MemNet: A Persistent Memory Network for Image Restoration

Ying Tai* [1], Jian Yang[1], Xiaoming Liu[2], and Chunyan Xu[1]

[1]Department of Computer Science and Engineering, Nanjing University of Science and Technology
[2]Department of Computer Science and Engineering, Michigan State University
{taiying, csjyang, cyx}@njust.edu.cn, liuxm@cse.msu.edu

Abstract

Recently, very deep convolutional neural networks (CNNs) have been attracting considerable attention in image restoration. However, as the depth grows, the long-term dependency problem is rarely realized for these very deep models, which results in the prior states/layers having little influence on the subsequent ones. Motivated by the fact that human thoughts have persistency, we propose a very deep persistent memory network (MemNet) that introduces a memory block, consisting of a recursive unit and a gate unit, to explicitly mine persistent memory through an adaptive learning process. The recursive unit learns multi-level representations of the current state under different receptive fields. The representations and the outputs from the previous memory blocks are concatenated and sent to the gate unit, which adaptively controls how much of the previous states should be reserved, and decides how much of the current state should be stored. We apply MemNet to three image restoration tasks, i.e., image denosing, super-resolution and JPEG deblocking. Comprehensive experiments demonstrate the necessity of the MemNet and its unanimous superiority on all three tasks over the state of the arts. Code is available at `https://github.com/tyshiwo/MemNet`.

1. Introduction

Image restoration [29] is a classical problem in low-level computer vision, which estimates an uncorrupted image from a noisy or blurry one. A corrupted low-quality image $\mathbf{x}$ can be represented as: $\mathbf{x} = D(\tilde{\mathbf{x}}) + \mathbf{n}$, where $\tilde{\mathbf{x}}$ is a high-quality version of $\mathbf{x}$, D is the degradation function and $\mathbf{n}$ is the additive noise. With this mathematical model, extensive studies are conducted on many image restoration tasks, e.g., image denoising [2, 5, 9, 37], single-image super-resolution (SISR) [15, 38] and JPEG deblocking [18, 26].

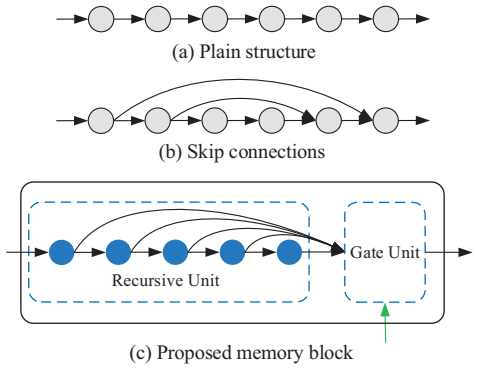


Figure 1. Prior network structures (a,b) and our memory block (c). The blue circles denote a recursive unit with an unfolded structure which generates the short-term memory. The green arrow denotes the long-term memory from the previous memory blocks that is directly passed to the gate unit.

As three classical image restoration tasks, image denoising aims to recover a clean image from a noisy observation, which commonly assumes additive white Gaussian noise with a standard deviation σ; single-image super-resolution recovers a high-resolution (HR) image from a low-resolution (LR) image; and JPEG deblocking removes the blocking artifact caused by JPEG compression [7].

Recently, due to the powerful learning ability, very deep convolutional neural network (CNN) is widely used to tackle the image restoration tasks. Kim et al. construct a 20-layer CNN structure named VDSR for SISR [20], and adopts residual learning to ease training difficulty. To control the parameter number of very deep models, the authors further introduce a recursive layer and propose a Deeply-Recursive Convolutional Network (DRCN) [21]. To mitegate training difficulty, Mao et al. [27] introduce symmetric skip connections into a 30-layer convolutional auto-encoder

*This work was supported by the National Science Fund of China under Grant Nos. 91420201, 61472187, 61502235, 61233011, 61373063 and 61602244, the 973 Program No. 2014CB349303, Program for Changjiang Scholars, and partially sponsored by CCF-Tencent Open Research Fund. Jian Yang and Xiaoming Liu are corresponding authors.

network named RED for image denoising and SISR. Moreover, Zhang et al. [40] propose a denoising convolutional neural network (DnCNN) to tackle image denoising, SISR and JPEG deblocking simultaneously.

The conventional plain CNNs, e.g., VDSR [20], DRCN [21] and DnCNN [40] (Fig. 1(a)), adopt the single-path feed-forward architecture, where one state is mainly influenced by its direct former state, namely *short-term memory*. Some variants of CNNs, RED [27] and ResNet [12] (Fig. 1(b)), have skip connections to pass information across several layers. In these networks, apart from the short-term memory, one state is also influenced by a *specific* prior state, namely *restricted long-term memory*. In essence, recent evidence suggests that mammalian brain may protect previously-acquired knowledge in neocortical circuits [4]. However, none of above CNN models has such mechanism to achieve persistent memory. As the depth grows, they face the issue of lacking long-term memory.

To address this issue, we propose a very deep persistent memory network (MemNet), which introduces a memory block to explicitly mine persistent memory through an adaptive learning process. In MemNet, a Feature Extraction Net (FENet) first extracts features from the low-quality image. Then, several memory blocks are stacked with a *densely connected structure* to solve the image restoration task. Finally, a Reconstruction Net (ReconNet) is adopted to learn the residual, rather than the direct mapping, to ease the training difficulty.

As the key component of MemNet, a memory block contains a *recursive unit* and a *gate unit*. Inspired by neuroscience [6, 25] that recursive connections ubiquitously exist in the neocortex, the recursive unit learns *multi-level* representations of the current state under different receptive fields (blue circles in Fig. 1(c)), which can be seen as the *short-term memory*. The *short-term memory* generated from the recursive unit, and the *long-term memory* generated from the previous memory blocks [1] (green arrow in Fig. 1(c)) are *concatenated* and sent to the gate unit, which is a non-linear function to maintain persistent memory. Further, we present an extended multi-supervised MemNet, which fuses all intermediate predictions of memory blocks to boost the performance.

