Deep Convolutional Dictionary Learning for Image Denoising

Hongyi Zheng\(^{a,b,*}\), Hongwei Yong\(^{a,b,*}\), Lei Zhang\(^{a,b,†}\)

\(^{a}\)The Hong Kong Polytechnic University \(^{b}\)DAMO Academy, Alibaba Group

\{cshzheng, cshyong, csizhang\}@comp.polyu.edu.hk

Abstract

Inspired by the great success of deep neural networks (DNNs), many unfolding methods have been proposed to integrate image modeling techniques, such as dictionary learning (DicL) and sparse coding, into DNNs for image restoration. However, the performance of such methods remains limited for several reasons. First, the unfolded architectures do not strictly follow the image representation model of DicL and lose the desired physical meaning. Second, handcrafted priors are still used in most unfolding methods without effectively utilizing the learning capability of DNNs. Third, a universal dictionary is learned to represent all images, reducing the model representation flexibility. We propose a novel framework of deep convolutional dictionary learning (DCDicL), which follows the representation model of DicL strictly, learns the prior meaning. DCDicL demonstrates leading denoising performance in various image restoration applications, such as denoising [30, 36, 71, 64].

The DicL model can be formulated as follows:

$$\min_{D, X} \frac{1}{2} \left| \left| DX - Y \right| \right|^2_2 + \lambda_X \phi(X) + \lambda_D \phi(D)$$

(1)

where \(Y \in \mathbb{R}^{m \times N}\) is a set of \(N\) training samples and each column of it is a stretched image patch vector; \(X \in \mathbb{R}^{d \times N}\) is the representation coefficient matrix of \(Y\) over dictionary \(D\); \(\psi(\cdot)\) and \(\phi(\cdot)\) denote the sparsity prior on coefficient \(X\) and the dictionary, respectively. The most widely used priors of \(\psi(\cdot)\) are sparsity priors, such as \(\|X\|_1\) and \(\|X\|_2\), and the corresponding DicL models are often called Sparse DicL. K-SVD [3, 71] is the most representative Sparse DicL method. It alternatively performs two steps to learn the dictionary: fix \(D\) and perform sparse coding (SC) to compute \(X\), and update \(D\) through singular value decomposition (SVD).

Inspired by K-SVD, many DicL methods have been proposed [36, 65, 14, 71, 27, 46, 45] and successfully used in various image restoration applications, such as denoising [16, 9] and super-resolution [63, 62, 61]. One problem of the patch-based DicL model in Eq. (1) is its lack of shift-invariant property, and convolutional dictionary learning (CDicL) [19] was proposed to address this issue by using the convolution operation to replace the matrix multiplication in signal representation. Specifically, the objective function of CDicL can be written as:

$$\min_{D, \{X_i\}} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \left| \left| D@X_i - Y_i \right| \right|^2_2 + \lambda_X \psi(X_i) + \lambda_D \phi(D)$$

(2)

where \(D@X_i = \sum_{c=1}^{C} D_c * X_{i,c} \ast\) is the 2D convolution operator, and \(C\) is the number of channels; \(D = \{D_c\}_{c=1}^{C}\) is the convolutional dictionary and \(D_c \in \mathbb{R}^{k \times k}\) is the \(c\)-th 2D dictionary atom (i.e., filter); \(X_i = \{X_{i,c}\}_{c=1}^{C}\) is the representation coefficient (also called feature map) of image \(Y_i \in \mathbb{R}^{h \times w}\) and \(X_{i,c} \in \mathbb{R}^{h \times w}\) is the \(c\)-th channel of \(X_i\).

1. Introduction

How to represent an image signal plays a key role in traditional image processing applications [9, 41, 42, 15, 14]. One popular approach is to represent an image patch vector \(y \in \mathbb{R}^m\) as a linear combination of atomic bases, i.e., \(y = Dx\), where \(D \in \mathbb{R}^{m \times d}\) is the dictionary of atoms, and \(x \in \mathbb{R}^d\) is the representation coefficient vector. In the early stage, cosine functions [4], wavelets [5] and contourlets [11] are commonly used as the dictionary atoms. However, such dictionaries are manually designed under some mathematical constraints and are not flexible enough to represent the complex natural image structures. Later on, researchers turned to learn the dictionary directly from image data, and many dictionary learning (DicL) methods have been developed [30, 36, 71, 64].
In CDicL, the sparse prior is commonly used for the feature map $X_i$ (e.g., $\|X_i\|_1$) and convolutional sparse coding (CSC) [8, 58] is used to solve the feature map. CDicL has demonstrated its advantages over patch-based DicL in several image processing tasks [34, 19, 21, 32].

