-2}, \dots, 1 \times 10^{-1}, 2 \times 10^{-1}, 3 \times 10^{-1}, \dots]$. This setting empirically works well.

Since the IPAD and its thresholds are responsible for passing essential information from the input, the learned IPAD should span the signal subspace of its input training data. Therefore, a strong rank constraint should be imposed. The rank constraint parameter for IPAD learning is set to d_{li} .

The CAD and its thresholds are responsible for identifying the missing high-frequency information of the input signal. Different parameter combinations (κ, μ) have been evaluated to find the best parameter setting for learning CAD. Around 20,000 training LR and HR patch pairs have been extracted from the training dataset. For each combination of κ and μ , a 1-layer DeepAM which has 256 atoms in the analysis dictionary is learned using the extracted training data and evaluated on *Set5* [40]. The number of conjugate gradient descent iterations used for learning IPAD and CAD is set to 500. The halting condition is when the maximum number

⁵This is also applied to deep neural network with soft-thresholding operator learned using back-propagation, but not applied to deep neural network with ReLU operator learned using back-propagation due to a degraded performance.

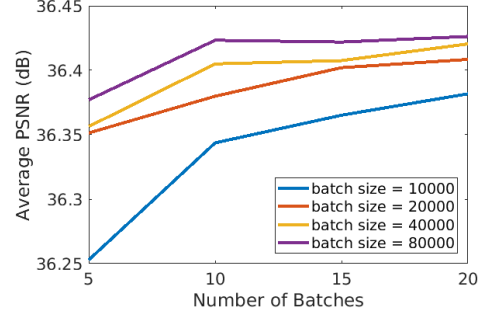


Fig. 8. The performance of learned DeepAM using different batch sizes and number of batches for training evaluated on *Set5*. All the models are with 3 layer and each layer is with 256 atoms.

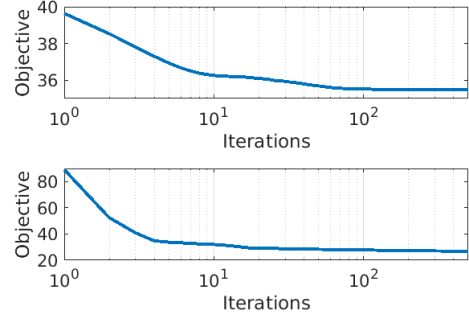


Fig. 9. The objective minimization in IPAD learning of 35 atoms (top) and in CAD learning of 221 atoms (bottom) with the default parameter setting of GOAL+ algorithm.

of iterations is reached. The evaluation results for different parameter combinations are shown in Fig. 7.

From Fig. 7, we can see that both the rank constraint $h(\cdot)$ and the joint sparsifying constraint $p(\cdot)$ are essential to learn a DeepAM with good performance. When the rank constraint is not used (i.e. the rank constraint parameter $\kappa = 0$), the performance of the learned DeepAM is around 35.4 dB and is around 0.6 dB worse than the result of the best parameter combination. When the joint sparsifying constraint is not used in the optimization (i.e. the joint sparsifying constraint parameter $\mu = 0$), the performance of the learned DeepAM is around 0.3 dB worse than the result of the best parameter combination. The rank constraint $h(\cdot)$ and the joint sparsifying constraint $p(\cdot)$ are therefore both important for learning a good CAD. In the rest of the paper, the parameter combination (κ, μ) for learning CAD is set to $(0.1 \times d_{Ci}, 100)$.

Table I summarizes the parameters setting of GOAL+ algorithm for learning the i -th layer Information Preserving Analysis Dictionary (IPAD) and Clustering Analysis Dictionary (CAD). Both the IPAD and the CAD are initialized with i.i.d. Gaussian random entries. Fig. 9 shows the objective minimization of the IPAD learning and the CAD learning with the default settings and when learning from around 20,000 training samples.

We further explore the dependency of the performance of learned DeepAM on the training data size. Fig. 8 shows the performance of DeepAM when using different batch size and

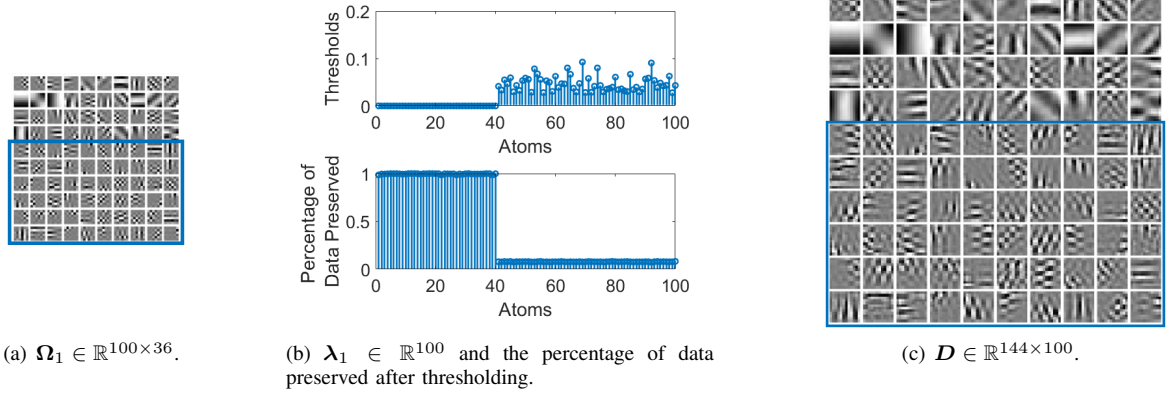


Fig. 10. An example of a learned 1-layer DeepAM. Each atom is displayed as a 2D patch. The atoms within the blue box are the clustering atoms. In Ω_1 , the first 40 atoms are the information preserving atoms and the remaining 60 atoms are the clustering atoms.

