Learning Deep Analysis Dictionaries—Part I: Unstructured Dictionaries

Jun-Jie Huang, *Student 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. Simulation results show that the proposed deep analysis dictionary model achieves comparable performance with a Deep Neural Network which has the same structure and is optimized using back-propagation.**

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

I. Introduction

DEEP Neural Networks (DNNs) [3] are complex architectures composed of a cascade of multiple linear and non-linear layers. Back-propagation algorithm [4] 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 [5], [6] 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 [7] 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 [8], 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.* [9] have shown that the number of cells is exponentially larger than the degrees of freedom. Giryes *et al.* [10] 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.

Contrary to DNNs, the sparse representation framework [11] is much more established. Sparse representation over an over-complete dictionary can serve as an effective and robust representation 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 [12].

A synthesis model [12] represents a signal $\boldsymbol{x} = \boldsymbol{D}\boldsymbol{\gamma}$ as a linear combination of a small number of column atoms from a redundant synthesis dictionary $\boldsymbol{D} \in \mathbb{R}^{n\times m}$ with $n < m$ and $\|\boldsymbol{\gamma}\|_0 = k \ll m$. Sparse pursuit is to seek the sparsest representation $\boldsymbol{\gamma}$ given the input signal $\boldsymbol{x}$ and the dictionary $\boldsymbol{D}$. Sparse pursuit algorithms include greedy algorithms [13]–[15] and convex relaxation based algorithms [16]–[19]. The greedy algorithms, like Orthogonal Matching Pursuit (OMP) [13], 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) [17] or iterative algorithms, like Iterative Soft-Thresholding Algorithm (ISTA) [18], [19]. The synthesis dictionary learning algorithms [20], [21] mainly take an alternating minimization strategy and iterate between a sparse-coding stage and a dictionary update stage. With well established theories and algorithms, sparse representation over redundant synthesis model has been extensively used in signal and image processing.

Recently, the analysis model [12], [22] has attracted more research interests. A redundant analysis dictionary $\boldsymbol{\Omega} \in \mathbb{R}^{m\times n}$ is a tall matrix with $m > n$ where each row of $\boldsymbol{\Omega}$ is an atom of the dictionary. The expectation is that the analysis coefficients $\boldsymbol{\Omega x}$ 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 a regularization term $\lambda||\boldsymbol{\Omega x}||_1$ 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 [22]–[26]. Analysis K-SVD [22] iterates between

This work was in part presented in [1], [2].

an analysis pursuit operation and a K-SVD dictionary update stage. Yaghoobi *et al.* [23] proposed a uniformly-normalized tight frame constraint for learning analysis operators. Analysis Simultaneous Codeword Optimization (ASimCO) algorithm [24] enforces a K-sparse constraint on the sparse-coding stage and updates multiple dictionary atoms simultaneously in the dictionary update stage. Sparsifying transform learning [25], [26] proposed to constrain the analysis operator to be full rank and well-conditioned. The GeOmetric Analysis Operator Learning (GOAL) algorithm [27], [28] 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 [29] 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 [30] 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. A Multi-Layer Convolutional Sparse Coding (ML-CSC) model is proposed in [31], [32] and gives a new interpretation on the working of the 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. Tariyal *et al.* [33] proposed a greedy layer-wise deep dictionary learning method which performs synthesis dictionary learning layer-by-layer. A parametric approach is proposed in [34] 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 three-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 facilitate such discrimination.
- We propose learning algorithms for DeepAM. To achieve the two different goals of IPAD and CAD, different learning criteria are introduced. We validate our proposed DeepAM on the single image super-resolution task. Simulation results show that the proposed deep dictionary model achieves comparable performance with a DNN which has the same structure but is optimized using back-propagation.

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 $\widehat{\boldsymbol{Y}}$ from an observed (low-resolution) LR image $\boldsymbol{X}$.

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 [35]–[41] 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 [35]–[37], [39]–[44] 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 $\left\{\left(\boldsymbol{x}_i^0, \boldsymbol{y}_i\right)\right\}_{i=1}^N$ 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 $\boldsymbol{\mathcal{X}}^0 = [\boldsymbol{x}_1^0, \cdots, \boldsymbol{x}_N^0] \in \mathbb{R}^{d_0 \times N}$ and the corresponding ground-truth HR training data matrix $\boldsymbol{\mathcal{Y}} = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_N] \in \mathbb{R}^{d_{L+1} \times N}$.

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,

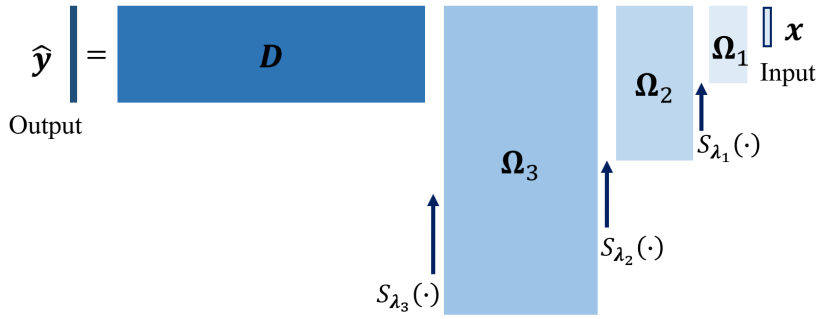


Fig. 1. A 3-layer deep analysis dictionary model. There are 3 layers of analysis dictionaries $\{\boldsymbol{\Omega}_i\}_{i=1}^3$ with element-wise soft-thresholding operators $\{\mathcal{S}_{\boldsymbol{\lambda}_i}(\cdot)\}_{i=1}^3$ and a layer of synthesis dictionary $\boldsymbol{D}$. The output signal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

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 $\left\{\boldsymbol{\Omega}_i \in \mathbb{R}^{d_i \times d_{i-1}}\right\}_{i=1}^{L}$ with $d_i \geq d_{i-1}$. The row atoms $\left\{\boldsymbol{\omega}_{i,j}^T\right\}_{j=1}^{d_i}$ of the analysis dictionary $\boldsymbol{\Omega}_i$ are of unit norm. The non-linear operator used in DeepAM is the element-wise soft-threshold[1] $\{\mathcal{S}_{\boldsymbol{\lambda}_i}(\cdot)\}_{i=1}^{L}$ where $\boldsymbol{\lambda}_i \in \mathbb{R}^{d_i}$ denotes the threshold vector at layer i. The dictionary $\boldsymbol{D}$ 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:

$$\widehat{\boldsymbol{y}} = \boldsymbol{D}\mathcal{S}_{\boldsymbol{\lambda}_L}\left(\boldsymbol{\Omega}_L\mathcal{S}_{\boldsymbol{\lambda}_{L-1}}\left(\cdots\boldsymbol{\Omega}_2\mathcal{S}_{\boldsymbol{\lambda}_1}\left(\boldsymbol{\Omega}_1\boldsymbol{x}^0\right)\cdots\right)\right), \quad (1)$$