With the rapid development of deep learning (DL) techniques in recent years, many deep neural network (DNN) based image restoration methods have been proposed [67, 69, 22, 13, 12]. Driven by a large amount of training data and the strong learning capacity of DNN, these methods have surpassed traditional image restoration methods, including those DicL based ones, by a large margin. Nonetheless, due to the black-box nature of DNN, there lacks a clear interpretation for its success in image restoration, while DicL has good interpretability. Therefore, researchers have attempted to integrate DicL, SC and DL for both good performance and clear physical meaning. These methods, often called deep unfolding methods, unfold the traditional SC and DicL models through certain algorithms, and parameterize the model by DNN in an end-to-end learning manner. Representative methods include DKSVD [47], Learned-CSC [52], CSCNet [50], DCSC [18], etc.

However, the existing deep unfolding methods usually fail to compete with DL methods for several reasons. First, the unfolded architectures do not strictly follow the original DicL models, which impairs the physical meaning and sacrifices the advantages of DicL. Second, most of them [52, 50, 18] still use the handcrafted priors, e.g., $L_1$ (sparsity) prior, instead of learning the priors from data, wasting the learning capacity of DNN architectures. Third, they usually learn a universal dictionary for all images, reducing the model’s representation capability. In this work, we propose a new unfolding framework, called deep convolutional dictionary learning (DCDicL), which resolves the above issues of previous unfolding methods. The contributions of this paper are summarized as follows:

- DCDicL learns the priors for both dictionary and representation coefficients from the training data, overcoming the disadvantages of handcrafted priors.
- DCDicL learn’s a specific dictionary for each image, which is adaptive to the image content. This endows DCDicL with more powerful capability for recovering image subtle structures.
- To testify the effectiveness of our framework, we apply DCDicL on the image denoising task. It achieves leading denoising performance over not only previous unfolding methods but also DL methods.

2. Related Works

2.1. Dictionary learning

Dictionary learning (DicL) is an important image modeling and representation learning approach and it has been widely studied in image restoration [63, 16, 19, 21, 32]. DicL aims to optimize a dictionary of atoms for representing the signal with handcrafted priors such as the sparsity prior on representation coefficients. In the seminal work of K-SVD [3, 71], the dictionary is optimized alternatively in two steps. The SC step employs the greedy orthogonal matching pursuit method to estimate the coefficients with $L_0$ constraint, while the singular value decomposition is used in the second step to update the dictionary. Many methods have been proposed to improve K-SVD [36, 65, 14, 71, 27, 46, 45]. For example, Mairal et al. [36] extended K-SVD to color image restoration. Zhang et al. [65] used group sparsity to make the learned dictionary more structured. Dong et al. [14] introduced the non-local self-similarity prior into DicL for image restoration.

DicL is a patch-based image modeling method and it lacks the shift-invariant property. Convolution dictionary learning (CDicL) [19] was proposed to address this issue. It replaces the dictionary atoms with a set of filters and reconstructs the original image by convolutional operation instead of matrix multiplication. The sparsity priors are imposed on the convolution feature maps, which can be solved by CSC [8, 58]. CDicL takes advantage of shift-invariant property and exploits better the image global information, exhibiting better performance than patch-based DicL in various image restoration applications [34, 19, 21, 21, 32].

2.2. Deep learning

The great success of deep learning (DL) in image recognition [31, 51, 23] facilitates its application to image restoration and enhancement tasks. Mao et al. [37] proposed a residual encoder-decoder network for image restoration. Dong et al. [12] proposed to use a 3-layer convolutional neural network (CNN), called SRCNN, for single image super-resolution (SISR). With the rapid development of deep neural network (DNN) training techniques, in [28] a 20-layer CNN, namely VDSR, was trained, which outperforms significantly traditional SISR methods. Zhang et al. [67] proposed the DnCNN model, which is a milestone of image denoising. The FFDNet [69] was developed for fast and flexible image denoising with multiple noise levels. Tai et al. [53] proposed the persistent memory network (MemNet) for image denoising. Jia et al. [26] proposed FOCNet, which solves a fractional optimal control problem for image denoising. N3Net [40], NLRN [33], RNAN [72] adopt the non-local modules to exploit the non-local image prior for noise removal.

