Block-Matching Convolutional Neural Network (BMCNN): Improving CNN-Based Denoising by Block-Matched Inputs

Byeongyong Ahn* Yoonsik Kim†, Guyong Park†, and Nam Ik Cho†
* Samsung Electronics Co., Ltd, Suwon, Korea
E-mail: by89.ahn@samsung.com Tel/Fax: +82-31-301-0787
† INMC, Seoul National University, Seoul, Korea
E-mail: nicho@snu.ac.kr Tel/Fax: +82-02-880-8480

***Abstract*—There are two main streams in up-to-date image denoising algorithms: non-local self similarity (NSS) prior based methods and convolutional neural network (CNN) based methods. The NSS based methods are favorable on images with regular and repetitive patterns while the CNN based methods perform better on irregular structures. In this paper, we propose a block-matching convolutional neural network (BMCNN) method that combines NSS prior and CNN. Initially, similar local patches in the input image are integrated into a 3D block. In order to prevent the noise from messing up the block matching, we first apply an existing denoising algorithm on the noisy image. The denoised image is employed as a pilot signal for the block matching, and then denoising function for the block is learned by a CNN structure. Experimental results show that the proposed BMCNN algorithm achieves state-of-the-art performance. In detail, BMCNN can restore both repetitive and irregular structures.**

I. Introduction

With current image capturing technologies, noise is inevitable especially in low light conditions. Moreover, captured images can be affected by additional noises through the compression and transmission procedures. Since the noise degrades visual quality, compression performance, and also the performance of computer vision algorithms, the image denoising has been extensively studied over several decades [1]–[11].

In order to estimate a clean image from its noisy observation, a number of methods that take account of certain image priors have been developed. Among various image priors, the NSS is considered a remarkable one such that it is employed in most of current state-of-the-art methods. The NSS implies that some patterns occur repeatedly in an image and the image patches that have similar patterns can be located far from each other. The nonlocal means filter [1] is a seminal work that exploits this NSS prior. The employment of NSS prior has boosted the performance of image denoising significantly, and many up-to-date denoising algorithms [3], [7], [12]–[14] can be categorized as NSS based methods. Most NSS based methods consist of following steps. First, they find similar patches and integrate them into a 3D block. Then the block is denoised using some other priors such as low-band prior [3], sparsity prior [13], [14] and low-rank prior [7]. Since the patch similarity can be easily disrupted by noise, the NSS based methods are usually implemented as two-step or iterative procedures.

Although the NSS based methods show high denoising performance, they have some limitations. First, since the block denoising stage is designed considering a specific prior, it is difficult to satisfy mixed characteristics of an image. For example, some methods work very well with the regular structures (such as stripe pattern) whereas some do not. Furthermore, since the prior is based on human observation, it can hardly be optimal. The NSS based methods also contain some parameters that have to be tuned by a user, and it is difficult to find the optimal parameters for the overall tasks. Lastly, many NSS based methods such as LSSC [12], NCSR [5] and WNNM [7] involve complex optimization problems. These optimizations are very time-consuming, and also very difficult to be parallelized.

Some researchers, meanwhile, developed discriminative learning methods for image denoising, which learn the image prior models and corresponding denoising function. Schmidt et al. [15] proposed a cascade of shrinkage fields (CSF) that unifies the random field-based model and quadratic optimization. Chen et al. [16] proposed a trainable nonlinear reaction diffusion (TNRD) model which learns parameters for a diffusion model by the gradient descent procedure. Although these methods find optimal parameters in a data-driven manner, they are limited to the specific prior model. Recently, neural network based denoising algorithms [2], [9], [17] are attracting considerable attentions for their performance and fast processing by GPU. They trained networks which take noisy patches as input and estimate noise-free original patch. These networks consist of series of convolution operations and non-linear activations. Since the neural network denoising algorithms are also based on the data-driven framework, they can learn at least locally optimal filters for the local regions provided that sufficiently large number of training patches from abundant dataset are available. It is believed that the networks can also learn the priors which were neglected by human observers or difficult to be implemented. However, the patch-based denoising is basically a local processing and the

existing neural network methods did not consider the NSS prior. Hence, they often show inferior performance than the NSS based methods especially in the case of regular and repetitive structures [18], which lowers the overall performance.

In this paper, a combined denoising framework named block-matching convolutional neural network (BMCNN) is presented. Fig. 1 illustrates the difference between the existing denoising algorithms and the proposed method. As shown in the figure, the proposed method finds similar patches and stack them as a 3D input like BM3D [3], which is illustreated in Fig. 1(d). By using the set of similar patches as the input, the network is able to consider the NSS prior in addition to the local prior that the conventional neural networks could train. Compared to the conventional NSS based algorithms, the BMCNN is a data-driven framework and thus can find more accurate denoising function for the given input. Finally, it will be explained that some of the conventional methods can also be interpreted as a kind of BMCNN framework.