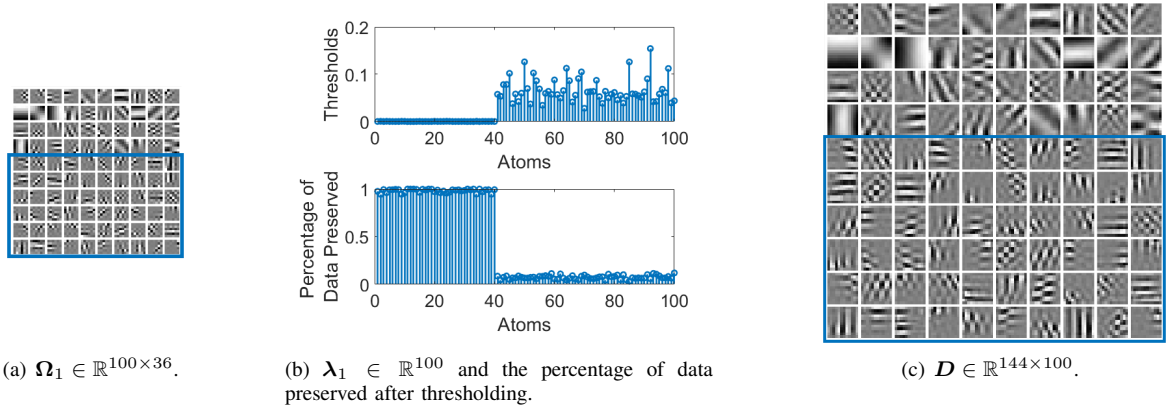


Fig. 11. The 1-layer DeepAM further fine-tuned using back-propagation. The dictionary atoms seem more localized. The thresholds are in general larger than those in Fig. 10(b).

number of batches for training. The total number of training samples is around 80,000. For each batch, 100 iterations of conjugate gradient descent are performed to minimize the objective function. We can find that using large batch size for training is beneficial and learning using around 15 batches is sufficient to achieve a good performance. Therefore, in the rest of the paper, we set the batch size as $\min(N, 4 \times 10^4)$ and set the number of batches as 15.

Given the default settings determined in this section, the computation time for learning a layer of DeepAM with 256 atoms using 320,000 training data pairs is about 20 minutes on a Intel Core i7-8700 6-core 3.20GHz with 32G RAM CPU machine.

C. Visualization of the Learned DeepAM

In this section, we will show and analyze the learned DeepAM with the implementation details as described in the previous section.

Fig. 10 shows an example of a learned 1-layer DeepAM. It contains an analysis dictionary Ω_1 , thresholds λ_1 and a synthesis dictionary D . Each atom is displayed in a 2D patch in which black and white corresponds to the smallest and the largest value, respectively. The number of the information preserving atoms is set to 40 which is larger than the rank of the input data. The thresholds depicted in Fig. 10(b) show a

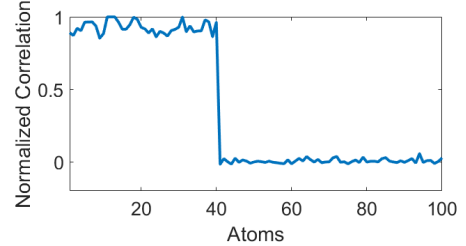


Fig. 12. The normalized inner product between the HR projection of an analysis atom $H^\dagger \omega$ and its corresponding synthesis atom d .

clear bimodal behaviour. The first 40 thresholds are close to zero, while the remaining 60 thresholds are relatively large. After thresholding, almost all coefficients corresponding to IPAD are non-zero, and the percentage of non-zero coefficients of different CAD atoms are similar and are around 8%. This indicates that modelling the distribution of the analysis coefficients as a Laplacian distribution is a good approximation. The atoms within the blue box are the clustering atoms. The atoms in IPAD shown in Fig. 10(a) are similar to the LR versions of their corresponding synthesis atoms in Fig. 10(c). The CAD atoms look like directional filters and are more localized. There is little low-frequency information. The corresponding

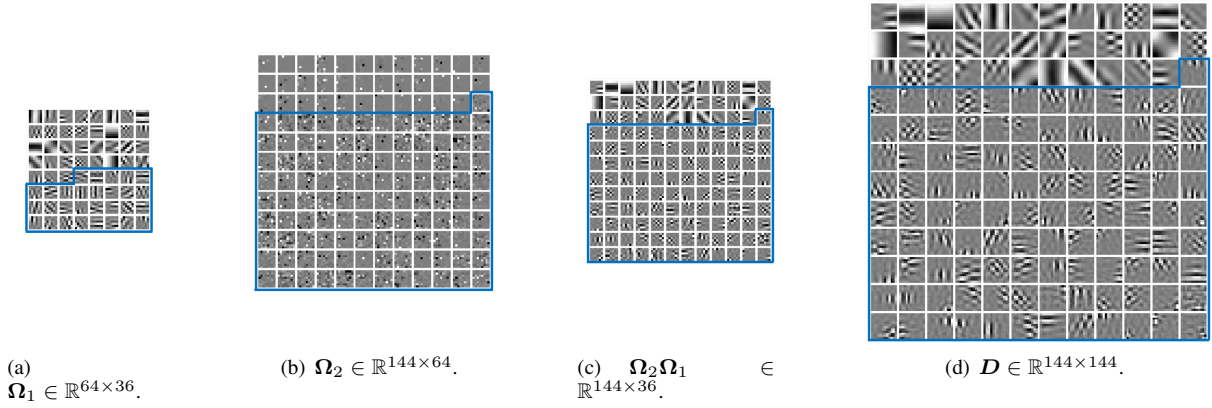


Fig. 13. An example of a learned 2-layer DeepAM. Each atom is displayed as a 2D patch. The atoms within the blue box are the clustering atoms.