In summary, the main contributions of this work include:

$\diamond$ A memory block to accomplish the *gating mechanism* to help bridge the long-term dependencies. In each memory block, the gate unit adaptively learns different weights for different memories, which controls how much of the long-term memory should be reserved, and decides how much of the short-term memory should be stored.

$\diamond$ A very deep end-to-end persistent memory network (*80* convolutional layers) for image restoration. The densely connected structure helps compensate mid/high-frequency signals, and ensures maximum information flow between memory blocks as well. To the best of our knowledge, it is by far the *deepest* network for image restoration.

$\diamond$ The *same* MemNet structure achieves the state-of-the-art performance in image denoising, super-resolution and JPEG deblocking. Due to the strong learning ability, our MemNet can be trained to handle different levels of corruption even using a *single* model.

[1] For the first memory block, its long-term memory comes from the output of FENet.

2. Related Work

The success of AlexNet [22] in ImageNet [31] starts the era of deep learning for vision, and the popular networks, GoogleNet [33], Highway network [32], ResNet [12], reveal that the network depth is of crucial importance.

As the early attempt, Jain et al. [17] proposed a simple CNN to recover a clean natural image from a noisy observation and achieved comparable performance with the wavelet methods. As the pioneer CNN model for SISR, super-resolution convolutional neural network (SRCNN) [8] predicts the nonlinear LR-HR mapping via a fully deep convolutional network, which significantly outperforms classical shallow methods. The authors further proposed an extended CNN model, named Artifacts Reduction Convolutional Neural Networks (ARCNN) [7], to effectively handle JPEG compression artifacts.

To incorporate task-specific priors, Wang et al. adopted a cascaded sparse coding network to fully exploit the natural sparsity of images [36]. In [35], a deep dual-domain approach is proposed to combine both the prior knowledge in the JPEG compression scheme and the practice of dual-domain sparse coding. Guo et al. [10] also proposed a dual-domain convolutional network that jointly learns a very deep network in both DCT and pixel domains.

Recently, very deep CNNs become popular for image restoration. Kim et al. [20] stacked 20 convolutional layers to exploit large contextual information. Residual learning and adjustable gradient clipping are used to speed up the training. Zhang et al. [40] introduced batch normalization into a DnCNN model to jointly handle several image restoration tasks. To reduce the model complexity, the DRCN model introduced recursive-supervision and skip-connection to mitigate the training difficulty [21]. Using symmetric skip connections, Mao et al. [27] proposed a very deep convolutional auto-encoder network for image denoising and SISR. Very Recently, Lai et al. [23] proposed LapSRN to address the problems of speed and accuracy for SISR, which operates on LR images directly and progressively reconstruct the sub-band residuals of HR images. Tai et al. [34] proposed deep recursive residual network (DRRN) to address the problems of model parameters and accuracy, which recursively learns the residual unit in a multi-path model.

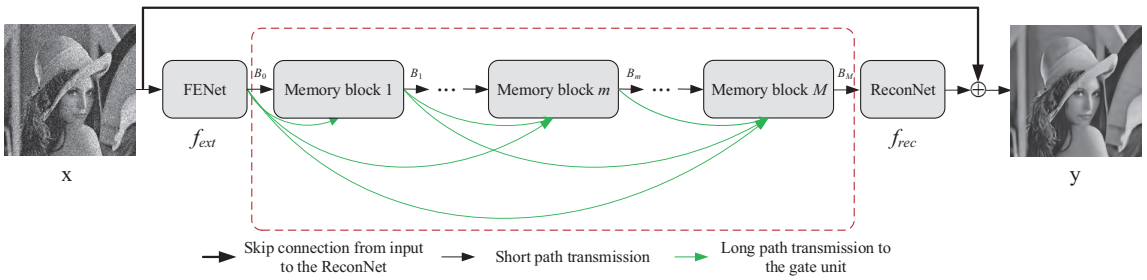


Figure 2. Basic MemNet architecture. The red dashed box represents multiple stacked memory blocks.

3. MemNet for Image Restoration

3.1. Basic Network Architecture

Our MemNet consists of three parts: a feature extraction net FENet, multiple stacked memory blocks and finally a reconstruction net ReconNet (Fig. 2). Let's denote $\mathbf{x}$ and $\mathbf{y}$ as the input and output of MemNet. Specifically, a convolutional layer is used in FENet to extract the features from the noisy or blurry input image,

$$B_0 = f_{ext}(\mathbf{x}), \tag{1}$$

where f_{ext} denotes the feature extraction function and B_0 is the extracted feature to be sent to the first memory block. Supposing M memory blocks are stacked to act as the feature mapping, we have

$$B_m = \mathcal{M}_m(B_{m-1}) = \mathcal{M}_m(\mathcal{M}_{m-1}(...(\mathcal{M}_1(B_0))...)), \tag{2}$$

where $\mathcal{M}_m$ denotes the m-th memory block function and B_{m-1} and B_m are the input and output of the m-th memory block respectively. Finally, instead of learning the direct mapping from the low-quality image to the high-quality image, our model uses a convolutional layer in ReconNet to reconstruct the residual image [20, 21, 40]. Therefore, our basic MemNet can be formulated as,

$$\begin{aligned}\mathbf{y} &= \mathcal{D}(\mathbf{x}) \\ &= f_{rec}(\mathcal{M}_M(\mathcal{M}_{M-1}(...(\mathcal{M}_1(f_{ext}(\mathbf{x})))...))) + \mathbf{x},\end{aligned} \tag{3}$$

where f_{rec} denotes the reconstruction function and $\mathcal{D}$ denotes the function of our basic MemNet.

Given a training set $\{\mathbf{x}^{(i)}, \tilde{\mathbf{x}}^{(i)}\}_{i=1}^N$, where N is the number of training patches and $\tilde{\mathbf{x}}^{(i)}$ is the ground truth high-quality patch of the low-quality patch $\mathbf{x}^{(i)}$, the loss function of our basic MemNet with the parameter set Θ, is

$$\mathcal{L}(\Theta) = \frac{1}{2N}\sum_{i=1}^{N} \|\tilde{\mathbf{x}}^{(i)} - \mathcal{D}(\mathbf{x}^{(i)})\|^2, \tag{4}$$

3.2. Memory Block

We now present the details of our memory block. The memory block contains a recursive unit and a gate unit.