The rest of this paper is organized as follows. In Section II, researches that are related with the proposed framework are reviewed. There are two main topics: image denoising based on NSS, and image restoration based on neural network. Section III presents the proposed BMCNN network. In Section IV, experimental results and the comparison with the state-of-the-art methods are presented. This paper is concluded in Section V.

II. Related works

A. Image Denoising based on Nonlocal Self-similarity

Most state-of-the-art denoising methods [1], [3], [4], [7], [19] employ NSS prior of natural images. The nonlocal means filter (NLM) [1] was the first to employ this prior to estimate a clean pixel from the relations of similar non-neighboring blocks. Some other algorithms estimate the denoised patch rather than estimating each pixel separately. For instance, Dabov et al. [3] proposed the BM3D algorithm that exploits non-local similar patches for denoising a patch. The similar patches are found by block matching and they are stacked to be a 3D block, which is then denoised in the 3D transform domain. Dong et al. [4], [5] solved the denoising problem by using the sparsity prior of natural images. Since the matrix formed by similar patches is of low rank, finding the sparse representation of noisy group results in a denoised patch. Nejati et al. [13] proposed a low-rank regularized collaborative filtering. Gu et al. [7] also considered denoising as a kind of low-rank matrix approximation problem which is solved by weighted nuclear norm minimization (WNNM). In addition to the low-rank nature, they took advantage of the prior knowledge that large singular value of the low-rank approximation represents the major components of the image. Specifically, the WNNM algorithm adopted the term that prevents large singular values from shrinking in addition to the conventional nuclear norm minimization (NNM) [20]. Recently, Zha et al. [14] proposed to use group sparsity residual constraint. Their method estimates a group sparse code instead of denoising the group directly.

B. Image restoration based on Neural Network

Since Lecun et al. [21] showed that their CNN performs very well in digit classification problem, various CNN structures and related algorithms have been developed for diverse computer vision problems ranging from low to high-level tasks. Among these, this section introduces some neural network algorithms for image enhancement problems. In the early stage of this work, some multilayer perceptrons (MLP) were adopted for image processing. Burger et al. [2] showed that a plain MLP can compete the state-of-the-art image denoising methods (BM3D) provided that huge training set, deep network and numerous neuron are available. Their method was tested on several type of noise: Gaussian noise, salt-and-pepper noise, compression artifact, etc. Schuler et al. [22] trained the same structure to remove the artifacts that occur from non-blind image deconvolution.

Meanwhile, many researchers have developed CNN based algorithms. Jain et al. [9] proposed a CNN for denoising, and discussed its relationship with the Markov random field (MRF) [23]. Dong et al. [24] proposed a SRCNN, which is a convolutional network for image super-resolution. Although their network was lightweight, it achieved superior performance to the conventional non-CNN approaches. They also showed that some conventional super-resolution methods such as sparse coding [25] can be considered a special case of deep neural network. Their work was continued to the compression artifact reduction [26]. Kim et al. [27], [28] proposed two algorithms for image super-resolution. In [28], they presented skip-connection from input to output layer. Since the input and output are highly correlated in super-resolution problem, the skip-connection was very effective. In [27], they introduced a network with repeated convolution layers. Their recursive structure enabled a very deep network without huge model and prevented exploding/vanishing gradients [29].

Recently, some techniques such as residual learning [30] and batch normalization [31] have made considerable contributions in developing CNN based image processing algorithms. The techniques contribute to stabilizing the convergence of the network and improving the performance. For some examples, Timofte et al. [32] adopted residual learning for image super-resolution, and Zhang et al. [17] proposed a deep CNN using both batch normalization and residual learning.

III. Block Matching Convolutional Neural Network

In this section, we present the BMCNN that esimtates the original image X from its noisy observation $Y = X + V$, where we concentrate on additive white gaussian noise (AWGN) that follows $V \sim N(0, \sigma^2)$. The overview of our algorithm is illustrated in Fig. 2. First, we apply an existing denoising method to the noisy image. The denoised image is regraded as a pilot signal for block matching. That is, we find a group of similar patches from the input and pilot images, which is denoised by a CNN. Finally, the denoised patches are aggregated to form the output image.

