

# Image Restoration Using Convolutional Auto-encoders with Symmetric Skip Connections

Xiao-Jiao Mao, Chunhua Shen, Yu-Bin Yang

**Abstract**—Image restoration, including image denoising, super resolution, inpainting, and so on, is a well-studied problem in computer vision and image processing, as well as a test bed for low-level image modeling algorithms. In this work, we propose a very deep fully convolutional auto-encoder network for image restoration, which is a encoding-decoding framework with symmetric convolutional-deconvolutional layers. In other words, the network is composed of multiple layers of convolution and de-convolution operators, learning end-to-end mappings from corrupted images to the original ones. The convolutional layers capture the abstraction of image contents while eliminating corruptions. Deconvolutional layers have the capability to upsample the feature maps and recover the image details. To deal with the problem that deeper networks tend to be more difficult to train, we propose to symmetrically link convolutional and deconvolutional layers with skip-layer connections, with which the training converges much faster and attains better results. The skip connections from convolutional layers to their mirrored corresponding deconvolutional layers exhibit two main advantages. First, they allow the signal to be back-propagated to bottom layers directly, and thus tackles the problem of gradient vanishing, making training deep networks easier and achieving restoration performance gains consequently. Second, these skip connections pass image details from convolutional layers to deconvolutional layers, which is beneficial in recovering the clean image. Significantly, with the large capacity, we show it is possible to cope with different levels of corruptions using a single model. Using the same framework, we train models on tasks of image denoising, super resolution removing JPEG compression artifacts, non-blind image deblurring and image inpainting. Our experiment results on benchmark datasets show that our network can achieve best reported performance on all of the four tasks, and set new state-of-the-art.

**Index Terms**—Image restoration, auto-encoder, convolutional/de-convolutional networks, skip connection, image denoising, super resolution, image inpainting.

## CONTENTS

<b>1</b>	<b>Introduction</b>	2
<b>2</b>	<b>Related work</b>	2
<b>3</b>	<b>Very deep convolutional auto-encoder for image restoration</b>	3
3.1	Architecture . . . . .	3
3.2	Deconvolution decoder . . . . .	3
3.3	Skip connections . . . . .	4
3.4	Training . . . . .	5
3.5	Testing . . . . .	6
<b>4</b>	<b>Discussions</b>	6
4.1	Analysis on the architecture . . . . .	6
4.2	Gradient back-propagation . . . . .	6
4.3	Training with symmetric skip connections . . . . .	6
4.4	Comparison with the deep residual net-work [1] . . . . .	7
4.5	Testing efficiency . . . . .	7

<b>5</b>	<b>Experiments</b>	8
5.1	Network parameters . . . . .	8
5.2	Image denoising . . . . .	9
5.3	Image super-resolution . . . . .	10
5.4	JPEG deblocking . . . . .	12
5.5	Non-blind deblurring . . . . .	13
5.6	Image inpainting . . . . .	13

<b>6</b>	<b>Conclusions</b>	14
----------	--------------------	----

<b>References</b>	14
-------------------	----

- X.-J. Mao and Y.-B. Yang are with the State Key Laboratory for Novel Software Technology, Nanjing University, China. E-mail: xjmgl.nju@gmail.com, yangyubin@nju.edu.cn.
- C. Shen is with the School of Computer Science, University of Adelaide, Australia. E-mail: chunhua.shen@adelaide.edu.au.
- X.-J. Mao's contribution was made when visiting University of Adelaide. Correspondence should be addressed to C. Shen.

## 1 INTRODUCTION

Image restoration [2], [3], [4], [5], [6] is a classical problem in low-level vision, which has been widely studies in the literature. Yet, it remains an active research topic and provides a test bed for many image modeling techniques.

Generally speaking, image restoration is the operation of taking a corrupted image and estimating the original image, which is known to be an ill-posed inverse problem. A corrupted image  $Y$  can be represented as

$$y = H(x) + n \quad (1)$$

where  $x$  is the clean version of  $y$ ;  $H$  is the degradation function and  $n$  is the additive noise. By accommodating different types of degradation operators and noise distributions, the same mathematical model applies to most low-level imaging problems such as image denoising [7], [8], [9], [10], super-resolution [11], [12], [13], [14], [15], [16], [17], inpainting [18], [19], [20] and recovering raw images from compressed images [21], [22], [23]. In the past decades, extensive studies have been carried out to develop various of image restoration methods.

Recently, deep neural networks (DNNs) have shown their superior performance in image processing and computer vision tasks, ranging from high-level recognition, semantic segmentation to low-level denoising and super-resolution. One of the early deep learning models which has been used for image denoising is the Stacked Denoising Auto-encoders (SdA) [24]. It is an extension of the stacked auto-encoder [25] and was originally designed for unsupervised feature learning. Denoising auto-encoders can be stacked to form a deep network by feeding output of the previous layer to the current layer as input. Jain and Seung [26] proposed to use Convolutional Neural Networks (CNN) to denoise natural images. Their framework is the same as the recent Fully Convolutional Neural Networks (FCN) for semantic image segmentation [27] and other tasks such as super-resolution [28], although their network is not as deep as today's models. Their network accepts an image as the input and produces an entire image as the output through four hidden layers of convolutional filters. The weights are learned by minimizing the difference between the output and the clean image.

By observing recent superior performance of CNN on image processing tasks, here we propose a very deep fully convolutional CNN-based framework for image restoration. The input of our framework is a corrupted image, and the output is its clean version. We observe that it is beneficial to train a very deep model for low-level tasks like denoising, super-resolution and JPEG deblocking. Our network is much deeper compared to that in [26] and recent low-level image processing models such as [28]. Instead of using image priors, the proposed framework learns fully convolutional and deconvolutional mappings from corrupted images to the clean ones in an end-to-end fashion. The network is composed of multiple layers of convolution and deconvolution operators. As deeper networks tend to be more difficult to train, we further propose to symmetrically link convolutional and deconvolutional layers with multiple skip-layer connections, with which the training converges much faster and better performance is achieved.

Our main contributions can be summarized as follows.

- We propose a very deep network architecture for image restoration. The network consists of a chain of symmetric convolutional layers and deconvolutional layers. The convolutional layers act as the feature extractor which encode the primary components of image contents while eliminating the corruptions. The deconvolutional layers then decode the image abstraction to recover the image content details. To the best of our knowledge, the proposed framework is the first attempt to used both convolution and deconvolution for low-level image restoration.
- To better train the deep network, we propose to add skip connections between corresponding convolutional and deconvolutional layers. These shortcuts divide the network into several blocks. These skip connections help to back-propagate the gradients to bottom layers and pass image details to the top layers. These two characteristics make training of the end-to-end mapping from corrupted image to the clean one easier and more effective, and thus achieve performance improvement while the network going deeper.
- We apply the same network for tasks such as image denoising, image super-resolution, JPEG deblocking, non-blind image deblurring and image inpainting. Experiments on a few widely-used benchmark datasets demonstrate the advantages of our network over other recent state-of-the-art methods. Moreover, relying on the large capacity and fitting ability, our network can be trained to obtain good restoration performance on different levels of corruption even using a single model.

The remaining content is organized as follows. We provide a brief review of related work in Section 2. We present the architecture of the proposed network, as well as training, testing details in Section 3. In Section 4, we discuss some relevant issues. Experimental results and analysis are provided in Section 5.

## 2 RELATED WORK

In the past decades, extensive studies have been conducted to develop various image restoration methods. See detailed reviews in a survey [29]. Traditional methods such as the BM3D algorithm [30] and dictionary learning based methods [31], [16], [32] have shown very promising performance on image restoration topics such as image denoising and super-resolution. Due to the ill-posed nature of image restoration, image prior knowledge formulated as regularization techniques are widely used. Classic regularization models, such as total variation [33], [34], [35], are effective in removing noise artifacts, but also tend to over-smooth the images. As an alternative, sparse representation [36], [3], [37], [38], [39] based prior modeling is popular, too. Mathematically, the sparse representation model assumes that a data point  $x$  can be linearly reconstructed by an over-completed dictionary, and most of the coefficients are zero or close to zero.

An active (and probably more promising) category for image restoration is the neural network based methods. The most significant difference between neural network methods and other methods is that they typically learn parameters for image restoration directly from training data (e.g., pairs of clean and corrupted images) rather than relying on pre-defined image priors.

Stacked denoising auto-encoder [24] is one of the most well-known deep neural network models which can be used for image denoising. Unsupervised pre-training, which minimizes the reconstruction error with respect to inputs, is done for one layer at a time. Once all layers are pre-trained, the network goes through a fine-tuning stage. Xie et al. [18] combined sparse coding and deep networks pre-trained with denoising auto-encoder for low-level vision tasks such as image denoising and inpainting. The main idea is that the sparsity-inducing term for regularization is proposed for improved performance. Deep network cascade (DNC) [40] is a cascade of multiple stacked collaborative local auto-encoders for image super-resolution. High frequency texture enhanced image patches are fed into the network to suppress the noises and collaborate the compatibility of the overlapping patches.

Other neural network based image restoration methods using networks such as multi-layer perceptron. Early works, such as a multi-layer perceptron with a multilevel sigmoidal function [41], have been proposed and proved to be effective in image restoration tasks. Burger et al. [42] presented a patch-based algorithm learned on a large dataset with a plain multi-layer perceptron and is able to compete with the state-of-the-art traditional image denoising methods such as BM3D. They also concluded that with large networks, large training data, neural networks can achieve state-of-the-art image denoising performance, which is confirmed in the work here.

Compared to auto-encodes and multilayer perceptron, it seems that convolutional neural networks have achieved even more significant success in the field of image restoration. Jain and Seung [26] proposed fully convolutional CNN for denoising. The network is trained by minimizing the loss between a clean image and its corrupted version by adding noises on it. They found that CNN works well on both blind and non-blind image denoising, providing comparable or even superior performance to wavelet and Markov Random Field (MRF) methods. Recently, Dong et al. [28] proposed to directly learn an end-to-end mapping between the low/high-resolution images for image super-resolution. They observed that convolutional neural networks are essentially related to sparse coding based methods, i.e., the three layers in their network can be viewed as patch representation extractor, non-linear mapping and image reconstructor. They also proposed variant networks for other image restoration tasks such as JPEG debloking [21]. Wang et al. [17] argued that domain expertise represented by the conventional sparse coding is still valuable and can be combined to achieve further improved results in image super-resolution. Instead of training with different levels of scaling factors, they proposed to use a cascade structure to repeatedly enlarge the low-resolution image by a fixed scale until reaching a desired size. In general, DNN-based methods learn restoration parameters directly from data,

which tends to be more effective in real-world image restoration applications.