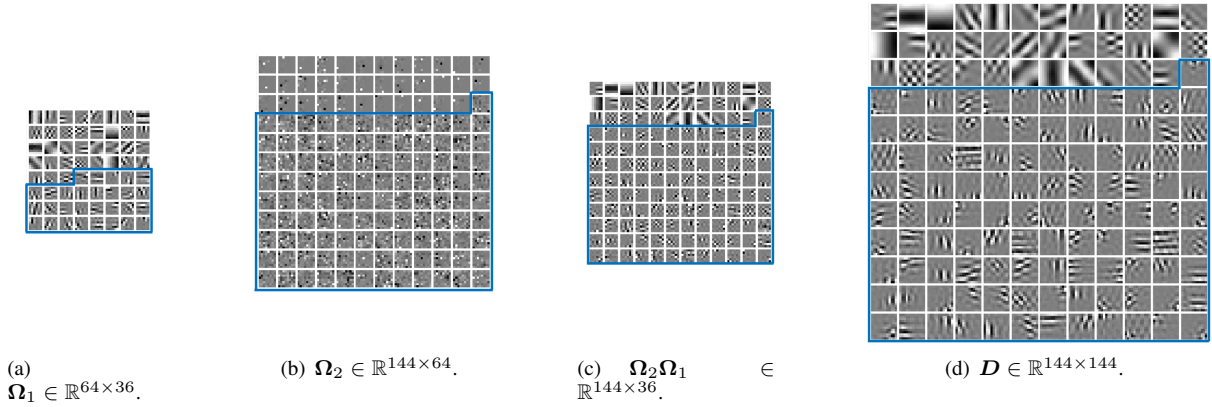


Fig. 14. The dictionaries of the 2-layer DeepAM further fine-tuned using back-propagation.

synthesis atoms are correlated to the CAD atoms, however, they are not the HR counterpart. Fig. 12 shows the normalized inner product between the HR projection of an analysis atom $H^\dagger \omega$ and its corresponding synthesis atom d . We can see that the analysis atoms and synthesis atoms corresponding to the IPAD part have high inner products, while the inner products between the analysis atoms and synthesis atoms corresponding to the CAD are very small. This shows that, in line with our objective, the synthesis atoms which correspond to the CAD part are nearly orthogonal to the LR data subspace.

Back-propagation can be used to further update all the parameters (both dictionaries and thresholds) in our learned DeepAM. The back-propagation update is implemented using Pytorch with Adam optimizer [58], batch size 1024, initial learning rate 10^{-3} , learning rate decay step 20, and decay rate 0.1. The parameter setting has been tuned to achieve the best performance.

Fig. 11 shows the 1-layer DeepAM after updating using back-propagation. With back-propagation, the performance of DeepAM has a rapid improvement with the first 5 epochs and converges within 20 epochs. After back-propagation update, the average PSNR evaluated on *Set5* has improved by approximately 0.3 dB. We can find that the different characteristics of the IPAD part and the CAD part are preserved on the updated DeepAM. There are subtle differences on the updated dictionaries. In general, the IPAD atoms have no visible

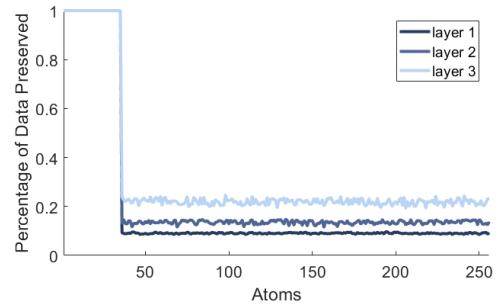


Fig. 15. The percentage of data preserved after thresholding for the atoms in 3 different layers of the 3-layer DeepAM in Table V.

changes, while the CAD atoms have become more localized. The thresholds continue to have a bimodal behaviour. There is only a slight change on the percentage of non-zero coefficients of different atoms.

Fig. 13 shows the dictionaries of a learned 2-layer DeepAM including two analysis dictionaries Ω_1 , Ω_2 and a synthesis dictionary D . The first analysis dictionary Ω_1 is similar to that in Fig. 10(a), while its CAD part mainly contains directional filters due to a smaller number of clustering atoms. The second analysis dictionary Ω_2 is shown in Fig. 13(b) and is a sparse dictionary where the sparse atoms can be consid-

Training Data Size	5,000			320,000		
Method	DNN-R	DNN-S	DeepAM	DNN-R	DNN-S	DeepAM
<i>Set5</i>	35.06	35.62	35.96	36.32	36.62	36.44
<i>Set14</i>	31.36	31.63	31.87	32.01	32.30	32.17

TABLE II

AVERAGE PSNR (dB) BY DIFFERENT METHODS LEARNED USING A SMALL AND A LARGE TRAINING DATASET EVALUATED ON *Set5* [40] AND *Set14* [41]. ALL THE MODELS ARE WITH 3 LAYERS AND EACH LAYER HAS 256 ATOMS. (THE BEST RESULT IN EACH ROW IS IN BOLD.)

ered as indicating a weighted combination of the first layer analysis dictionary atoms if the soft-thresholding operation is neglected. The effective dictionary $\Omega_{21} = \Omega_2 \Omega_1$ shown in Fig. 13(c) can partially show the effective atoms applied to the input LR data whose IPAD part is similar to that in Ω_1 and CAD part contains more localized atoms when compared to those in Ω_1 . Similar observations can be found in a deeper analysis dictionary in DeepAM. The synthesis dictionary has similar characteristics as the one in the 1-layer DeepAM.

Fig. 14 shows the dictionaries of the 2-layer DeepAM after updating with back-propagation. The back-propagation slightly updates the dictionaries and converges within 20 epochs. The average PSNR evaluated on *Set5* improves by 0.2 dB after the first 5 epochs and achieved a 0.3 dB improvement after convergence. As in the 1-layer DeepAM case, after back-propagation, there is still a clear difference between the IPAD atoms and the CAD atoms. The IPAD atoms did not change significantly, while the CAD atoms in Ω_1 and the effective dictionary Ω_{21} have become more localized.