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

Let us denote with $\boldsymbol{x}^i = \mathcal{S}_{\boldsymbol{\lambda}_i}(\boldsymbol{\Omega}_i\boldsymbol{x}^{i-1})$ the output of the i-th layer. This means that the input signal $\boldsymbol{x}^{i-1}$ is multiplied with the analysis dictionary $\boldsymbol{\Omega}_i$ and then passed through the element-wise soft-thresholding operator $\mathcal{S}_{\boldsymbol{\lambda}_i}(\cdot)$. The thresholded output signal $\boldsymbol{x}^i$ will be a sparse signal and is expected to be better at predicting the ground-truth signal $\boldsymbol{y}$ than $\boldsymbol{x}^{i-1}$. After L layers, $\boldsymbol{x}^L$ 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 Unite (ReLU) non-linearity[2]. As ReLU can be considered as the one-sided version of soft-thresholding, there exists an equivalence between a layer of analysis dictionary with soft-thresholding and a layer of Neural Networks with ReLU:

$$\boldsymbol{\Omega}_{i+1}\mathcal{S}_{\boldsymbol{\lambda}_i}\left(\boldsymbol{\Omega}_i\boldsymbol{x}^{i-1}\right) = \boldsymbol{\Omega}_{i+1}^{\text{HC}}\text{ReLU}(\boldsymbol{\Omega}_i^{\text{VC}}\boldsymbol{x}^{i-1}, \boldsymbol{\lambda}_i^{\text{VC}}), \quad (2)$$

[1]Soft-thresholding is defined as $\mathcal{S}_\lambda(a) = \text{sgn}(a)\max(|a|-\lambda, 0)$.
[2]ReLU is defined as $\text{ReLU}(a,\lambda) = \max(a-\lambda, 0)$.

where $\boldsymbol{\Omega}_{i+1}^{\text{HC}} = [\boldsymbol{\Omega}_{i+1}, -\boldsymbol{\Omega}_{i+1}]$, $\boldsymbol{\Omega}_i^{\text{VC}} = [\boldsymbol{\Omega}_i; -\boldsymbol{\Omega}_i]$ and $\boldsymbol{\lambda}_i^{\text{VC}} = [\boldsymbol{\lambda}_i; \boldsymbol{\lambda}_i]$. The superscripts HC and VC stand for horizontal concatenate and vertical concatenate, respectively.

From Eqn. (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.

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:

$$\min_{\boldsymbol{\theta}} \left\|\boldsymbol{\mathcal{Y}} - \boldsymbol{D}\mathcal{S}_{\boldsymbol{\lambda}_L}\left(\cdots\boldsymbol{\Omega}_2\mathcal{S}_{\boldsymbol{\lambda}_1}\left(\boldsymbol{\Omega}_1\boldsymbol{\mathcal{X}}^0\right)\cdots\right)\right\|_F^2, \quad (3)$$

where $\boldsymbol{\theta} = \left\{\boldsymbol{D}, \{\boldsymbol{\Omega}_i\}_{i=1}^L, \{\boldsymbol{\lambda}_i\}_{i=1}^L\right\}$ denotes all the parameters of the L-layer DeepAM.

Optimizing Eqn. (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 [4]. However this would lead to effective but difficult to interpret results.

Our objective instead is to build a deep dictionary model with a higher interpretability through understanding the purpose of 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:

$$\widehat{\boldsymbol{y}}_i = \boldsymbol{D}\mathcal{S}_{\boldsymbol{\lambda}_1}(\boldsymbol{\Omega}_1\boldsymbol{x}_i^0), \quad (4)$$

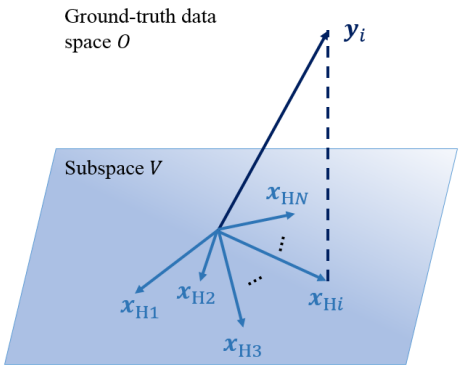


Fig. 2. In image super-resolution, the input data spans a low-dimensional subspace V within the HR data space O. The objective is to estimate the unknown HR signal $\boldsymbol{y}_i$ based on the input LR signal $\boldsymbol{x}_{\mathrm{H}i}$. The dashed line represents the residual signal $\boldsymbol{r}_i = \boldsymbol{y}_i - \boldsymbol{x}_{\mathrm{H}i}$ which is orthogonal to the subspace spanned by the input data.

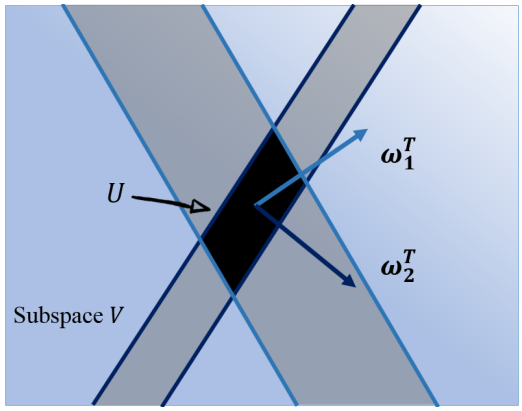


Fig. 3. The analysis and soft-thresholding operations partition the input data subspace V. There are two pairs of analysis atom and soft-thresholding operator. After soft-thresholding, the data in the gray region is with 1 zero coefficient and the data in the convex polyhedron U (i.e. the black region)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

where $\boldsymbol{x}_i^0 \in \mathbb{R}^{d_0}$ is one of the elements of $\boldsymbol{\mathcal{X}}$, $\widehat{\boldsymbol{y}}_i \in \mathbb{R}^{d_2}$ with $d_2 > d_0$ and $\boldsymbol{\Omega}_1 \in \mathbb{R}^{d_1 \times d_0}$.

We assume, for the sake of argument, that the degradation model is linear. That is, there exists a degradation matrix $\boldsymbol{H} \in \mathbb{R}^{d_0 \times d_2}$ such that $\boldsymbol{x}_i^0 = \boldsymbol{H}\boldsymbol{y}_i$. Denote $\boldsymbol{x}_{\mathrm{H}i} = \boldsymbol{H}^\dagger \boldsymbol{x}_i^0 \in \mathbb{R}^{d_2}$ as the projection of the LR signal $\boldsymbol{x}^0$ onto the HR signal space with the pseudo-inverse matrix $\boldsymbol{H}^\dagger$.

As shown in Fig. 2, the signal $\boldsymbol{x}_{\mathrm{H}i}$ 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 $\boldsymbol{\Omega}_1$ and the non-linear soft-thresholding operation $\mathcal{S}_{\boldsymbol{\lambda}_1}(\cdot)$ in a way that facilitates the recovery of the information of $\boldsymbol{y}$ in $V^\perp$.

When we multiply $\boldsymbol{x}^0$ with $\boldsymbol{\Omega}_1$, the analysis dictionary atoms $\{\boldsymbol{\omega}_{1,j}^T\}_{j=1}^{d_1}$ project the input LR data onto specific directions. After soft-thresholding, the resulting signal $\boldsymbol{x}^1 = \mathcal{S}_{\boldsymbol{\lambda}_1}(\boldsymbol{\Omega}_1 \boldsymbol{x}^0)$ 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 $\boldsymbol{D}$.