### 3 VERY DEEP CONVOLUTIONAL AUTO-ENCODER FOR IMAGE RESTORATION

The proposed framework mainly contains a chain of convolutional layers and symmetric deconvolutional layers, as shown in Figure 1. Skip connections are connected symmetrically from convolutional layers to deconvolutional layers. We term our method “RED-Net”—very deep Residual Encoder-Decoder Networks.

#### 3.1 Architecture

The framework is fully convolutional (and deconvolutional). Deconvolution is essentially unsampling convolution). Rectification layers are added after each convolution and deconvolution. For low-level image restoration problems, we use neither pooling nor unpooling in the network as usually pooling discards useful image details that are essential for these tasks. It is worth mentioning that since the convolutional and deconvolutional layers are symmetric, the network is essentially pixel-wise prediction, thus the size of input image can be arbitrary. The input and output of the network are images of the same size  $w \times h \times c$ , where  $w$ ,  $h$  and  $c$  are width, height and number of channels.

Our main idea is that the convolutional layers act as a feature extractor, which preserve the primary components of objects in the image and meanwhile eliminating the corruptions. After forwarding through the convolutional layers, the corrupted input image is converted into a “clean” one. The subtle details of the image contents may be lost during this process. The deconvolutional layers are then combined to recover the details of image contents. The output of the deconvolutional layers is the recovered clean version of the input image. Moreover, we add skip connections from a convolutional layer to its corresponding mirrored deconvolutional layer. The passed convolutional feature maps are summed to the deconvolutional feature maps element-wise, and passed to the next layer after rectification. Deriving from the above architecture, we have used two networks in our experiments, which are of 20 layers and 30 layers respectively, for image denoising, image super-resolution, JPEG deblocking and image inpainting.

#### 3.2 Deconvolution decoder

Architectures combining layers of convolution and deconvolution [43], [44] have been proposed for semantic segmentation recently. In contrast to convolutional layers, in which multiple input activations within a filter window are fused to output a single activation, deconvolutional layers associate a single input activation with multiple outputs. Deconvolution is usually used as *learnable up-sampling layers*.

In our network, the convolutional layers successively down-sample the input image content into a small size abstraction. Deconvolutional layers then up-sample the abstraction back into its original resolution.

Besides the use of skip connections, a main difference between our model and [43], [44] is that our network is

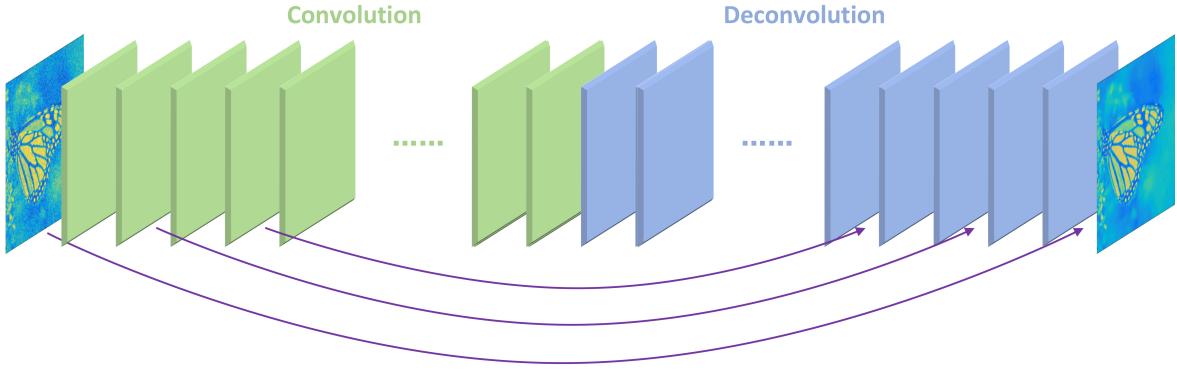


Fig. 1. The overall architecture of our proposed network. The network contains layers of symmetric convolution (encoder) and deconvolution (decoder). Skip shortcuts are connected every a few (in our experiments, two) layers from convolutional feature maps to their mirrored deconvolutional feature maps. The response from a convolutional layer is directly propagated to the corresponding mirrored deconvolutional layer, both forwardly and backwardly.

fully convolutional and deconvolutional, i.e., without pooling and un-pooling. The reason is that for low-level image restoration, the aim is to eliminate low level corruption while preserving image details instead of learning image abstractions. Different from high-level applications such as segmentation or recognition, pooling typically eliminates the abundant image details and can deteriorate restoration performance.

One can simply replace deconvolution with convolution, which results in an architecture that is very similar to recently proposed very deep fully convolutional neural networks [27], [28]. However, there exist essential differences between a fully convolution model and our model. Take image denoising as an example. We compare the 5-layer and 10-layer fully convolutional network with our network (combining convolution and deconvolution, but without skip connection). For fully convolutional networks, we use padding or up-sampling the input to make the input and output be of the same size. For our network, the first 5 layers are convolutional and the second 5 layers are deconvolutional. All the other parameters for training are identical, i.e., trained with SGD and learning rate of  $10^{-6}$ , noise level  $\sigma = 70$ . The Peak Signal-to-Noise Ratio (PSNR) on the validation set is reported, which shows that using deconvolution works better than the fully convolutional counterpart, as shown in Figure 2.

Furthermore, in Figure 3, we visualize some results that are outputs of layer 2, 5, 8 and 10 from the 10-layer fully convolutional network and ours. In the fully convolution case, the noise is eliminated step by step, i.e., the noise level is reduced after each layer. During this process, the details of the image content may be lost. Nevertheless, in our network, convolution preserves the primary image content. Then deconvolution is used to compensate the details.

### 3.3 Skip connections

An intuitive question is that, is a network with deconvolution able to recover image details from the image abstraction only? We find that in shallow networks with only a few layers of convolution layers, deconvolution is able to recover the details. However, when the network goes deeper or using operations such as max pooling, even with

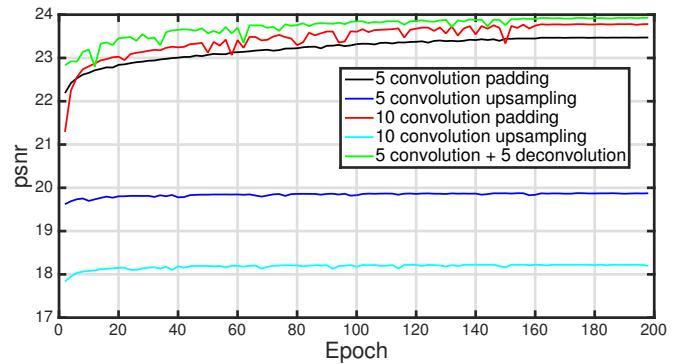


Fig. 2. PSNR values on the validation set during training. Our model exhibits better PSNR than the compared ones upon convergence.

deconvolution layers, it does not work that well, possibly because too much details are already lost in the convolution and pooling.

The second question is that, when our network goes deeper, does it achieve performance gain? We observe that deeper networks in image restoration tasks tend to easily suffer from performance degradation. The reason may be two folds. First of all, with more layers of convolution, a significant amount of image details could be lost or corrupted. Given only the image abstraction, recovering its details is an under-determined problem. Secondly, in terms of optimization, deep networks often suffer from gradients vanishing and become much harder to train—a problem that is well addressed in the literature of neural networks.

To address the above two problems, inspired by highway networks [45] and deep residual networks [1], we add skip connections between two corresponding convolutional and deconvolutional layers as shown in Figure 1. A building block is shown in Figure 4. There are two reasons for using such connections. First, when the network goes deeper, as mentioned above, image details can be lost, making deconvolution weaker in recovering them. However, the feature maps passed by skip connections carry much image detail, which helps deconvolution to recover an improved clean version of the image. Second, the skip connections also achieve benefits on back-propagating the gradient to

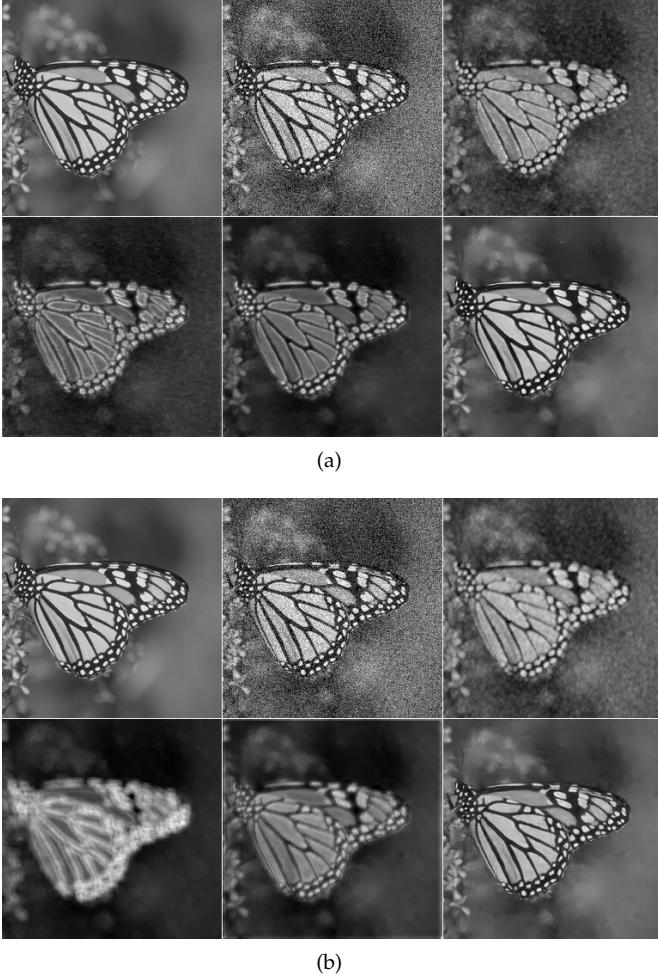


Fig. 3. (a) Visualization of the 10-layer fully convolutional network. The images from top-left to bottom-right are: clean image, noisy image, output of conv-2, output of conv-5, output of conv-8 and output of conv-10, where “conv- $i$ ” stands for the  $i$ -th convolutional layer; (b) Visualization of the 10-layer convolutional and deconvolutional network. The images from top-left to bottom-right are: clean image, noisy image, output of conv-2, output of conv-5, output of deconv-3 and output of deconv-5, where “deconv- $i$ ” stands for the  $i$ -th deconvolutional layer.

bottom layers, which makes training deeper network much easier as observed in [45] and [1].

Note that our skip layer connections are very different from the ones proposed in [45] and [1], where the only concern is on the optimization side. In our case, we want to pass information of the convolutional feature maps to the corresponding deconvolutional layers. The very deep highway networks [45] are essentially feedforward long short-term memory (LSTMs) with forget gates, and the CNN layers of deep residual network [1] are feedforward LSTMs without gates. Note that our networks are in general not in the format of standard feedforward LSTMs.