Fig. 15 further shows the percentage of non-zero coefficients for each atom in 3 different layers of the 3-layer DeepAM. We can find that the percentage of non-zero coefficients has a bimodal behaviour in all three layers which is the same to that shown in Fig. 10(b). After thresholding, the percentage of non-zero coefficients corresponding to CAD atoms are almost the same in each layer. The percentage of non-zero coefficients for CAD atoms in layer 1, 2 and 3 is around 9%, 14% and 22%, respectively. This means the feature representation becomes less sparse with the increase of layers. A denser signal representation is helpful for modelling more complex signals which requires the use of more synthesis atoms for a good reconstruction quality.

D. Comparison with Deep Neural Networks

To have a better understanding of DeepAM, in this section, we compare our proposed DeepAM method with deep neural networks (DNNs) learned using back-propagation algorithm. We will first discuss the ability to learn from different training data size, in particular, small training data. We will then compare different methods on image super-resolution with noise to promote a better understanding of DeepAM.

For training DeepAM, we use the default settings discussed in Section V-A. Unless otherwise specified, the number of IPAD atoms in each layer is set to be K_{LR} which is the rank of the input LR data.

For comparison, DNNs are learned with the same training data using gradient descent with back-propagation. Let us denote with DNN-R and DNN-S the DNN with ReLU (with bias terms) as non-linearity and soft-thresholding (with soft-thresholds) as non-linearity, respectively. Note that all the

dictionaries, bias terms in DNN-R and the soft-thresholds in DNN-S are learnable. The architecture of DNN-S is the same as our DeepAM. The implementation of DNNs is based on Pytorch with Adam optimizer [58], using batch size 1024, initial learning rate 5×10^{-3} , and decay rate 0.1. The parameter setting has been tuned to achieve the best performance. The parameters of the DNNs are initialized using the default method in Pytorch.

1) *Image Super-Resolution Learning from Small and Large Training Samples:* In this part, we will discuss the ability of DeepAM to learn from training datasets of different sizes.

Table II shows the performance of DNN-R, DNN-S and DeepAM which are learned using a small and a large set of training samples, respectively. All the models are with 3 layers and each layer has 256 atoms (neurons). For the training of DNNs, when the number of training samples is large, the total number of epochs is set to 250 and the learning rate decay step is 50, and when the number of training samples is small, the total number of epochs is set to 2500 and the learning rate decay step is 500.

From the table, we can see that DNNs and DeepAM perform differently when learned from a small and a large number of training samples. In general, the performance of DeepAM is more stable compared to DNN-R and DNN-S. When the number of training samples is sufficiently large, DNN-S is able to achieve better performances compared to DNN-R and DeepAM. The performance of DeepAM is around 0.2 dB and 0.1 dB lower than that of DNN-S on *Set5* [40] and *Set14* [41], respectively. When the number of training samples is relatively small, DeepAM achieves a higher PSNR compared to DNN-R and DNN-S. For example, when evaluated on *Set5* [40], the performance of DeepAM is around 0.9 dB and 0.3 dB higher than that of DNN-R and DNN-S, respectively. This suggests that the proposed DeepAM learning method can be very useful for applications with limited number of training samples.

We further compare DeepAM with DNN-R and DNN-S on the self-example (or self-learning) image super-resolution task [59], [60] which is an example of applications with limited number of training samples. In self-learning, these samples are obtained from the input low-resolution image and its down-sampled version since the high-resolution image is not available. The learned image super-resolution model is then applied on the input low-resolution image to perform up-sampling. In self-example image super-resolution, the number of training samples is usually very small. For example, around 3200 training sample pairs can be extracted from an input low-resolution image of resolution 128×128 pixels and its down-sampled version. As the number of training data is small, in order to achieve good performance, the total number of epochs for training DNNs is increased to 2500 with learning rate

Images	Bicubic	DNN-R	DNN-S	DeepAM
baby	36.89	38.22	38.26	38.41
bird	36.73	39.53	40.39	40.71
butterfly	27.58	30.16	30.93	31.88
head	34.70	35.44	35.54	35.56
woman	32.30	34.56	34.74	35.16
Average	33.64	35.58	35.97	36.34

TABLE III

PSNR (dB) OF DIFFERENT METHODS FOR SELF-EXAMPLE IMAGE SUPER-RESOLUTION EVALUATED ON *Set5* [40]. (THE BEST RESULT IN EACH ROW IS IN BOLD.)

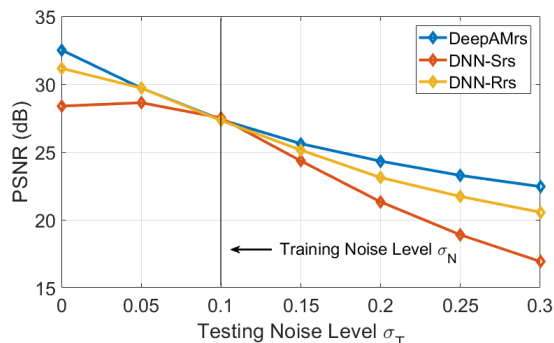


Fig. 16. The performance of DeepAM_{rs}, DNN-R_{rs}, and DNN-S_{rs} under different levels of noise evaluated on *Set5* [40]. The training noise level is $\sigma_N = 0.1$, and the testing noise level σ_T ranges from 0 to 0.3. The pixel values are within $[0, 1]$.

decay step 500. Similarly, we increase the number of iterations for training DeepAM to 5000.

