A DEEP DICTIONARY MODEL TO PRESERVE AND DISENTANGLE KEY FEATURES IN A SIGNAL

Jun-Jie Huang and Pier Luigi Dragotti

Communications and Signal Processing Group (CSP), Imperial College London, UK

ABSTRACT

We propose a deep dictionary model for single image super-resolution (SISR) made of multiple layers of analysis dictionaries interlaced with corresponding soft-thresholding operations and a single synthesis dictionary. In this paper, we introduce a novel method for learning analysis dictionary and thresholding pairs as building block for the deep dictionary model. Each analysis dictionary contains two sub-dictionaries: an information preserving analysis dictionary (IPAD) and a clustering analysis dictionary (CAD). The IPAD and thresholding pair passes the key information from the previous layer, while the CAD and thresholding pair gives a sparse representation of its input data that facilitates discrimination of key features. Simulation results show that the proposed deep dictionary model achieves comparable performance with a deep neural network which has the same structure and is optimized using backpropagation.

***Index Terms*—** Sparse representation, dictionary learning, sparse dictionary, deep learning

1. INTRODUCTION

Deep Neural Networks (DNNs) are computational models which are composed of multiple layers of linear transforms and point-wise non-linearities. Backpropagation algorithm [1] is usually applied to optimize 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, a clear understanding of the functioning of the linear transforms and the non-linearites is still lacking.

Contrary to DNNs, the sparse representation theory [2] is much more established. Therefore building a deep model using sparse representation and redundant dictionaries could be a way to facilitate the interpretation of DNNs. The sparse representation model [3] can be divided into synthesis model and analysis model. In the synthesis model [2], an input signal is represented as a sparse combination of column vectors of a redundant synthesis dictionary. In the analysis model [3, 4] , a redundant analysis dictionary $\mathbf{\Omega}$ is a tall matrix where each row is an atom of the dictionary. The analysis model aims to sparsify the analysis representation $\boldsymbol{\alpha} = \mathbf{\Omega}\boldsymbol{x}$.

A Multi-Layer Convolutional Sparse Coding (ML-CSC) model is proposed in [5, 6] and gives a new interpretation about the workings 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. Tariyal *et al.* [7] proposed a greedy layer-wise deep dictionary learning method which performs synthesis dictionary learning layer-by-layer. A parametric approach is proposed in [8] 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. Instead of using synthesis dictionary as building block, Huang and Dragotti [9] proposed a deep dictionary model (DDM) for the image super-resolution task which is composed of multiple layers of analysis dictionaries with associated soft-thresholding operators and a synthesis dictionary. During testing, the estimated high-resolution patch is obtained by multiplying a synthesis dictionary with a feature representation of the input low-resolution patch which is obtained using the multi-layer analysis dictionary. The forward model of DDM is similar to that of DNNs.

In this paper, we follow the same architecture as [9] but propose a novel method to learn the analysis dictionary and soft-thresholding pairs. Each analysis dictionary consists of two sub-dictionaries: an information preserving analysis dictionary (IPAD) and a clustering analysis dictionary (CAD). The IPAD together with the thresholding step preserves the key information from its previous layer, while the CAD with thresholding generates a disentangled representation. With multiple layers of analysis dictionary and thresholding, the input signal is non-linearly transformed into a sparse vector which is both informative and discriminative. Finally, the regression task is achieved by multiplying this sparse vector with a synthesis dictionary. Single image super-resolution (SISR) [10–16] will be used as a sample application to validate the proposed method.

The rest of the paper is organized as follows: Section 2 introduces our proposed method. Section 3 presents simulation results and Section 4 draws conclusions.