Instead of directly learning the mappings from the input  $X$  to the output  $Y$ , we would like the network to fit the residual [1] of the problem, which is denoted as  $\mathcal{F}(X) = Y - X$ . Such a learning strategy is applied to inner blocks of the encoding-decoding network to make training more effective. Skip connections are passed every two convolutional layers to their mirrored deconvolutional layers. Other configurations are possible and our experiments show

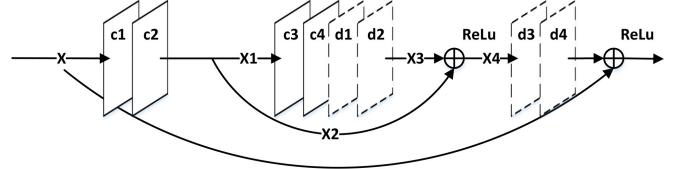


Fig. 4. An example of a building block in the proposed framework. The rectangle in solid and dotted lines denote convolution and deconvolution respectively.  $\oplus$  denotes element-wise sum of feature maps.

that this configuration already works very well. Using such shortcuts makes the network easier to be trained and gains restoration performance by increasing the network depth.

### 3.4 Training

In general, there are three types of layers in our network: convolution, deconvolution and element-wise sum. Each layer is followed by a Rectified Linear Unit (ReLU) [46]. Let  $X$  be the input, the convolutional and deconvolutional layers are expressed as:

$$F(X) = \max(0, W_k * X + B_k), \quad (2)$$

where  $W_k$  and  $B_k$  represent the filters and biases, and  $*$  denotes either convolution or deconvolution operation for the convenience of formulation. For element-wise sum layer, the output is the element-wise sum of two inputs of the same size, followed by the ReLU activation:

$$F(X_1, X_2) = \max(0, X_1 + X_2) \quad (3)$$

Learning the end-to-end mapping from corrupted images to clean images needs to estimate the weights  $\Theta$  represented by the convolutional and deconvolutional kernels. Specifically, given a collection of  $N$  training sample pairs  $\{X^i, Y^i\}$ , where  $X^i$  is a noisy image and  $Y^i$  is the clean version as the groundtruth. We minimize the following Mean Squared Error (MSE):

$$\mathcal{L}(\Theta) = \frac{1}{N} \sum_{i=1}^N \| \mathcal{F}(X^i; \Theta) - Y^i \|_F^2. \quad (4)$$

Traditionally, a network can learn the mapping from the corrupted image to the clean version directly. However, our network learns for the additive corruption from the input since there is a skip connection between the input and the output of the network. We found that optimizing for the corruption converges better than optimizing for the clean image. In the extreme case, if the input is a clean image, it would be easier to push the network to be zero mapping (learning the corruption) than to fit an identity mapping (learning the clean image) with a stack of nonlinear layers.

We implement and train our network using Caffe [47]. Empirically, we find that using Adam [48] with base learning rate of  $10^{-4}$  for training converges faster than traditional stochastic gradient descent (SGD). The base learning rate for all layers are the same, different from [28], [26], in which a smaller learning rate is set for the last layer. This is not necessary in our network. Specifically, gradients with respect to the parameters of  $i$ th layer is firstly computed as:

$$g = \nabla_{\theta_i} \mathcal{L}(\theta_i). \quad (5)$$

Then, the two momentum vectors are computed as:

$$m = \beta_1 m + (1 - \beta_1)g, \quad v = \beta_2 v + (1 - \beta_2)g^2. \quad (6)$$

The update rule is:

$$\alpha = \alpha \sqrt{1 - \beta_2^t} / (1 - \beta_1^t), \quad \theta_i = \theta_i - \alpha m / (\sqrt{v} + \epsilon). \quad (7)$$

$\beta_1, \beta_2$  and  $\epsilon$  are set as the recommended values in [48].

300 images from the Berkeley Segmentation Dataset (BSD) [49] are used to generate image patches as the training set for each image restoration task.

### 3.5 Testing

Although trained on local patches, our network can perform restoration on images of arbitrary sizes. Given a testing image, one can simply go forward through the network, which is already able to outperform existing methods. To achieve even better results, we propose to process a corrupted image on multiple orientations. Different from segmentation, the filter kernels in our network only eliminate the corruptions, which is usually not sensitive to the orientation of image contents in low level restoration tasks. Therefore, we can rotate and mirror flip the kernels and perform forward multiple times, and then average the output to achieve an ensemble of multiple tests. We see that this can lead to slightly better performance.

## 4 DISCUSSIONS

### 4.1 Analysis on the architecture

Assume that we have a network with  $L$  layers, and skip connections are passed every layer in the first half of the network. For the convenience of presentation, we denote  $F_c$  and  $F_d$  the convolution and deconvolution operation in each layer and do not use ReLU. According to the architecture described in the last section, we can obtain the output of the  $i$ -th layer as follows:

$$X_i = \begin{cases} X_{L-i} + F_d(X_{i-1}), & i \geq L/2; \\ F_c(X_{i-1}). & i < L/2. \end{cases} \quad (8)$$

It is easy to observe that our skip connections indicate identity mapping. The output of the network is:

$$X_L = X_0 + F_d(X_{L-1}). \quad (9)$$

Recursively, we can compute  $X_L$  more specifically as follows according to Equation (8):

$$\begin{aligned} X_L &= X_0 + F_d(X_{L-1}) \\ &= X_0 + F_d(X_1 + F_d(X_{L-2})) \\ &= X_0 + F_d(X_1) + F_d^2(X_2 + F_d(X_{L-3})) \\ &\dots \\ &= X_0 + F_d(X_1) + F_d^2(X_2) + \dots + F_d^{L/2-1}(X_{L/2-1}) \\ &+ F_d^{L/2}(X_{L/2}). \end{aligned} \quad (10)$$

Since  $F_d^{L/2}(X_{L/2})$  can be expressed as  $F_d^{L/2}(F_c^{L/2}(X_0))$ , we convert Equation (10) as:

$$X_L = F_d^{L/2}(F_c^{L/2}(X_0)) + \sum_{i=0}^{L/2-1} F_d^i(X_i). \quad (11)$$

In Equation (11), the term  $F_d^{L/2}(F_c^{L/2}(X_0))$  is actually the output of the given network without skip connections. The difference here is that by adopting the skip connection, we decode each feature maps  $X_i, 0 \leq i < L/2$  in the first half network and integrate them to the output. The most significant benefit is that they carry important image details, which helps to reconstruct clean image. Moreover, the term  $\sum_{i=0}^{L/2-1} F_d^i(X_i)$  indicates that these details are represented at different levels. It is intuitive to see the following fact. It may not be easy to tell what information is needed for reconstructing clean images using only one feature maps encoding the image abstraction; but much easier if there are multiple feature maps encoding different levels of image abstraction.

### 4.2 Gradient back-propagation

For back-propagation, a layer receives gradients from the layers that it is connected to. As an example shown in Figure 4,  $X$  is the input of the first layer, after two convolutional layers  $c1$  and  $c2$ , the output is  $X_1$ . To update the parameters represented as  $\theta_2$  of  $c2$ , we compute the derivative of  $\mathcal{L}$  with respect to  $\theta_2$  as follows:

$$\nabla_{\theta_2} \mathcal{L}(\theta_2) = \frac{\partial \mathcal{L}}{\partial X_1} \frac{\partial X_1}{\partial \theta_2} + \frac{\partial \mathcal{L}}{\partial X_2} \frac{\partial X_2}{\partial \theta_2} \quad (12)$$

where using  $X_1$  and  $X_2$  is only for the clarity of presentation, they are essentially the same. We can further formulate (12) as:

$$\nabla_{\theta_2} \mathcal{L}(\theta_2) = \frac{\partial \mathcal{L}}{\partial X_4} \frac{\partial X_4}{\partial X_3} \frac{\partial X_3}{\partial X_1} \frac{\partial X_1}{\partial \theta_2} + \frac{\partial \mathcal{L}}{\partial X_4} \frac{\partial X_4}{\partial X_2} \frac{\partial X_2}{\partial \theta_2}. \quad (13)$$

Only  $\frac{\partial \mathcal{L}}{\partial X_4} \frac{\partial X_4}{\partial X_3} \frac{\partial X_3}{\partial X_1} \frac{\partial X_1}{\partial \theta_2}$  is computed if we do not use skip connections, and its magnitude may become very small after back-propagating through many layers from the top in very deep networks. However,  $\frac{\partial \mathcal{L}}{\partial X_4} \frac{\partial X_4}{\partial X_2} \frac{\partial X_2}{\partial \theta_2}$  carries larger gradients since it does not have to go through layers of  $d2, d1, c4$  and  $c3$  in this example. Thus with the first term only, it is more unlikely to approach zero gradients. As we can see, the skip connection helps to update the filters in bottoms layers, and thus makes training easier.

### 4.3 Training with symmetric skip connections

The aim of restoration is to eliminate corruption while preserving the image details as much as possible. Previous works typically use shallow networks for low-level image restoration tasks. The reason may be that deeper networks can destroy the image details, which is undesired for pixel-wise dense regression. Even worse, using very deep networks may easily suffer from training issues such as gradient vanishing. Using skip connections in a very deep network can address both of the above two problems.

Firstly, we design experiments to show that using skip connections is beneficial for image detail preserving. Specifically, two networks are trained for image denoising with a noise level of  $\sigma = 70$ .

(a) In the first network, we use 5 layers of  $3 \times 3$  convolution with stride 3. The input size of training data is  $243 \times 243$ , which results in a vector after 5 layers of convolution, encoding the very high level abstraction of the image. Then deconvolution is used to recover the input from the feature

vector. The results are shown in Figure 5. We can observe that it is challenging for deconvolution to recover details from only a vector encoding the abstraction of the input. This phenomenon implies that simply using deep networks for image restoration may not lead to satisfactory results.

(b) The second network uses the same settings as the first one, but adding skip connections. The results are show in Figure 5. Compared to the first network, the one with skip connections can recover the input and achieves much better PSNR values. This is easy to understand since the feature maps with abundant details at bottom layers are directly passed to the top layers.

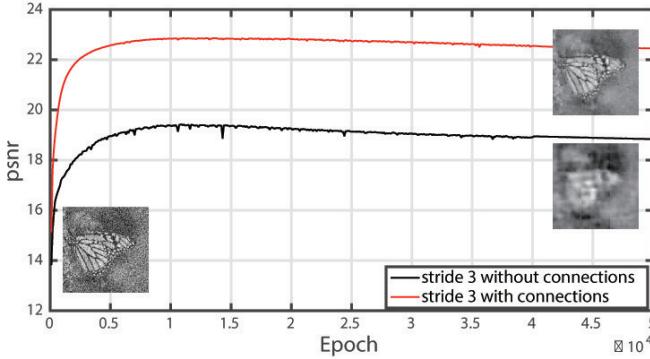


Fig. 5. Recovering image details using deconvolution and skip connections. Skip connections are beneficial in recovering image details.