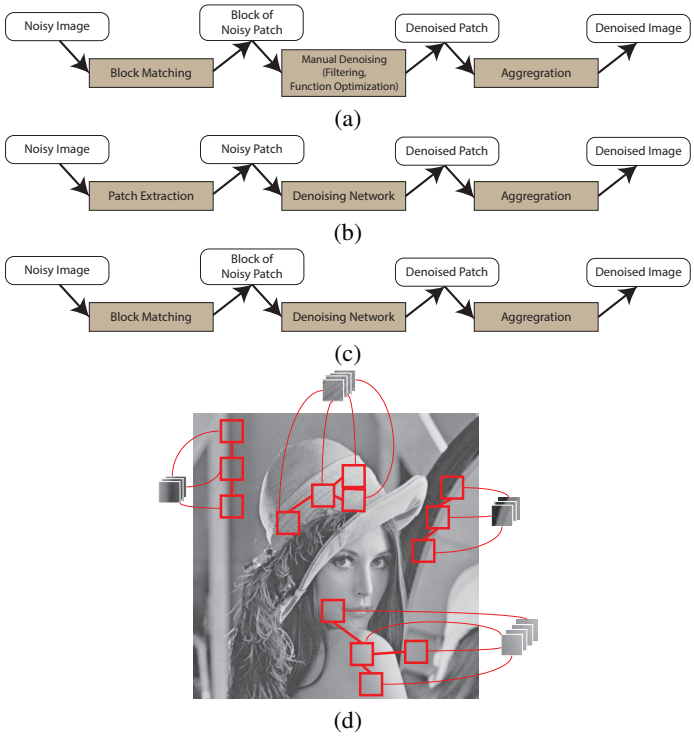


Fig. 1. Flow chart of (a) conventional NSS based system, (b) NN based system and (c) the proposed BMCNN. (d) Illustration of block-matching step.

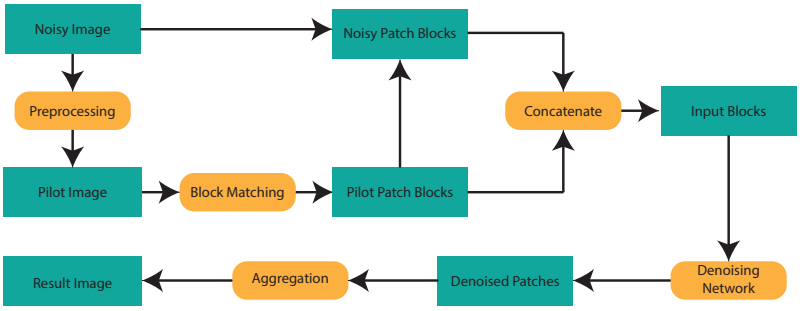


Fig. 2. The flowchart of proposed BMCNN denoising algorithm.

A. Patch Extraction and Block Matching

The proposed method first extracts overlapping patches $\{y_p\}$ of fixed size $N_{patch} \times N_{patch}$ from Y. In general, the patch-based denoising algorithms show the best performance when they process all possible overlapping patches (i.e., the patches are extracted with the stride 1), which is obviously computationally demanding. Hence, many previous studies [2], [3], [7] suggested to use some larger stride that decreases computations while not much degrading the performance. Accordingly, the proposed algorithm extracts the reference patches (patches to be denoised) with the stride $N_{step} > 1$.

For each reference patch y_p with the center pixel at p, the proposed method searches for similar patches in its neighborhood. The searching step is based on a block matching method proposed in [3], where the dissimilarity between two patches y_p, y_q is measured by their l^2 distance

$$d(y_p, y_q) = \|y_p - y_q\|_2^2. \quad (1)$$

Based on the l^2 distance, k patches nearest to the reference patch including y_p itself are selected and stacked, which forms a 3D block $\{Y_p\}$ of size $N_{patch} \times N_{patch} \times k$.

1) Pilot Signal for Block Matching: The proposed framework is aimed to take advantage of NSS by gathering similar patching through block-matching. But this can be disrupted by severe noise when only the noisy image Y is available so

that the distance between the patches (1) is corrupted. In [3], it is presented that the distance is a non-central chi-squared random variable with the mean

$$E(d(y_p, y_q)) = d(x_p, x_q) + 2\sigma^2 N_{patch}^2 \tag{2}$$

and variance

$$V(d(y_p, y_q)) = 8\sigma^2 N_{patch}^2(\sigma^2 + d(x_p, x_q)). \tag{3}$$

As shown, the variance grows with $O(\sigma^4)$, and thus the block matching results are likely to depend on the noise distribution as well as the ground-truth block distance, especially with a large σ. In order to alleviate this problem, we adopt an existing denoising algorithm as a preprocessing step. The preprocessing step estimates a denoised image $\hat{X}_{basic}$, which is named *pilot image* and used for our algorithm as follows:

- Block-matching is performed on $\hat{X}_{basic}$. Since the preprocessing attenuates the noise, the block-matching on the pilot image provides more accurate results.
- The group is formed by stacking both the similar patches in the pilot image and the corresponding noisy patches. Since some information can be lost by denoising, noisy input patch can help reconstructing the details of the image.