Recursive Unit is used to model a non-linear function that acts like a recursive synapse in the brain [6, 25]. Here, we use a *residual building block*, which is introduced in ResNet [12] and shows powerful learning ability for object recognition, as a recursion in the recursive unit. A residual building block in the m-th memory block is formulated as,

$$H_m^r = \mathcal{R}_m(H_m^{r-1}) = \mathcal{F}(H_m^{r-1}, W_m) + H_m^{r-1}, \tag{5}$$

where H_m^{r-1}, H_m^r are the input and output of the r-th residual building block respectively. When $r = 1$, $H_m^0 = B_{m-1}$. $\mathcal{F}$ denotes the residual function, W_m is the weight set to be learned and $\mathcal{R}$ denotes the function of residual building block. Specifically, each residual function contains two convolutional layers with the pre-activation structure [13],

$$\mathcal{F}(H_m^{r-1}, W_m) = W_m^2\tau(W_m^1\tau(H_m^{r-1})), \tag{6}$$

where τ denotes the activation function, including batch normalization [16] followed by ReLU [30], and $W_m^i, i = 1, 2$ are the weights of the i-th convolutional layer. The bias terms are omitted for simplicity.

Then, several recursions are recursively learned to generate multi-level representations under different receptive fields. We call these representations as the short-term memory. Supposing there are R recursions in the recursive unit, the r-th recursion in recursive unit can be formulated as,

$$H_m^r = \mathcal{R}_m^{(r)}(B_{m-1}) = \underbrace{\mathcal{R}_m(\mathcal{R}_m(...(\mathcal{R}_m}_{r}(B_{m-1}))...)), \tag{7}$$

where r-fold operations of $\mathcal{R}_m$ are performed and $\{H_m^r\}_{r=1}^R$ are the multi-level representations of the recursive unit. These representations are concatenated as the short-term memory: $B_m^{short} = [H_m^1, H_m^2, ..., H_m^R]$. In addition, the long-term memory coming from the previous memory blocks can be constructed as: $B_m^{long} = [B_0, B_1, ..., B_{m-1}]$. The two types of memories are then concatenated as the input to the gate unit,

$$B_m^{gate} = [B_m^{short}, B_m^{long}]. \tag{8}$$

Gate Unit is used to achieve persistent memory through an adaptive learning process. In this paper, we adopt a 1×1 convolutional layer to accomplish the *gating mechanism* that can learn adaptive weights for different memories,

$$B_m = f_m^{gate}(B_m^{gate}) = W_m^{gate}\tau(B_m^{gate}), \tag{9}$$

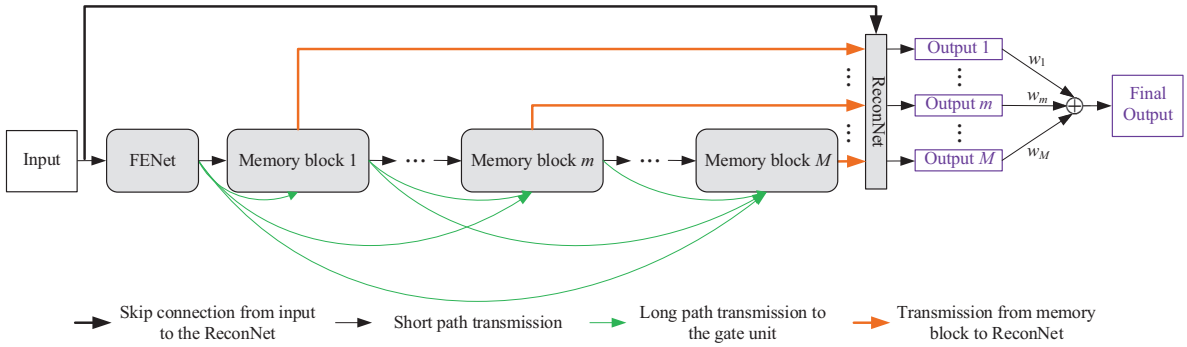


Figure 3. Multi-supervised MemNet architecture. The outputs with purple color are supervised.

where f_m^{gate} and B_m denote the function of the 1×1 convolutional layer (parameterized by W_m^{gate}) and the output of the m-th memory block, respectively. As a result, the weights for the long-term memory controls how much of the previous states should be reserved, and the weights for the short-term memory decides how much of the current state should be stored. Therefore, the formulation of the m-th memory block can be written as,

$$\begin{aligned} B_m &= \mathcal{M}_m(B_{m-1}) \\ &= f_{gate}([\mathcal{R}_m(B_{m-1}), ..., \mathcal{R}_m^{(R)}(B_{m-1}), B_0, ..., B_{m-1}]). \end{aligned} \tag{10}$$

3.3. Multi-Supervised MemNet

To further explore the features at different states, inspired by [21], we send the output of each memory block to the *same* reconstruction net f_{rec} to generate

$$\mathbf{y}_m = \hat{f}_{rec}(\mathbf{x}, B_m) = \mathbf{x} + f_{rec}(B_m), \tag{11}$$

where $\{\mathbf{y}_m\}_{m=1}^M$ are the intermediate predictions. All of the predictions are supervised during training, and used to compute the final output via weighted averaging: $\mathbf{y} = \sum_{m=1}^M w_m \cdot \mathbf{y}_m$ (Fig. 3). The optimal weights $\{w_m\}_{m=1}^M$ are automatically learned during training and the final output from the ensemble is also supervised. The loss function of our multi-supervised MemNet can be formulated as,

$$\begin{aligned} \mathcal{L}(\Theta) = &\frac{\alpha}{2N} \sum_{i=1}^N \|\tilde{\mathbf{x}}^{(i)} - \sum_{m=1}^M w_m \cdot \mathbf{y}_m^{(i)}\|^2 \\ &+ \frac{1-\alpha}{2MN} \sum_{m=1}^M \sum_{i=1}^N \|\tilde{\mathbf{x}}^{(i)} - \mathbf{y}_m^{(i)}\|^2, \end{aligned} \tag{12}$$

where α denotes the loss weight.