Table III shows the performance of DeepAM, DNN-S and DNN-R applied to self-example image super-resolution. From the table, we can see that DeepAM significantly outperforms DNN-S and DNN-R. The average PSNR of the proposed DeepAM is around 0.8 dB and 0.4 dB higher than that of DNN-R and DNN-S, respectively. In particular, for the image butterfly, the PSNR achieved by DeepAM is around 1.7 dB and 0.9 dB higher than that of DNN-R and DNN-S, respectively. The results further indicates that DeepAM can be very useful for applications with limited number of training samples.

2) *Image Super-Resolution with Noise*: In this part, we compare DeepAM with DNN-R and DNN-S in the noisy image super-resolution task. We will see that the learned DeepAM can be well adapted to testing images with unseen noise levels by rescaling the thresholds of IPAD and CAD at the first layer.

We use a noisy training dataset where noisy images are obtained by adding i.i.d. zero mean Gaussian noise $\mathcal{N}(0, \sigma_N^2)$ with $\sigma_N = 0.1$ and around 320,000 training patch pairs are used for training. As the input LR patches are noisy, we set the number of IPAD atoms in the first layer to $3 \times K_{LR}$ and the number of IPAD atoms in other layers to K_{LR} . DNNs are learned with weight decay 10^{-6} to achieve a more robust model.

The test images are corrupted with i.i.d. zero mean Gaussian noise $\mathcal{N}(0, \sigma_T^2)$ and σ_T might be different from σ_N . We assume that the testing noise levels is known since there are methods [61], [62] that can estimate the noise level from noisy

images. When $\sigma_T^2 \neq \sigma_N^2$, we are faced with a case of mismatch between testing and training. However, since we know the task that each part of DeepAM has to fulfill, we can adjust the thresholds accordingly at testing stage without the need to retrain the network. This is something more difficult to achieve with a generic DNN.

From the analysis in Section IV-B, the IPAD thresholds should be proportional to the variance of the noise. Based on that, we can adapt the learned DeepAM to different noise levels by rescaling the IPAD thresholds of the first layer with a factor σ_T^2/σ_N^2 . The IPAD thresholds rescaling should be able to enhance the model robustness to a different noise level by passing essential information from the input signal. With noisy input image, the function of CAD at the first layer and its thresholds can be interpreted as identifying meaningful signals from the noisy input. As the CAD and its thresholds are already more selective than IPAD part, we therefore rescale the CAD thresholds of the first layer with a factor σ_T/σ_N . We denote DeepAM_{rs} as the DeepAM with rescaled IPAD and CAD thresholds.

We can also apply bias terms (thresholds) rescaling to DNN-R and DNN-S to enhance the model robustness to unseen noise levels, and we denote with DNN-R_{rs} and DNN-S_{rs} as DNN-R and DNN-S with rescaled bias terms (thresholds) at the first layer with a factor σ_T^2/σ_N^2 , respectively. All the thresholds are rescaled with the same factor as we can not identify the function of the neurons in DNNs.

Fig. 16 shows the performance of DeepAM_{rs}, DNN-R_{rs} and DNN-S_{rs} evaluated with different noise levels. The testing images are from *Set5* [40] and are with additive i.i.d. zero mean Gaussian noise $\mathcal{N}(0, \sigma_T^2)$ with σ_T ranging from 0 to 0.3. From Fig. 16, we can see that DeepAM achieves similar performance as DNN-R and DNN-S when evaluated on the testing noise level $\sigma_T = \sigma_N$. By rescaling IPAD thresholds and CAD thresholds, DeepAM_{rs} achieves a robust adaptation of DeepAM to testing images with unseen noise levels. The IPAD and its rescaled thresholds perform a denoising operation and pass essential information of the input signal to the next layer. The CAD and its rescaled thresholds shows a stronger ability to select meaningful signals from the input signals. The results of DeepAM_{rs} shows the importance of understanding the function of each component in DeepAM. Rescaling the bias terms (thresholds) of DNN-R and DNN-S can also improve their robustness to unseen noise levels, but cannot lead to the same performance as DeepAM_{rs}.

Fig. 17 show two examples of image super-resolution results of DNN-R_{rs}, DNN-S_{rs} and DeepAM_{rs} on testing image *bird* from *Set5* [41] with $\sigma_T = 0$ and $\sigma_T = 0.2$, respectively. DeepAM_{rs} is able to achieve effective image super-resolution on both cases. The rescaled thresholds can successfully pass essential information and remove noise from the input signals. With rescaled thresholds, all methods can adapt to testing images with unseen noise levels. Taking the results with $\sigma_T = 0.2$ as an example, DNN-R_{rs}, DNN-S_{rs} and DeepAM_{rs} improves DNN-R, DNN-S and DeepAM by 4.06 dB, 1.64 dB and 5.05 dB, respectively.