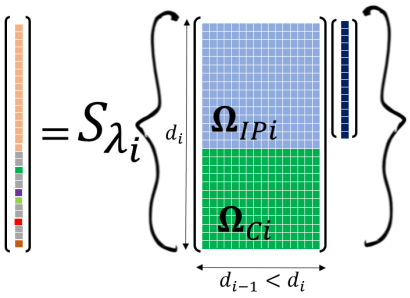


Fig. 1: The layer i of the proposed deep dictionary architecture which consists of an analysis dictionary followed by soft-thresholding. The analysis dictionary $\boldsymbol{\Omega}_i$ consists of an information preserving dictionary $\boldsymbol{\Omega}_{IPi}$ and a clustering dictionary $\boldsymbol{\Omega}_{Ci}$. The soft-thresholds corresponding to $\boldsymbol{\Omega}_{Ci}$ are much higher than those used for $\boldsymbol{\Omega}_{IPi}$.

2. PROPOSED METHOD

Good signal representations should be both informative and discriminative. In deep dictionary model, an input signal goes through multiple layers of linear transformations and pointwise non-linearities. The signal representation $\boldsymbol{\alpha}_i$ at layer i should also be informative and discriminative with respect to its input signal $\boldsymbol{\alpha}_{i-1}$. That is, $\boldsymbol{\alpha}_i$ should contain sufficient information of $\boldsymbol{\alpha}_{i-1}$, and the signal space of $\boldsymbol{\alpha}_i$ should be more disentangled when compared to that of $\boldsymbol{\alpha}_{i-1}$.

In this paper, we achieve that by learning a novel analysis dictionary and soft-thresholding pair $(\boldsymbol{\Omega}, \boldsymbol{\lambda})$. Here $\boldsymbol{\Omega}$ is composed of an Information Preserving Analysis Dictionary (IPAD) and a Clustering Analysis Dictionary (CAD) which are followed by a different set of thresholds $\boldsymbol{\lambda}_{IP}$ and $\boldsymbol{\lambda}_C$ respectively leading to the two pairs of operators $(\boldsymbol{\Omega}_{IP}, \boldsymbol{\lambda}_{IP})$ and $(\boldsymbol{\Omega}_C, \boldsymbol{\lambda}_C)$. The IPAD with its corresponding set of thresholds is responsible for passing the key information from its previous layer, while the CAD and thresholding pair facilitates the separation of key feature in the signal. We learn $\boldsymbol{\Omega}_{IP}$ and $\boldsymbol{\lambda}_{IP}$ using [9, 17], whereas we propose a new approach to learn $\boldsymbol{\Omega}_C$ and $\boldsymbol{\lambda}_C$. Given the CAD is used to discriminate features, the learned thresholds tend to be much higher than $\boldsymbol{\lambda}_{IP}$. Figure 1 illustrates the analysis dictionary and the thresholding operation at layer i.

2.1. Background

Our L-layer deep dictionary model [9] is composed of $L-1$ redundant analysis dictionaries $\left\{\boldsymbol{\Omega}_i \in \mathbb{R}^{d_i \times d_{i-1}}\right\}_{i=1}^{L-1}$ with the corresponding soft-threshold $\left\{\boldsymbol{\lambda}_i \in \mathbb{R}^{d_i}\right\}_{i=1}^{L-1}$ and a redundant synthesis dictionary $\boldsymbol{D} \in \mathbb{R}^{d_L \times d_{L-1}}$. The row atoms of the analysis dictionary are assumed to be of unit norm. The forward model of the deep dictionary model can be expressed as:

$$\boldsymbol{y} = \boldsymbol{D}\mathcal{S}_{\boldsymbol{\lambda}_{L-1}}\left(\boldsymbol{\Omega}_{L-1}\mathcal{S}_{\boldsymbol{\lambda}_{L-2}}\left(\cdots\boldsymbol{\Omega}_2\mathcal{S}_{\boldsymbol{\lambda}_1}\left(\boldsymbol{\Omega}_1\boldsymbol{x}\right)\cdots\right)\right), \quad (1)$$

where $\boldsymbol{x}$ is the input LR patch in vectorized form, $\boldsymbol{y}$ is the estimated HR patch vector, and $\mathcal{S}_{\boldsymbol{\lambda}}(\cdot)$ is the soft-thresholding operator.