3.4. Dense Connections for Image Restoration

Now we analyze why the *long-term dense connections* in MemNet may benefit the image restoration. In very deep networks, some of the mid/high-frequency information can get lost at latter layers during a typical feedforward CNN process, and dense connections from previous layers can compensate such loss and further enhance

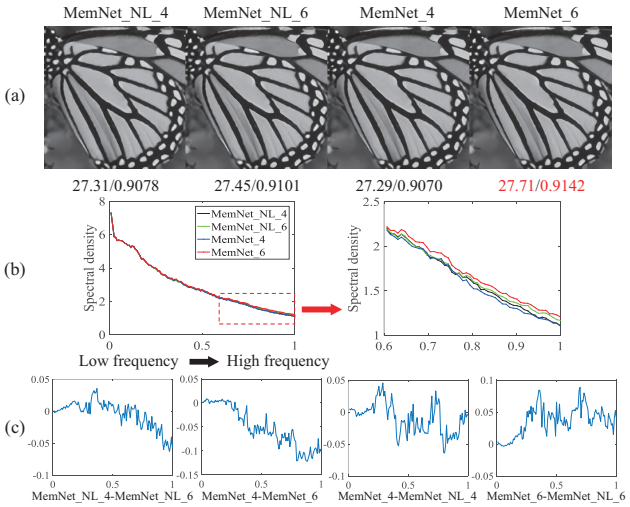


Figure 4. (a) $\times 4$ super-resolved images and PSNR/SSIMs of different networks. (b) We convert 2-D power spectrums to 1-D spectral densities by integrating the spectrums along each concentric circle. (c) Differences of spectral densities of two networks.

high-frequency signals. To verify our intuition, we train a 80-layer MemNet without long-term connections, which is denoted as MemNet_NL, and compare with the original MemNet. Both networks have 6 memory blocks leading to 6 intermediate outputs, and each memory block contains 6 recursions. Fig. 4(a) shows the 4th and 6th outputs of both networks. We compute their power spectrums, center them, estimate spectral densities for a continuous set of frequency ranges from low to high by placing concentric circles, and plot the densities of four outputs in Fig. 4(b).

We further plot *differences* of these densities in Fig. 4(c). From left to right, the first case indicates the earlier layer does contain some mid-frequency information that the latter layers lose. The 2nd case verifies that with dense connections, the latter layer absorbs the information carried from the previous layers, and even generate more mid-frequency information. The 3rd case suggests in earlier layers, the frequencies are similar between two models. The last case demonstrates the MemNet recovers more high frequency than the version without long-term connections.

4. Discussions

Difference to Highway Network First, we discuss how the memory block accomplishes the gating mechanism and present the difference between MemNet and Highway Network – a very deep CNN model using a gate unit to regulate information flow [32].

To avoid information attenuation in very deep plain networks, inspired by LSTM, Highway Network introduced the bypassing layers along with gate units, i.e.,

$$\mathbf{b} = \mathcal{A}(\mathbf{a}) \cdot \mathcal{T}(\mathbf{a}) + \mathbf{a} \cdot (1 - \mathcal{T}(\mathbf{a})), \tag{13}$$

where $\mathbf{a}$ and $\mathbf{b}$ are the input and output, $\mathcal{A}$ and $\mathcal{T}$ are two non-linear transform functions. $\mathcal{T}$ is the *transform gate* to control how much information produced by $\mathcal{A}$ should be stored to the output; and $1 - \mathcal{T}$ is the *carry gate* to decide how much of the input should be reserved to the output.

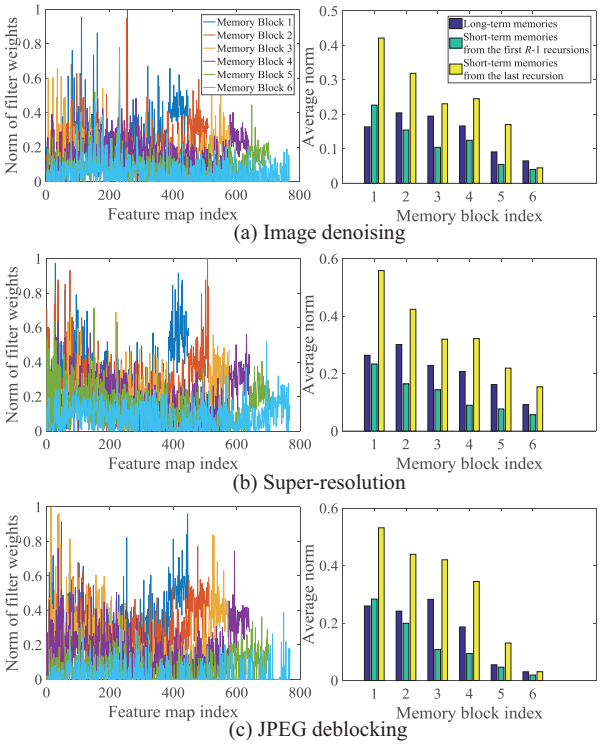
In MemNet, the short-term and long-term memories are concatenated. The 1×1 convolutional layer adaptively learns the weights for different memories. Compared to Highway Network that learns specific weight for each *pixel*, our gate unit learns specific weight for each *feature map*, which has two advantages: (1) to reduce model parameters and complexity; (2) to be less prone to overfitting.

Difference to DRCN There are three main differences between MemNet and DRCN [21]. The first is the design of the basic module in network. In DRCN, the basic module is a *convolutional layer*; while in MemNet, the basic module is a *memory block* to achieve persistent memory. The second is in DRCN, the weights of the basic modules (i.e., the convolutional layers) are *shared*; while in MemNet, the weights of the memory blocks are *different*. The third is there are no dense connections among the basic modules in DRCN, which results in a *chain* structure; while in MemNet, there are long-term dense connections among the memory blocks leading to the *multi-path* structure, which not only helps information flow across the network, but also encourages gradient backpropagation during training. Benefited from the good information flow ability, MemNet could be easily trained without the multi-supervision strategy, which is *imperative* for training DRCN [21].