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, $\mathcal{S}_{\boldsymbol{\lambda}_1}(\boldsymbol{\Omega}_1 \boldsymbol{x}^0) = \boldsymbol{0}$, $\forall \boldsymbol{x}^0 \in U$ (see the central black region in Fig. 3). Therefore, the corresponding estimation $\widehat{\boldsymbol{y}}$ 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

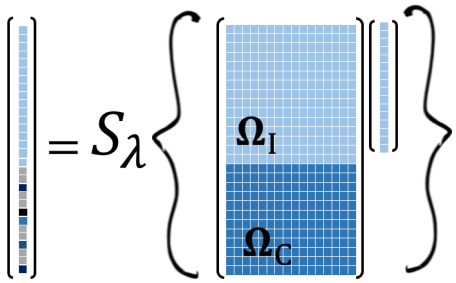


Fig. 4. The analysis dictionary $\boldsymbol{\Omega}$ is designed to consist of an information preserving analysis dictionary $\boldsymbol{\Omega}_\mathrm{I}$ and a clustering analysis dictionary $\boldsymbol{\Omega}_\mathrm{C}$. The soft-thresholds corresponding to $\boldsymbol{\Omega}_\mathrm{C}$ are much higher than those used for $\boldsymbol{\Omega}_\mathrm{I}$ and result in a sparser representation.

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 within the same cluster has the same sparsity pattern and shares the same linear model 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 $\boldsymbol{\Omega}$ into two sub-dictionaries $\boldsymbol{\Omega} = [\boldsymbol{\Omega}_\mathrm{I}; \boldsymbol{\Omega}_\mathrm{C}]$, and similarly divide each threshold vector into two parts $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_\mathrm{I}; \boldsymbol{\lambda}_\mathrm{C}]$. The Information Preserving Analysis Dictionary (IPAD) $\boldsymbol{\Omega}_\mathrm{I}$ with its thresholds $\boldsymbol{\lambda}_\mathrm{I}$ aims at passing key information of the input signal to the next layer. The Clustering Analysis Dictionary (CAD) $\boldsymbol{\Omega}_\mathrm{C}$ with its threshold $\boldsymbol{\lambda}_\mathrm{C}$ is to facilitate the separation of key feature in the signal. The CAD and thresholding operators provide a highly non-linear 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 $\{(\mathbf{\Omega}_{\mathrm{I}i}, \boldsymbol{\lambda}_{\mathrm{I}i})\}_{i=1}^{L}$ 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 $\boldsymbol{x}_{\mathrm{I}}^{L}$ 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 $\{(\mathbf{\Omega}_{\mathrm{C}i}, \boldsymbol{\lambda}_{\mathrm{C}i})\}_{i=1}^{L}$ are the main source of non-linearity in DeepAM. The feature representation $\boldsymbol{x}_{\mathrm{C}}^{L}$ generated by the last layer of CAD should be able to well represent the signal components of $\boldsymbol{y}$ 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 four 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) there are no pairs of row atoms that are linearly dependent; (4) the row atoms are of unit norm.

Our proposed learning algorithm is an extension of the GeOmetric Analysis Operator Learning (GOAL) algorithm [27], [28] 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 $\mathbf{\Omega} \in \mathbb{R}^{m\times n}$, the j-th atom of $\mathbf{\Omega}$ as $\boldsymbol{\omega}_j^T$ and the training data as $\boldsymbol{\mathcal{X}} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N] \in \mathbb{R}^{n\times N}$. We assume that the data $\boldsymbol{\mathcal{X}}$ span a K dimensional subspace $V \in \mathbb{R}^n$. Let us denote with $\boldsymbol{W} \in \mathbb{R}^{n\times K}$ an orthogonal basis of V, and with $\boldsymbol{U} \in \mathbb{R}^{n\times(n-K)}$ the orthogonal basis of the orthogonal complement $V^{\perp}$.

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

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

Together with the unit norm constraint, the feasible set of the analysis dictionary $\mathbf{\Omega}$ is therefore defined as $\Theta = \mathbb{S}_{n-1}^{m} \cap \boldsymbol{U}^{\perp}$ 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 $\mathbf{\Omega}$ to be of unit norm and excludes the contributions from $\boldsymbol{U}$. Eqn. (5) is a generalization of the log-det constraint term applied in GOAL [27], [28]. This is because in our case, $\boldsymbol{W}$ 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 [27], [28], $\boldsymbol{W}$ defines a much larger subspace and is with $K = n$ or $K = n-1$.

We achieve the constraint $\mathbf{\Omega}^T \in \Theta$ by performing orthogonal projection onto the tangent space $\mathrm{T}_{\mathbf{\Omega}}(\Theta)$ of the manifold Θ at location $\mathbf{\Omega}$. For a row atom $\boldsymbol{\omega}^T$, the operation of the orthogonal projection onto the tangent space $\mathrm{T}_{\boldsymbol{\omega}}(\Theta)$ can be represented by the projection matrix $\boldsymbol{P}_{\boldsymbol{\omega}}$ [28]:

$$\boldsymbol{P}_{\boldsymbol{\omega}} = \mathbf{I}_n - \boldsymbol{Q}_{\boldsymbol{\omega}}^{\dagger}\boldsymbol{Q}_{\boldsymbol{\omega}}, \quad (6)$$

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

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:

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

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

Linearly dependent row atoms (e.g. $\boldsymbol{\omega}_i^T = \pm\boldsymbol{\omega}_j^T$) are undesirable in the learned dictionary. A logarithm barrier term $l(\mathbf{\Omega})$ is used to penalize linearly dependent row atoms:

$$l(\mathbf{\Omega}) = -\frac{1}{m(m-1)}\sum_{1\le i<j\le m}\log\left(1-\left(\boldsymbol{\omega}_i^T\boldsymbol{\omega}_j\right)^2\right). \quad (8)$$

The combination of the information preservation constraint in Eqn. (5), sparsifying constraint in Eqn. (7), and linearly

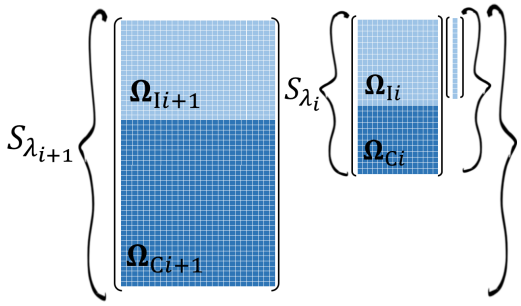


Fig. 5. Two consecutive layers in DeepAM. The IPAD and threshold pairs create an information flow channel from input to output, and the CAD and threshold pairs combine information from the previous layer and generate a feature representation that can well represent the residual part.

Algorithm 1: GOAL+ Algorithm

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

dependent penalty term in Eqn. (8) leads to the objective function of GOAL+:

$$\mathbf{\Omega} = \arg\min_{\mathbf{\Omega}^T \in \Theta} f(\mathbf{\Omega}), \tag{9}$$

where $f(\mathbf{\Omega}) = g(\mathbf{\Omega}) + \kappa h(\mathbf{\Omega}) + \upsilon l(\mathbf{\Omega})$ with κ and υ being the regularization parameters.