Let us denote with $\boldsymbol{\mathcal{X}}_0 \in \mathbb{R}^{d_0 \times N}$ and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{d_L \times N}$ the matrices which contain the training LR patches and the corresponding HR patches, respectively. The i^{th} column of $\boldsymbol{\mathcal{X}}_0$ and $\boldsymbol{\mathcal{Y}}$ is a pair of training LR and HR patch vector $(\boldsymbol{x}_i, \boldsymbol{y}_i)$. At layer i, the thresholded training data is denoted as $\boldsymbol{\mathcal{X}}_i = \mathcal{S}_{\boldsymbol{\lambda}_i}\left(\boldsymbol{\Omega}_i\boldsymbol{\mathcal{X}}_{i-1}\right)$ for $1 \leq i < L$.

2.2. IPAD Learning

In deep dictionary model [9], the analysis dictionary learning algorithm is an extension of the geometric analysis operator learning method [17]. The obtained analysis dictionary is able to sparsify the input data and has unit norm row atoms. For a detailed description of learning objective function and the optimization method we refer to [9, 17].

The i^{th} layer Information Preserving Analysis Dictionary (IPAD) $\boldsymbol{\Omega}_{IP,i}$ is obtained by applying the dictionary learning method in [9] with the training data $\boldsymbol{\mathcal{X}}_{i-1}$. With the learned analysis dictionary, the distribution of inner product between a row atom and the input data can be well characterized by an i.i.d. zero-mean Laplacian distribution. The threshold in the soft-thresholding operator is set to be proportional to the inverse of the standard deviation σ of the Laplacian distribution. The soft-threshold for $\boldsymbol{\Omega}_{IP,i}$ is then defined as:

$$\boldsymbol{\lambda}_{IP,i} = \rho\left[\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \cdots, \frac{1}{\sigma_m}\right]^T,$$

where the standard deviation σ_i of the i^{th} coefficient can be estimated using the obtained IPAD $\boldsymbol{\Omega}_{IP,i}$ and its input data.

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

$$\hat{\rho} = \arg\min_{\rho} \left\|\boldsymbol{\mathcal{Y}} - \boldsymbol{G}\mathcal{S}_{\rho\boldsymbol{\lambda}}\left(\boldsymbol{\Omega}_{IP,i}\boldsymbol{\mathcal{X}}_{i-1}\right)\right\|_F^2, \quad (2)$$

where $\boldsymbol{\lambda} = \left[1/\sigma_1, 1/\sigma_2, \cdots, 1/\sigma_m\right]^T$, $\boldsymbol{G} = \boldsymbol{\mathcal{Y}}\ \boldsymbol{\mathcal{Z}}^T(\boldsymbol{\mathcal{Z}}\boldsymbol{\mathcal{Z}}^T)^{-1}$ with $\boldsymbol{\mathcal{Z}} = \mathcal{S}_{\rho\boldsymbol{\lambda}}(\boldsymbol{\Omega}_{IP,i}\boldsymbol{\mathcal{X}}_{i-1})$, and ρ belongs to a discrete set of values.

The obtained IPAD and its threshold pair $(\boldsymbol{\Omega}_{IP,i}, \boldsymbol{\lambda}_{IP,i})$ 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. The soft-thresholding applied to the IPAD can be interpreted as a "denoising" operation.

2.3. CAD Learning

In this paper, we propose to couple IPAD with a Clustering Analysis Dictionary (CAD) $\boldsymbol{\Omega}_C$ and its associated vector $\boldsymbol{\lambda}_C$ of soft-thresholds. The expectation is that $(\boldsymbol{\Omega}_C, \boldsymbol{\lambda}_C)$ can partition the input parameter space into pieces where the prediction can be easily performed.

