Image Denoising Using Very Deep Fully Convolutional Encoder-Decoder Networks with Symmetric Skip Connections[*]

Xiao-Jiao Mao[†], Chunhua Shen[⋆], Yu-Bin Yang[†]

[†]State Key Laboratory for Novel Software Technology, Nanjing University, China

[⋆]School of Computer Science, University of Adelaide, Australia

Abstract

Image denoising is a long-standing problem in computer vision and image processing, as well as a test bed for low-level image modeling algorithms. In this paper, we propose a very deep encoding-decoding framework for image denoising. Instead of using image priors, the proposed framework learns end-to-end fully convolutional mappings from noisy images to the clean ones. The network is composed of multiple layers of convolution and de-convolution operators. With the observation that deeper networks improve denoising performance, we propose to use significantly deeper networks than those employed previously for low-level image processing tasks such as denoising.

However, deeper networks tend to be more difficult to train. We propose to symmetrically link convolutional and de-convolutional layers with skip-layer connections, with which the training converges much faster and attains a higher-quality local optimum. From the image processing point of view, those symmetric connections help preserve image details. The convolutional layers act as the feature extractor, which capture the abstraction of image contents while eliminating noise. De-convolutional layers are then used to recover the image details. For complex situations in which the network is very deep, only using convolution may not suffice to well recover the details because much of information is already lost with many feed-forward convolutional layers. The skip connections from convolutional layers to their mirrored corresponding de-convolutional layers exhibit two main advantages. First, they allow the signal being back-propagated to bottom layers directly, and thus tackles the problem of gradient vanishing, making training deep networks easier and achieving denoising performance gains consequently. Second, these skip connections pass image details from convolutional layers to de-convolutional layers, which is beneficial in recovering the clean image. Experiments are carried out on benchmark datasets, and the results show that our network achieves better performance than all previously reported state-of-the-art methods.

Contents

[*]This work was done when the first author was visiting The University of Adelaide.

1 Introduction

Image denoising is a classical problem in low-level vision, which has been widely studies in the literature. Yet, it remains an active topic and provides a test bed for many image modeling techniques.

Generally speaking, a noisy image Y can be represented as $Y = X + N$, where X is the clean version of Y and N is the additive noise, which is often assumed to be Gaussian. By accommodating different types of noise distributions, the same mathematical model applies to most other low-level imaging problems such as image deblur, super-resolution, inpainting, and recovering raw images from compressed images. Traditional methods including bilateral filters [27], total variation [24], sparse representation based methods [11, 9] have been long studied.

Recently, deep neural networks have shown their superior performance in image processing and computer vision tasks, ranging from high-level recognition, semantic segmentation to low-level denoising, super-resolution and deblur. One of the early deep learning models which has been used for image denoising is the Stacked Denoising Auto-encoders (SdA) [28]. It is an extension of the stacked auto-encoder [2] and was originally designed for unsupervised feature learning. Denoising autoencoders can be stacked to form a deep network by feeding output of the previous layer to the current as input. Extensions of SdAs have also been well studied, such as combining sparse coding [29] with SdA for image denoising. Jain and Seung [15] proposed to use Convolutional Neural Networks (CNN) to denoise natural images. Their framework is the same as the recent Fully Convolutional Neural Networks (FCNN) for segmentation [19] and other tasks such as super-resolution [8], although their network is not as deep as these day's models: their network accepts an image as input and produces an entire image as output through 4 hidden layers of convolutional filters. The weights are learned by minimizing the difference between the output and the clean image.

Observing recent superior performance of CNN on image processing tasks, we propose a very deep CNN-based framework for image denoising. The input of our framework is a noisy 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, Our network can be much more deeper compared to that in [15] and recent low-level image processing models such as [8]. Our model is a very deep encoding-decoding framework for image denoising. Instead of using image priors, the proposed framework learns end-to-end fully convolutional mappings from noisy images to the clean ones. The network is composed of multiple layers of convolution and *de-convolution* operators. As deeper networks tend to be more difficult to train, we propose to symmetrically link convolutional and de-convolutional layers with skip-layer connections, with which the training converges much faster and attains a higher-quality local optimum.

Our main contributions are briefly outlined as follows:

1. We propose a very deep network architecture for image denoising. 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 noise. 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 image denoising.

2. To better train the deep network, we propose to add skip connections between corresponding convolutional and de-convolutional 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. The two characteristics make training of the end-to-end mapping from noisy image to the clean one more easier and effective, and thus achieve performance improvement while the network going deeper.

3. We carry out experiments on common benchmark images. The results demonstrate the advantages of our network over other recent state-of-the-art methods on image denoising. Our proposed denoising framework sets new record on image denoising.

The rest of this paper is organized as follows. We provide a review of related work in Section 2. Section 3 presents the architecture of the proposed network, as well as training, testing details and discussions. Experimental results and analysis are provided in Section 4.

2 Related work

Image denoising is an extensively studied topic in image processing and computer vision yet remains a challenging problem. Traditional methods based on wavelet shrinkage, such as soft-thresholding [10], model the wavelet transform coefficients as Laplacian distributions. Total variation [24, 22] based methods assume, e.g., Laplacian distributions of image gradients and have shown some success for image denoising.

Many statistical models of wavelet coefficients have been proposed in the literature, such as Gaussian scale mixture model [23] and generalized Gaussian [7]. Instead of modeling the statistics, models based on image priors on patches are popular too. One representative is the dictionary based methods. Due to the success the K-SVD [1], learning dictionaries for image denoising [4, 11] has been studied extensively.

Natural images usually exhibit local patterns. Since an image patch often has many similar patches across the entire image, nonlocal self-similarity is one of the most

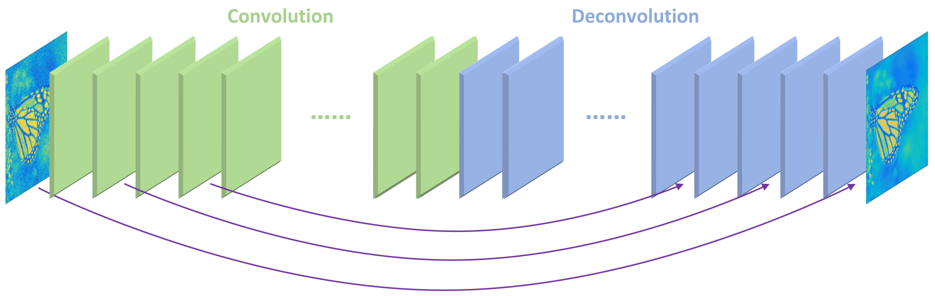


Figure 1: The overall architecture of our proposed network. The network contains layers of symmetric convolution (encoder) and deconvolution (decoder). Skip shortcuts are connected every a few (in our experiments, two) layers from convolutional feature maps to their mirrored deconvolutional feature maps. The skip connections divide the network into several blocks, the size of which is 4, i.e., 2 convolutional and deconvolutional layer respectively. The response from a convolutional layer is directly propagated to the corresponding mirrored deconvolutional layer, both forwardly and backwardly.

successful priors for image denoising. An example is the BM3D algorithm [6] that groups nonlocal similar patches by collaborative filtering in a transformed domain. BM3D has become a benchmark in image denoising. Dong et al. [9] proposed a nonlocally centralized sparse representation model, in which the nonlocal mean is subtracted in the sparse domain in order to regularize the sparse coding. Since image denoising is normally an ill-posed problem, the use of regularization has been proved to be essential. Zoran and Weiss [31] learn clean image patches using Gaussian mixture models and reconstruct them by maximizing the expected patch log-likelihood. In the method of weighted nuclear norm minimization (WNNM) [12], low-rank regularization is used to recover the latent structure of matrix of noisy patches. To further exploit the nonlocal self-similarity of clean images, the patch group prior based denosing (PGPD) method [30] learns explicit nonlocal self-similarity models from clean images and then apply them to noisy images for denoising. Chen et al. [5] proposed a new patch clustering based low-rank regularization framework, in which patch clustering is guided by a learned Gaussian mixture prior to obtain the latent patch subspace with different structures.