The objective function defined in Eqn. (9) is optimized using a geometric conjugate gradient descent method [27], [45]. The analysis dictionary learning algorithm GOAL+ is summarized in **Algorithm 1**. At iteration t, the gradient of the objective function $\nabla f(\mathbf{\Omega}^{(t)})$ is computed and orthogonal projected on the tangent space of the manifold Θ at location $\mathbf{\Omega}^{(t)}$. The orthogonal projection of $\nabla f(\mathbf{\Omega})$ onto the tangent space $\mathrm{T}_{\mathbf{\Omega}}(\Theta)$ can be expressed as $\Pi_{\mathrm{T}_{\mathbf{\Omega}}(\Theta)}(\nabla f(\mathbf{\Omega})) = [\boldsymbol{P}_{\boldsymbol{\omega}_1}\nabla f(\boldsymbol{\omega}_1), \cdots, \boldsymbol{P}_{\boldsymbol{\omega}_{d_{\mathrm{I}i}}}\nabla f(\boldsymbol{\omega}_{d_{\mathrm{I}i}})]$. Let us denote $\mathcal{G} \doteq \Pi_{\mathrm{T}_{\mathbf{\Omega}^{(t)}}(\Theta)}(\nabla f(\mathbf{\Omega}^{(t)}))$, 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 $\mathbf{\Omega}^{(t+1)}$ 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 [27] with the exception of the orthogonal projection step as described in Eqn. (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 [27], [45]. Now that the overall objectives of GOAL+ have been introduced, we can focus on how to tailor the optimization in Eqn. (9) to achieve the objectives of IPAD and CAD respectively.

B. Learning IPAD and Threshold Pairs

The function of the IPAD and threshold pair $(\mathbf{\Omega}_{\mathrm{I}i}, \boldsymbol{\lambda}_{\mathrm{I}i})$ is to pass key information of the input data $\boldsymbol{\mathcal{X}}^0$ to deeper layers. The learned IPADs create a channel that enables the information flow from the input signal to the estimated output signal.

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

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

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

There are related works in the literature about thresholding for redundant representations [46]–[48]. Elad [46] 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 [47] proposed a denoising scheme for redundant representations based on Stein's Unbiased Risk Estimator. Lin and Lee [48] proposed a Bayesian framework for finding the optimal l_1-norm regularization.

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

$$\min_{\boldsymbol{z}} \frac{1}{2}||\boldsymbol{x} - \mathbf{\Omega}^T\boldsymbol{z}||_2^2 + \sum_{j=1}^{m} \lambda_{\mathrm{I}j}|z_j|, \tag{10}$$

where z_j is the j-th coefficient of the sparse vector $\boldsymbol{z}$, and $\lambda_{\mathrm{I}j}$ is the corresponding regularization parameter.

Selecting the soft-threshold $\boldsymbol{\lambda}_{\mathrm{I}i}$ is equivalent to finding suitable regularization parameters in Eqn. (10) as the soft-thresholding operation $\mathcal{S}_{\boldsymbol{\lambda}}(\mathbf{\Omega}\boldsymbol{x})$ can be interpreted as the first iteration of the Iterative Soft-Thresholding Algorithm (ISTA) [18] for solving Eqn. (10):

$$\boldsymbol{z}^{(1)} = \mathcal{S}_{\boldsymbol{\lambda}}\left(\boldsymbol{z}^{(0)} + \mathbf{\Omega}\left(\boldsymbol{x} - \mathbf{\Omega}^T\boldsymbol{z}^{(0)}\right)\right), \tag{11}$$

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

Lin and Lee [48] proposed a method to choose the optimal regularization parameters based on a Bayesian analysis. They assume that the data $\boldsymbol{x}$ is with additive i.i.d. zero mean Gaussian noise with variance σ^2:

$$P(\boldsymbol{x}|\mathbf{\Omega}^T, \boldsymbol{z}, \sigma^2) = \frac{1}{(2\pi\sigma)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\left\|\boldsymbol{x} - \mathbf{\Omega}^T\boldsymbol{z}\right\|_2^2\right) \tag{12}$$

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

$$P(\boldsymbol{z}|\boldsymbol{\lambda}) = \prod_{j=1}^{m} \frac{\lambda_j}{2} \exp\left(-\lambda_j|z_j|\right). \tag{13}$$

Empirically, we have found that the prior distribution $P(\boldsymbol{z})$ can be well characterized by an i.i.d. zero-mean Laplacian distribution. Based on the analysis in [48], the optimal regularization parameters for Eqn. (10) can be set as inversely proportional to the variance of the Laplacian distribution:

$$\boldsymbol{\lambda} \propto \left[\frac{1}{\sigma_1}, \cdots, \frac{1}{\sigma_m}\right]^T, \tag{14}$$

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

From Eqn. (14), the soft-threshold associated with IPAD $\boldsymbol{\Omega}_{\mathrm{I}i}$ is:

$$\boldsymbol{\lambda}_{\mathrm{I}i} = \rho_{\mathrm{I}} \left[\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \cdots, \frac{1}{\sigma_{d_{\mathrm{I}i}}}\right]^T, \tag{15}$$

where ρ_{I} is a scaling parameter, and the variance σ_j of the j-th coefficient can be estimated using the obtained IPAD $\boldsymbol{\Omega}_{\mathrm{I}i}$ and its input data.

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:

$$\hat{\rho}_{\mathrm{I}} = \arg\min_{\rho \in \mathcal{D}} \left\| \boldsymbol{\mathcal{Y}} - \boldsymbol{G}\mathcal{S}_{\rho\boldsymbol{\lambda}}\left(\boldsymbol{\Omega}_{\mathrm{I}i}\boldsymbol{\mathcal{X}}^{i-1}\right)\right\|_F^2, \tag{16}$$

where $\boldsymbol{\lambda} = [1/\sigma_1, 1/\sigma_2, \cdots, 1/\sigma_{d_{\mathrm{I}i}}]^T$, $\boldsymbol{G} = \boldsymbol{\mathcal{Y}}\boldsymbol{Z}^T(\boldsymbol{Z}\boldsymbol{Z}^T)^{-1}$ with $\boldsymbol{Z} = \mathcal{S}_{\rho\boldsymbol{\lambda}}(\boldsymbol{\Omega}_{\mathrm{I}i}\boldsymbol{\mathcal{X}}^{i-1})$, and $\mathcal{D}$ is a discrete set of values.

The obtained pair $(\boldsymbol{\Omega}_{\mathrm{I}i}, \boldsymbol{\lambda}_{\mathrm{I}i})$ 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 $(\boldsymbol{\Omega}_{\mathrm{C}i}, \boldsymbol{\lambda}_{\mathrm{C}i})$ 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: Different from IPAD, learning CAD $\boldsymbol{\Omega}_{\mathrm{C}i}$ requires supervision from both the input training data $\boldsymbol{\mathcal{X}}^{i-1}$ and the ground-truth HR training data $\boldsymbol{\mathcal{Y}}$.

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

$$\boldsymbol{D}_i = \arg\min_{\boldsymbol{D}} ||\boldsymbol{D}\boldsymbol{\mathcal{X}}^{i-1} - \boldsymbol{\mathcal{Y}}||_F^2. \tag{17}$$

It has a closed-form solution given by:

$$\boldsymbol{D}_i = \boldsymbol{\mathcal{Y}}{\boldsymbol{\mathcal{X}}^{i-1}}^T \left(\boldsymbol{\mathcal{X}}^{i-1}{\boldsymbol{\mathcal{X}}^{i-1}}^T\right)^{-1}. \tag{18}$$

The learning objective for CAD $\boldsymbol{\Omega}_{\mathrm{C}i}$ is that its atoms should be able to project $\boldsymbol{\mathcal{X}}^{i-1}$ 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 $\boldsymbol{\Psi}_i \in \mathbb{R}^{d_{\mathrm{C}i} \times d_{L+1}}$ which acts on the data $\boldsymbol{\mathcal{Y}}^i$ and is able to jointly sparsify the data $\boldsymbol{\mathcal{Y}}^i$ and the residual data $\boldsymbol{\mathcal{E}}^i$. That is, the atoms in $\boldsymbol{\Psi}_i$ are able to identify the data in $\boldsymbol{\mathcal{Y}}^i$ with large residual energy. The i-th layer CAD is then re-parameterized as:

$$\boldsymbol{\Omega}_{\mathrm{C}i} = \boldsymbol{\Psi}_i\boldsymbol{D}_i. \tag{19}$$

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

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 $\boldsymbol{\mathcal{Y}}^i$ and $\boldsymbol{\mathcal{E}}^i$:

$$p(\boldsymbol{\Psi}) = c\sum_{j=1}^{N}\sum_{l=1}^{d_{\mathrm{C}i}} \log\left(1 + \nu\left(\left(\boldsymbol{\psi}_l^T\boldsymbol{y}_j^i\right)^2 - \left(\boldsymbol{\psi}_l^T\boldsymbol{e}_j^i\right)^2\right)^2\right), \tag{20}$$

where $c = 1/Nd_{\mathrm{C}1}\log(1+\nu)$, and ν is a tunable parameter.