A Gaussian Mixture Model (GMM) [18] will be learned using the input training data of the current layer. Within the learned GMM, the data belonging to some Gaussian models are with very low prediction error. Let us denote these

Gaussian models as low error Gaussian models (LEGM). Similarly, we denote the other Gaussian models as high error Gaussian models (HEGM). Each model will lead to one atom and $\boldsymbol{\Omega}_C$ contains K atoms in total. The CAD and threshold pair $(\boldsymbol{\Omega}_C, \boldsymbol{\lambda}_C)$ will be used to differentiate the data belonging to the HEGM from that belonging to the LEGMs. They assign non-zero coefficients to those in HEGM and zero coefficients to those of LEGMs. The intuition is that HEGM might be better at discovering latent features and so only patches that contain these features will have a large inner product with the atom related to the appropriate HEGM and will be preserved.

With the training data pair $(\boldsymbol{\mathcal{X}}_i, \boldsymbol{\mathcal{Y}})$, a linear model can be fitted using least squares $\boldsymbol{R} = \boldsymbol{\mathcal{Y}}\boldsymbol{\mathcal{X}}_i^T(\boldsymbol{\mathcal{X}}_i\boldsymbol{\mathcal{X}}_i^T)^{-1}$. The prediction error of a training data pair $\left(\boldsymbol{x}_j, \boldsymbol{y}_j\right)$ is defined as $e_j = \left\|\boldsymbol{y}_j - \boldsymbol{R}\boldsymbol{x}_j\right\|_2^2$.

GMM models the data as a mixture of Gaussians. Each model is a multivariate Gaussian $\mathcal{N}\left(\cdot|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)$ with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$ and is characterized by a model weight π_k:

$$p\left(\boldsymbol{x}_i|\boldsymbol{\theta}\right) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right), \tag{3}$$

where K is the number of Gaussian models in GMM, and $\boldsymbol{\theta} = \{\boldsymbol{\theta}_k\}_{k=1}^K$ with $\boldsymbol{\theta}_k = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ is the model parameters of the GMM with $\pi_k \in \mathbb{R}^+$ and $\sum_{k=1}^K \pi_k = 1$.

We assume that all Gaussian models are with zero mean, i.e. $\boldsymbol{\mu}_k = \boldsymbol{0}$. The reason is that the data distribution of the image patches is centered at the origin since we remove the mean from each patch. Imposing zero mean forces the learned Gaussian models to capture the symmetric distribution. The GMM is learned using the online Expectation Maximization (EM) algorithm which is able to access a larger number of training data and generalizes better when compared to the standard EM algorithm. At iteration t, a random batch of M data pairs is drawn from the training data matrix $\boldsymbol{\mathcal{X}}_i$ and is used to update the model parameters of GMM.

In the E-step, given the GMM parameter $\boldsymbol{\theta}^{(t-1)}$ of the previous iteration, the membership probability r_{ik} can be updated as:

$$r_{ik} = \frac{\pi_k^{(t-1)}\mathcal{N}(\boldsymbol{x}_i|\boldsymbol{0}, \boldsymbol{\Sigma}_k^{(t-1)})}{\sum_j \pi_j^{(t-1)}\mathcal{N}(\boldsymbol{x}_i|\boldsymbol{0}, \boldsymbol{\Sigma}_j^{(t-1)})}. \tag{4}$$

In the M-Step, the mixture weights and the covariance matrices are updated as a linear combination of the previous model parameter and the current iteration estimation:

$$\begin{cases} \pi_k^{(t)} = (1-\eta)\pi_k^{(t-1)} + \eta\frac{1}{M}\sum\limits_i r_{ik}, \\ \boldsymbol{\Sigma}_k^{(t)} = (1-\eta)\boldsymbol{\Sigma}_k^{(t-1)} + \eta\frac{\sum_i r_{ik}\boldsymbol{x}_i\boldsymbol{x}_i^T}{\sum\limits_i r_{ik}}, \end{cases} \tag{5}$$

where $\eta \in (0, 1]$ is the step size for the online EM algorithm, and M is the batch data size.