Generally speaking, the above DNN models act as an implicit regularizer and learn the image priors from training data, surpassing the handcrafted priors used in traditional methods by a large margin. Nonetheless, most of the DNN based image restoration methods lack good interpretability.
### 2.3. Deep unfolding

Deep unfolding methods attempt to integrate the merits of model-based SC and DiC-L methods (e.g., good interpretability) and the merits of DL (e.g., strong learning capability). They unfold certain optimization algorithms, such as iterative shrinkage-threshold [6, 50, 18], alternating direction method of multipliers [7], half-quadratic splitting [2, 66] and primal-dual [1], parameterize the unfolded model, and update the learnable parameters by DNN. For the deep unfolding of DiC-L, the iterative shrinkage-threshold algorithm (ISTA) is usually used in the unfolding process. The DKSVD [47] model first replaces the $L_0$ prior in K-SVD by $L_1$ prior, then unfolds the SC process by ISTA iterations. Finally, it parameterizes the dictionaries by multi-layer perceptron (MLP) modules. The Learned-CSC [52] unfolds the CSC process by ISTA, which can be integrated with DNN. Simon et al. [50] improved the Learned-CSC through strided convolution. Fu et al. [18] introduced a multi-scale feature extraction module before ISTA unfolding, and applied it to JPEG artifacts removal.

Though improving the interpretability of DL, current deep unfolding methods have some problems and their performances still lag behind DL based methods. For example, the unfolded structures mismatch the original DiC-L model, fixed priors are adopted without fully utilizing the learning capability of DNN, and the learned dictionary is universal but not image adaptive. We will discuss these issues in Section 3 and present a new framework of deep unfolding.

### 3. Methodology

#### 3.1. Problems of current deep unfolding methods

Most of the deep unfolding methods for DiC-L [47] and CDic-L [52, 50, 18] assume $L_1$ prior on coefficient $X$. Without loss of generality, we focus on the unfolding of CDic-L in the following discussion. The objective function is a special case of Eq. (2), which can be written as:

$$
\min_D \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \left\| D \odot X_i - Y_i \right\|_2^2 + \lambda_X \left\| X_i \right\|_1
$$

With a set of $N$ training sample pairs $\{X_i, Y_i\}$, where $Y_i$ is the clean ground truth of noisy image $Y_i$. Eq. (3) can be formulated as a bi-level optimization problem:

$$
\min_D \frac{1}{N} \sum_{i=1}^{N} L(D \odot X_i, Y_i) \quad (4a)
$$

s.t. $X_i = \arg\min_{X} \frac{1}{2 \sigma_i^2} \left\| D \odot X - Y_i \right\|_2^2 + \lambda_X \left\| X \right\|_1 \quad (4b)$

where $L(\cdot, \cdot)$ measures the loss and $\sigma_i$ is the noise level of image $Y_i$. Eq. (4b) can be solved by ISTA [10] iteratively:

$$
X = S_{\lambda, \eta} \left( X - \frac{1}{\eta} \text{rot} 180(D) \odot (D \odot X - Y_i) \right)
$$

where $S$ is the shrinkage function, $\lambda$ and $\eta$ are hyper-parameters, and rot$180(D)$ rotates $D$ by 180°. Existing deep unfolding methods [47, 52, 50, 18] parameterize rot$180(D)$ and $D$ by two DNN modules to build the architecture. In the forward pass, Eq. (4b) is solved to estimate $X$. In the backward pass, $L(\cdot, \cdot)$ is calculated to update dictionary $D$ as weight matrices of convolutional layers.