Secondly, we train and compare five different networks to show that using skip connections help to back-propagate gradient in training to better fit the end-to-end mapping, as shown in Figure 6. The five networks are: 10, 20 and 30 layer networks without skip connections; and 20, 30 layer networks with skip connections. As can be seen, the training loss increases when the network going deeper without shortcuts (similar phenomenon is also observed in [1]). On the validation set, deeper networks without shortcuts achieve lower PSNR and we even observe overfitting for the 30-layer network. These results may be due to the gradient vanishing problem. However, we obtain smaller training errors on the training set and higher PSNR and better generalization capability on the testing set when using skip connections.

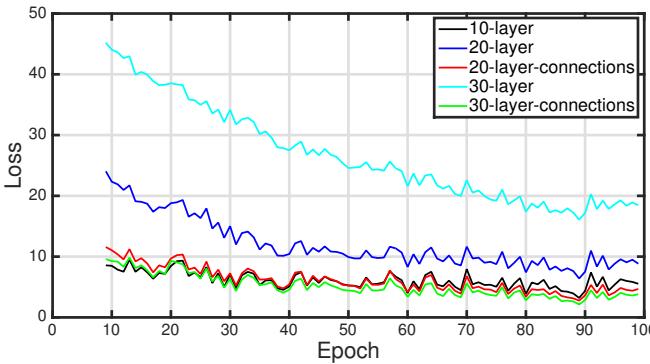


Fig. 6. The training loss on the training set during training.

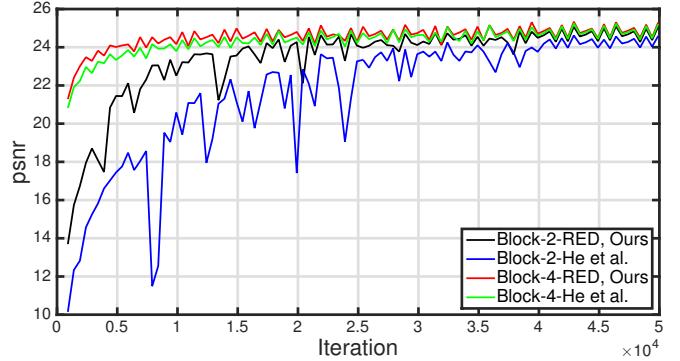


Fig. 7. Comparisons of skip connections in [1] and our model, where “Block- $i$ -RED” is the connections in our model with block size  $i$  and “Block- $i$ -He et al.” is the connections in He et al. [1] with block size  $i$ ; The PSNR values on the validation set during training: the PSNR at the last iteration for the curves are: 25.08, 24.59, 25.30 and 25.21.

#### 4.4 Comparison with the deep residual network [1]

One may use different types of skip connections in our network. A straightforward alternate is that in [1]. In [1], skip connections are added to divide the network into sequential blocks. A benefit of our model is that our skip connections have element-wise correspondence, which can be very important in pixel-wise prediction problems such as image denoising. We carry out experiments to compare these two types of skip connections. Here the block size indicates the span of the connections. The results are shown in Figure 7. We can observe that our connections often converge to a better optimum, demonstrating that element-wise correspondence can be important. Meanwhile, our long range skip connections pass the image detail directly from bottom layers to top layers. If we use the skip connection type in [1], the network may still lose some image details.

#### 4.5 Testing efficiency

To apply deep learning models on devices with limited computing power such as mobile phones, one has to speed-up the testing phase. For our network, we propose to use down-sampling in convolutional layers to reduce the size of the feature maps. In order to obtain an output of the same size as the input, deconvolution is used to up-sample the feature maps in the symmetric deconvolutional layers. Thus, the testing efficiency can be well improved with almost negligible performance degradation.

In specific, we use stride = 2 in convolutional layers to down-sample the feature maps. Down-sampling at different convolutional layers are tested on image denoising, as shown in Figure 8. We test an image of size 160×240 on an i7-2600 CPU, the testing time for “no down-sample”, “down-sample at conv1”, “down-sample at conv5”, “down-sample at conv9”, “down-sample at conv5,9” are 3.17s, 0.84s, 1.43s, 2.00s and 1.17s respectively.

The main observation is that the testing PSNRs may slightly degrade according to the scale reduction of the feature map in the entire network. The down-sampling in the first convolutional layer reduces the size of the feature maps to 1/4, which leads to almost 4x faster in testing, but the PSNR only degrades less than 0.1 compared to the network

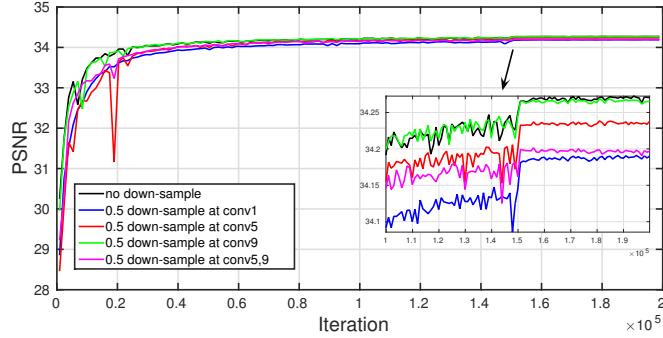


Fig. 8. The PSNRs on the validation set with different down-sampling strategies. "down-sample at conv- $i$ " denotes that down-sampling is used in the  $i$ th convolutional layer, and up-sampling is used in its symmetric deconvolutional layer.

without down-sampling. The down-sampling in "conv9" reduces 1/3 of the testing time, but the performance is almost as well as that without down-sampling. As a result, an "earlier" down-sampling may lead to slightly worse performance, but it achieves much faster testing efficiency. It should be a trade-off in different application situations.

## 5 EXPERIMENTS

In this section, we first provide some experimental results and analysis on different parameters, including filter number, filter size, training patch size and skip connection step size, of the network.

Then, evaluation of image restoration tasks including image denoising, image super-resolution, JPEG image de-blocking, non-blind image deblurring and image inpainting are conducted and compared against a few existing state-of-the-art methods in each topic. Peak Signal-to-Noise Ratio (PSNR) and Structural SIMilarity (SSIM) index are calculated for evaluation. For our method, which is denoted as RED-Net, we implement three versions: RED10 contains 5 convolutional and deconvolutional layers without shortcuts, RED20 contains 10 convolutional and deconvolutional layers with shortcuts of step size 2, and RED30 contains 15 convolutional and deconvolutional layers with shortcuts of step size 2.

### 5.1 Network parameters

Although we have observed that deeper networks tend to achieve better image restoration performance, there exist more problems related to different parameters to be investigated. We carried out image denoising experiments on three folds: (a) filter number, (b) filter size, (c) training patch size and (d) step size of skip connections, to show the effects of different parameters.

For different filter numbers, we fix the filter size as  $3 \times 3$ , training patch size as  $50 \times 50$  and skip connection step size as 2. Different filter numbers of 32, 64 and 128 are tested, and the PSNR values recorded on the validation set during training are shown in Figure 9. To converge, the training iterations for different number of filters are similar, but better optimum can be obtained with more filters. However, a smaller number of filters is preferred if a fast testing speed is desired.

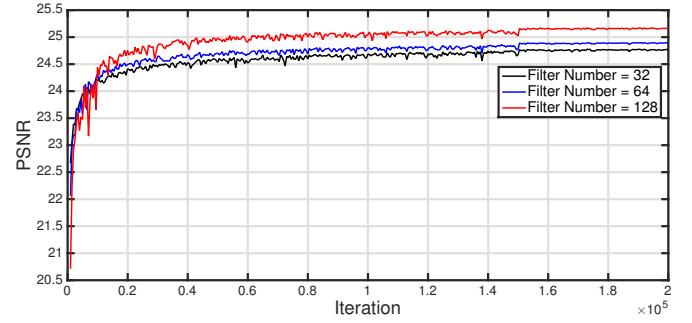


Fig. 9. The PSNR values on the validation set during training with different number of filters.

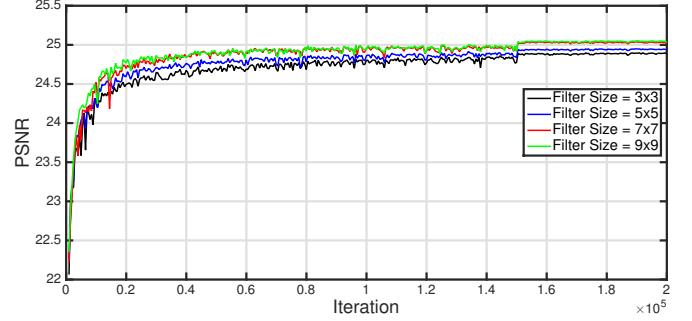


Fig. 10. The PSNR values on the validation set during training with different size of filters.

For the experiments on filter size, we set the filter number to be 64, training patch size as  $50 \times 50$ , skip connection step size as 2.

Filter size of  $3 \times 3$ ,  $5 \times 5$ ,  $7 \times 7$ ,  $9 \times 9$  are tested. Figure 10 show the PSNR values on the validation set while training. It is clear that larger filter size leads to better performance. Different from high-level tasks [50], [51], [52] which favor smaller filter sizes, larger filter size tends to obtain better performance in low-level image restoration applications.

However, there may exist a bottle neck as the performance of  $9 \times 9$  is almost as the same as  $7 \times 7$  in our experiments. The reason may be that for high-level tasks, the networks have to learn image abstraction for classification, which is usually very different from the input pixels. Larger filter size may result in larger respective fields, but also made the networks more difficult to train and converge to a poor optimum. Using smaller filter size is mainly beneficial for convergence in such complex mappings.

In contrast, for low-level image restoration, the training is not as difficult as that in high-level applications since only a bias is needed to be learned to revise the corrupted pixel. In this situation, utilizing neighborhood information in the mapping stage is more important, since the desired value for a pixel should be predicted from its neighbor pixels. However, using larger filter size inevitably increases the complexity (e.g., filter size of  $9 \times 9$  is 9 times more complex as  $3 \times 3$ ) and training time.

For the training patch size, we set the filter number to be 64, filter size as  $3 \times 3$ , skip connection step size as 2. Then we test different training patch sizes of  $25 \times 25$ ,  $50 \times 50$ ,  $75 \times 75$ ,  $100 \times 100$ , as shown in Figure 11.

Better performance is achieved with larger training patch

size. The reason can be two folds. First of all, since the network essentially performs pixel-wise prediction, if the number of training patches are the same, larger size of training patch results in more pixels to be used, which is equivalent to using more training data. Secondly, the corruptions in image restoration tasks can be described as some types of latent distributions. Larger size of training patch contains more pixels that better capture the latent distributions to be learned, which consequently helps the network to fit the corruptions better.

As we can see, the “width” of the network is as crucial as the “depth” in training a network with satisfactory image restoration performance. However, one should always make a trade-off between the performance and speed.