The second active (and probably more promising) category for image denoising is the neural network based methods. The most significant difference between neural network methods between the others is that they typically learn parameters for denoising directly from training data (pairs of clean and noisy images) rather than relying on priors. Stacked denoising auto-encoder [28] 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. [29] 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 better performance. Other neural network based denoising methods using multi-layer perceptron or convolutional neural networks. Burger et al. [3] presented a patch-based algorithm learned on a large dataset with a plain multi-layer perceptron. They also concluded that with large networks, large training data, neural networks can achieve state-of-the-art image denoising performance. Jain and Seung [15] proposed convolutional networks for denoising. They found that convolutional networks provide comparable or even superior performance to wavelet and Markov Random Field (MRF) methods.

The advantage of using neural networks for denoising is that they learn denoising parameters directly from data, i.e., end-to-end learning and making no assumptions about natural images. This advantage has been widely witnessed with the success of deep learning in computer vision. Therefore, in this paper we propose to use the neural network architecture for image denoising. However, different from the above methods, here we show that when the network going deeper, we are able to train the network end-to-end and achieve much better denoising performance than existing methods. Our network is a skip connection based encoding-decoding framework, in which both convolution and deconvolution are used for image denoising.

3 Very deep RED-Net for image denoising

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

3.1 Architecture

The framework is fully convolutional and deconvolutional. For low-level image restoration problems, we use neither

Table 1: Configurations of the 20 and 30 layer networks. "conv3" and "deconv3" stand for convolution and deconvolution kernels of size 3×3. 128, 256 and 512 is the number of feature maps after each convolution and deconvolution. "c" is the number of channels of input and output image. In this work, we test on gray-scale images, i.e., $c = 1$. However, it is straightforward to apply to color images.

RED-Net20	RED-Net30
(conv3-128)$\times$4	(conv3-128)$\times$6
(conv3-256)$\times$3	(conv3-256)$\times$6
(conv3-512)$\times$3	(conv3-512)$\times$3
(deconv3-512)$\times$2	(deconv3-512)$\times$2
(deconv3-256)$\times$3	(deconv3-512)$\times$6
(deconv3-128)$\times$4	(deconv3-512)$\times$6
(deconv3-c)	(deconv3-c)

pooling nor unpooling in the network as usually pooling discards useful image details that are essential for denoising and other image restoration tasks.

Rectification layers are added after each convolution and deconvolution. Motivated by the VGG model [25], the kernel size for convolution and deconvolution is set to 3×3, which has shown excellent image recognition performance. The convolutional layers act as feature extractor, which preserve the primary components of objects in the image and meanwhile eliminating the noise. After forwarding through the convolutional layers, the input noisy image is converted into a clean one, but 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 clean version of the input noisy image. It is worth mentioning that since the convolutional and deconvolutional layers are symmetric, the size of input image can be arbitrary. Moreover, skip connections are also added 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 propose two networks, which are 20-layer and 30-layer respectively. The specific configurations of the two networks are described in Table 1.

3.1.1 Deconvolution decoder

Architectures combining layers of convolution and deconvolution [21, 14] have been proposed for semantic segmentation lately. 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. Besides the use of skip connections, the other difference between our model and [21, 14] is that our network is fully convolutional and deconvolutional, i.e., without pooling and un-pooling. The reason is that for image denoising, the aim is to eliminate noise while preserving image details instead of learning image abstractions for classification. Different from high-level applications such as segmentation or recognition, pooling tends to deteriorate denoising performance.

One can simply replace deconvolution with convolution, which results in a architecture that is very similar to recently proposed very deep fully convolutional neural networks [19, 8]. However, there exist essential differences between a fully convolution model and our model. 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. 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 and up-sample the input to make the input and output 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 the same, 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. Figure 3 and Figure 4 visualize some results that are outputs of layer 2, 5, 8 and 10 from the 10-layer fully convolutional network and ours. As we can see, the fully convolutional network reduces noise layer by layer, and our network preserve primary image contents by convolution and recover some details by using deconvolution.

3.1.2 Skip connections

An intuitive question is that, is 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, deconvolution does not work so well, possibly because too much details are already lost in the convolution. The second question is that, when our network goes deeper, does it achieve performance gain? We observe that deeper networks often suffer from gradients vanishing and become hard to train—a problem that is well addressed in the literature.

To address the above two problems, inspired by highway networks [26] and deep residual networks [13], we add skip connections between two corresponding convolutional and deconvolutional layers as shown in Figure 1. A building block is shown in Figure5. There are two reasons for using

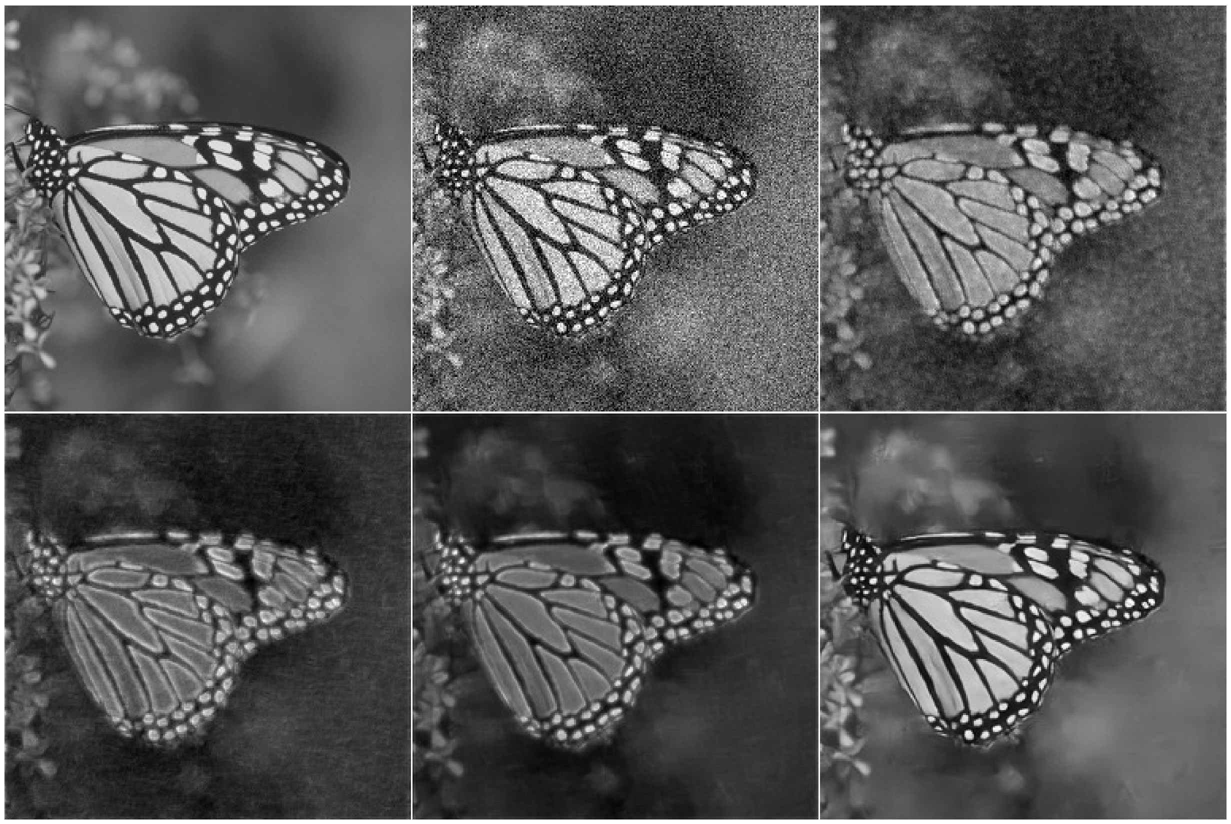


Figure 3: Visualization of the 10-layer fully convolutional network. The images from top-left to bottom-right are: clean image, noisy image, output of conv-2, output of conv-5, output of conv-8 and output of conv-10, where "conv-i" stands for the i-th convolutional layer.