The above unfolding scheme, however, has some inherent problems as listed below:

- First, rot$180(D)$ and $D$ are parameterized as two independent Conv modules, violating the mathematical constraint (i.e., rot$180(D)$ and $D$ are the rotation of each other) and losing the physical meaning of CDic-L.
- Second, the learning capability of DNN is misused. In Eq. (3), given the coefficient $X$ and the signal $Y_i$, the dictionary $D$ can be solved explicitly and there is no need to update them using DNN. On the other hand, the DNN should be used to learn the complex priors of $X$, whereas the handcrafted sparsity priors (e.g., $L_1$-sparsity) are used, which is far less effective.
- Third, a universal dictionary $D$ is parameterized to represent all images in the existing unfolding scheme, which impairs the flexibility of image representation.

The above three problems of deep CDic-L unfolding limits its performance in image restoration, lagging behind those standard DNN based methods.

#### 3.2. Deep convolutional dictionary learning

To solve the limitations of current deep CDic-L unfolding methods, we propose a new deep unfolding framework, namely DCDic-L, which can effectively integrate the advantages of CDic-L and DL. Instead of unfolding the objective function in Eq. (3), which enforces handcrafted priors on $X$ and employs a universal $D$, we unfold the general objective function in Eq. (2), employ adaptive $D$ on each image, and learn the deep priors of both $X$ and $D$ from data. The learning model of DCDic-L can be written as:

$$
\min_{\{D_i, X_i\}} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \left\| D_i \odot X_i - Y_i \right\|_2^2 + \lambda_X \psi(X_i) + \lambda_D \phi(D_i)
$$

(6)

We rewrite Eq. (6) as the bi-level optimization problem:

$$
\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(D_i \odot X_i, Y_i^{gt}) \quad (7a)
$$

s.t. $\{D_i, X_i\} = \arg\min_{D, X} \frac{1}{2 \sigma_i^2} \left\| D \odot X - Y_i \right\|_2^2 + \lambda_X \psi(X) + \lambda_D \phi(D) \quad (7b)$

where $\theta$ is the learnable parameters.

The architecture of DCDic-L can be derived by unfolding the inner objective in Eq. (7b). For the convenience of expression, we omit the subscript “i” in the following development. To separate the data term and prior term, we introduce two auxiliary variables $X'$ and $D'$, and solve the following two objective functions:

$$
\min_{X, X'} \frac{1}{2 \sigma_i^2} \left\| D \odot X' - Y \right\|_2^2 + \lambda_X \psi(X), \text{ s.t. } X = X' \quad (8a)
$$

$$
\min_{D, D'} \frac{1}{2 \sigma_i^2} \left\| D' \odot X - Y \right\|_2^2 + \lambda_D \phi(D), \text{ s.t. } D = D' \quad (8b)
$$
According to the Half Quadratic Splitting (HQS) algorithm [24], solving Eq. (8) is equivalent to minimizing the following objective functions:

\[
\begin{align*}
\min_{X,Y} & \frac{1}{2\sigma^2} \|D \odot (X' - Y)\|_2^2 + \lambda_X \psi(X) + \frac{\mu_X}{2} \|X - X'\|_2^2 \\
\min_{D'Y} & \frac{1}{2\sigma^2} \|D' \odot (X' - Y)\|_2^2 + \lambda_D \phi(D') + \frac{\mu_D}{2} \|D - D'\|_2^2
\end{align*}
\]

where \(\mu_D\) and \(\mu_X\) are the hyper-parameters for \(D\) and \(X\), respectively. When \(\mu_D\) and \(\mu_X\) are large enough, the constraint in Eq. (8) can be met.

Eq. (9) can be solved iteratively. In the \(t\)-th iteration (stage), \(X'_t\), \(X_t\), \(D'_t\), \(D_t\) are solved as follows:

\[
\begin{align*}
X'_t &= \text{Solve}_X(Y, D_{t-1}, X_{t-1}, \alpha_X) \\
&= \arg\min_{X'} \frac{1}{2} \|D_{t-1} \odot X' - Y\|_2^2 + \frac{\alpha_X}{2} \|X' - X_{t-1}\|_2^2 \\
X_t &= \text{Net}_X(X'_t, \beta_X) \\
&= \arg\min_{X} \psi(X') + \frac{\beta_X}{2} \|X'_t - X_t\|_2^2 \\
D'_t &= \text{Solve}_D(Y, D_{t-1}, X_t, \alpha_D) \\
&= \arg\min_{D'} \frac{1}{2} \|D_{t-1} \odot X_t - Y\|_2^2 + \frac{\alpha_D}{2} \|D' - D_{t-1}\|_2^2 \\
D_t &= \text{Net}_D(D'_t, \beta_D) \\
&= \arg\min_{D} \phi(D') + \frac{\beta_D}{2} \|D'_t - D_t\|_2^2
\end{align*}
\]