Let us denote with $\left(\boldsymbol{\mathcal{X}}_{(k)}, \boldsymbol{\mathcal{Y}}_{(k)}\right)$ the data pairs which belong to Gaussian model $\boldsymbol{\theta}_k$ and E_k the average prediction error for $\left(\boldsymbol{\mathcal{X}}_{(k)}, \boldsymbol{\mathcal{Y}}_{(k)}\right)$. Based on the definition, a LEGM is with a small E_k, while a HEGM is with a large E_k. A threshold τ is set to differentiate the data belonging to HEGM from those from LEGMs based on the average prediction error. Each HEGM (with $E_k > \tau$) will lead to a single dictionary atom and a threshold. When the number of HEGM is smaller than K, a new Gaussian model will be added into the current GMM such that K pairs of clustering atom and threshold can be obtained. The online EM algorithm stops when the model parameters converge.

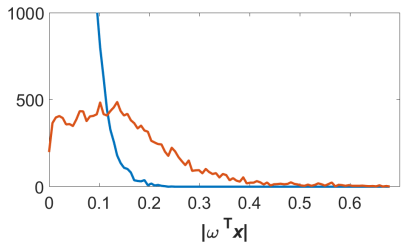


Fig. 2: The histogram of $\left|\boldsymbol{\omega}^T\boldsymbol{x}\right|$ for the data from LEGMs (blue) and from a HEGM (orange).

For the i^{th} HEGM $\boldsymbol{\theta}_k$, a pair of clustering atom $\boldsymbol{\omega}_{C,i}^T$ and threshold $\lambda_{C,i}$ will be learned and used to distinguish $\boldsymbol{\mathcal{X}}_{(k)}$ from the data from all LEGMs denoted as $\boldsymbol{\mathcal{X}}_{(s)}$ with $s = \{k|E_k \leq \tau\}$. Since the soft-thresholding operator is used for separation, we formulate the clustering atom learning problem as follows:

$$\begin{aligned} \boldsymbol{\omega}^T = \arg\max_{\boldsymbol{\omega}^T} & \sum_{\boldsymbol{x}_i \in \boldsymbol{\mathcal{X}}_{(k)}} \frac{|r_{ik}\boldsymbol{\omega}^T\boldsymbol{x}_i|}{R_k} - \sum_{\boldsymbol{x}_i \in \boldsymbol{\mathcal{X}}_{(s)}} \frac{|r_{js}\boldsymbol{\omega}^T\boldsymbol{x}_j|}{R_s}, \\ \text{s.t. } & \boldsymbol{\omega}^T\boldsymbol{\omega} = 1, \end{aligned} \tag{6}$$

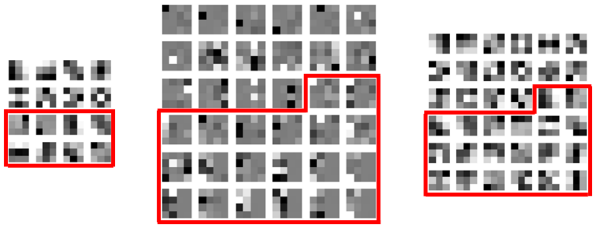
where $r_{js} = \sum_{k \in s} r_{jk}$, $R_k = \sum_i r_{ik}$, $R_s = \sum_j r_{js}$ and r_{ik} are the membership probabilities.

The constrained form of Eqn. (6) can be transformed into an unconstrained one by using Lagrangian multiplier. Although Eqn. (6) does not have a closed-form solution, it can be solved in an iterative manner by using a re-weighted formulation. At each iteration, an updated atom $\boldsymbol{\omega}^T$ is obtained by solving:

$$\boldsymbol{\omega}^T = \arg\max_{\boldsymbol{\omega}^T} \boldsymbol{\omega}^T \left(\hat{\boldsymbol{\Sigma}}_k - \boldsymbol{\Sigma}_s\right)\boldsymbol{\omega} - \zeta\left(\boldsymbol{\omega}^T\boldsymbol{\omega} - 1\right), \tag{7}$$