We also provide the experiments of different step sizes of shortcuts, as shown in Figure 12. A smaller step size of shortcuts achieves better performance than a larger one. We believe that a smaller step size of shortcuts makes it easier to back-propagate the gradient to bottom layers, thus tackle the gradient vanishing issue better. Meanwhile, a small step size of shortcuts essentially passes more direct information.

## 5.2 Image denoising

Image denoising experiments are performed on two datasets: 14 common benchmark images [9], [8], [7], [10], as show in Figure 13 and the BSD dataset.

As a common experimental setting in the literature, additive Gaussian noises with zero mean and standard deviation  $\sigma$  are added to the image to test the performance of denoising methods. In this paper we test noise level  $\sigma$



Fig. 13. The 14 testing images for denoising.

of 10, 30, 50 and 70. BM3D [30], NCSR [3], EPLL [6], PCLR [8], PGPD [9] and WMMN [10] are compared with our method. For these methods, we use the source code released by their authors and test on the images with their default parameters.

**Evaluation on the 14 images** Table 1 presents the PSNR and SSIM results of  $\sigma$  10, 30, 50, and 70. We can make some observations from the results. First of all, the 10 layer convolutional and deconvolutional network has already achieved better results than the state-of-the-art methods, which demonstrates that combining convolution and deconvolution for denoising works well, even without any skip connections.

Moreover, when the network goes deeper, the skip connections proposed in this paper help to achieve even better denoising performance, which exceeds the existing best method WNNM [10] by 0.32dB, 0.43dB, 0.49dB and 0.51dB on noise levels of  $\sigma$  being 10, 30, 50 and 70 respectively. While WNNM is only slightly better than the second best existing method PCLR [8] by 0.01dB, 0.06dB, 0.03dB and 0.01dB respectively, which shows the large improvement of our model.

Last, we can observe that the more complex the noise is, the more improvement our model achieves than other methods. Similar observations can be made on the evaluation of SSIM.

**Evaluation on BSD200** For the BSD dataset, 300 images are used for training and the remaining 200 images are used for denoising to show more experimental results. For efficiency, we convert the images to gray-scale and resize them to smaller images. Then all the methods are run on the dataset to get average PSNR and SSIM results of  $\sigma$  10, 30, 50, and 70, as shown in Table 2. For existing methods, their denoising performance does not differ much, while our model achieves 0.38dB, 0.47dB, 0.49dB and 0.42dB higher of PSNR over WNNM [10].

**Blind denoising** We also perform blind denoising to show the superior performance of our network. In blind denoising, the training set consists of image patches of different levels of noises, and a 30-layer network is trained on this training set. In the testing phase, we test noisy images with  $\sigma$  of 10, 30, 50 and 70 using this model. The evaluation results are shown in Table 3. Although training with different levels of corruption, we can observe that the performance of our network degrades comparing to the case in which using separate models for denoising. This is reasonable because the network has to fit much more complex mappings. However, it still beats the existing methods. For PSNR evaluation, our blind denoising model achieves the same performance as WNNM [10] on  $\sigma = 10$ , and outperforms WNNM [10] by 0.35dB, 0.43dB

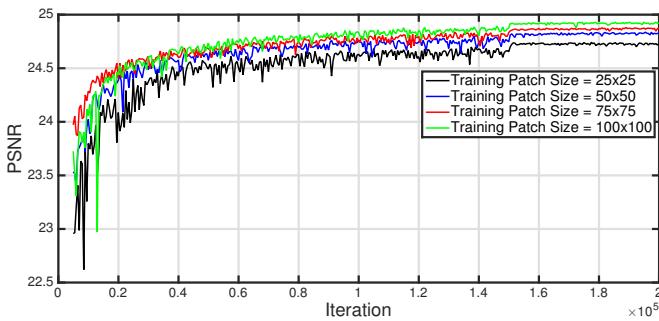


Fig. 11. The PSNR values on the validation set during training with different size of training patch.

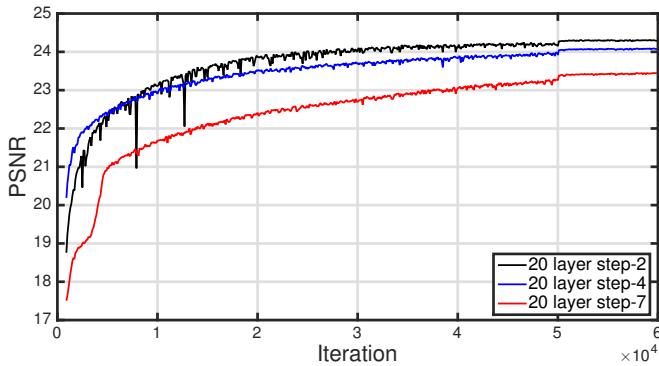


Fig. 12. The PSNR values on the validation set during training.

TABLE 1  
Average PSNR and SSIM results of  $\sigma = 10, 30, 50, 70$  on 14 images.

	PSNR								
	BM3D	EPLL	NCSR	PCLR	PGPD	WNNM	RED10	RED20	RED30
$\sigma = 10$	34.18	33.98	34.27	34.48	34.22	34.49	34.62	34.74	<b>34.81</b>
$\sigma = 30$	28.49	28.35	28.44	28.68	28.55	28.74	28.95	29.10	<b>29.17</b>
$\sigma = 50$	26.08	25.97	25.93	26.29	26.19	26.32	26.51	26.72	<b>26.81</b>
$\sigma = 70$	24.65	24.47	24.36	24.79	24.71	24.80	24.97	25.23	<b>25.31</b>
	SSIM								
$\sigma = 10$	0.9339	0.9332	0.9342	0.9366	0.9309	0.9363	0.9374	0.9392	<b>0.9402</b>
$\sigma = 30$	0.8204	0.8200	0.8203	0.8263	0.8199	0.8273	0.8327	0.8396	<b>0.8423</b>
$\sigma = 50$	0.7427	0.7354	0.7415	0.7538	0.7442	0.7517	0.7571	0.7689	<b>0.7733</b>
$\sigma = 70$	0.6882	0.6712	0.6871	0.6997	0.6913	0.6975	0.7012	0.7177	<b>0.7206</b>

TABLE 2  
Average PSNR and SSIM results of  $\sigma = 10, 30, 50, 70$  on BSD.

	PSNR								
	BM3D	EPLL	NCSR	PCLR	PGPD	WNNM	RED10	RED20	RED30
$\sigma = 10$	33.01	33.01	33.09	33.30	33.02	33.25	33.49	33.59	<b>33.63</b>
$\sigma = 30$	27.31	27.38	27.23	27.54	27.33	27.48	27.79	27.90	<b>27.95</b>
$\sigma = 50$	25.06	25.17	24.95	25.30	25.18	25.26	25.54	25.67	<b>25.75</b>
$\sigma = 70$	23.82	23.81	23.58	23.94	23.89	23.95	24.13	24.33	<b>24.37</b>
	SSIM								
$\sigma = 10$	0.9218	0.9255	0.9226	0.9261	0.9176	0.9244	0.9290	0.9310	<b>0.9319</b>
$\sigma = 30$	0.7755	0.7825	0.7738	0.7827	0.7717	0.7807	0.7918	0.7993	<b>0.8019</b>
$\sigma = 50$	0.6831	0.6870	0.6777	0.6947	0.6841	0.6928	0.7032	0.7117	<b>0.7167</b>
$\sigma = 70$	0.6240	0.6168	0.6166	0.6336	0.6245	0.6346	0.6367	0.6521	<b>0.6551</b>

and 0.40dB on  $\sigma = 30, 50$  and 70 respectively, which is still marginal improvements. For SSIM evaluation, our network is 0.0005, 0.0141, 0.0199 and 0.0182 higher than WNNM. The performance improvement is more obvious on BSD dataset. The 30-layer network outperforms the second best method WNNM [10] by 0.13dB, 0.4dB, 0.43dB, 0.41dB for PSNR and 0.0036, 0.0173, 0.0191, 0.0198 for SSIM.

**Visual results** Some visual results are shown in Figure 14. We highlight some details of the clean image and the recovered ones by different methods. The first observation is that our method better recovers the image details, as we can see from the third and fourth rows, which is due to the high PSNR we achieve by minimizing the pixel-wise Euclidean loss.

Moreover, we can observe from the first and second rows that our network obtains more visually smooth results than other methods. This may due to the testing strategy which average the output of different orientations.

### 5.3 Image super-resolution

For super-resolution, The high-resolution image is first down-sampled with scaling factor parameters of 2, 3 and

4 respectively. Since the size of the input and output of our network are the same, we up-sample the low-resolution image to its original size as the input of our network. We compare our network with SRCNN [28], NBSRF [53], CSCN [17], CSC [16], TSE [54] and ARFL+ [55] on three dataset: Set5, Set14 and BSD100.

The results of the compared methods are either cited from their original papers or obtained using the released source code by the authors.

**Evaluation on Set 5** The evaluation on Set5 is shown in Table 4. In general, our 10-layer network already outperforms the compared methods, and we achieve better performance with deeper networks.

The second best method is CSCN, which is also a recently proposed neural network based method. Compared to CSCN, our 30-layer network exceeds it by 0.52dB, 0.56dB, 0.47dB on PSNR and 0.0032, 0.0063, 0.0094 on SSIM respectively.

The larger scaling parameter is, the better improvement our method can make, which demonstrates that our network is better at fitting complex corruptions than other methods.

**Evaluation on Set 14** The evaluation on Set14 is shown in Table 5. The improvement on Set14 is not as significant as that on Set5, but we can still observe that the 30-layer network achieves higher PSNR and SSIM than the second best CSCN for 0.23dB, 0.06dB, 0.1dB and 0.0049, 0.0070, 0.0098. The performance on 10-layer, 20-layer and 30-layer RED-Net also does not improve that much as on Set5, which may imply that Set14 is more difficult to perform image super-resolution.

**Evaluation on BSD 100** We also evaluate super-resolution results on BSD100, as shown in Table 6. The overall results are very similar than those on Set5. CSCN

TABLE 3

Average PSNR and SSIM results for image denoising using a single 30-layer network.