where \(\{\alpha_X, \alpha_D, \beta_X, \beta_D\} = \{\mu_X, \mu_D, \mu_X, \mu_D\}\). For data term subproblems in Eqs. (10a) and (10c), we have closed-form fast solutions, which will be presented in Sections 3.3 and 3.4. For prior term subproblems in Eqs. (10b) and (10d), we learn two DNNs to solve them. For hyper-parameters \(\{\alpha_X, \alpha_D, \beta_X, \beta_D\}\), we use a simple network (HyperNet) with input \(\sigma\) to predict them for each stage.

The unfolding process of DCDicL is depicted in Algorithm 1. We use a simple network (HeadNet) to get an initial estimation for \(X_0\) and simply use the zero initialization for \(D_0\). The overall architecture of DCDicL is illustrated in Fig. 1. During the forward pass, DCDicL solves Eq. (7a) by addressing the four subproblems in Eq. (10) iteratively, and obtains the final estimations of \(X\) and \(D\) for each image adaptively. While during the backward pass, DCDicL calculates \(L(D \odot X, Y^p)\) and solves Eq. (7a), via which the image priors are updated from training data. The details of the network design will be presented in Section 3.5.

### 3.3. Solving X

For the convenience of expression, we omit the subscript \((t)\) in the following development. The closed-form solution of \(X'\) can be derived by solving Eq. (10a). According to [8], Eq. (10a) can be efficiently solved using Fast Fourier Transform (FFT). Denote by \(\mathcal{F}(\cdot)\) the 2D FFT, and let \(D = \mathcal{F}(D), X^* = \mathcal{F}(X^*), Y = \mathcal{F}(Y)\) and \(X = \mathcal{F}(X)\). Taking the derivative of Eq. (10a) w.r.t. \(X^*\), letting the derivative be zero and using the Sherman-Morrison formula [49], we have the following closed-form solution:

\[
X' = \frac{1}{\alpha_X} \mathcal{F}^{-1} \left\{ \mathcal{Z} - \mathcal{D} \mathcal{C} \left( \frac{\mathcal{D} \odot \mathcal{Z}}{\alpha_X + (\mathcal{D} \odot \mathcal{D})^\top} \right) \right\}
\]

where \(\mathcal{Z} = \mathcal{D} \mathcal{C} (Y^p \mathcal{C}) + \alpha_X \mathcal{X}\), \(\mathcal{F}^{-1}(\cdot)\) denotes the inverse FFT, \(\mathcal{D}\) denotes complex conjugate of \(\mathcal{D}\), \(\mathcal{C}\) is the Hadamard product, \(\mathcal{A} \circ \mathcal{B} = \sum_{c=1}^{C} A_c B_c\), \(\mathcal{A}^\top\) expands the channel dimension of \(A\) to \(C\), and \(\mathcal{X}\) is the Hadamard product.
division. The detailed derivation can be found in the supplementary file.

3.4. Solving $D$

As in practice the dimension of $D'$ is much lower than that of $X$, the system is overdetermined and can be solved by least squares method. To utilize the modern least squares solvers, we unfold Eq. (10c) from the form of convolution into the form of matrix multiplication. The details can be found in the supplementary file. Here, we use $\mathbf{x}$ and $\mathbf{y}$ to denote the unfolded results of original matrices $X$ and $Y$. The objective function in Eq. (10c) can be re-written as:

$$\arg\min d^* \frac{1}{2} \| \mathbf{x} d - \mathbf{y} \|_2^2 + \frac{\alpha}{2} \| \mathbf{d}^* - \mathbf{d} \|_2^2$$

where $d^* = \text{vec}(D^*)$, $d = \text{vec}(D)$, and $\text{vec}(\cdot)$ is the vectorization operator.