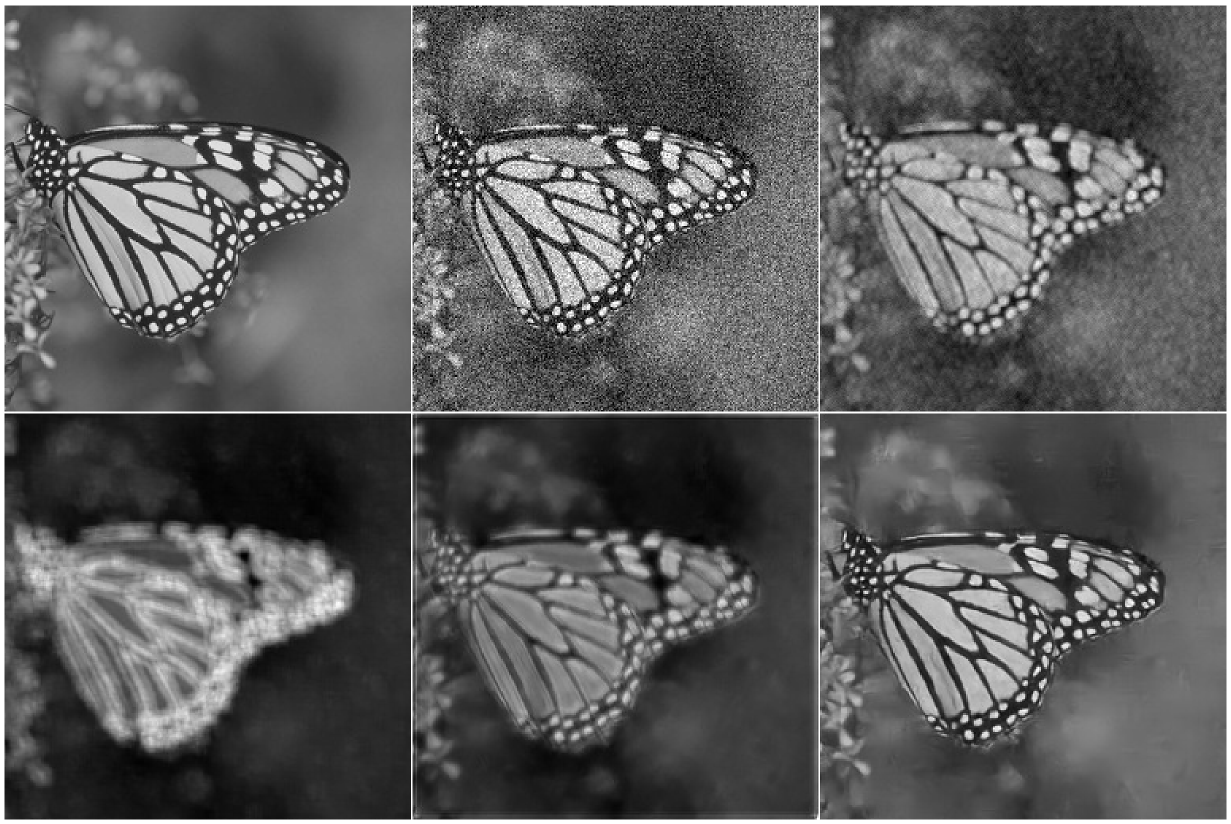


Figure 4: Visualization of the 10-layer convolutional and deconvolutional network. The images from top-left to bottom-right are: clean image, noisy image, output of conv-2, output of conv-5, output of deconv-3 and output of deconv-5, where "conv-i" and "deconv-i" stand for the i-th convolutional and deconvolutional layer respectively.

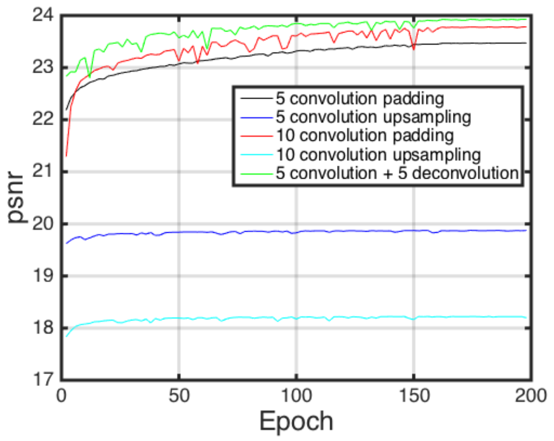


Figure 2: PSNR values on the validation set during training. Our model exhibit better PSNR than compared ones after convergence.

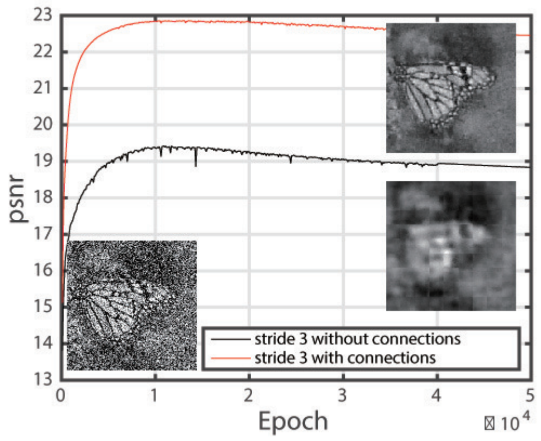


Figure 6: Recovering image details using deconvolution and skip connections. Skip connections are beneficial in recovering image details.

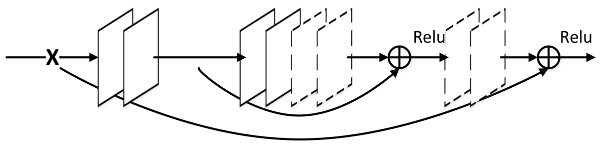


Figure 5: An example of a building block in the proposed framework. The rectangle in solid and dotted lines denote convolution and deconvolution respectively. $\oplus$ denotes element-wise sum of feature maps.

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 a better clean image. Second, the skip connections also achieve benefits on back-propagating the gradient to bottom layer, which makes training deeper network much easier as observed in [26] and [13]. Note that our skip layer connections are very different from the ones proposed in [26] and [13], 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.

Instead of directly learning the mappings from input X to the output Y, we would like the network to fit the residual [13] of the problem, which is denoted as $\mathcal{F}(X) = Y - X$. Such 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 that this configuration already works very well. Using such shortcuts makes the network easier to be trained and gains denoising performance via increasing network depth.

The very deep highway networks [26] are essentially feedforward long short-term memory (LSTMs) with forget gates; and the CNN layers of deep residual network [13] are feedforward LSTMs without gates. Note that our deep residual networks are in general not in the format of standard feedforward LSTMs.

3.2 Discussions

The aim of denoising is to eliminate noise while preserving the image details as mush as possible. In our network, convolution eliminates the noisy signals while learning abstraction of image contents. Then, we use deconvolution to recover the details of image. However, for simple situations, such as the network is not that deep, deconvolution works well since not much details are lost. However, if too much of them are lost, we observe that deconvolution does not work well any more. We design experiments to show this observation, in which two networks are trained for denoising noises of $\sigma = 70$. In the first network, we use 5 layers of 3×3 convolution with stride 3. The input size of training data is 243×243, which results in a vector after 5 layers of convolution. Then deconvolution is used to recover the input. The second network uses the same settings as the first one, except for adding skip connections. The results are show in Figure 6. We can observe that it is hard for deconvolution to recover details from only a vector encoding the abstraction of the input, which shows that the ability on recovering image details for deconvolution is limited. However, if we use skip connections, the network can still recover the input, because details are passed to topper layers in the network. On the other hand, they help to back-propagate gradient in training to better fit the end-to-end mapping.