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

$$\boldsymbol{\Psi}_i = \arg\min_{\boldsymbol{\Psi}^T \in \Theta} f(\boldsymbol{\Psi}), \tag{21}$$

where $f(\boldsymbol{\Psi}) = g(\boldsymbol{\Psi}) + \kappa h(\boldsymbol{\Psi}) + \upsilon l(\boldsymbol{\Psi}) + \mu p(\boldsymbol{\Psi})$ with κ, υ and μ being the regularization parameters. Here, $g(\cdot)$, $h(\cdot)$ and $l(\cdot)$ are those defined in Eqn. (7), Eqn. (5) and Eqn. (8), respectively.

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

The new objective function in Eqn. (21) is optimized using GOAL+ algorithm. With the learned analysis dictionary $\boldsymbol{\Psi}_i$, the i-th layer CAD $\boldsymbol{\Omega}_{\mathrm{C}i}$ is obtained as in Eqn. (19).

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 in the form of:

$$\boldsymbol{\lambda}_{\mathrm{C}i} = \rho_{\mathrm{C}}\left[\sigma_1, \sigma_2, \cdots, \sigma_{d_{\mathrm{C}i}}\right], \tag{22}$$

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 coefficients 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 Eqn. (16) by solving a 1-dimensional search problem. The optimization problem for ρ_{C} is formulated as:

$$\rho_{\mathrm{C}} = \arg\min_{\rho \in \mathcal{D}} \left\| \boldsymbol{\mathcal{Y}}_R - \boldsymbol{G}\mathcal{S}_{\rho\boldsymbol{\lambda}}\left(\boldsymbol{\Omega}_{\mathrm{C}i}\boldsymbol{\mathcal{X}}^{i-1}\right)\right\|_F^2, \tag{23}$$

Algorithm 2: DeepAM Learning Algorithm

Input: Training data pair $(\boldsymbol{\mathcal{X}}^0, \boldsymbol{\mathcal{Y}})$, the number of layers L, and the structure of DeepAM;
for $i \leftarrow 1$ **to** L **do**
  Learning $\boldsymbol{\Omega}_{\mathrm{I}i}$ using GOAL+ with training data $\boldsymbol{\mathcal{X}}^{i-1}$ and objective function Eqn. (9);
  Learning $\boldsymbol{\Omega}_{\mathrm{C}i}$ using GOAL+ with training data $(\boldsymbol{\mathcal{X}}^{i-1}, \boldsymbol{\mathcal{Y}})$ and objective function Eqn. (21);
  Learning thresholds $\boldsymbol{\lambda}_{\mathrm{I}i}$ and $\boldsymbol{\lambda}_{\mathrm{C}i}$;
  $\boldsymbol{\Omega}_i \leftarrow [\boldsymbol{\Omega}_{\mathrm{I}i}; \boldsymbol{\Omega}_{\mathrm{C}i}]$, $\boldsymbol{\lambda}_i \leftarrow [\boldsymbol{\lambda}_{\mathrm{I}i}; \boldsymbol{\lambda}_{\mathrm{C}i}]$;
  Update input training data as $\boldsymbol{\mathcal{X}}^i = \mathcal{S}_{\boldsymbol{\lambda}_i}(\boldsymbol{\Omega}_i \boldsymbol{\mathcal{X}}^{i-1})$;
end
Learning the synthesis dictionary $\boldsymbol{D}$ as in Eqn. (24).
Output: Learned DeepAM $\left\{\{\boldsymbol{\Omega}_i, \boldsymbol{\lambda}_i\}_{i=1}^{L}, \boldsymbol{D}\right\}$.

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

In the simulation results in Section V, 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 $\boldsymbol{\mathcal{X}}^0$ is consistently non-linearly transformed to a high dimensional feature representation $\boldsymbol{\mathcal{X}}^L$ with good descriptive and discriminative properties. The synthesis dictionary $\boldsymbol{D}$ models the linear transformation from $\boldsymbol{\mathcal{X}}^L$ to the desired HR counterpart $\boldsymbol{\mathcal{Y}}$. Similar to Eqn. (18), the synthesis dictionary is learned using least squares:

$$\boldsymbol{D} = \boldsymbol{\mathcal{Y}} \boldsymbol{\mathcal{X}}^{L^T} \left(\boldsymbol{\mathcal{X}}^L \boldsymbol{\mathcal{X}}^{L^T}\right)^{-1}. \tag{24}$$

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 and with other single image super-resolution algorithms.

Parameters	ν	κ	υ	μ
IPAD	$100 \times d_{i-1}$	$d_{\mathrm{I}i}$	$0.01 \times d_{\mathrm{I}i}$	—
CAD	$100 \times d_{i-1}$	$0.01 \times d_{\mathrm{C}i}$	$0.01 \times d_{\mathrm{C}i}$	100

TABLE I
Parameters setting of GOAL+ algorithm for learning the i-th layer IPAD $\boldsymbol{\Omega}_{\mathrm{I}i} \in \mathbb{R}^{d_{\mathrm{I}i} \times d_{i-1}}$ and CAD $\boldsymbol{\Omega}_{\mathrm{C}i} \in \mathbb{R}^{d_{\mathrm{C}i} \times d_{i-1}}$.

A. Implementation Details

We use the standard 91 training images [35] as training dataset and use *Set5* [35] and *Set14* [36] as the testing dataset. The Peak Signal-to-Noise Ratio (PSNR)[3] 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. Around $N = 3 \times 10^5$ LR-HR patch pairs have been collected for training. During testing, full patch overlapping is applied to reconstruct the HR images.