By taking the derivative of the above objective function w.r.t. $d^*$ and letting the derivative be zero, we can obtain the closed-form solution of $D'$:

$$D' = \text{vec}^{-1} \left( \mathbf{x}^T \mathbf{x} + \alpha_d I \right)^{-1} \left( \mathbf{x}^T \mathbf{y} + \alpha_d \mathbf{d} \right)$$

where $\text{vec}^{-1}(\cdot)$ reverses the vectorization. Eq. (13) can be efficiently solved by modern least square solvers such as LU solver provided by PyTorch. Notice that unfolding $X$ to $\mathbf{x}$ would increase the memory overhead by $k^2$, which is not desirable in practice. Fortunately, we can efficiently compute $\mathbf{x}^T \mathbf{x}$ and $\mathbf{x}^T \mathbf{y}$ from $X$ and $Y$ without explicitly storing $\mathbf{x}$. The details are in the supplementary file.

3.5. Network design

As shown in Fig. 1, our proposed DCDicL framework has four sub-networks, including HeadNet, $Net_X$, $Net_D$ and HypaNet, whose architectures are illustrated in Fig. 2.

HeadNet takes the noisy image $Y$ and noise level $\sigma$ as input to initialize coefficient $X_0$. It consists of 2 Conv layers (64 channels each layer) with ReLU activation.

$Net_X$ learns the prior on coefficients $X$. It acts as an implicit regularizer as those end-to-end networks in DL based denoising methods [67, 69, 26, 40]. We adopt the U-Net [43] architecture for its effectiveness in image restoration tasks [66, 73, 57]. Specifically, our $Net_X$ consists of 7 blocks. The first 3 blocks down-sample the feature maps through strided convolution, and the last 3 blocks up-sample the feature maps by transposed convolution. Each block consists of several residual units, while each residual unit consists of 2 Conv layers with ReLU activation and a skip connection. The Conv layers in the first 4 blocks have 64, 128, 256 and 512 channels, respectively. The selection of the number of residual units is discussed in section 4.2.

$Net_D$ learns the prior on dictionary $D$. As $D$ has a much smaller spatial size than $X$, a shallower network is enough to provide sufficient receptive field and learning capability. Our $Net_D$ consists of 6 Conv layers with ReLU activation, and there is no ReLU after the last Conv layer. Each Conv layer has 16 channels, and there is a skip connection between the first and the last units.

HypaNet takes noise level $\sigma$ as input and predicts the hyperparameters for each stage. It consists of 2 Conv layers (kernel size 1) and a SoftPlus layer, ensuring all hyperparameters are positive.

4. Experiments

4.1. Training details

We follow [39] to use the combination of WED [35], DIV2K training set [55] and BSD400 [38, 67], for training. The noisy image $Y$ is obtained by adding additive white Gaussian noise of standard deviation $\sigma$ on the ground truth image $Y^gt$. Patches of size $128 \times 128$ are randomly cropped from $Y^gt$ and $Y$ image pairs for training.