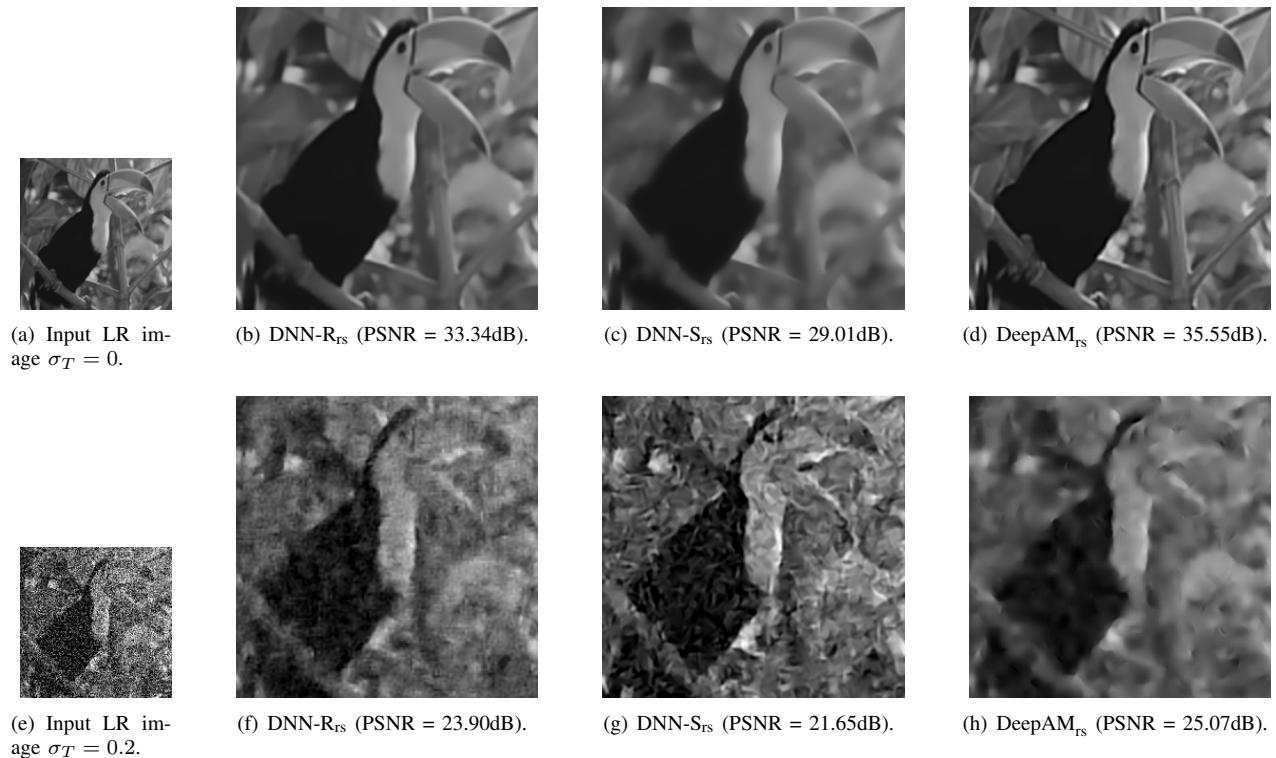


Fig. 17. Image super-resolution with different noise levels using DNN-R_{rs}, DNN-S_{rs} and DeepAM_{rs}. All the models are learned with training images with noise level $\sigma_N = 0.1$. The thresholds (bias terms) are rescaled to adapt to the testing noise levels.

Methods	Bicubic	SC [41]	ANR [42]	A+ [43]	SRCNN [47]	DNN-S	DeepAM	DeepAM _{bp}
<i>Set5</i>	33.66	35.24	35.36	34.44	35.87	35.62	35.96	36.03
<i>Set14</i>	30.23	31.50	31.49	31.26	31.88	31.63	31.87	31.92

TABLE IV

PSNR (dB) OF DIFFERENT METHODS LEARNED USING A SMALL TRAINING DATASET EVALUATED ON *Set5* [40] AND *Set14* [41]. (THE BEST RESULT IN EACH ROW IS IN BOLD.)

E. Comparison with Single Image Super-Resolution Methods

In this section, we will compare our proposed DeepAM method with some existing single image super-resolution methods including Bicubic interpolation, SC-based method [41], Anchored Neighbor Regression (ANR) method [42], Adjusted Anchored Neighborhood Regression (A+) method [43], Super-Resolution Convolutional Neural Network (SRCNN) method [47], and DNN-S.

The SC-based method [41] is a synthesis dictionary based method with a coupled LR and HR dictionary. The LR dictionary is learned using K-SVD [19] and has 1024 atoms, and the HR dictionary is learned using least squares. It assumes that a LR patch and its corresponding HR patch share the same sparse code which is retrieved using OMP [11]. The input LR feature is the concatenation of the intensity, the first-order derivatives, and the second-order derivatives of the LR data and is further compressed using Principal Component Analysis (PCA). The ANR method [42] and the A+ method [43] use the same feature representation as [41]. They apply a learned LR synthesis dictionary for LR patch clustering and have a regression model for each dictionary atom. The super-resolution algorithm finds the nearest neighbor atom for each input LR signal and apply the corresponding regression

model for HR signal prediction. The dictionary has 1024 atoms and thus there are 1024 regression models. The A+ method [43] represented the state-of-the-art before the emergence of methods based on deep convolutional neural networks. The aforementioned methods [41]–[43] are all patch-based. The Super-Resolution Convolutional Neural Network (SRCNN) method [23], [47] is the first to use convolutional neural network for single image super-resolution. SRCNN [47] has 3 layers and is with 64 filters with spatial size 9×9 , 32 filters with spatial size 1×1 and 32 filters with spatial size 5×5 for layer 1, 2 and 3, respectively. It takes the Bicubic up-scaled image as input and is able to upscale the input LR image without dividing the input image into patches. SRCNN [63] has achieved further improvement by using deeper and wider networks, and learning from ImageNet with 395,909 images. DeepAM_{bp} represents the 3-layer DeepAM (each layer with 256 atoms) that is refined using back-propagation. The input data for DeepAM and DeepAM_{bp} is the intensity of the LR image patches.

In Table IV, we compare different methods in a small training dataset setting in which the patch-based methods [41]–[43] learn from 5000 patch pairs, and the SRCNN method [47] learns from 512 patch pairs of size 36×36 extracted from the

Methods	Bicubic	SC [41]	ANR [42]	A+ [43]	SRCNN [63]	DNN-S	DeepAM	DeepAM _{bp}
<i>Set5</i>	33.66	35.78	35.82	36.23	36.66	36.62	36.44	36.71
<i>Set14</i>	30.23	31.79	31.78	32.04	32.45	32.30	32.17	32.40

TABLE V

PSNR (dB) OF DIFFERENT METHODS LEARNED USING LARGE TRAINING DATASETS EVALUATED ON *Set5* [40] AND *Set14* [41]. (THE BEST RESULT IN EACH ROW IS IN BOLD.)