Difference to DenseNet Another related work to MemNet is DenseNet [14], which also builds upon a densely connected principle. In general, DenseNet deals with object recognition, while MemNet is proposed for image restoration. In addition, DenseNet adopts the densely connected structure in a *local* way (i.e., inside a dense block), while MemNet adopts the densely connected structure in a *global* way (i.e., across the memory blocks). In Secs. 3.4 and 5.2, we analyze and demonstrate the long-term dense connections in MemNet indeed play an important role in image restoration tasks.

Methods	MemNet_NL	MemNet_NS	MemNet
$\times 2$	37.68/0.9591	37.71/0.9592	37.78/0.9597
$\times 3$	33.96/0.9235	34.00/0.9239	34.09/0.9248
$\times 4$	31.60/0.8878	31.65/0.8880	31.74/0.8893

Table 1. Ablation study on effects of long-term and short-term connections. Average PSNR/SSIMs for the scale factor $\times 2$, $\times 3$ and $\times 4$ on dataset Set5. Red indicates the best performance.



(a) Image denoising

(b) Super-resolution

(c) JPEG deblocking

Figure 5. The norm of filter weights v_m^l vs. feature map index l. For the curve of the mth block, the left ($m \times 64$) elements denote the long-term memories and the rest ($L_m - m \times 64$) elements denote the short-term memories. The bar diagrams illustrate the average norm of long-term memories, short-term memories from the first $R-1$ recursions and from the last recursion, respectively. E.g., each yellow bar is the average norm of the short-term memories from the last recursion in the recursive unit (i.e., the last 64 elements in each curve).

5. Experiments

5.1. Implementation Details

Datasets For image denoising, we follow [27] to use 300 images from the Berkeley Segmentation Dataset (BSD) [28], known as the train and val sets, to generate image patches as the training set. Two popular benchmarks, a dataset with 14 common images and the BSD test set with 200 images, are used for evaluation. We generate the input noisy patch by adding Gaussian noise with one of the three noise levels ($\sigma = 30$, 50 and 70) to the clean patch.

For SISR, by following the experimental setting in [20], we use a training set of 291 images where 91 images are from Yang et al. [38] and other 200 are from BSD train set. For testing, four benchmark datasets, Set5 [1], Set14 [39],

Dataset	VDSR [20]	DRCN [21]	RED [27]	MemNet		
Depth	20	20	30	80		
Filters	64	256	128	64		
Parameters	665K	1,774K	4,131K	677K		
Traing images	291	91	300	91	91	291
Multi-supervision	No	Yes	No	No	Yes	Yes
PSNR	33.66	33.82	33.82	33.92	33.98	34.09

Table 2. SISR comparisons with start-of-the-art networks for scale factor $\times 3$ on Set5. Red indicates the fewest number or best performance.

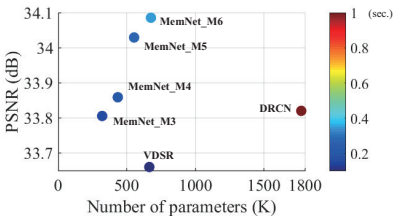


Figure 6. PSNR, complexity vs. speed.

BSD100 [28] and Urban100 [15] are used. Three scale factors are evaluated, including $\times 2$, $\times 3$ and $\times 4$. The input LR image is generated by first bicubic downsampling and then bicubic upsampling the HR image with a certain scale.

For JPEG deblocking, the same training set for image denoising is used. As in [7], Classic5 and LIVE1 are adopted as the test datasets. Two JPEG quality factors are used, i.e., 10 and 20, and the JPEG deblocking input is generated by compressing the image with a certain quality factor using the MATLAB JPEG encoder.

Training Setting Following the method [27], for image denoising, the *grayscale* image is used; while for SISR and JPEG deblocking, the *luminance* component is fed into the model. The input image size can be arbitrary due to the fully convolution architecture. Considering both the training time and storage complexities, training images are split into 31×31 patches with a stride of 21. The output of MemNet is the estimated high-quality patch with the same resolution as the input low-quality patch. We follow [34] to do data augmentation. For each task, we train a *single* model for all different levels of corruption. E.g., for image denoising, noise augmentation is used. Images with different noise levels are all included in the training set. Similarly, for super-resolution and JPEG deblocking, scale and quality augmentation are used, respectively.

We use Caffe [19] to implement two 80-layer MemNet networks, the basic and the multi-supervised versions. In both architectures, 6 memory blocks, each contains 6 recursions, are constructed (i.e., M6R6). Specifically, in multi-supervised MemNet, 6 predictions are generated and used to compute the final output. α balances different regularizations, and is empirically set as $\alpha = 1/(M+1)$.

The objective functions in Eqn. 4 and Eqn. 12 are optimized via the mini-batch stochastic gradient descent (SGD) with backpropagation [24]. We set the mini-batch size of SGD to 64, momentum parameter to 0.9, and weight decay to 10^{-4}. All convolutional layer has 64 filters. Except the 1×1 convolutional layers in the gate units, the kernel size of other convolutional layers is 3×3. We use the method in [11] for weight initialization. The initial learning rate is set to 0.1 and then divided 10 every 20 epochs. Training a 80-layer basic MemNet by 91 images [38] for SISR roughly takes 5 days using 1 Tesla P40 GPU. Due to space constraint and more recent baselines, we focus on SISR in Sec. 5.2, 5.4 and 5.6, while all three tasks in Sec. 5.3 and 5.5.

5.2. Ablation Study

Tab. 1 presents the ablation study on the effects of long-term and short-term connections. Compared to MemNet, MemNet_NL removes the long-term connections (green curves in Fig. 3) and MemNet_NS removes the short-term connections (black curves from the first $R-1$ recursions to the gate unit in Fig. 1. Connection from the last recursion to the gate unit is *reserved* to avoid a broken interaction between recursive unit and gate unit). The three networks have the same depth (80) and filter number (64). We see that, long-term dense connections are very important since MemNet significantly outperforms MemNet_NL. Further, MemNet achieves better performance than MemNet_NS, which reveals the short-term connections are also useful for image restoration but less powerful than the long-term connections. The reason is that the long-term connections skip much more layers than the short-term ones, which can carry some mid/high frequency signals from very early layers to latter layers as described in Sec. 3.4.