	14 images			
	$\sigma = 10$	$\sigma = 30$	$\sigma = 50$	$\sigma = 70$
PSNR	34.49	29.09	26.75	25.20
SSIM	0.9368	0.8414	0.7716	0.7157
BSD200				
	$\sigma = 10$	$\sigma = 30$	$\sigma = 50$	$\sigma = 70$
PSNR	33.38	27.88	25.69	24.36
SSIM	0.9280	0.7980	0.7119	0.6544

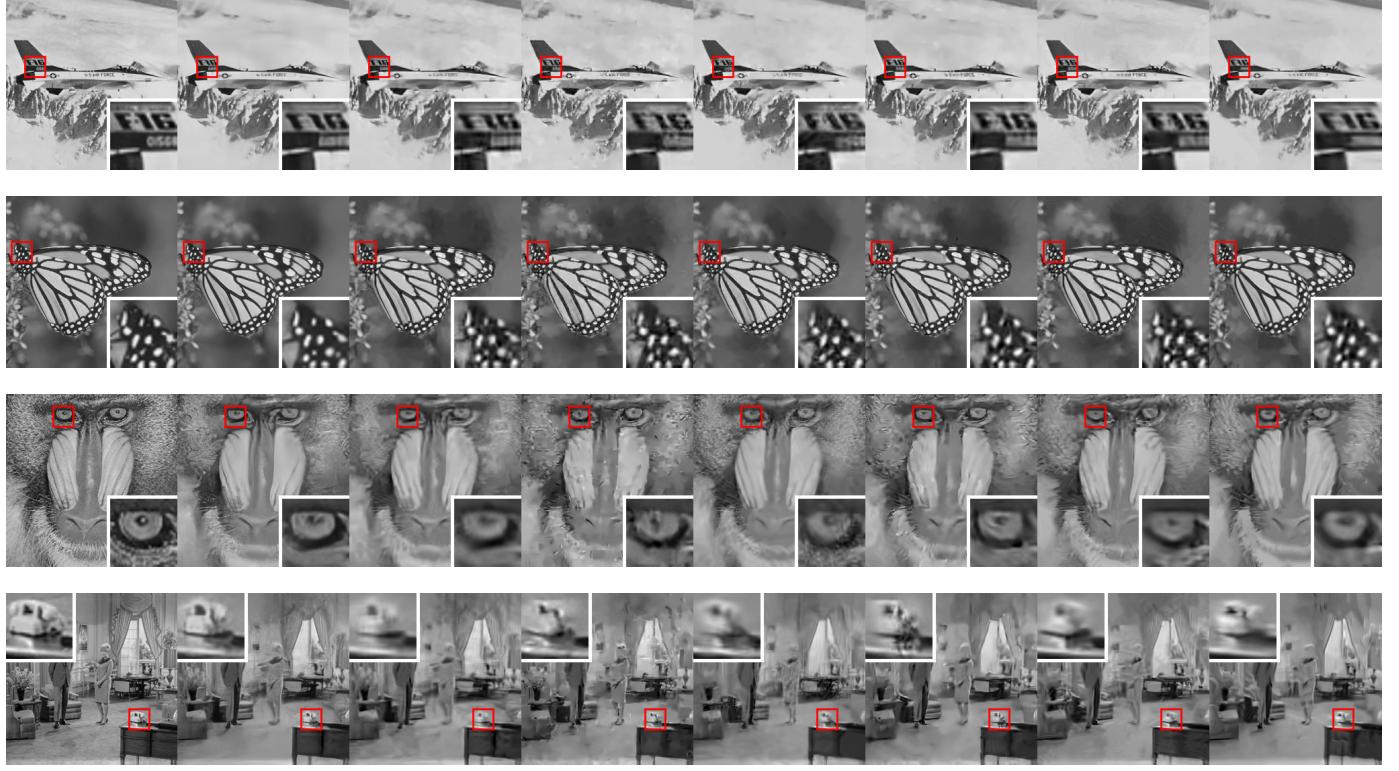


Fig. 14. Visual results of image denoising. Images from left to right column are: clean image; the recovered image of RED30, BM3D, EPLL, NCSR, PCLR, PGPD, WNNM.

TABLE 4  
Average PSNR and SSIM results of scaling 2, 3 and 4 on Set5.

	PSNR									
	SRCNN	NBSRF	CSCN	CSC	TSE	ARFL+	RED10	RED20	RED30	
$s = 2$	36.66	36.76	37.14	36.62	36.50	36.89	37.43	37.62	<b>37.66</b>	
$s = 3$	32.75	32.75	33.26	32.66	32.62	32.72	33.43	33.80	<b>33.82</b>	
$s = 4$	30.49	30.44	31.04	30.36	30.33	30.35	31.12	31.40	<b>31.51</b>	
	SSIM									
$s = 2$	0.9542	0.9552	0.9567	0.9549	0.9537	0.9559	0.9590	0.9597	<b>0.9599</b>	
$s = 3$	0.9090	0.9104	0.9167	0.9098	0.9094	0.9094	0.9197	0.9229	<b>0.9230</b>	
$s = 4$	0.8628	0.8632	0.8775	0.8607	0.8623	0.8583	0.8794	0.8847	<b>0.8869</b>	

TABLE 5  
Average PSNR and SSIM results of scaling 2, 3 and 4 on Set14.

	PSNR									
	SRCNN	NBSRF	CSCN	CSC	TSE	ARFL+	RED10	RED20	RED30	
$s = 2$	32.45	32.45	32.71	32.31	32.23	32.52	32.77	32.87	<b>32.94</b>	
$s = 3$	29.30	29.25	29.55	29.15	29.16	29.23	29.42	29.61	<b>29.61</b>	
$s = 4$	27.50	27.42	27.76	27.30	27.40	27.41	27.58	27.80	<b>27.86</b>	
	SSIM									
$s = 2$	0.9067	0.9071	0.9095	0.9070	0.9036	0.9074	0.9125	0.9138	<b>0.9144</b>	
$s = 3$	0.8215	0.8212	0.8271	0.8208	0.8197	0.8201	0.8318	0.8343	<b>0.8341</b>	
$s = 4$	0.7513	0.7511	0.7620	0.7499	0.7518	0.7483	0.7654	0.7697	<b>0.7718</b>	

is still the second best method and outperforms other compared methods by large margin, but its performance is not as good as our 10-layer network. Our deeper networks obtain performance gains. Compared to CSCN, the 30-layer network achieves higher PSNR for 0.45dB, 0.38dB, 0.29dB and higher SSIM for 0.0066, 0.0084, 0.0099.

**Comparisons with VDSR [56] and DRCN [57]** Concurrent to our work [58], networks [56], [57] which incorporate residual learning for image super-resolution are

proposed. In [56], a fully convolutional network termed VDSR is proposed to learn the residual image for image super-resolution. The loss layer takes three inputs: residual estimate, low-resolution input and ground truth high-resolution image, and Euclidean loss is computed between the reconstructed image (the sum of network input and output) and ground truth. DRCN [57] proposed to use a recursive convolutional block, which does not increase the

TABLE 6  
Average PSNR and SSIM results of scaling 2, 3 and 4 on BSD100

	PSNR								
	SRCNN	NBSRF	CSCN	CSC	TSE	ARFL+	RED10	RED20	RED30
$s = 2$	31.36	31.30	31.54	31.27	31.18	31.35	31.85	31.95	<b>31.99</b>
$s = 3$	28.41	28.36	28.58	28.31	28.30	28.36	28.79	28.90	<b>28.93</b>
$s = 4$	26.90	26.88	27.11	26.83	26.85	26.86	27.25	27.35	<b>27.40</b>
	SSIM								
$s = 2$	0.8879	0.8876	0.8908	0.8876	0.8855	0.8885	0.8953	0.8969	<b>0.8974</b>
$s = 3$	0.7863	0.7856	0.7910	0.7853	0.7843	0.7851	0.7975	0.7993	<b>0.7994</b>
$s = 4$	0.7103	0.7110	0.7191	0.7101	0.7108	0.7091	0.7238	0.7268	<b>0.7290</b>

number of parameters while increasing the depth of the network. To ease the training, firstly each recursive layer is supervised to reconstruct the target high-resolution image (HR). The second proposal is to use a skip-connection from input to the output. During training, the network has  $D$  outputs, in which the  $d$ th output  $y_d = x + \text{Rec}(H_d)$ .  $x$  is the input low-resolution image,  $\text{Rec}()$  denotes the reconstruction layer and  $H_d$  is the output of  $d$ th recursive layer. The final loss includes three parts: (a) the Euclidean loss between the ground truth and each  $y_d$ ; (b) the Euclidean loss between the ground truth and the weighted sum of all  $y_d$ ; and (c) the L2 regularization on the network weights. Although skip connections are used in our network, VDSR and DCRN to perform identity mapping, their differences are significant.

Firstly, both VDSR and DRCN use one path of connections between the input and output, which actually models the corruptions. In VDSR, the network itself is standard fully convolutional. DRCN uses recursive convolutional layers that lead to multiple losses, which is different from VDSR. The skip connections in VDSR and DRCN model the super-resolution problem as learning the residual image, which actually learns the corruption as in image restoration. In other words, the residual learning is only conducted in the input-output level (low-resolution and high-resolution images) in VDSR and DRCN. In contrast, our network uses multiple skip connections that divide the network into multiple blocks for residual learning. Secondly, our skip connections pass image abstraction of different levels from multiple convolutional layers forwardly. No such information is used in VDSR and DRCN. In VDSR and DRCN, the skip connection only pass the input image. However, in our network, different levels of image abstraction are obtained after the convolutional layers, and they are passed to the deconvolutional layers for reconstruction. At last, our skip connections help to back-propagate gradients in different layers. In VDSR and DCRN, the skip connections do not involve in back-propagating gradients since they connect the input and output, and there are no weights to be updated for the input low-resolution image. The image super-resolution comparisons of VDSR, DRCN and our network on Set5, Set14 and BSD100 are provided in Table 7.

**Blind super-resolution** The results of blind super-resolution are shown in Table 8. Among the compared methods, CSCN can also deal with different scaling parameters by repeatedly enlarging the image by a smaller scaling factor.

Our method is different from CSCN. Given a low-resolution image as input and the output size, we first upsample the input image to the desired size, resulting in an

image with poor details. Then the image is fed into our network. The output is an image of the same size with fine details. The training set consists of image patches of different scaling parameters and a single model is trained. Except that CSCN works slightly better on Set 14 with scaling factors 3 and 4, our network outperforms the existing methods, showing that our network works much better in image super-resolution even using only one single model to deal with complex corruptions.

TABLE 8  
Average PSNR and SSIM results for image super-resolution using a single 30 layer network.

	Set5		
	$s = 2$	$s = 3$	$s = 4$
PSNR	37.56	33.70	31.33
SSIM	0.9595	0.9222	0.8847
Set14			
$s = 2$	$s = 3$	$s = 4$	
PSNR	32.81	29.50	27.72
SSIM	0.9135	0.8334	0.7698
BSD100			
$s = 2$	$s = 3$	$s = 4$	
PSNR	31.96	28.88	27.35
SSIM	0.8972	0.7993	0.7276

**Visual results** Some visual results in grey-scale images are shown in Figure 15. Note that it is straightforward to perform super-resolution on color images.

We can observe from the second and third rows that our network is better at obtaining high resolution edges and text. Meanwhile, our results seem much more smooth than others. For faces such as the fourth row, our network still obtains better visually results.