where $\hat{\boldsymbol{\Sigma}}_k = \sum\limits_{\boldsymbol{x}_i \in \boldsymbol{\mathcal{X}}_{(k)}} \frac{r_{ik}\boldsymbol{x}_i\boldsymbol{x}_i^T}{R_k\left|\boldsymbol{\omega}_0^T\boldsymbol{x}_i\right|}$, $\boldsymbol{\Sigma}_s = \sum\limits_{\boldsymbol{x}_j \in \boldsymbol{\mathcal{X}}_{(s)}} \frac{r_{js}\boldsymbol{x}_j\boldsymbol{x}_j^T}{R_s\left|\boldsymbol{\omega}_0^T\boldsymbol{x}_j\right|}$ with $\boldsymbol{\omega}_0^T$ being the previous iteration estimated $\boldsymbol{\omega}^T$, and ζ is the Lagrange multiplier.

The solution of Eqn. (7) is the eigenvector which corresponds to the largest eigenvalue of $\left(\hat{\boldsymbol{\Sigma}}_k - \boldsymbol{\Sigma}_s\right)$. The algorithm will converge in a few iterations. Figure 2 shows an example of the histogram of $\left|\boldsymbol{\omega}^T\boldsymbol{x}\right|$ using a learned clustering atom for data from LEGMs (blue) and from a HEGM (orange).



(a) $\mathbf{\Omega}_1$ (b) $\mathbf{\Omega}_2$ (c) $\mathbf{\Omega}_2\mathbf{\Omega}_1$

Fig. 3: Learned analysis dictionaries. Each atom is displayed as a 2D patch. The atoms within red box are clustering atoms.

A threshold value $\lambda_{C,i}$ will be learned and used to set a non-zero coefficient for a data that is from a HEGM and set zeros for other data. We formulate the threshold learning problem as follows:

$$\lambda = \arg\max_{\lambda} \frac{|S_{k,\lambda}|}{|S_{s,\lambda}| + \epsilon}, \tag{8}$$

where $S_{t,\lambda} = \left\{ \boldsymbol{x}_j \in \boldsymbol{\mathcal{X}}_{(k)} : \left| \boldsymbol{\omega}_{C,i}^T \boldsymbol{x}_j \right| > \lambda \right\}$ with $t = k$ or s, $|S|$ denotes the cardinally of a set S and ϵ is a small number.

A clustering analysis dictionary contains all the learned clustering atoms, i.e. $\mathbf{\Omega}_C = [\boldsymbol{\omega}_{C,1}, \cdots, \boldsymbol{\omega}_{C,K}]^T$. Similarly, the clustering threshold is denoted as $\boldsymbol{\lambda}_C = [\lambda_{C,1}, \cdots, \lambda_{C,K}]^T$. After $(\mathbf{\Omega}_C, \boldsymbol{\lambda}_C)$ pair has been learned, the threshold values will be fine-tuned to minimize the prediction error. The threshold updating is performed in an iterative manner by fixing all the other threshold values and optimizing one. A threshold value $\lambda_{C,i}$ is updated to the direction that can further minimize the prediction error. The threshold updating procedure stops when the prediction error can not be further reduced.

3. SIMULATION RESULTS

In this section, we report the simulation results of our proposed method and the comparison methods. The standard 91 training images [10] are applied for training and *Set 14* [11] is used for evaluation. The up-scaling factor is set to 2. The LR and HR patch size is 3×3 and 6×6, respectively. The input LR feature is the raw pixel values with removed mean.