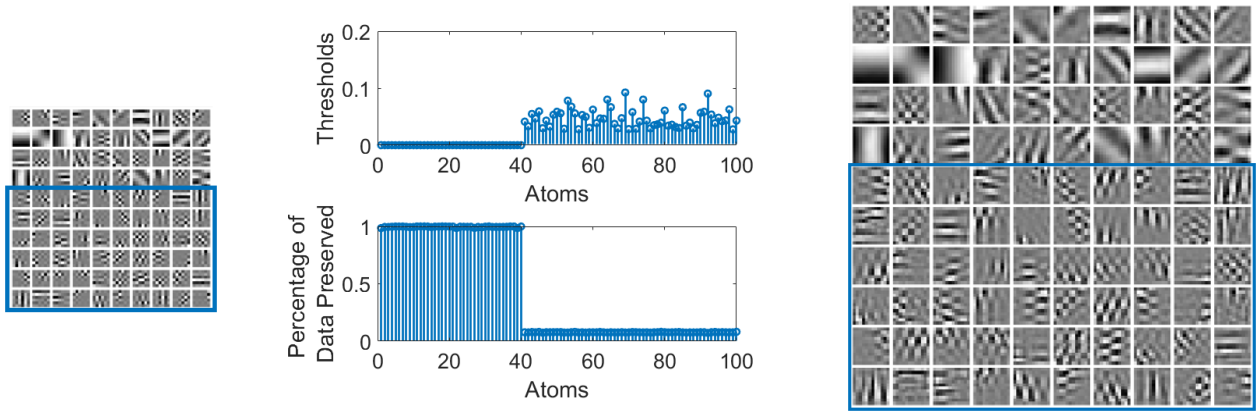
Table I shows the parameters setting of GOAL+ algorithm for learning the i-th layer Information Preserving Analysis Dictionary (IPAD) and Clustering Analysis Dictionary (CAD). Both the IPAD and the CAD are initialized with i.i.d. Gaussian random entries. We apply batch training for GOAL+ algorithm. The training data has been divided into batches with size $N_b = 10^4$. During training, the GOAL+ algorithm is sequentially applied to batches until the learned dictionary converges. For each batch, $\tau = 100$ iterations of conjugate gradient descent are performed to update the dictionary. The discrete set $\mathcal{D}$ used for searching the scaling parameter of the thresholds is set to be $\mathcal{D} = [\cdots, 10^{-2}, 2 \times 10^{-2}, \cdots, 10^{-1}, 2 \times 10^{-1}, \cdots]$.

B. 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.

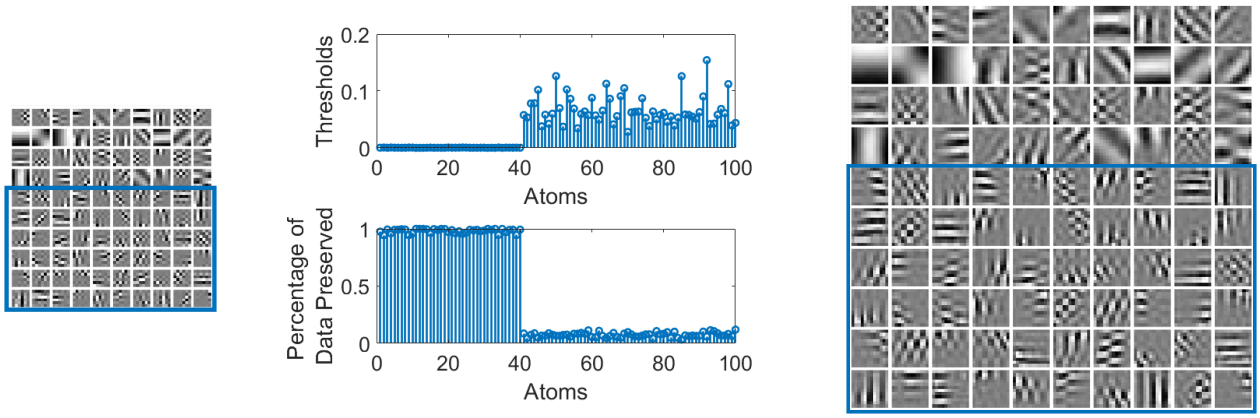
Figure 6 shows an example of a learned 1-layer DeepAM. It contains an analysis dictionary $\boldsymbol{\Omega}_1$, thresholds $\boldsymbol{\lambda}_1$ and a synthesis dictionary $\boldsymbol{D}$. Each atom is displayed in a 2D patch in which black and white corresponds to the smallest and the largest 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. 6(b) show a 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

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



(a) $\boldsymbol{\Omega}_1 \in \mathbb{R}^{100\times 36}$. (b) $\boldsymbol{\lambda}_1 \in \mathbb{R}^{100}$ and the percentage of data preserved after thresholding. (c) $\boldsymbol{D} \in \mathbb{R}^{144\times 100}$.

Fig. 6. 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 $\boldsymbol{\Omega}_1$, the first 40 atoms are the information preserving atoms and the remaining 60 atoms are the clustering atoms.



(a) $\boldsymbol{\Omega}_1 \in \mathbb{R}^{100\times 36}$. (b) $\boldsymbol{\lambda}_1 \in \mathbb{R}^{100}$ and the percentage of data preserved after thresholding. (c) $\boldsymbol{D} \in \mathbb{R}^{144\times 100}$.

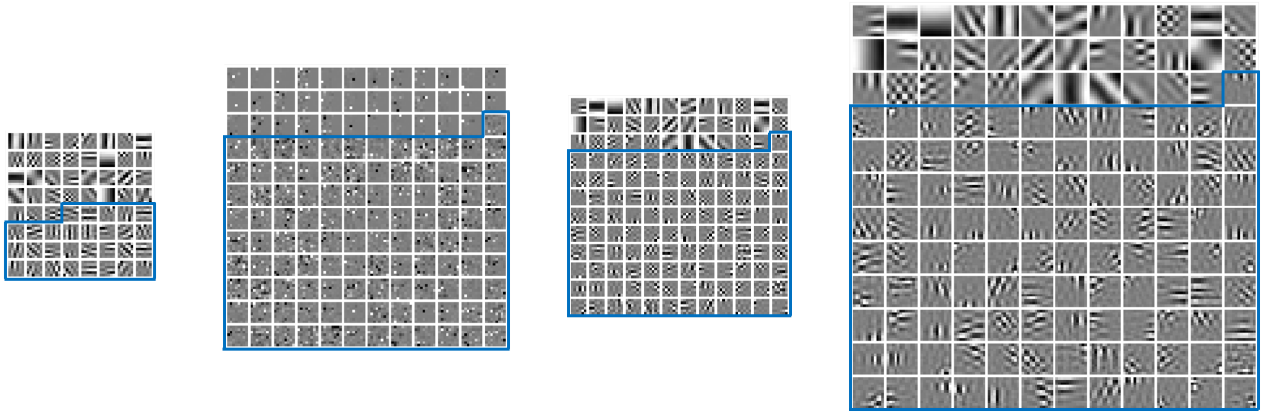
Fig. 7. The 1-layer DeepAM further fine-tuned using back-propagation. The dictionary atoms seem more localized. The thresholds are in general larger than those in Fig. 6(b).

atoms within the blue box are the clustering atoms. The atoms in IPAD shown in Fig. 6(a) are similar to the LR versions of their corresponding synthesis atoms in Fig. 6(c). The CAD atoms look like directional filters and are more localized. There is little low-frequency information. The corresponding synthesis atoms are correlated to the CAD atoms, however, they are not the HR counterpart. The inner product between the HR projection of a CAD atom and its corresponding synthesis atom $\langle \boldsymbol{H}^{\dagger}\boldsymbol{\omega}, \boldsymbol{d}\rangle$ is close to zero. 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 the parameters in our learned DeepAM. The back-propagation update is implemented using Pytorch with Adam optimizer [49], 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.