$L_1$ loss is used as the loss function $L(\cdot, \cdot)$ on the output of each stage. As in previous multi-stage learning work [60], we set the weight of the loss imposed on the last stage as 1, and set the weight as $\frac{1}{n - 1}$ for all the other $T - 1$ stages. The Adam optimizer [29] is used for updating the learnable parameters. The batch size is 32 and we train it for 1e6 iterations. The learning rate starts from $1e^{-4}$ and decays by a factor of 0.5 for every 2e5 iterations. In order to speed up and stabilize the training, we first train a 1-stage model and reload its weights into the $T$-stage model for fine-tuning. (The selection of $T$ is discussed in Section 4.2.) All the $T$ stages share the same parameters and we train a shared model for all noise levels, which are set to $\{15, 25, 50\}$ as in [67, 26, 69]. The number of atoms in $D$ is determined by the number of feature maps in $Net_X$ (i.e., 64), while the spatial size of each atom is discussed in Section 4.2.
Table 1: Grayscale image denoising results in PSNR(dB)/SSIM(\%). “-” means that the result is not available.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>(\sigma)</th>
<th>BM3D</th>
<th>WNNM</th>
<th>DnCNN</th>
<th>N3Net</th>
<th>NLRN</th>
<th>RNAN</th>
<th>FOCNet</th>
<th>IRCNN</th>
<th>FFDNet</th>
<th>DKSVD</th>
<th>CSCNet</th>
<th>DCDicL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set12</td>
<td>15</td>
<td>32.3</td>
<td>89.52</td>
<td>32.7</td>
<td>90.89</td>
<td>32.8</td>
<td>90.24</td>
<td>-</td>
<td>33.1</td>
<td>90.70</td>
<td>-</td>
<td>33.0</td>
<td>76.90</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>29.9</td>
<td>88.04</td>
<td>30.2</td>
<td>85.57</td>
<td>30.4</td>
<td>86.17</td>
<td>30.5</td>
<td>86.56</td>
<td>30.8</td>
<td>86.89</td>
<td>-</td>
<td>30.7</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>26.7</td>
<td>26.76</td>
<td>26.5</td>
<td>76.76</td>
<td>26.7</td>
<td>18.98</td>
<td>27.4</td>
<td>43.79</td>
<td>31.6</td>
<td>78.09</td>
<td>27.7</td>
<td>68.0</td>
</tr>
<tr>
<td>BSD68</td>
<td>15</td>
<td>31.0</td>
<td>78.17</td>
<td>31.3</td>
<td>78.88</td>
<td>31.0</td>
<td>78.87</td>
<td>29.2</td>
<td>93.92</td>
<td>29.4</td>
<td>91.33</td>
<td>29.6</td>
<td>41.33</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>28.5</td>
<td>76.00</td>
<td>28.8</td>
<td>80.87</td>
<td>26.2</td>
<td>37.81</td>
<td>26.3</td>
<td>39.73</td>
<td>26.4</td>
<td>47.92</td>
<td>26.4</td>
<td>47.92</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>25.6</td>
<td>65.86</td>
<td>25.7</td>
<td>65.89</td>
<td>25.6</td>
<td>65.86</td>
<td>25.6</td>
<td>65.89</td>
<td>25.6</td>
<td>65.86</td>
<td>25.6</td>
<td>65.86</td>
</tr>
<tr>
<td>Urban100</td>
<td>15</td>
<td>32.5</td>
<td>92.20</td>
<td>32.9</td>
<td>92.72</td>
<td>32.6</td>
<td>92.46</td>
<td>-</td>
<td>33.5</td>
<td>94.35</td>
<td>-</td>
<td>33.5</td>
<td>94.35</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>29.7</td>
<td>87.77</td>
<td>30.3</td>
<td>98.85</td>
<td>29.7</td>
<td>87.77</td>
<td>30.9</td>
<td>98.85</td>
<td>30.9</td>
<td>98.85</td>
<td>30.9</td>
<td>98.85</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>25.9</td>
<td>77.91</td>
<td>26.5</td>
<td>80.47</td>
<td>26.1</td>
<td>78.56</td>
<td>26.2</td>
<td>81.48</td>
<td>26.7</td>
<td>83.32</td>
<td>27.0</td>
<td>40.2</td>
</tr>
</tbody>
</table>

Figure 3: Ablation studies on (a) \(k\); (b) \(n_r\) and \(T\).

4.2. Ablation studies

In DCDicL, the hyperparameters can be end-to-end learned by HypaNet. In this section, we perform ablation studies on the selection of spatial size \(k\) of \(D\), the no. of stages \(T\), and the no. of residual units, denoted by \(n_r\). We perform the ablation study on the Set12 dataset with noise level \(\sigma=25\). The results are illustrated in Fig. 3.

Selection of \(k\). We select \(k\) among \{3, 5, 7\}. It can be seen that the PSNR index rises with the increase of \(k\); however, the improvement becomes minor when \(k=7\). On the other hand, the inference time boosts when \(k=7\). To balance the performance and efficiency, we set \(k=5\) in the experiment.

Selection of \(T\) and \(n_r\). The depth of our DCDicL model can be adjusted by 2 key factors. The first factor is the no. of unrolling stages \(T\). The second factor is the size of subnetworks. Since NetD, HeadNet and HypaNet are much smaller than NetX, we can simply control the model size by adjusting the no. of residual units \(n_r\), which consists of 2 Conv layers, in each block of NetX.

It can be seen that the PSNR index rises with the increases of both \(n_r\) and \(T\). However, the improvement becomes minor when \(n_r=6\) and \(T=6\). To guarantee the efficiency, we set \(n_r=4\) and \(T=4\) in the experiment.