Comparison to deep residual network [13] One may use different types of skip connections in our network, a straightforward alternate is that in [13]. In [13], the 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

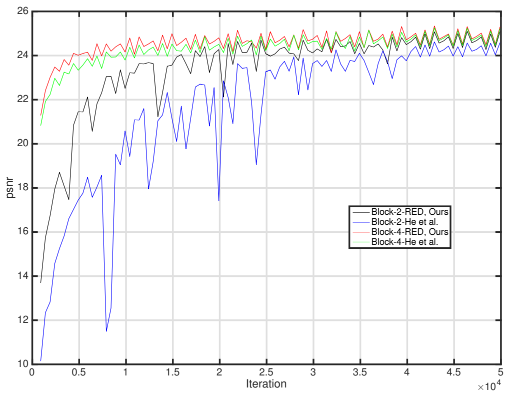


Figure 7: Comparisons of skip connections in [13] and our model, where "Block-i-RED" is the connections in our model with block size i and "Block-i-He et al." is the connections in He et al. [13] with block size i. The PSNR at the last iteration for the curves are: 25.08, 24.59, 25.30 and 25.21.

be very important in pixel-wise prediction problems such image denoising. We carry out experiments to compare the 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.

3.3 Training

Learning the end-to-end mapping from noisy images to clean images needs to estimate the weights Θ represented by the convolutional and deconvolutional kernels. This is achieved by minimizing the Euclidean loss between the outputs of the network and the clean image. In specific, 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):

$$\mathcal{L}(\Theta) = \frac{1}{n} \sum_{n=1}^{N} \|\mathcal{F}(X_i; \Theta) - Y_i\|_F^2. \tag{1}$$

We implement and train our network using Caffe [16]. The filter weights are initialized using the "Xavier" strategy. In practice, we find that using Adam [17] with learning rate $10^{(}-4)$ for training converges faster than traditional stochastic gradient descent (SGD). The base learning rate for all layers are the same, different from [8, 15], in which a smaller learning rate is set for the last layer. This trick is not necessary in our network.

Gray-scale images are used for experiments in this paper, although it is easy to generalize to color images for our network. 300 images from the Berkeley Segmentation Dataset (BSD) [20] are used to generate the training set. For each image, non-overlapping patches of size 50×50 are sampled as ground truth. We observe that using a smaller patch size leads to worse performance, which is probably because it is not sufficient to well represent the distribution of noise with fewer pixels. For each patch, we add Gaussian noise as the noisy image. To obtain a large dataset for training, a patch is used multiple times to generate noisy image, thus a training set of size 0.5M is formed. For different noise levels, we form different training data to train corresponding models.

3.4 Denoising images

The network is trained to perform denoising on images of arbitrary size. Given a testing noisy image, one can simply forward through the network, which is able to obtain a better performance than existing methods. To achieve more smooth denoising results, we propose to denoise a noisy image on multiple orientations. Different from segmentation, the filter kernels in our network only eliminate the noisy signal, which is not sensitive to the orientation of image contents. Therefore, we can rotate and mirror, flip the kernels and perform forward multiple times, and then average the output to get a more smooth denoised image. We see that this can lead to slightly better denoising performance.

4 Experiments

In this section, we first provide an analysis on skip connections. Then evaluation of denoising performance of our models against a few existing state-of-the-art methods is carried out.

Denoising experiments are performed on two datasets: 14 common benchmark images [30, 5, 18, 12], as show in Figure 11 and the BSD dataset. As a common experimental setting in the literature, additive Gaussian noises with zero mean and standard deviation σ are added to the image to test the performance of denoising methods.

In this paper we test noise level σ of 10, 30, 50 and 70. BM3D [6], NCSR [9], EPLL [31], PCLR [5], PDPD [30] and WMMN [12] are compared with our method, which is denoted as RED-Net. For these methods, we use the source code released by their authors and test on the images with their default parameters. Peak Signal-to-Noise Ratio (PSNR) and Structural SIMilarity (SSIM) index are calculated for evaluation. Visual quality comparisons are also provided. For our method, 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.

4.1 Analysis on skip connections

Experimental comparisons are shown in Figures 9 and 10, in which we compare 5 different networks including 10-

Figure 8: The testing images for denoising.

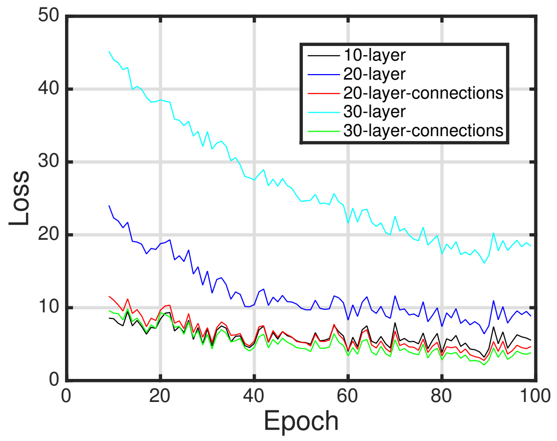


Figure 9: The training loss on the training set during training.

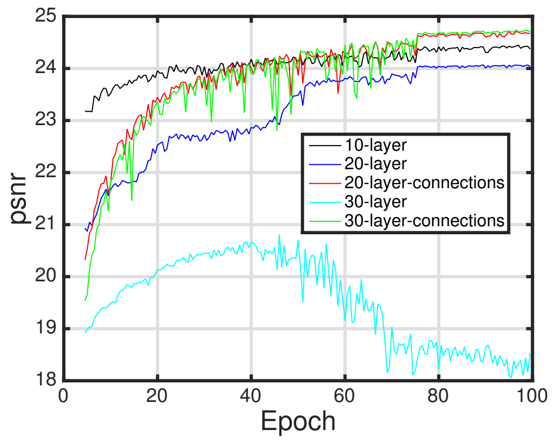


Figure 10: PSNR values on the validation set during training.

layer, 20-layer, 30-layer, 20-layer with skip connections and 30-layer with skip connections. On the training set, the training loss increases when the network going deeper without shortcuts, but we obtain smaller loss when using skip connections. On the validation set, deeper networks without shortcuts achieve lower PSNR and we even observe over-fitting for the 30-layer network. However, using shortcuts obtains higher PSNR and better generalization ability when networks go deeper.

We also provide the experiments of different step sizes of shortcuts, as shown in Figure 11 and Figure 12. As we can see, a smaller step size of shortcuts achieves better performance than a bigger one. The reason may be 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.

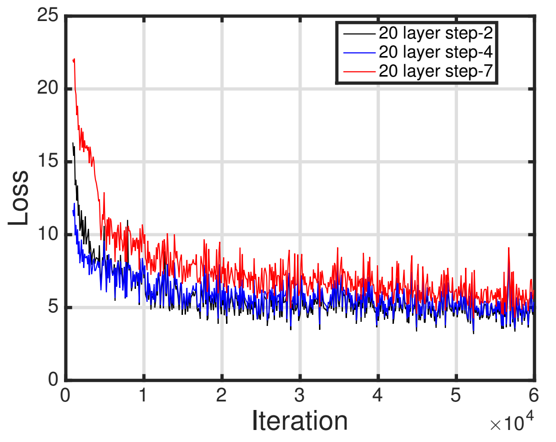


Figure 11: Loss on the training set during training.

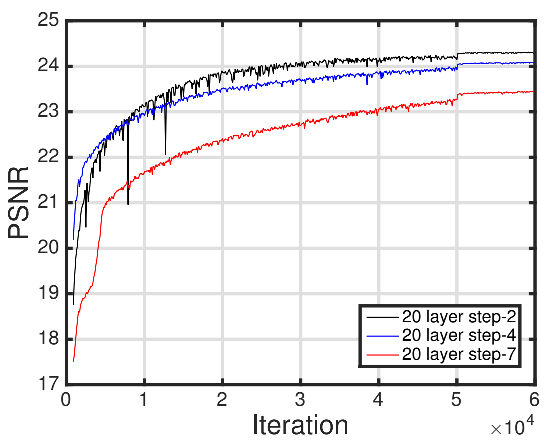


Figure 12: PSNR values on the validation set during training.