91 training images [40]. In this small training dataset setting, DeepAM already achieves better performance than the comparison methods and DeepAM_{bp} further improves DeepAM and achieves the highest average PSNR. In Table V, we further compare different methods in a large training dataset setting. The patch-based methods [41]–[43] learn from 320000 patch pairs extracted from the 91 training images [40], and the SRCNN [63] learns using more than 5 million patch pairs of size 33×33 from the ILSVRC 2013 ImageNet detection training partition. In this large training dataset setting, the performance of DeepAM is around 0.2 dB lower than DNN-S. With further fine-tuning, DeepAM_{bp} achieves the highest PSNR and its performance is comparable to that of SRCNN [63]. The fact that DeepAM and DNN perform similarly suggests that back-propagation optimization probably does something similar to our optimization strategy. Moreover, the good performance of DeepAM_{bp} suggests that our approach provides an effective initialization for back-propagation.

VI. CONCLUSIONS

In this paper, we proposed a Deep Analysis Dictionary Model (DeepAM) which consists of multiple layers of analysis dictionary and soft-thresholding operators and a layer of synthesis dictionary. Each analysis dictionary has been designed to contain two sub-dictionaries: an Information Preserving Analysis Dictionary (IPAD) and a Clustering Analysis Dictionary (CAD). The IPAD and threshold pairs are to pass key information from the input to deeper layers. The function of the CAD and threshold pairs is to facilitate discrimination of key features. We proposed an extension of GOAL [26] to perform dictionary learning for both the IPAD and the CAD. The thresholds have been efficiently set according to simple principles, while leading to effective models. Simulation results show that our proposed DeepAM outperforms DNNs with similar number of trainable parameters for small training datasets. We also show that, due to the interpretability of the components of DeepAM architecture, it is easier to adjust some parameters of the model to handle noisy test images with unseen levels of noise.

REFERENCES

- [1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.
- [2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.
- [3] S. Mallat, “Group invariant scattering,” *Communications on Pure and Applied Mathematics*, vol. 65, no. 10, pp. 1331–1398, 2012.
- [4] J. Bruna and S. Mallat, “Invariant scattering convolution networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1872–1886, 2013.
- [5] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *European conference on computer vision*. Springer, 2014, pp. 818–833.
- [6] N. Lei, Z. Luo, S.-T. Yau, and D. X. Gu, “Geometric understanding of deep learning,” *arXiv preprint arXiv:1805.10451*, 2018.
- [7] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, “On the number of linear regions of deep neural networks,” in *Advances in neural information processing systems*, 2014, pp. 2924–2932.
- [8] R. Giryes, G. Sapiro, and A. M. Bronstein, “Deep neural networks with random Gaussian weights: A universal classification strategy?” *IEEE Trans. Signal Processing*, vol. 64, no. 13, pp. 3444–3457, 2016.
- [9] M. Elad, “Sparse and redundant representations: From theory to applications in signal and image processing,” *Springer*, 2010.
- [10] M. Elad, P. Milanfar, and R. Rubinstein, “Analysis versus synthesis in signal priors,” *Inverse problems*, vol. 23, no. 3, p. 947, 2007.
- [11] Y. C. Pati, R. Rezaifar, and P. S. Krishnaprasad, “Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition,” in *Proceedings of 27th Asilomar conference on signals, systems and computers*. IEEE, 1993, pp. 40–44.
- [12] W. Dai and O. Milenkovic, “Subspace pursuit for compressive sensing signal reconstruction,” *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2230–2249, 2009.
- [13] T. Blumensath and M. E. Davies, “Iterative hard thresholding for compressed sensing,” *Applied and computational harmonic analysis*, vol. 27, no. 3, pp. 265–274, 2009.
- [14] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.
- [15] S. S. Chen, D. L. Donoho, and M. A. Saunders, “Atomic decomposition by basis pursuit,” *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.
- [16] I. Daubechies, M. Defrise, and C. De Mol, “An iterative thresholding algorithm for linear inverse problems with a sparsity constraint,” *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 57, no. 11, pp. 1413–1457, 2004.
- [17] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.
- [18] K. Engan, S. O. Aase, and J. H. Husoy, “Method of optimal directions for frame design,” in *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, vol. 5. IEEE, 1999, pp. 2443–2446.
- [19] M. Aharon, M. Elad, A. Bruckstein *et al.*, “K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation,” *IEEE Transactions on Signal Processing*, vol. 54, no. 11, p. 4311, 2006.
- [20] A.-K. Seghouane and A. Iqbal, “Consistent adaptive sequential dictionary learning,” *Signal Processing*, vol. 153, pp. 300–310, 2018.
- [21] R. Rubinstein, T. Peleg, and M. Elad, “Analysis K-SVD: A dictionary-learning algorithm for the analysis sparse model,” *IEEE Transactions on Signal Processing*, vol. 61, no. 3, pp. 661–677, 2013.
- [22] M. Yaghoobi, S. Nam, R. Gribonval, and M. E. Davies, “Constrained overcomplete analysis operator learning for cosparsely signal modelling,” *IEEE Transactions on Signal Processing*, vol. 61, no. 9, pp. 2341–2355, 2013.
- [23] J. Dong, W. Wang, W. Dai, M. D. Plumbley, Z.-F. Han, and J. Chambers, “Analysis simco algorithms for sparse analysis model based dictionary learning,” *IEEE Transactions on Signal Processing*, vol. 64, no. 2, pp. 417–431, 2015.
- [24] S. Ravishanker and Y. Bresler, “Learning sparsifying transforms,” *IEEE Transactions on Signal Processing*, vol. 61, no. 5, pp. 1072–1086, 2012.
- [25] L. Pfister and Y. Bresler, “Learning filter bank sparsifying transforms,” *IEEE Transactions on Signal Processing*, vol. 67, no. 2, pp. 504–519, 2018.
- [26] S. Hawe, M. Kleinstuber, and K. Diepold, “Analysis operator learning and its application to image reconstruction,” *IEEE Transactions on Image Processing*, vol. 22, no. 6, pp. 2138–2150, 2013.