4.3. Comparison with state of the arts

In this section, we compare DCDicL with state of the art image denoising methods. Since some competing methods only provide the codes or results for grayscale images or color images, we compare DCDicL with different methods on different datasets. Since the sizes of testing images are often different from that of training patches (i.e., \(128 \times 128\)), the regularization strength on \(D\) in Eq. (10c) should be different. Hence, in testing we scale \(\alpha_D\) by \(\frac{h_{test}^2}{h_{test}^2 + w_{test}^2}\) to normalize the regularization strength, where \(h_{test}\) and \(w_{test}\) are the spatial sizes of testing image.

For grayscale image denoising, we adopt 3 widely used testing datasets, including Set12, BSD68 [44] and Urban100 [25], in the experiments. We compare the proposed DCDicL with representative model-based methods (i.e., BM3D [9], WNNM [20], DL based methods (i.e., DnCNN [67], N3Net [40], NLRN [33], RNAN [72], FOCNet [26], IRCNN [68], FFDNet [69]), and deep unfolding methods (DKSVD [47], CSCNet [50]). The experimental results are shown in Table 1. For color image denoising methods (DKSVD [47], CSCNet [50]), “-” means that the result is not available.
noising, four widely used color image datasets, including CBSD68 [44], Kodak24 [17], McMaster [70] and Urban100 are used here. We compare DCDicL with model-based method (i.e., CBM3D), DL based methods (i.e., DnCNN, IRCNN, FFDNet, RNAN, RPCNN [59], BRDNet [54], DSNet [39]), and deep unfolding method (i.e., CSCNet). The experimental results are shown in Table 2.

It can be seen that DCDicL achieves the best PSNR and
DCCidL performs extraordinary well on the Urban100 dataset, whose images contain lots of fine-scale repetitive structures and textures. It surpasses the second best method by 0.59dB in grayscale image denoising and 0.8dB in color image denoising even when $\sigma=50$. This is because, by solving the dictionary $D$ in Eq. (10e), DCCidL perceives the global information from input image $Y$ and exploits the image self-similarity adaptively and effectively. Figs. 4, 5 and 6 show the denoising results on images from Set12, McMaster and CSBD68. It can be seen that DCCidL restores many subtle edges and textures that cannot be restored by other competing methods. More visualizations can be found in the supplementary file.

We further compare the inference time of DCCidL and competing methods. All experiments are done on Set12 ($\sigma=50$) with a GTX 1080Ti GPU. As can be seen from Fig. 7, DCCidL is slower than DnCNN, FFDNet, IRCNN and DKSVD, but achieves much higher denoising performance. RNAN, N3Net and NLRN adopt complex non-local modules for improving the performance, but still fail to compete with DCCidL in terms of both denoising performance and inference speed. Overall, we can conclude that DCCidL provides a good solution in terms of both effectiveness and efficiency.

### Analysis on adaptive dictionaries

To demonstrate the effectiveness of the adaptive dictionaries learned by DCCidL, we replace the NetD in DCCidL with a universal dictionary (i.e., weight matrix in a Conv layer) to train a new (vanilla U-Net) model, called DCCidL-U. We compare DCCidL and DCCidL-U on color image denoising. The denoising results (PSNR/SSIM) are shown in Table 3, and Fig. 8 shows an example. One can see that the adaptive dictionaries learned by NetD improve the denoising performance significantly. DCCidL recovers many image fine textures, which are lost by the universal dictionary. Fig. 9 visualizes the dictionaries learned on two images. It can be seen that DCCidL can adaptively adjust the dictionary according to the content of input image. More visualizations on the dictionaries and denoising results can be found in the supplementary file.

### 5. Conclusion

We proposed a new deep unfolding method, namely deep convolutional dictionary learning (DCCidL), and validated its effectiveness on image denoising. Following the mathematical modeling of dictionary learning strictly, DCCidL learns the priors on $X$ and $D$ from training data, and learns a specific dictionary for each image. The dictionary is not only adaptive to image content but also perceives image global information, endowing DCCidL with strong capability to recover subtle image structures even under severe noise. Extensive experiments on benchmark datasets demonstrated that DCCidL surpasses previous deep unfolding and deep learning based methods in terms of both quantitative metrics and visual quality. DCCidL provides new insight on deep image modeling, and can be extended to more image restoration tasks.
References