5.3. Gate Unit Analysis

We now illustrate how our gate unit affects different kinds of memories. Inspired by [14], we adopt a weight norm as an approximate for the dependency of the current layer on its preceding layers, which is calculated by the corresponding weights from all filters w.r.t. each feature map: $v_m^l = \sqrt{\sum_{i=1}^{64}(W_m^{gate}(1,1,l,i))^2},\ l = 1, 2, ..., L_m$, where L_m is the number of the input feature maps for the m-th gate unit, l denotes the feature map index, W_m^{gate} stores the weights with the size of $1 \times 1 \times L_m \times 64$, and v_m^l is the weight norm of the l-th feature map for the m-th gate unit. Basically, the larger the norm is, the stronger dependency it has on this particular feature map. For better visualization, we normalize the norms to the range of 0 to 1. Fig. 5 presents the norm of the filter weights $\{v_m^l\}_{m=1}^6$ vs. feature map index l. We have three observations: (1) Different tasks have different norm distributions. (2) The average and variance of the weight norms become smaller as the memory block number increases. (3) In general, the short-term memories from the last recursion in recursive unit (the last 64 elements in each curve) contribute most than the other two memories, and the long-term memories seem to play a more important role in late memory blocks to recover useful signals than the short-term memories from the first $R-1$ recursions.

Dataset	Noise	BM3D [5]	EPLL [41]	PCLR [2]	PGPD [37]	WNNM [9]	RED [27]	MemNet
14 images	30	28.49/0.8204	28.35/0.8200	28.68/0.8263	28.55/0.8199	28.74/0.8273	29.17/0.8423	29.22/0.8444
	50	26.08/0.7427	25.97/0.7354	26.29/0.7538	26.19/0.7442	26.32/0.7517	26.81/0.7733	26.91/0.7775
	70	24.65/0.6882	24.47/0.6712	24.79/0.6997	24.71/0.6913	24.80/0.6975	25.31/0.7206	25.43/0.7260
BSD200	30	27.31/0.7755	27.38/0.7825	27.54/0.7827	27.33/0.7717	27.48/0.7807	27.95/0.8019	28.04/0.8053
	50	25.06/0.6831	25.17/0.6870	25.30/0.6947	25.18/0.6841	25.26/0.6928	25.75/0.7167	25.86/0.7202
	70	23.82/0.6240	23.81/0.6168	23.94/0.6336	23.89/0.6245	23.95/0.6346	24.37/0.6551	24.53/0.6608

Table 3. Benchmark image denoising results. Average PSNR/SSIMs for noise level 30, 50 and 70 on 14 images and BSD200. Red color indicates the best performance and blue color indicates the second best performance.

Dataset	Scale	Bicubic	SRCNN [8]	VDSR [20]	DRCN [21]	DnCNN [40]	LapSRN [23]	DRRN [34]	MemNet
Set5	×2	33.66/0.9299	36.66/0.9542	37.53/0.9587	37.63/0.9588	37.58/0.9590	37.52/0.959	37.74/0.9591	37.78/0.9597
	×3	30.39/0.8682	32.75/0.9090	33.66/0.9213	33.82/0.9226	33.75/0.9222	–/–	34.03/0.9244	34.09/0.9248
	×4	28.42/0.8104	30.48/0.8628	31.35/0.8838	31.53/0.8854	31.40/0.8845	31.54/0.885	31.68/0.8888	31.74/0.8893
Set14	×2	30.24/0.8688	32.45/0.9067	33.03/0.9124	33.04/0.9118	33.03/0.9128	33.08/0.913	33.23/0.9136	33.28/0.9142
	×3	27.55/0.7742	29.30/0.8215	29.77/0.8314	29.76/0.8311	29.81/0.8321	–/–	29.96/0.8349	30.00/0.8350
	×4	26.00/0.7027	27.50/0.7513	28.01/0.7674	28.02/0.7670	28.04/0.7672	28.19/0.772	28.21/0.7721	28.26/0.7723
BSD100	×2	29.56/0.8431	31.36/0.8879	31.90/0.8960	31.85/0.8942	31.90/0.8961	31.80/0.895	32.05/0.8973	32.08/0.8978
	×3	27.21/0.7385	28.41/0.7863	28.82/0.7976	28.80/0.7963	28.85/0.7981	–/–	28.95/0.8004	28.96/0.8001
	×4	25.96/0.6675	26.90/0.7101	27.29/0.7251	27.23/0.7233	27.29/0.7253	27.32/0.728	27.38/0.7284	27.40/0.7281
Urban100	×2	26.88/0.8403	29.50/0.8946	30.76/0.9140	30.75/0.9133	30.74/0.9139	30.41/0.910	31.23/0.9188	31.31/0.9195
	×3	24.46/0.7349	26.24/0.7989	27.14/0.8279	27.15/0.8276	27.15/0.8276	–/–	27.53/0.8378	27.56/0.8376
	×4	23.14/0.6577	24.52/0.7221	25.18/0.7524	25.14/0.7510	25.20/0.7521	25.21/0.756	25.44/0.7638	25.50/0.7630

Table 4. Benchmark SISR results. Average PSNR/SSIMs for scale factor ×2, ×3 and ×4 on datasets Set5, Set14, BSD100 and Urban100.

Dataset	Quality	JPEG	ARCNN [7]	TNRD [3]	DnCNN [40]	MemNet
Classic5	10	27.82/0.7595	29.03/0.7929	29.28/0.7992	29.40/0.8026	29.69/0.8107
	20	30.12/0.8344	31.15/0.8517	31.47/0.8576	31.63/0.8610	31.90/0.8658
LIVE1	10	27.77/0.7730	28.96/0.8076	29.15/0.8111	29.19/0.8123	29.45/0.8193
	20	30.07/0.8512	31.29/0.8733	31.46/0.8769	31.59/0.8802	31.83/0.8846

Table 5. Benchmark JPEG deblocking results. Average PSNR/SSIMs for quality factor 10 and 20 on datasets Classic5 and LIVE1.