#### 5.4 JPEG deblocking

Lossy compression, such as JPEG, introduces complex compression artifacts, particularly the blocking artifacts, ringing effects and blurring. In this section, we carry out deblocking experiments to recover high quality images from their JPEG compression. As in other compression artifacts reduction methods, standard JPEG compression schemes of JPEG quality settings  $q = 10$  and  $q = 20$  in MATLAB JPEG encoder are used. The LIVE1 dataset is used for evaluation, and we have compared our method with AR-CNN [21], SA-DCT [22] and deeper SRCNN [21].

The results are shown in Table 9. We can observe that since the Euclidean loss favors a high PSNR, our network outperforms other methods. Compared to AR-CNN, the 30-layer network exceeds it by 0.37dB and 0.44dB on compression quality of 10 and 20. Meanwhile, we can see that

TABLE 7

Comparisons between RED30 (ours), VDSR [56] and DRCN [57]: Average PSNR and SSIM results of scaling 2, 3 and 4 on Set5, Set14 and BSD100.

Dataset	Scale	VDSR (PSNR/SSIM)	DRCN (PSNR/SSIM)	RED30 (PSNR/SSIM)
Set5	$\times 2$	37.53/0.9587	37.63/0.9588	<b>37.66/0.9599</b>
	$\times 3$	33.66/0.9213	33.82/0.9226	<b>33.82/0.9230</b>
	$\times 4$	31.35/0.8838	<b>31.53/0.8854</b>	31.51/0.8869
Set14	$\times 2$	33.03/0.9124	<b>33.04/0.9118</b>	32.94/0.9144
	$\times 3$	<b>29.77/0.8314</b>	29.76/0.8311	29.61/0.8341
	$\times 4$	28.01/0.7674	<b>28.02/0.7670</b>	27.86/0.7718
BSD100	$\times 2$	31.90/0.8960	31.85/0.8942	<b>31.99/0.8974</b>
	$\times 3$	28.82/0.7976	28.80/0.7963	<b>28.93/0.7994</b>
	$\times 4$	27.29/0.7251	27.23/0.7233	<b>27.40/0.7290</b>

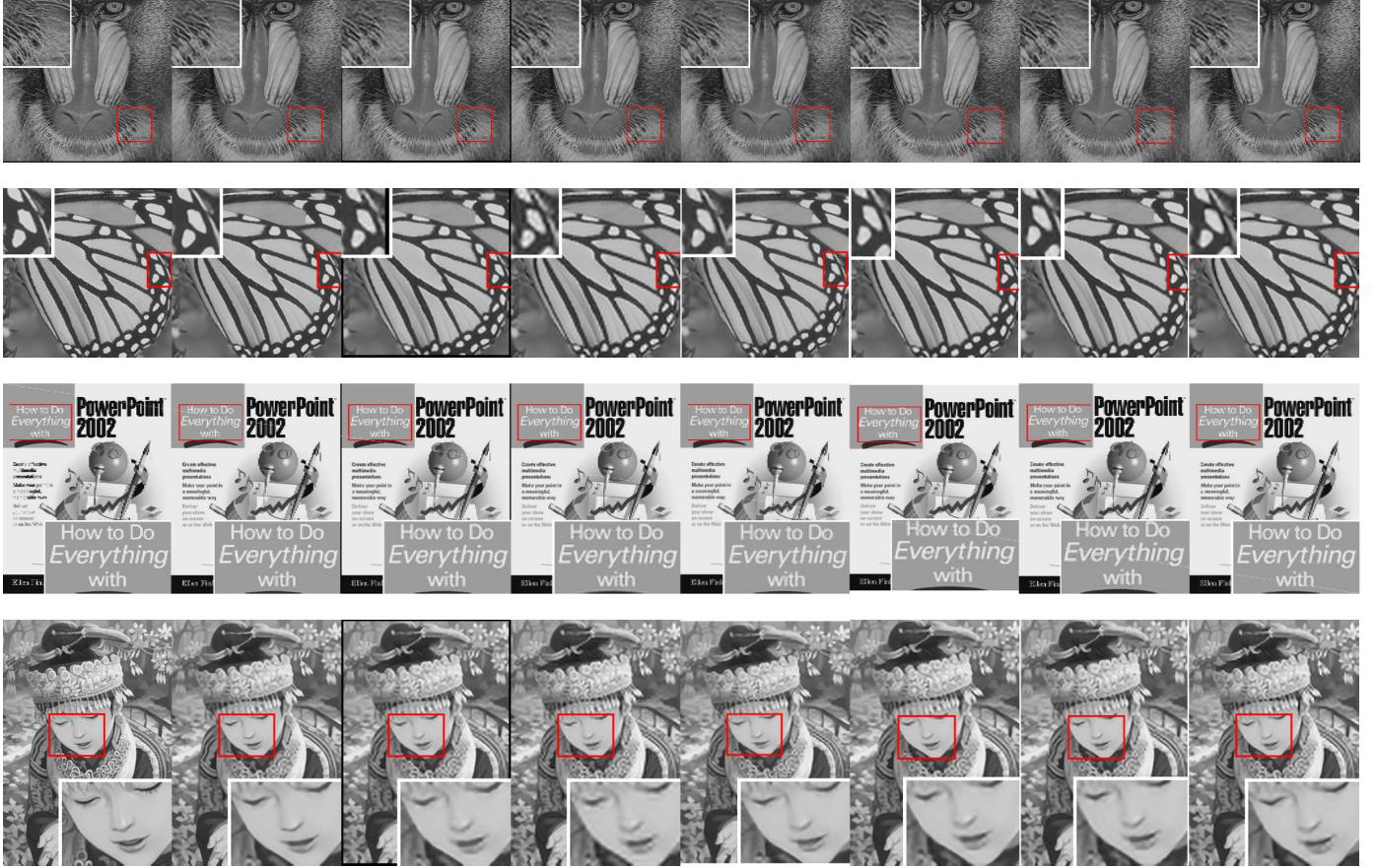


Fig. 15. Visual results of image super-resolution. Images from left to right column are: High resolution image; the recovered image of RED30, ARFL+, CSC, C SCN, NBSRF, SRCNN, TSE.

compared to shallow networks, using significantly deeper networks does improve the deblurring performance.

### 5.5 Non-blind deblurring

We mainly follow the experimental protocols as in [59] for evaluation of non-blind deblurring. The performance on deblurring “disk”, “motion” and “gaussian” kernels are compared, as shown in Table 10. We generate blurred image patches with the corresponding kernels, and train end-to-end mapping with pairs of blurred and non-blurred image patches. As we can see from the results, our network outperforms those compared methods with significant improvements. Figure 16 shows some visual comparisons. We can observe from the visual examples that our network

works better than the compared methods on recovering the image details, as well as achieving visually more appealing results on low frequency image contents.

### 5.6 Image inpainting

In this section, we conduct text removal for experiments of image inpainting. Text is added to the original image from the LIVE1 dataset with font size of 10 and 20. We have compared our method with FoE [19]. For our model, we extract image patches with text on them and learn a mapping from them to the original patches. For FoE, we provide both images with text and masks indicating which pixel is corrupted.

The average PSNR and SSIM for font size 10 and 20 on LIVE are: 38.24dB, 0.9869 and 34.99dB, 0.9828 using 30-layer

TABLE 9  
JPEG compression deblock: average PSNR results of LIVE1.

	SA-DCT	Deeper SRCNN	AR-CNN	RED10	RED20	RED30
Quality = 10	28.65	28.92	28.98	29.24	29.33	<b>29.35</b>
Quality = 20	30.81	-	31.29	31.63	31.71	<b>31.73</b>

TABLE 10  
PSNR results on non-blind deblurring.

kernel type	Krishnan et al. [60]	Levin et al. [61]	Cho et al. [62]	Schuler et al. [63]	Xu et al. [59]	RED30
disk	25.94	24.54	23.97	24.67	26.01	<b>32.13</b>
motion	30.34	37.80	33.25	-	-	<b>38.84</b>
gaussian	27.90	32.34	30.09	30.97	-	<b>34.49</b>



Fig. 16. Visual comparisons on non-blind deblurring. Images from left to right are: blurred images, the results of Cho [62], Krishnan [60], Levin [61], Schuler [63], Xu [59] and our method.

RED-Net, and they are much better than those of FoE, which are 34.59dB, 0.9762 and 31.10dB, 0.9510. For scratch removal, we randomly draw scratch on the clean image and test with our network and FoE. The PSNR and SSIM for our network are 39.41dB and 0.9923, which is much better than 32.92dB and 0.9686 of FoE.

Figure 17 shows some visual comparisons of our method between FoE. We can observe from the examples that our network is better at recovering text, logos, faces and edges in the natural images. Looking on the first example, one may wonder why the text in the original image is not eliminated. For traditional methods such as FoE, this problem is addressed by providing a mask, which indicates the location of corrupted pixels. While our network is trained on specific distributions of corruptions, i.e., the text of font sizes 10 and 20 that are added. It is equivalent to distinguishing corrupted and non-corrupted pixels of different distributions.

## 6 CONCLUSIONS

In this paper we have proposed a deep encoding and decoding framework for image restoration. Convolution and deconvolution are combined, modeling the restoration problem by extracting primary image content and recovering details.

More importantly, we propose to use skip connections, which helps on recovering clean images and tackles the

optimization difficulty caused by gradient vanishing, and thus obtains performance gains when the network goes deeper. Experimental results and our analysis show that our network achieves better performance than state-of-the-art methods on image denoising, image super-resolution, JPEG deblocking and image inpainting.

## REFERENCES

- [1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016.
- [2] J. Mairal, F. R. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2009, pp. 2272–2279.
- [3] W. Dong, L. Zhang, G. Shi, and X. Li, "Nonlocally centralized sparse representation for image restoration," *IEEE Trans. Image Process.*, vol. 22, no. 4, pp. 1620–1630, 2013.
- [4] U. Schmidt, J. Jancsary, S. Nowozin, S. Roth, and C. Rother, "Cascades of regression tree fields for image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 4, pp. 677–689, 2016.
- [5] U. Schmidt and S. Roth, "Shrinkage fields for effective image restoration," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2014, pp. 2774–2781.
- [6] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2011, pp. 479–486.
- [7] H. Liu, R. Xiong, J. Zhang, and W. Gao, "Image denoising via adaptive soft-thresholding based on non-local samples," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015, pp. 484–492.