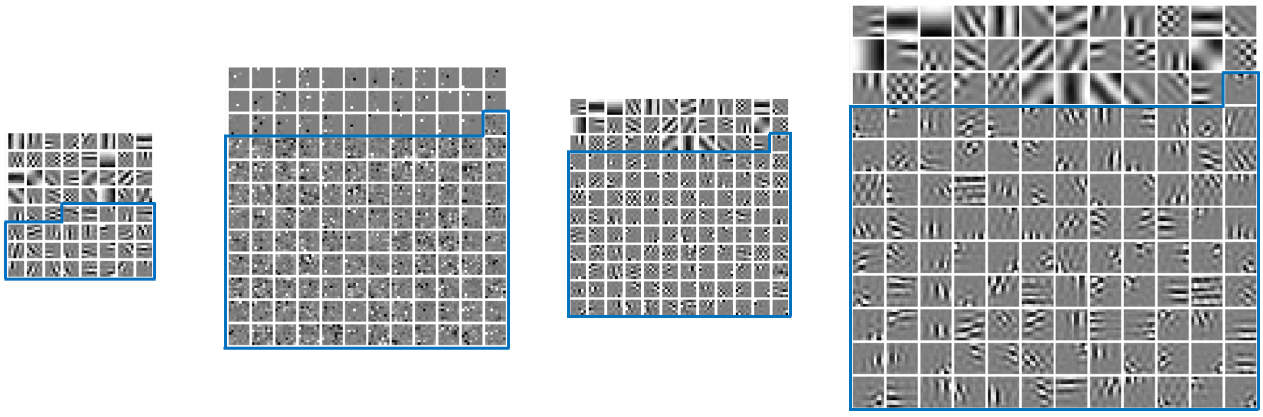
Figure 7 shows the 1-layer DeepAM after updating using back-propagation. With back-propagation, the performance of DeepAM has a rapid improvement with the first 5 epochs and converges within 20 epochs. After back-propagation update, the average PSNR evaluated on *Set5* has improved by approximately 0.3 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 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.

Figure 8 shows the dictionaries of a learned 2-layer DeepAM including two analysis dictionaries $\boldsymbol{\Omega}_1$, $\boldsymbol{\Omega}_2$ and a synthesis dictionary $\boldsymbol{D}$. The first analysis dictionary $\boldsymbol{\Omega}_1$ is similar to that in Figure 6(a), while its CAD part mainly contains directional filters due to a smaller number of clustering atoms. The second analysis dictionary $\boldsymbol{\Omega}_2$ is shown in Figure 8(b) and is a sparse dictionary where the sparse atoms can be considered as indicating a weighted combination of the first layer analysis dictionary atoms if the soft-thresholding operation is neglected. The effective dictionary $\boldsymbol{\Omega}_{21} = \boldsymbol{\Omega}_2\boldsymbol{\Omega}_1$



(a) $\mathbf{\Omega}_1 \in \mathbb{R}^{64\times 36}$. (b) $\mathbf{\Omega}_2 \in \mathbb{R}^{144\times 64}$. (c) $\mathbf{\Omega}_2\mathbf{\Omega}_1 \in \mathbb{R}^{144\times 36}$. (d) $\boldsymbol{D} \in \mathbb{R}^{144\times 144}$.

Fig. 8. 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.



(a) $\mathbf{\Omega}_1 \in \mathbb{R}^{64\times 36}$. (b) $\mathbf{\Omega}_2 \in \mathbb{R}^{144\times 64}$. (c) $\mathbf{\Omega}_2\mathbf{\Omega}_1 \in \mathbb{R}^{144\times 36}$. (d) $\boldsymbol{D} \in \mathbb{R}^{144\times 144}$.

Fig. 9. The dictionaries of the 2-layer DeepAM further fine-tuned using back-propagation.

shown in Figure 8(c) can partially show the effective atoms applied to the input LR data whose IPAD part is similar to that in $\mathbf{\Omega}_1$ and CAD part contains more localized atoms when compared to those in $\mathbf{\Omega}_1$. 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.

Figure 9 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 $\mathbf{\Omega}_1$ and the effective dictionary $\mathbf{\Omega}_{21}$ have become more localized.

C. Comparison with Deep Neural Networks

In this section, we compare our proposed DeepAM method with Deep Neural Networks (DNNs). The number of IPAD atoms in each layer is set to be 35 which is the rank of the input LR data since a DeepAM with more CAD atoms provides better performance when the input data is noiseless. For comparison, DNNs are learned with the same training data using gradient descent with back-propagation. Let us denote DNN-R and DNN-S as the DNN with ReLU as non-linearity and soft-thresholding as non-linearity, respectively. The architecture of DNN-S is the same as our DeepAM. The implementation of DNNs is based on Pytorch with Adam optimizer [49], batch size 1024, initial learning rate 5×10^{-3}, learning rate decay step 50, and decay rate 0.1. The total number of epoches for training is 250. The parameter setting has been tuned to achieve the best performance. The parameters of the DNNs are initialized using the default method in Pytorch.

Table II reports the average PSNR (dB) of DNN-R, DNN-S and the proposed DeepAM method evaluated on *Set5* [35] and *Set14* [36]. There are three different model sizes which correspond to DNNs with 1 hidden layer, 2 hidden layers and 3 hidden layers where the number of neurons in each layer is fixed to be 256.

With a deeper model, the performance of our proposed DeepAM method improves even when the size of the final feature representation is the same. This indicates that the depth of DeepAM is important for the final performance. Figure 10

Model Size	$36 \times 256 \times 144$			$36 \times 256 \times 256 \times 144$			$36 \times 256 \times 256 \times 256 \times 144$		
Method	DNN-R	DNN-S	DeepAM	DNN-R	DNN-S	DeepAM	DNN-R	DNN-S	DeepAM
Set5	35.83	36.26	35.90	36.14	36.50	36.12	36.19	36.54	36.22
Set 14	31.80	32.06	31.83	32.00	32.23	31.96	32.01	32.25	32.02

TABLE II
AVERAGE PSNR (DB) BY DIFFERENT METHODS EVALUATED ON *Set5* [35] AND *Set 14* [36].

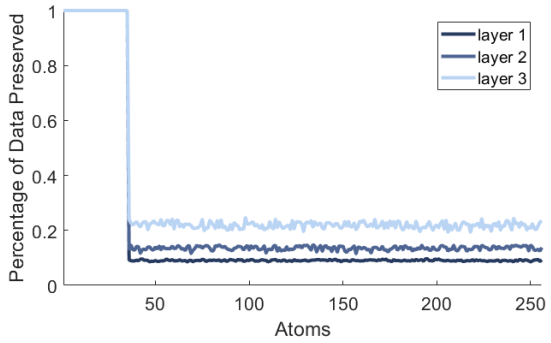


Fig. 10. The percentage of data preserved after thresholding for the atoms in 3 different layers of the 3-layer DeepAM in Table II.

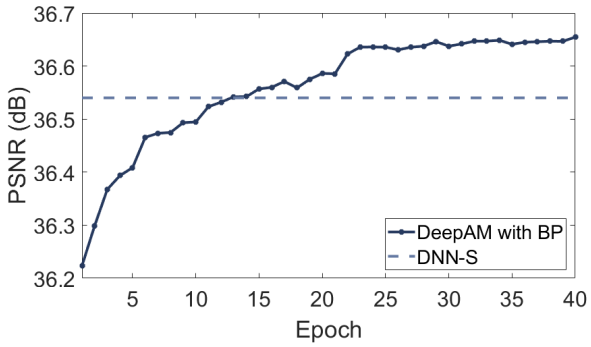


Fig. 11. The average PSNR (dB) of the DeepAM updated using back-propagation evaluated on *Set5* [35].

further shows the percentage of non-zero coefficients for each atom in 3 different layers of the 3-layer DeepAM in Table II. 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 Figure 6(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. This could be the reason for an improved performance with the increase of depth.