4.2 Denoising results and discussions

4.2.1 Evaluation on 14 images

Table 2 and Table 3 present the PSNR and SSIM results of σ 10, 30, 50, and 70 image-wise. 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 [12] by 0.32dB, 0.43dB, 0.49dB and 0.51dB on noise levels of σ being 10, 30, 50 and 70 respectively. While WNNM is only slightly better than the second best existing method PCLR [5] 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.

4.2.2 Evaluation on the BSD dataset

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 σ 10, 30, 50, and 70, as shown in Table 4. 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.

4.2.3 Visual quality

Figures 13 and 14 show some visual quality comparisons of our model and existing methods. At a lower noise level, e.g., $\sigma = 30$, besides that our model achieves higher PSNR, it also shows visually better results than others. For higher level of noises, existing methods can easily fail to recover the details of objects, but our model achieves better results than others.

5 Conclusions

In this paper we have proposed a deep encoding and decoding framework for image denoising. Convolution and deconvolution are combined, modeling the denoising problem by extracting primary image content and recovering details, which achieves better performance than using fully convolutions. Meanwhile, we propose to use skip connections for training, 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 denoising performance than state-of-the-art methods. We are applying this framework to other low-level image processing tasks such as image super-resolution, inpaintinng and deblur.

Acknowledgements

C. Shen's participation was in part supported by an ARC Future Fellowship (FT120100969). C. Shen is the corresponding author (e-mail: `chunhua.shen@adelaide.edu.au`).

References

[1] M. Aharon, M. Elad, and A. Bruckstein. SVDD: An algorithm for designing overcomplete dictionaries for sparse representation. *Trans. Sig. Proc.*, 54(11):4311–4322, November 2006.

[2] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 153–160, 2006.

[3] 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.*, pages 2392–2399, 2012.

[4] P. Chatterjee and P. Milanfar. Clustering-based denoising with locally learned dictionaries. *IEEE Trans. Image Processing*, 18(7):1438–1451, 2009.

[5] F. Chen, L. Zhang, and H. Yu. External patch prior guided internal clustering for image denoising. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 603–611, 2015.

[6] K. Dabov, A. Foi, V. Katkovnik, and K. O. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Trans. Image Processing*, 16(8):2080–2095, 2007.

[7] M. N. Do and M. Vetterli. Wavelet-based texture retrieval using generalized gaussian density and kullback-leibler distance. *IEEE Trans. Image Processing*, 11(2):146–158, 2002.

[8] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2):295–307, 2016.

Table 2: PSNR results of different σ.

	$\sigma = 10$														
	Plane	Baboon	Hill	Barbara	Boat	C.Man	Couple	House	Lake	Lena	Man	Pepper	Monarch	C.house	Average
BM3D	34.35	30.89	33.44	34.71	33.50	35.34	33.57	37.51	32.83	35.36	33.11	35.57	35.08	33.19	34.18
EPLL	34.35	30.94	33.45	33.59	33.44	35.05	33.34	36.71	32.93	35.10	33.22	35.45	35.10	33.08	33.98
NCSR	34.53	30.94	33.54	34.67	33.58	35.44	33.53	37.48	32.98	35.49	33.28	35.62	35.43	33.21	34.27
PCLR	34.72	31.03	33.69	34.78	33.81	35.73	33.75	37.65	33.26	35.63	33.44	35.92	35.94	33.36	34.48
PGPD	34.49	30.96	33.54	34.26	33.52	35.27	33.60	37.31	33.04	35.51	33.27	35.68	35.29	33.29	34.22
WNNM	34.71	31.05	33.60	35.23	33.77	35.70	33.69	37.45	33.14	35.71	33.43	35.90	36.09	33.44	34.49
RED10	34.99	31.15	33.81	34.17	34.04	35.77	33.92	37.49	33.42	35.87	33.72	36.11	36.62	33.56	**34.62**
RED20	35.07	31.22	33.89	34.39	34.14	35.95	34.07	37.64	33.48	36.06	33.79	36.26	36.81	33.64	**34.74**
RED30	35.11	31.24	33.95	34.64	34.21	35.95	34.17	37.72	33.51	36.14	33.82	36.27	36.84	33.70	**34.81**

	$\sigma = 30$														
	Plane	Baboon	Hill	Barbara	Boat	C.Man	Couple	House	Lake	Lena	Man	Pepper	Monarch	C.house	Average
BM3D	28.57	24.88	28.40	28.99	27.71	29.89	27.63	32.58	26.74	29.63	27.80	30.00	28.65	27.45	28.49
EPLL	28.58	25.15	28.30	28.47	27.83	29.62	27.55	31.58	27.01	29.27	27.82	29.78	28.55	27.41	28.35
NCSR	28.53	24.99	28.07	28.78	27.56	29.80	27.26	32.78	26.81	29.81	27.65	29.88	28.99	27.31	28.44
PCLR	28.84	25.07	28.31	28.94	28.04	30.06	27.68	32.63	27.22	29.79	27.90	30.37	29.17	27.50	28.68
PGPD	28.69	24.98	28.30	28.93	27.82	30.00	27.61	32.60	26.97	29.75	27.82	30.09	28.72	27.45	28.55
WNNM	28.82	25.09	28.36	29.33	27.84	30.04	27.73	33.07	27.05	29.86	27.89	30.28	29.48	27.55	28.74
RED10	29.20	25.31	28.64	28.95	28.15	30.22	27.95	32.49	27.37	30.06	28.24	30.56	30.38	27.84	**28.95**
RED20	29.30	25.42	28.76	29.10	28.27	30.40	28.12	32.82	27.44	30.25	28.32	30.74	30.53	27.95	**29.10**
RED30	29.34	25.46	28.81	29.18	28.31	30.49	28.20	33.02	27.49	30.35	28.35	30.79	30.57	28.00	**29.17**

	$\sigma = 50$														
	Plane	Baboon	Hill	Barbara	Boat	C.Man	Couple	House	Lake	Lena	Man	Pepper	Monarch	C.house	Average
BM3D	26.06	22.97	26.28	26.75	25.35	27.38	25.11	30.35	24.26	27.19	25.50	27.16	25.81	24.99	26.08
EPLL	26.10	23.05	26.06	26.31	25.55	27.11	25.11	29.16	24.49	27.06	25.63	27.18	25.71	25.04	25.97
NCSR	25.80	23.13	26.09	26.47	25.09	27.08	24.83	30.23	24.28	27.11	25.18	27.17	25.80	24.79	25.93
PCLR	26.28	23.16	26.28	26.65	25.51	27.68	25.23	30.32	24.74	27.29	25.80	27.58	26.32	25.25	26.29
PGPD	26.15	23.08	26.15	26.78	25.39	27.56	25.29	30.23	24.49	27.34	25.65	27.47	26.10	24.94	26.19
WNNM	26.24	23.05	26.40	26.90	25.60	27.61	25.11	30.74	24.61	27.54	25.56	27.49	26.41	25.27	26.32
RED10	26.61	23.31	26.45	26.88	25.83	27.82	25.50	29.91	24.91	27.57	25.99	27.77	27.29	25.36	**26.51**
RED20	26.78	23.42	26.65	27.11	25.98	28.08	25.69	30.51	24.97	27.83	26.07	27.97	27.51	25.53	**26.72**
RED30	26.84	23.49	26.71	27.18	26.03	28.19	25.80	30.75	25.02	27.93	26.13	28.05	27.62	25.63	**26.81**

	$\sigma = 70$														
	Plane	Baboon	Hill	Barbara	Boat	C.Man	Couple	House	Lake	Lena	Man	Pepper	Monarch	C.house	Average
BM3D	24.52	22.22	24.87	25.44	24.03	25.66	23.85	28.47	22.96	25.77	24.16	25.37	24.07	23.64	24.65
EPLL	24.41	22.16	24.82	24.87	24.09	25.45	23.74	27.22	23.11	25.34	24.18	25.57	23.98	23.63	24.47
NCSR	24.39	21.99	24.66	24.95	23.74	25.46	23.43	28.15	22.76	25.58	23.89	25.23	23.59	23.26	24.36
PCLR	24.64	22.05	24.91	25.25	24.10	25.92	23.83	28.40	23.24	25.97	24.40	25.94	24.60	23.77	24.79
PGPD	24.60	22.25	25.10	25.31	24.04	26.11	23.72	28.34	23.03	25.84	24.30	25.61	23.94	23.80	24.71
WNNM	24.66	22.21	24.90	25.46	24.06	25.93	23.81	28.68	23.19	25.86	24.21	25.78	24.66	23.73	24.80
RED10	25.00	22.31	25.02	25.49	24.52	26.04	24.05	28.17	23.45	26.09	24.49	25.84	25.22	23.92	**24.97**
RED20	25.17	22.46	25.23	25.85	24.72	26.42	24.28	28.77	23.59	26.37	24.60	26.07	25.58	24.15	**25.23**
RED30	25.18	22.50	25.30	25.93	24.80	26.49	24.35	28.99	23.58	26.44	24.61	26.16	25.71	24.27	**25.31**

Table 3: SSIM results of different σ.