5.4. Comparision with Non-Persistent CNN Models

In this subsection, we compare MemNet with three existing non-persistent CNN models, i.e., VDSR [20], DRCN [21] and RED [27], to demonstrate the superiority of our persistent memory structure. VDSR and DRCN are two representative networks with the plain structure and RED is representative for skip connections. Tab. 2 presents the published results of these models along with their training details. Since the training details are different among different work, we choose DRCN as a baseline, which achieves good performance using the least training images. But, unlike DRCN that widens its network to increase the parameters (filter number: 256 vs. 64), we deepen our MemNet by stacking more memory blocks (depth: 20 vs. 80). It can be seen that, using the fewest training images (91), filter number (64) and relatively few model parameters (667K), our basic MemNet already achieves higher PSNR than the prior networks. Keeping the setting unchanged, our multi-supervised MemNet further improves the performance. With more training images (291), our MemNet significantly outperforms the state of the arts.

Since we aim to address the long-term dependency problem in networks, we intend to make our MemNet *very deep*. However, MemNet is also able to balance the model complexity and accuracy. Fig. 6 presents the PSNR of different intermediate predictions in MemNet (e.g., MemNet_M3 denotes the prediction of the 3rd memory block) for scale ×3 on Set5, in which the colorbar indicates the inference time (sec.) when processing a 288 × 288 image on GPU P40. Results of VDSR [20] and DRCN [21] are cited from their papers. RED [27] is skipped here since its high number of parameters may reduce the contrast among other methods. We see that our MemNet already achieve comparable result at the 3rd prediction using much fewer parameters, and significantly outperforms the state of the arts by slightly increasing model complexity.

5.5. Comparisons with State-of-the-Art Models

We compare multi-supervised 80-layer MemNet with the state of the arts in three restoration tasks, respectively.

Image Denoising Tab. 3 presents quantitative results on two benchmarks, with results cited from [27]. For BSD200 dataset, by following the setting in RED [27], the original image is resized to its half size. As we can see, our MemNet achieves the best performance on all cases. It should be noted that, for each test image, RED rotates and mirror flips the kernels, and performs inference multiple times. The outputs are then averaged to obtain the final result. They claimed this strategy can lead to better performance. However, in our MemNet, we *do not* perform any post-processing. For qualitative comparisons, we use public codes of PCLR [2], PGPD [37] and WNNM [9]. The results are shown in Fig. 7. As we can see, our MemNet handles Gaussian noise better than the previous state of the arts.

Super-Resolution Tab. 4 summarizes quantitative results on four benchmarks, by citing the results of prior methods. MemNet outperforms prior methods in almost all cases.

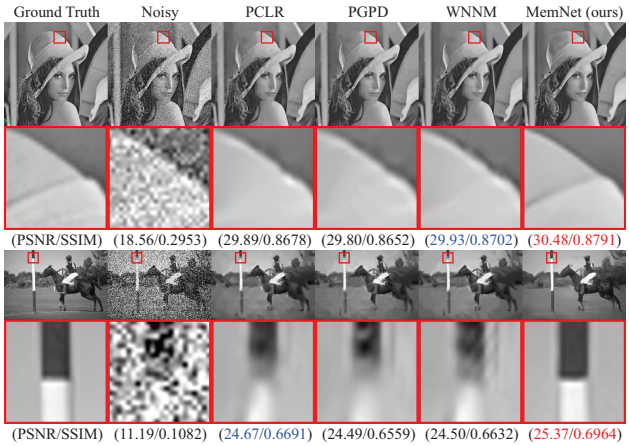


Figure 7. Qualitative comparisons of image denoising. The first row shows image "10" from 14-image dataset with noise level 30. Only MemNet recovers the fold. The second row shows image "206062" from BSD200 with noise level 70. Only MemNet correctly recovers the pillar. Please zoom in to see the details.

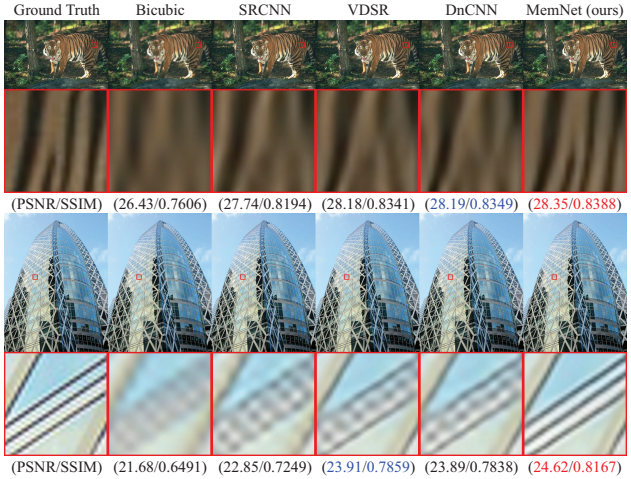


Figure 8. Qualitative comparisons of SISR. The first row shows image "108005" from BSD100 with scale factor $\times 3$. Only MemNet correctly recovers the pattern. The second row shows image "img_002" from Urban100 with scale factor $\times 4$. MemNet recovers sharper lines.

Since LapSRN doesn't report the results on scale $\times 3$, we use the symbol '$-$' instead. Fig. 8 shows the visual comparisons for SISR. SRCNN [8], VDSR [20] and DnCNN [40] are compared using their public codes. MemNet recovers relatively sharper edges, while others have blurry results.

JPEG Deblocking Tab. 5 shows the JPEG deblocking results on Classic5 and LIVE1, by citing the results from [40]. Our network significantly outperforms the other methods, and deeper networks do improve the performance compared to the shallow one, e.g., ARCNN. Fig. 9 shows the JPEG deblocking results of these three methods, which are generated by their corresponding public codes. As it can be seen, MemNet effectively removes the blocking artifact and recovers higher quality images than the previous methods.