- [27] M. Kiechle, T. Habigt, S. Hawe, and M. Kleinsteuber, "A bimodal co-sparse analysis model for image processing," *International Journal of Computer Vision*, vol. 114, no. 2-3, pp. 233–247, 2015.
- [28] R. Rubinstein and M. Elad, "Dictionary learning for analysis-synthesis thresholding," *IEEE Transactions on Signal Processing*, vol. 62, no. 22, pp. 5962–5972, 2014.
- [29] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1553–1564, 2010.
- [30] A.-K. Seghouane and A. Iqbal, "Basis expansion approaches for regularized sequential dictionary learning algorithms with enforced sparsity for fMRI data analysis," *IEEE Transactions on Medical Imaging*, vol. 36, no. 9, pp. 1796–1807, 2017.
- [31] V. Pappas, Y. Romano, and M. Elad, "Convolutional neural networks analyzed via convolutional sparse coding," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 2887–2938, 2017.
- [32] J. Sulam, V. Pappas, Y. Romano, and M. Elad, "Multilayer convolutional sparse modeling: Pursuit and dictionary learning," *IEEE Transactions on Signal Processing*, vol. 66, no. 15, pp. 4090–4104, 2018.
- [33] C. Garcia-Cardona and B. Wohlberg, "Convolutional dictionary learning: A comparative review and new algorithms," *IEEE Transactions on Computational Imaging*, vol. 4, no. 3, pp. 366–381, 2018.
- [34] I. Y. Chun and J. A. Fessler, "Convolutional analysis operator learning: Acceleration and convergence," *IEEE Transactions on Image Processing*, vol. 29, pp. 2108–2122, 2020.
- [35] I. Y. Chun, D. Hong, B. Adcock, and J. A. Fessler, "Convolutional analysis operator learning: Dependence on training data," *IEEE Signal Processing Letters*, vol. 26, no. 8, pp. 1137–1141, 2019.
- [36] S. Tariyal, A. Majumdar, R. Singh, and M. Vatsa, "Deep dictionary learning," *IEEE Access*, vol. 4, pp. 10 096–10 109, 2016.
- [37] S. Mahdizadehaghdam, A. Panahi, H. Krim, and L. Dai, "Deep dictionary learning: A parametric network approach," *arXiv preprint arXiv:1803.04022*, 2018.
- [38] J.-J. Huang and P. L. Dragotti, "A deep dictionary model for image super-resolution," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'18)*, March 2018.
- [39] —, "A deep dictionary model to preserve and disentangle key features in a signal," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'19)*, May 2019.
- [40] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.
- [41] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *International conference on curves and surfaces*. Springer, 2010, pp. 711–730.
- [42] R. Timofte, V. De Smet, and L. Van Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1920–1927.
- [43] —, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Asian Conference on Computer Vision*. Springer, 2014, pp. 111–126.
- [44] J.-J. Huang, W.-C. Siu, and T.-R. Liu, "Fast image interpolation via random forests," *IEEE Transactions on Image Processing*, vol. 24, no. 10, pp. 3232–3245, 2015.
- [45] J.-J. Huang and W.-C. Siu, "Learning hierarchical decision trees for single image super-resolution," *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.
- [46] J.-J. Huang, T. Liu, P. L. Dragotti, and T. Stathaki, "SRHRF+: Self-example enhanced single image super-resolution using hierarchical random forests," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on New Trends in Image Restoration and Enhancement*, July 2017.
- [47] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European Conference on Computer Vision*. Springer, 2014, pp. 184–199.
- [48] —, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.
- [49] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1646–1654.
- [50] A. Fawzi, M. Davies, and P. Frossard, "Dictionary learning for fast classification based on soft-thresholding," *International Journal of Computer Vision*, vol. 114, no. 2-3, pp. 306–321, 2015.
- [51] I. Y. Chun and J. A. Fessler, "Deep BCD-net using identical encoding-decoding CNN structures for iterative image recovery," in *2018 IEEE 13th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*. IEEE, 2018.
- [52] I. Y. Chun, X. Zheng, Y. Long, and J. A. Fessler, "BCD-net for low-dose CT reconstruction: Acceleration, convergence, and generalization," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2019.
- [53] I. Y. Chun, Z. Huang, H. Lim, and J. Fessler, "Momentum-net: Fast and convergent iterative neural network for inverse problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
- [54] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.
- [55] M. Elad, "Why simple shrinkage is still relevant for redundant representations?" *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5559–5569, 2006.
- [56] M. Raphan and E. P. Simoncelli, "Optimal denoising in redundant representations," *IEEE Transactions on Image Processing*, vol. 17, no. 8, pp. 1342–1352, 2008.
- [57] Y. Lin and D. D. Lee, "Bayesian l_1 -norm sparse learning," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 5. IEEE, 2006, pp. V–V.
- [58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [59] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. A. Morel, "Single-image super-resolution via linear mapping of interpolated self-examples," *IEEE Transactions on Image Processing*, vol. 23, no. 12, pp. 5334–5347, 2014.
- [60] A. Shocher, N. Cohen, and M. Irani, "zero-shot" super-resolution using deep internal learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3118–3126.
- [61] D. L. Donoho, "De-noising by soft-thresholding," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 613–627, 1995.
- [62] C. Liu, W. T. Freeman, R. Szeliski, and S. B. Kang, "Noise estimation from a single image," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1. IEEE, 2006, pp. 901–908.
- [63] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2015.