Our proposed DeepAM method achieves a similar average PSNR when compared to the DNN-R method over different model sizes. The DNN-S method achieves the highest average PSNR which is around 0.3 dB and 0.2 dB higher than that of our proposed DeepAM method when evaluated on *Set5* and *Set 14*, respectively. The slightly lower performance of DeepAM when compared to DNN-S is not surprising since DeepAM does not utilize joint optimization as DNNs and the thresholds are set according to simple principles. On the other hand, the simulation result validates the effectiveness of our proposed method. The better performance of DNN-S also suggests that DNNs with soft-thresholding as non-linearity can be more effective for image enhancement applications.

As shown in the previous section, back-propagation can be used to further improve the learned DeepAM. Figure 11 shows average PSNR of the 3-layer DeepAM in Table II which is updated using back-propagation and the performance of the baseline DNN-S. The performance of DeepAM improves significantly and outperforms DNN-S within 15 epochs. The converged performance of DeepAM is around 0.1 dB higher than that of DNN-S. The result shows that the parameters of DeepAM are not far from the final parameters. It also suggests that the DeepAM learning algorithm has the potential to be a good initialization method for DNNs.

D. 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 [36], Anchored Neighbor Regression (ANR) method [37], Adjusted Anchored Neighborhood Regression (A+) method [38], and Super-Resolution Convolutional Neural Network (SRCNN) method [42].

The SC-based method [36] is a synthesis dictionary based method with a coupled LR and HR dictionary. The LR dictionary is learned using K-SVD [21] 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 [13]. 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 [37] and the A+ method [38] use the same feature representation as [36]. 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 [38] represented the state-of-the-art before the emergence of methods based on deep convolutional neural networks. The aforementioned methods [36]–[38] are all patch-based. The Super-Resolution Convolutional Neural Network (SRCNN) method [42] is the first to use Convolutional Neural Network for single image super-resolution. SRCNN has 3 layers and is

Images	Bicubic	SC [36]	ANR [37]	A+ [38]	SRCNN [42]	DeepAM	DeepAM$_{bp}$
baboon	24.85	25.47	25.55	25.66	25.64	25.65	**25.70**
barbara	27.87	28.50	28.43	28.49	**28.53**	28.49	28.43
bridge	26.64	27.63	27.62	27.87	27.74	27.82	**27.93**
coastguard	29.16	30.23	30.34	30.34	30.43	30.44	**30.53**
comic	25.75	27.34	27.47	27.98	**28.17**	27.77	28.12
face	34.69	35.45	35.52	**35.63**	35.57	35.62	35.62
flowers	30.13	32.04	32.06	32.80	**32.95**	32.45	32.86
foreman	35.55	38.41	38.31	**39.45**	37.43	38.89	39.34
lenna	34.52	36.06	36.17	36.45	36.46	36.46	**36.51**
man	29.11	30.31	30.33	30.74	**30.78**	30.57	**30.78**
monarch	32.68	35.50	35.46	36.77	**37.11**	36.06	37.00
pepper	35.02	36.64	36.51	37.08	36.89	36.87	**37.11**
ppt3	26.58	29.00	28.67	29.79	**30.31**	29.13	29.70
zebra	30.41	33.05	32.91	33.45	33.14	33.34	**33.71**
Average	30.21	31.83	31.81	32.32	32.22	32.11	**32.38**

TABLE III
PSNR (DB) OF DIFFERENT METHODS EVALUATED ON *Set 14* [36]. (THE BEST RESULT IN EACH ROW IS IN BOLD.)

(a) Input LR image. (b) DeepAM (PSNR = 33.34 dB). (c) DeepAM$_{bp}$ (PSNR = 33.71 dB).

Fig. 12. The input LR and the reconstructed HR image using DeepAM and DeepAM$_{bp}$.

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.

In Table III, DeepAM$_{bp}$ and DeepAM represents the 3-layer DeepAM (model size is $36 \times 256 \times 256 \times 256 \times 64$) with and without back-propagation, respectively. The input data is the intensity of the LR image patches. Instead of predicting 12×12 HR for each input LR 6×6 patch, the output of DeepAM is the central 8×8 HR patch since the input LR patch does not contain sufficient information to predict the boundary pixels. DeepAM achieves a comparable performance to the existing methods. Its average PSNR is around 0.3 dB higher than that of the SC-based method and the ANR, while it is around 0.2 dB lower than that of A+ and is around 0.1 dB lower than that of SRCNN. DeepAM$_{bp}$ achieves the highest average PSNR. Figure 12 shows an example of the input LR image and the reconstructed HR images using DeepAM and DeepAM$_{bp}$.

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 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 [27] 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 achieves comparable performance with DNNs and other existing single image super-resolution methods.

REFERENCES

[1] 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.

[2] ——, "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.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[4] 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.

[5] S. Mallat, "Group invariant scattering," *Communications on Pure and Applied Mathematics*, vol. 65, no. 10, pp. 1331–1398, 2012.

[6] 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.

[7] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[8] N. Lei, Z. Luo, S.-T. Yau, and D. X. Gu, "Geometric understanding of deep learning," *arXiv preprint arXiv:1805.10451*, 2018.

[9] 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.

[10] 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.

[11] M. Elad, "Sparse and redundant representations: From theory to applications in signal and image processing," *Springer*, 2010.

[12] M. Elad, P. Milanfar, and R. Rubinstein, "Analysis versus synthesis in signal priors," *Inverse problems*, vol. 23, no. 3, p. 947, 2007.

[13] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[14] 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.

[15] 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.

[16] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] M. Yaghoobi, S. Nam, R. Gribonval, and M. E. Davies, "Constrained overcomplete analysis operator learning for cosparse signal modelling," *IEEE Transactions on Signal Processing*, vol. 61, no. 9, pp. 2341–2355, 2013.

[24] 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.

[25] S. Ravishankar and Y. Bresler, "Learning sparsifying transforms," *IEEE Transactions on Signal Processing*, vol. 61, no. 5, pp. 1072–1086, 2012.

[26] L. Pfister and Y. Bresler, "Learning filter bank sparsifying transforms," *IEEE Transactions on Signal Processing*, vol. 67, no. 2, pp. 504–519, 2018.

[27] S. Hawe, M. Kleinsteuber, 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.

[28] 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.

[29] R. Rubinstein and M. Elad, "Dictionary learning for analysis-synthesis thresholding," *IEEE Transactions on Signal Processing*, vol. 62, no. 22, pp. 5962–5972, 2014.

[30] 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.

[31] V. Papyan, 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. Papyan, 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] S. Tariyal, A. Majumdar, R. Singh, and M. Vatsa, "Deep dictionary learning," *IEEE Access*, vol. 4, pp. 10 096–10 109, 2016.

[34] S. Mahdizadehaghdam, A. Panahi, H. Krim, and L. Dai, "Deep dictionary learning: A parametric network approach," *arXiv preprint arXiv:1803.04022*, 2018.

[35] 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.

[36] 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.

[37] 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.

[38] ——, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Asian Conference on Computer Vision*. Springer, 2014, pp. 111–126.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] ——, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[44] 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.

[45] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[46] M. Elad, "Why simple shrinkage is still relevant for redundant representations?" *IEEE transactions on information theory*, vol. 52, no. 12, pp. 5559–5569, 2006.

[47] M. Raphan and E. P. Simoncelli, "Optimal denoising in redundant representations," *IEEE Transactions on Image Processing*, vol. 17, no. 8, pp. 1342–1352, 2008.

[48] 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.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.