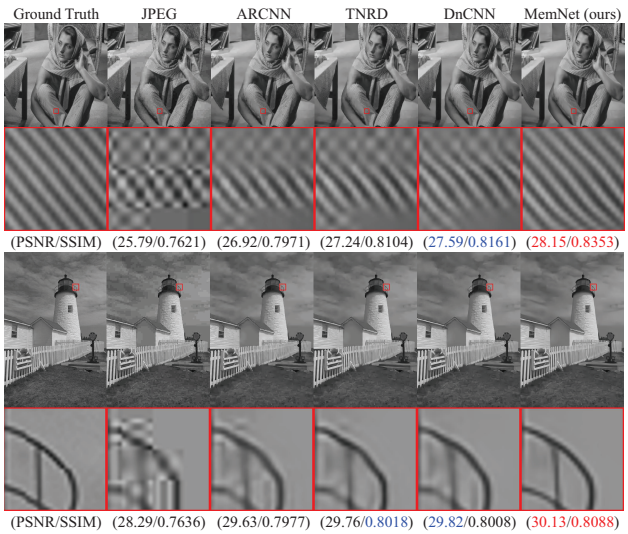


Figure 9. Qualitative comparisons of JPEG deblocking. The first row shows image "barbara" from Classic5 with quality factor 10. MemNet recovers the lines, while others give blurry results. The second row shows image "lighthouse" from LIVE1 with quality factor 10. MemNet accurately removes the blocking artifact.

Network	M4R6	M6R6	M6R8	M10R10
Depth	54	80	104	212
PSNR (dB)	34.05	34.09	34.16	34.23

Table 6. Comparison on different network depths.

5.6. Comparison on Different Network Depths

Finally, we present the comparison on different network depths, which is caused by stacking different numbers of memory blocks or recursions. Specifically, we test four network structures: M4R6, M6R6, M6R8 and M10R10, which have the depth 54, 80, 104 and 212, respectively. Tab. 6 shows the SISR performance of these networks on Set5 with scale factor $\times 3$. It verifies *deeper is still better* and the proposed deepest network M10R10 achieves 34.23 dB, with the improvement of 0.14 dB compared to M6R6.

6. Conclusions

In this paper, a very deep end-to-end persistent memory network (MemNet) is proposed for image restoration, where a memory block accomplishes the gating mechanism for tackling the long-term dependency problem in the previous CNN architectures. In each memory block, a recursive unit is adopted to learn multi-level representations as the short-term memory. Both the short-term memory from the recursive unit and the long-term memories from the previous memory blocks are sent to a gate unit, which adaptively learns different weights for different memories. We use the same MemNet structure to handle image denoising, super-resolution and JPEG deblocking simultaneously. Comprehensive benchmark evaluations well demonstrate the superiority of our MemNet over the state of the arts.

References

[1] C. M. Bevilacqua, A. Roumy, and M. Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC*, 2012.

[2] F. Chen, L. Zhang, and H. Yu. External patch prior guided internal clustering for image denoising. In *ICCV*, 2015.

[3] Y. Chen and T. Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Trans. on PAMI*, 2016.

[4] J. Cichon and W. Gan. Branch-specific dendritic ca2+ spikes cause persistent synaptic plasticity. *Nature*, 520(7546):180–185, 2015.

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

[6] P. Dayan and L. F. Abbott. Theoretical neuroscience. *Cambridge, MA: MIT Press*, 2001.

[7] C. Dong, Y. Deng, C. C. Loy, and X. Tang. Compression artifacts reduction by a deep convolutional network. In *ICCV*, 2015.

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

[9] S. Gu, L. Zhang, W. Zuo, and X. Feng. Weighted nuclear norm minimization with application to image denoising. In *CVPR*, 2014.

[10] J. Guo and H. Chao. Building dual-domain representations for compression artifacts reduction. In *ECCV*, 2016.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

[14] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[15] J.-B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, 2015.

[16] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[17] V. Jain and H. S. Seung. Natural image denoising with convolutional networks. In *NIPS*, 2008.

[18] J. Jancsary, S. Nowozin, and C. Rother. Loss-specific training of non-parametric image restoration models: A new state of the art. In *ECCV*, 2012.

[19] 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 *ACM MM*, 2014.

[20] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016.

[21] J. Kim, J. K. Lee, and K. M. Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR*, 2016.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[23] W. Lai, J. Huang, N. Ahuja, and M. Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, 2017.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 1998.

[25] M. Liang and X. Hu. Recurrent convolutional neural network for object recognition. In *CVPR*, 2015.

[26] X. Liu, X. Wu, J. Zhou, and D. Zhao. Data-driven sparsity-based restoration of jpeg-compressed images in dual transform-pixel domain. In *CVPR*, 2015.

[27] X. Mao, C. Shen, and Y. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *NIPS*, 2016.

[28] 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 *ICCV*, 2001.

[29] P. Milanfar. A tour of modern image filtering: new insights and methods, both practical and theoretical. *IEEE Signal Processing Magazine*, 30(1):106–128, 2013.

[30] V. Nair and G. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[32] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. In *NIPS*, 2015.

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, and S. Reed. Going deeper with convolutions. In *CVPR*, 2015.

[34] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *CVPR*, 2017.

[35] Z. Wang, D. Liu, S. Chang, Q. Ling, Y. Yang, and T. S. Huang. D^3: Deep dual-domain based fast restoration of jpeg-compressed images. In *CVPR*, 2016.

[36] Z. Wang, D. Liu, J. Yang, W. Han, and T. S. Huang. Deep networks for image super-resolution with sparse prior. In *ICCV*, 2015.

[37] J. Xu, L. Zhang, W. Zuo, D. Zhang, and X. Feng. Patch group based nonlocal self-similarity prior learning for image denoising. In *ICCV*, 2015.

[38] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE Trans. on IP*, 19(11):2861–2873, 2010.

[39] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. *Curves and Surfaces*, pages 711–730, 2012.

[40] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Trans. on IP*, 2017.

[41] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *ICCV*, 2011.