	σ = 10														
	Plane	Baboon	Hill	Barbara	Boat	C.Man	Couple	House	Lake	Lena	Man	Pepper	Monarch	C.house	Average
BM3D	0.9442	0.9053	0.9063	0.9390	0.9244	0.9415	0.9287	0.9361	0.9363	0.9457	0.9138	0.9436	0.9734	0.9358	0.9339
EPLL	0.9443	0.9099	0.9105	0.9194	0.9263	0.9440	0.9252	0.9333	0.9376	0.9416	0.9189	0.9433	0.9729	0.9376	0.9332
NCSR	0.9445	0.9031	0.9079	0.9379	0.9245	0.9435	0.9264	0.9377	0.9362	0.9460	0.9155	0.9427	0.9749	0.9372	0.9342
PCLR	0.9461	0.9079	0.9112	0.9395	0.9284	0.9446	0.9304	0.9382	0.9398	0.9474	0.9174	0.9466	0.9765	0.9387	0.9366
PGPD	0.9415	0.9011	0.9078	0.9273	0.9222	0.9350	0.9276	0.9327	0.9362	0.9437	0.9121	0.9419	0.9683	0.9358	0.9309
WNNM	0.9455	0.9074	0.9094	0.9447	0.9277	0.9438	0.9297	0.9331	0.9382	0.9493	0.9178	0.9462	0.9767	0.9384	0.9363
RED10	0.9475	0.9123	0.9129	0.9255	0.9318	0.9468	0.9315	0.9369	0.9417	0.9489	0.9233	0.9468	0.9771	0.9405	**0.9374**
RED20	0.9489	0.9140	0.9146	0.9285	0.9335	0.9481	0.9343	0.9381	0.9431	0.9519	0.9245	0.9483	0.9794	0.9416	**0.9392**
RED30	0.9495	0.9149	0.9153	0.9325	0.9347	0.9482	0.9363	0.9388	0.9435	0.9529	0.9249	0.9484	0.9803	0.9427	**0.9402**
	σ = 30														
	Plane	Baboon	Hill	Barbara	Boat	C.Man	Couple	House	Lake	Lena	Man	Pepper	Monarch	C.house	Average
BM3D	0.8632	0.6624	0.7663	0.8210	0.7851	0.8597	0.7871	0.8777	0.8230	0.8584	0.7746	0.8721	0.9201	0.8151	0.8204
EPLL	0.8608	0.6966	0.7688	0.8010	0.7936	0.8532	0.7843	0.8632	0.8313	0.8460	0.7777	0.8661	0.9139	0.8237	0.8200
NCSR	0.8645	0.6687	0.7547	0.8215	0.7789	0.8630	0.7685	0.8840	0.8260	0.8639	0.7668	0.8751	0.9299	0.8189	0.8203
PCLR	0.8712	0.6784	0.7622	0.8145	0.7905	0.8667	0.7884	0.8811	0.8385	0.8617	0.7805	0.8815	0.9323	0.8210	0.8263
PGPD	0.8626	0.6652	0.7627	0.8127	0.7866	0.8569	0.7845	0.8789	0.8285	0.8590	0.7726	0.8742	0.9192	0.8148	0.8199
WNNM	0.8660	0.6812	0.7648	0.8336	0.7884	0.8637	0.7957	0.8815	0.8364	0.8650	0.7736	0.8797	0.9306	0.8214	0.8273
RED10	0.8758	0.6998	0.7741	0.8154	0.7973	0.8673	0.7959	0.8792	0.8418	0.8692	0.7894	0.8826	0.9385	0.8309	**0.8327**
RED20	0.8797	0.7137	0.7814	0.8218	0.8056	0.8744	0.8068	0.8835	0.8456	0.8767	0.7947	0.8882	0.9447	0.8380	**0.8396**
RED30	0.8813	0.7167	0.7826	0.8246	0.8084	0.8781	0.8113	0.8861	0.8483	0.8801	0.7957	0.8907	0.9484	0.8399	**0.8423**
	σ = 50														
	Plane	Baboon	Hill	Barbara	Boat	C.Man	Couple	House	Lake	Lena	Man	Pepper	Monarch	C.house	Average
BM3D	0.8048	0.5086	0.6787	0.7527	0.6947	0.8129	0.6756	0.8508	0.7366	0.7959	0.6827	0.8099	0.8685	0.7250	0.7427
EPLL	0.7945	0.5361	0.6689	0.7296	0.7027	0.7900	0.6708	0.8146	0.7466	0.7787	0.6855	0.8013	0.8483	0.7273	0.7354
NCSR	0.8041	0.5189	0.6618	0.7421	0.6884	0.8129	0.6580	0.8564	0.7391	0.7986	0.6709	0.8213	0.8855	0.7233	0.7415
PCLR	0.8185	0.5323	0.6796	0.7449	0.6986	0.8229	0.6816	0.8544	0.7606	0.8012	0.6970	0.8259	0.8897	0.7455	0.7538
PGPD	0.8019	0.5177	0.6699	0.7459	0.6978	0.8069	0.6798	0.8486	0.7417	0.7961	0.6922	0.8200	0.8740	0.7265	0.7442
WNNM	0.8081	0.5208	0.6844	0.7553	0.7054	0.8146	0.6712	0.8570	0.7554	0.8098	0.6900	0.8205	0.8864	0.7443	0.7517
RED10	0.8195	0.5496	0.6826	0.7534	0.7130	0.8225	0.6899	0.8451	0.7636	0.8042	0.7020	0.8225	0.8832	0.7483	**0.7571**
RED20	0.8295	0.5609	0.6950	0.7654	0.7234	0.8324	0.7071	0.8596	0.7710	0.8182	0.7084	0.8349	0.8991	0.7591	**0.7689**
RED30	0.8313	0.5720	0.6985	0.7684	0.7280	0.8352	0.7150	0.8638	0.7751	0.8224	0.7114	0.8372	0.9029	0.7645	**0.7733**
	σ = 70														
	Plane	Baboon	Hill	Barbara	Boat	C.Man	Couple	House	Lake	Lena	Man	Pepper	Monarch	C.house	Average
BM3D	0.7533	0.4425	0.6114	0.7001	0.6367	0.7648	0.6088	0.8148	0.6820	0.7418	0.6190	0.7582	0.8246	0.6769	0.6882
EPLL	0.7273	0.4450	0.6020	0.6725	0.6337	0.7308	0.5939	0.7668	0.6748	0.7181	0.6174	0.7569	0.8002	0.6573	0.6712
NCSR	0.7639	0.4200	0.5999	0.6888	0.6211	0.7786	0.5840	0.8268	0.6853	0.7553	0.6141	0.7756	0.8332	0.6734	0.6871
PCLR	0.7728	0.4301	0.6045	0.6938	0.6430	0.7837	0.6077	0.8266	0.7084	0.7627	0.6316	0.7870	0.8589	0.6850	0.6997
PGPD	0.7597	0.4441	0.6235	0.6921	0.6353	0.7732	0.6025	0.8130	0.6765	0.7516	0.6295	0.7708	0.8295	0.6776	0.6913
WNNM	0.7623	0.4524	0.6193	0.7019	0.6390	0.7794	0.6084	0.8205	0.6973	0.7566	0.6240	0.7739	0.8459	0.6841	0.6975
RED10	0.7744	0.4440	0.6143	0.6987	0.6576	0.7838	0.6170	0.8170	0.7079	0.7521	0.6372	0.7759	0.8479	0.6895	**0.7012**
RED20	0.7869	0.4673	0.6326	0.7196	0.6700	0.7999	0.6382	0.8322	0.7187	0.7668	0.6471	0.7893	0.8684	0.7109	**0.7177**
RED30	0.7879	0.4725	0.6370	0.7216	0.6745	0.8034	0.6442	0.8379	0.7188	0.7678	0.6478	0.7902	0.8698	0.7153	**0.7206**