The deep dictionary model is set to have $L = 4$ layers. The dictionary size for $\mathbf{\Omega}_1$, ..., $\mathbf{\Omega}_3$ and $\boldsymbol{D}$ is set to 16×9, 36×16, 144×36, and 36×144, respectively. The online EM algorithm uses step size $\eta = 0.1$, and batch size $M = 10^5$. The threshold τ is set to be the average prediction error of all training data. The number of clustering atoms within the analysis dictionary from layer 1 to 3 is set to be 8, 20, and 108, respectively. Figure 3 shows the learned analysis dictionaries $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$, and the effective dictionary $\mathbf{\Omega}_2\mathbf{\Omega}_1$. In $\mathbf{\Omega}_1$, some atoms are directional and the others are localized. The atoms in $\mathbf{\Omega}_2$ are relatively sparse. There are novel patterns emerging in the effective dictionary $\mathbf{\Omega}_2\mathbf{\Omega}_1$.

For comparison, DNNs with the same structure are learned using the same training data. The number of neurons in DNNs from layer 1 to 4 is 16, 36, 144, and 36, respectively. Let us denote DNN-R and DNN-S as the DNN with ReLU non-linearity and soft-thresholding non-linearity, respectively. The forward model of DNN-S is the same as our DDM method. The implementation is based on Pytorch with Adam optimizer, batch size 256, initial learning rate 0.01, learning rate decay step 100, and decay rate 0.1. The total number of epoch for training is 500.

Image	Bicubic	DNN-R	DNN-S	DDM
baboon	24.86	25.46	25.48	25.42
barbara	27.88	28.41	28.41	28.43
bridge	26.62	27.37	27.45	27.40
costguard	29.26	30.17	30.21	30.17
comic	24.63	27.28	27.45	27.19
face	34.73	35.33	35.42	35.37
flowers	30.20	31.72	31.97	31.73
foreman	35.21	37.36	38.11	37.56
lenna	34.57	35.87	36.04	35.86
man	29.16	30.16	30.29	30.15
monarch	32.77	35.12	35.67	35.25
pepper	34.98	36.23	36.50	36.28
ppt3	24.66	28.31	28.47	28.12
zebra	28.03	32.61	32.84	32.59
Average	**29.83**	**31.53**	**31.74**	**31.54**

Table 1: PSNR (dB) by different methods evaluated on *Set 14* [11].

Table 1 reports the PSNR (dB) of Bicubic interpolation method, DNN-R method, DNN-S method and the proposed DDM method evaluated on *Set 14* [11]. Our proposed DDM method achieves the similar average PSNR when compared to the DNN-R method. This validates the effectiveness of our proposed deep dictionary model and shows that the simultaneous information preserving and clustering idea could be a good interpretation of the workings of DNNs. The DNN-S method achieves the highest average PSNR which is around 0.2 dB higher than that of the DDN-R method and our proposed DDM method. This suggests that DNNs with soft-thresholding as non-linearity is more effective for image enhancement applications. The result also indicates that our DDM method can be further improved. In particular, an optimization strategy needs to be devised to determine the ratio between the number of information preserving atoms and the number of clustering atoms.

4. CONCLUSIONS

In this paper, we proposed a novel method to learn a pair of analysis dictionary and soft-threshold which is used to construct the deep dictionary model for single image super-resolution. Each analysis dictionary contains two sub-dictionaries: an information preserving analysis dictionary and a clustering dictionary. The learned analysis dictionaries and together with the corresponding soft-thresholds can simultaneously preserve important information from the previous layer as well as facilitate discrimination of key features. Simulation results show that our proposed deep dictionary model achieves comparable performance with DNNs.

5. REFERENCES

[1] 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.

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

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

[4] 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.

[5] 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.

[6] J. Sulam, V. Papyan, Y. Romano, and M. Elad, "Multilayer convolutional sparse modeling: Pursuit and dictionary learning," *arXiv preprint arXiv:1708.08705*, 2017.

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

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

[9] 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)*. Calgary, Canada: IEEE, March 2018.

[10] 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.

[11] 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.

[12] 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.

[13] 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*, 2016.

[14] 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.

[15] 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.

[16] 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*. Hawaii, USA: IEEE, July 2017.

[17] 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.

[18] K. P. Murphy, "Machine learning: a probabilistic perspective," 2012.