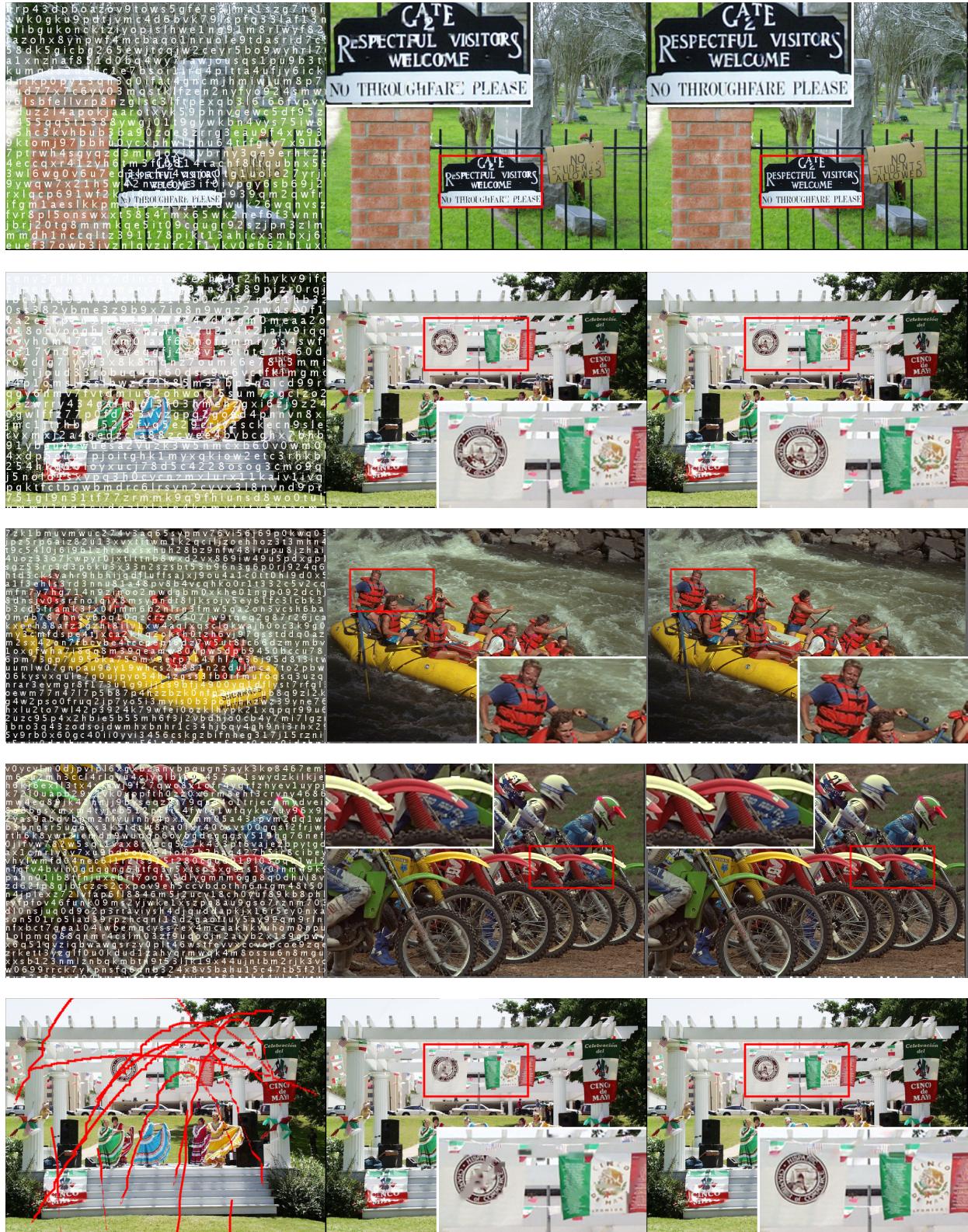


Fig. 17. Visual results of our method and FoE. Images from left to right are: Corrupted images, the inpainting results of FoE and the inpainting results of our method. We see better recovered details as shown in the zoomed patches.

- [8] F. Chen, L. Zhang, and H. Yu, "External patch prior guided internal clustering for image denoising," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2015, pp. 603–611.
- [9] J. Xu, L. Zhang, W. Zuo, D. Zhang, and X. Feng, "Patch group based nonlocal self-similarity prior learning for image denoising," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2015, pp. 244–252.
- [10] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted nuclear norm minimization with application to image denoising," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2014, pp. 2862–2869.
- [11] R. Timofte, V. D. Smet, and L. J. V. Gool, "A+: adjusted anchored neighborhood regression for fast super-resolution," in *Proc. Asian Conf. Comp. Vis.*, 2014, pp. 111–126.
- [12] J. Yang, Z. Lin, and S. Cohen, "Fast image super-resolution based on in-place example regression," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2013, pp. 1059–1066.
- [13] Y. Zhu, Y. Zhang, and A. L. Yuille, "Single image super-resolution using deformable patches," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2014, pp. 2917–2924.
- [14] Y. Zhu, Y. Zhang, B. Bonev, and A. L. Yuille, "Modeling deformable gradient compositions for single-image super-resolution," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015, pp. 5417–5425.
- [15] G. Riegler, S. Schulter, M. Rüther, and H. Bischof, "Conditioned regression models for non-blind single image super-resolution," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2015, pp. 522–530.
- [16] S. Gu, W. Zuo, Q. Xie, D. Meng, X. Feng, and L. Zhang, "Convolutional sparse coding for image super-resolution," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2015, pp. 1823–1831.
- [17] Z. Wang, D. Liu, J. Yang, W. Han, and T. S. Huang, "Deep networks for image super-resolution with sparse prior," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2015, pp. 370–378.
- [18] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Proc. Advances in Neural Inf. Process. Syst.*, 2012, pp. 350–358.
- [19] S. Roth and M. J. Black, "Fields of experts," *Int. J. Comput. Vision*, vol. 82, no. 2, pp. 205–229, 2009.
- [20] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEEE Trans. Image Process.*, vol. 17, no. 1, pp. 53–69, 2008.
- [21] C. Dong, Y. Deng, C. C. Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2015, pp. 576–584.
- [22] A. Foi, V. Katkovnik, and K. O. Egiazarian, "Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images," *IEEE Trans. Image Process.*, vol. 16, no. 5, pp. 1395–1411, 2007.
- [23] J. Jancsary, S. Nowozin, and C. Rother, "Loss-specific training of non-parametric image restoration models: A new state of the art," in *Proc. Eur. Conf. Comp. Vis.*, 2012, pp. 112–125.
- [24] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.
- [25] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. Advances in Neural Inf. Process. Syst.*, 2006, pp. 153–160.
- [26] V. Jain and H. S. Seung, "Natural image denoising with convolutional networks," in *Proc. Advances in Neural Inf. Process. Syst.*, 2008, pp. 769–776.
- [27] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015, pp. 3431–3440.
- [28] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, 2016.
- [29] P. Milanfar, "A tour of modern image filtering: New insights and methods, both practical and theoretical," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 106–128, 2013.
- [30] K. Dabov, A. Foi, V. Katkovnik, and K. O. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Trans. Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.
- [31] Z. Wang, Y. Yang, Z. Wang, S. Chang, J. Yang, and T. S. Huang, "Learning super-resolution jointly from external and internal examples," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 4359–4371, 2015.
- [32] P. Chatterjee and P. Milanfar, "Clustering-based denoising with locally learned dictionaries," *IEEE Trans. Image Process.*, vol. 18, no. 7, pp. 1438–1451, 2009.
- [33] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Phys. D*, vol. 60, no. 1-4, pp. 259–268, November 1992.
- [34] T. Chan, S. Esedoglu, F. Park, and A. Yip, "Recent developments in total variation image restoration," in *In Mathematical Models of Computer Vision*. Springer Verlag, 2005.
- [35] J. Oliveira, J. Bioucas-Dias, and M. A. T. Figueiredo, "Adaptive total variation image deblurring: a majorization-minimization approach," *Signal Processing*, vol. 89, no. 9, pp. 2479–2493, September 2009.
- [36] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, 2006.
- [37] W. Dong, L. Zhang, G. Shi, and X. Wu, "Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization," *IEEE Trans. Image Process.*, vol. 20, no. 7, pp. 1838–1857, 2011.
- [38] K. I. Kim and Y. Kwon, "Single-image super-resolution using sparse regression and natural image prior," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 6, pp. 1127–1133, 2010.
- [39] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2861–2873, 2010.
- [40] Z. Cui, H. Chang, S. Shan, B. Zhong, and X. Chen, "Deep network cascade for image super-resolution," in *Proc. Eur. Conf. Comp. Vis.*, 2014, pp. 49–64.
- [41] K. Sivakumar and U. B. Desai, "Image restoration using a multilayer perceptron with a multilevel sigmoidal function," *IEEE Trans. Signal Processing*, vol. 41, no. 5, pp. 2018–2022, 1993.
- [42] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with BM3D?" in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2012, pp. 2392–2399.
- [43] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2015, pp. 1520–1528.
- [44] S. Hong, H. Noh, and B. Han, "Decoupled deep neural network for semi-supervised semantic segmentation," in *Proc. Advances in Neural Inf. Process. Syst.*, 2015.
- [45] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Proc. Advances in Neural Inf. Process. Syst.*, 2015.
- [46] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
- [47] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM Int. Conf. Multimedia*, 2014, pp. 675–678.
- [48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learning Representations*, 2015.
- [49] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. IEEE Int. Conf. Comp. Vis.*, vol. 2, July 2001, pp. 416–423.
- [50] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comp. Vis.*, 2014.
- [51] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," *Proc. Int. Conf. Learn. Representations*, 2014.
- [52] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Proc. Int. Conf. Learn. Representations*, 2015.
- [53] J. Salvador and E. Perez-Pellitero, "Naive bayes super-resolution forest," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2015, pp. 325–333.
- [54] J. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015, pp. 5197–5206.
- [55] S. Schulter, C. Leistner, and H. Bischof, "Fast and accurate image upscaling with super-resolution forests," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015, pp. 3791–3799.
- [56] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016.
- [57] ———, "Deeply-recursive convolutional network for image super-resolution," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016.
- [58] X. Mao, C. Shen, and Y. Yang, "Image denoising using very deep fully convolutional encoder-decoder networks with symmetric

- skip connections," in *Proc. Advances in Neural Inf. Process. Syst.*, 2016.
- [59] L. Xu, J. S. J. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," in *Proc. Advances in Neural Inf. Process. Syst.*, 2014, pp. 1790–1798.
- [60] D. Krishnan and R. Fergus, "Fast image deconvolution using hyper-laplacian priors," in *Proc. Advances in Neural Inf. Process. Syst.*, 2009, pp. 1033–1041.
- [61] A. Levin, R. Fergus, F. Durand, and W. T. Freeman, "Image and depth from a conventional camera with a coded aperture," *ACM Trans. Graph.*, vol. 26, no. 3, p. 70, 2007.
- [62] S. Cho, J. Wang, and S. Lee, "Handling outliers in non-blind image deconvolution," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2011, pp. 495–502.
- [63] C. J. Schuler, H. C. Burger, S. Harmeling, and B. Schölkopf, "A machine learning approach for non-blind image deconvolution," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2013, pp. 1067–1074.