Table 4: Average PSNR and SSIM results of σ 10, 30, 50, 70 on 200 images from 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**	**33.63**
$\sigma = 30$	27.31	27.38	27.23	27.54	27.33	27.48	**27.79**	**27.90**	**27.95**
$\sigma = 50$	25.06	25.17	24.95	25.30	25.18	25.26	**25.54**	**25.67**	**25.75**
$\sigma = 70$	23.82	23.81	23.58	23.94	23.89	23.95	**24.13**	**24.33**	**24.37**
	SSIM								
$\sigma = 10$	0.9218	0.9255	0.9226	0.9261	0.9176	0.9244	**0.9290**	**0.9310**	**0.9319**
$\sigma = 30$	0.7755	0.7825	0.7738	0.7827	0.7717	0.7807	**0.7918**	**0.7993**	**0.8019**
$\sigma = 50$	0.6831	0.6870	0.6777	0.6947	0.6841	0.6928	**0.7032**	**0.7117**	**0.7167**
$\sigma = 70$	0.6240	0.6168	0.6166	0.6336	0.6245	0.6346	**0.6367**	**0.6521**	**0.6551**

[9] W. Dong, L. Zhang, G. Shi, and X. Li. Nonlocally centralized sparse representation for image restoration. *IEEE Trans. Image Processing*, 22(4):1620–1630, 2013.

[10] D. L. Donoho. De-noising by soft-thresholding. *IEEE Trans. Information Theory*, 41(3):613–627, 1995.

[11] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. Image Processing*, 15(12):3736–3745, 2006.

[12] 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.*, pages 2862–2869, 2014.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, volume abs/1512.03385, 2016.

[14] 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.

[15] V. Jain and H. S. Seung. Natural image denoising with convolutional networks. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 769–776, 2008.

[16] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learning Representations*, 2015.

[18] 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.*, pages 484–492, 2015.

[19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 3431–3440, 2015.

[20] 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.*, volume 2, pages 416–423, July 2001.

[21] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 1520–1528, 2015.

[22] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling & Simulation*, 4(2):460–489, 2005.

[23] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Trans. Image Processing*, 12(11):1338–1351, 2003.

[24] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268, November 1992.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. 2015.

[26] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In *Proc. Advances in Neural Inf. Process. Syst.*, 2015.

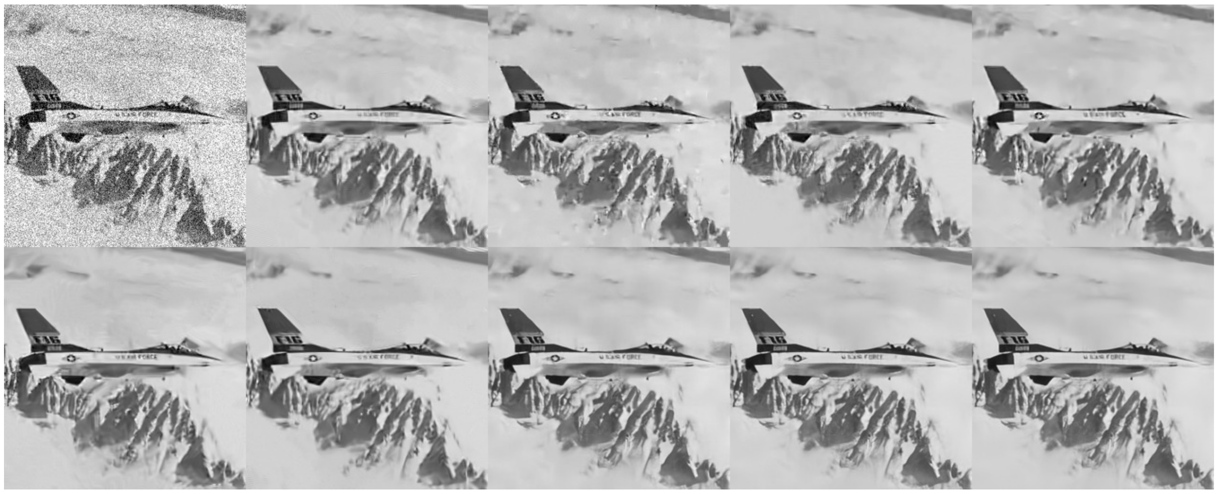
[27] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 839–846, 1998.

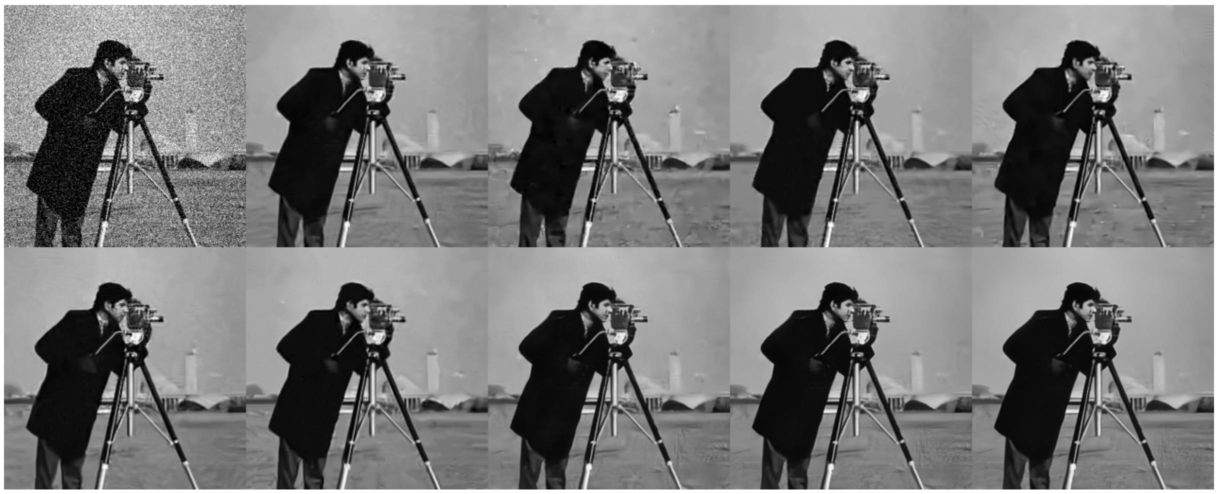
[28] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proc. Int. Conf. Mach. Learn.*, pages 1096–1103, 2008.

[29] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 350–358, 2012.

[30] 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.*, pages 244–252, 2015.

[31] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 479–486, 2011.


(a)


(b)

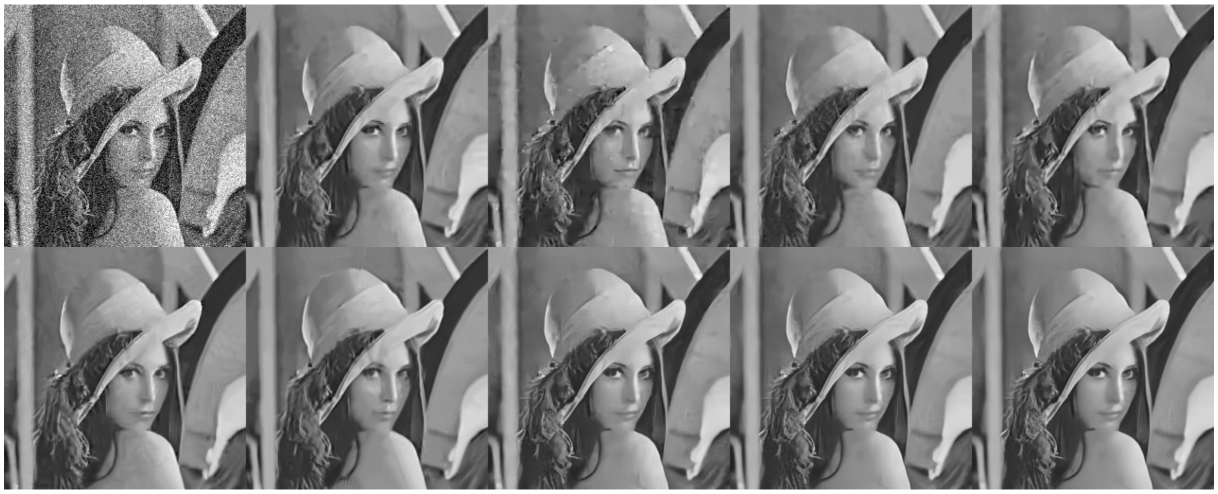
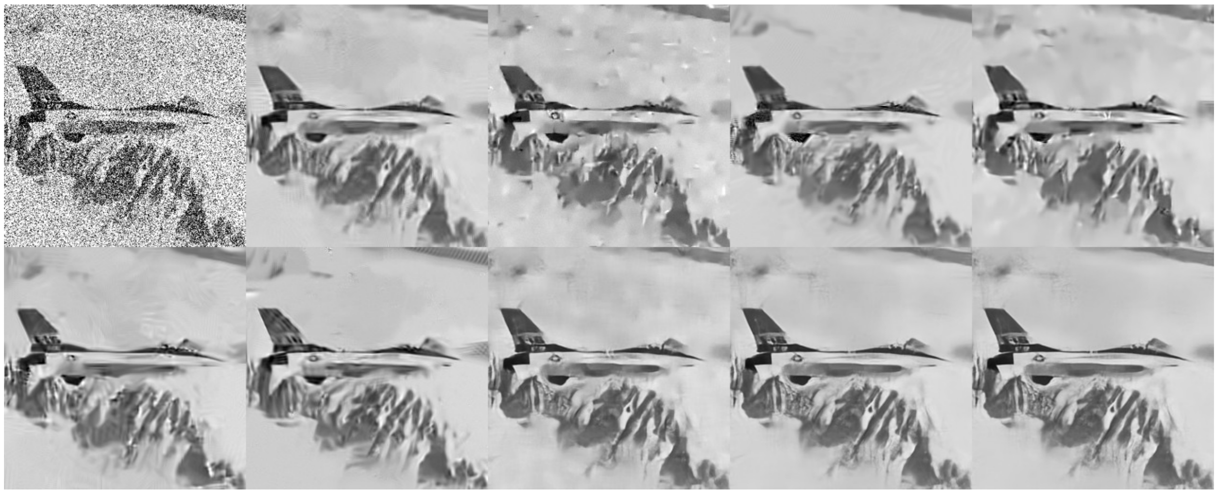
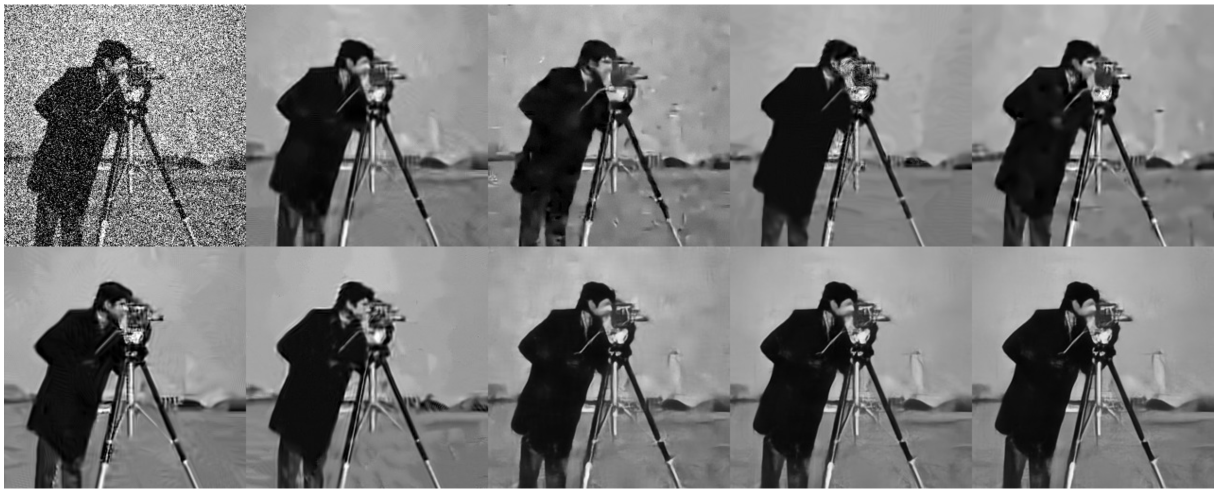

(c)

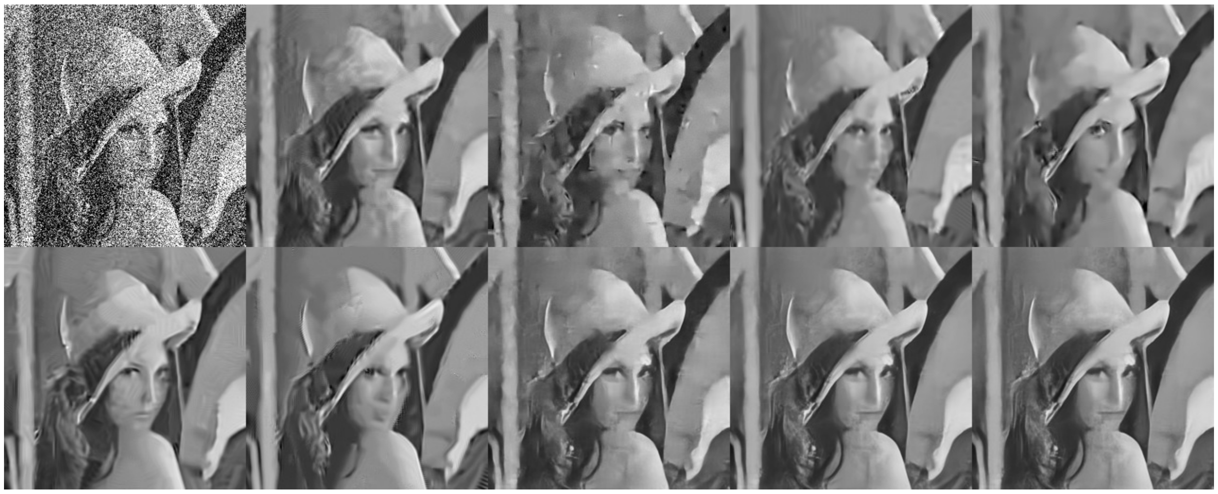
Figure 13: Denoising results on "Plane", "Camera Man" and "Lena" with $\sigma = 30$ respectively. The results of each image from top-left to right-bottom are: noisy image; BM3D; NPLL; NCSR; PCLR; PGPD; WNNM; RED10; RED20; RED-30.



(a)



(b)



(c)

Figure 14: Denoising results on “Plane”, “Camera Man” and “Lena” with $\sigma = 70$ respectively. The results of each image from top-left to right-bottom are: noisy image; BM3D; NPLL; NCSR; PCLR; PGPD; WNNM; RED10; RED20; RED-30.