In this paper, we mainly use DnCNN [17] to find the pilot image because of its promising denoising performance and short run-time on GPU. We also use BM3D as a preprocessing step, which show almost the same performance on the average. But the DnCNN and BM3D lead to somewhat different results for the individual image as will be explained with the experiments.

B. Network Structure

In CNN based methods, designing a network structure is an essential step that determines the performance. Simonyan et al. [33] pointed out that deep networks consisting of small convolutional filters with filter size 3×3 can achieve favorable performance in many computer vision tasks. Based on this principle, the DnCNN [17] employed only 3×3 filters, and our network is also consisted of 3×3 filters, with residual learning and batch normalization. The architecture of the network is illustrated in Fig. 3.

In our algorithm, the depth is set to 17, and the network is composed of three types of layers. The first layer generates 64 low-level feature maps using 64 filters of size $3 \times 3 \times 2k$, for the k patches from the input and another k patches from the preprocessed image. The feature maps are processed by a rectified linear unit (ReLU) to grant nonlinearity. This layer is also called *Stem*. The layers except for the first and the last layer (layer $2 \sim 16$) contain batch normalization between the convolution filters and ReLU operation. The batch normalization for feature maps is proven to offer some merits in many previous works [31], [34], [35], such as the alleviation of internal covariate shift. All the convolution operations for these layers use 64 filters of size $3 \times 3 \times 64$. The last layer consists of only a convolution layer. The layer uses a single $3 \times 3 \times 64$ filter to construct the output from the processed feature maps. In this paper, the network adopts the residual learning [30] $f(Y) = V$. Hence, the output of the last layer is the estimated noise component of the input and the denoised patch is obtained by subtracting the output from the input. These layers can also be categorized into three stages as follows.

1) Feature Extraction: At the first stage (layer $1 \sim 6$), the features of the patches are extracted. Fig. 4(a)$\sim$(c) show the function of the stage. The first layer transforms the input patches into low-level feature maps including the edges, and then the following layers generate gradually higher-level-features. The output of this stage contains complicated features and some features about the noise components.

2) Feature Refinement: The second stage (layer $7 \sim 11$) processes the feature maps to construct the target feature maps. In existing networks [24], [26], the refinement stage filters the noise component out because the main objective is to acquire a clean image. On the other hands, the target of our algorithm is a noise patch. Hence, the refined feature maps are comprised of the noise components as shown in Fig. 4(d).

3) Reconstruction: The last stage (layer $12 \sim 17$) makes the residual patch from the noise feature maps. The stage can be considered an inverse of the feature extraction stage in that the layers in the reconstruction stage gradually constructs lower-level features from high level feature maps as shown in Fig 4(d)$\sim$(f). Despite all the layers share the similar form, they contribute different operations throughout the network. It gives some intuitions in designing an end-to-end network for image processing.

C. Patch aggregation

In order to obtain the denoised image, it is straightforward to place the denoised patches $\hat{x}_p$ at the locations of their noisy counterparts y_p. However, as suggested in Sec. III-A, the step size N_{step} is smaller than the patch size N_{patch}, which yields an overcomplete result consequently. In other words, image pixels are estimated in multiple patches. Hence, a patch aggregation step that computes the appropriate value $\tilde{x}(i,j)$ for a pixel (i,j) from a number of estimates $\hat{x}_p(i,j)$ is required. The simplest method for aggregation is simply averaging the estimates as

$$\tilde{x}(i,j) = \frac{\sum_{(i,j)\in\hat{x}_p} \hat{x}_p(i,j)}{\sum_{(i,j)\in\hat{x}_p} 1}. \tag{4}$$

However, in some studies [2], [22], it is shown that weighting the patches $\hat{x}_p$ with a simple Gaussian window improves the aggregation results. Hence we also employ the Gaussian weighted aggregation

$$\tilde{x}(i,j) = \frac{\sum_{(i,j)\in\hat{x}_p} w_p(i,j)\hat{x}_p(i,j)}{\sum_{(i,j)\in\hat{x}_p} w_p(i,j)} \tag{5}$$

where the weights are determined as

$$w_p(i,j) = \frac{1}{\sqrt{2\pi\sigma_w^2}} \exp -\frac{|p-(i,j)|^2}{2\sigma_w^2} \tag{6}$$

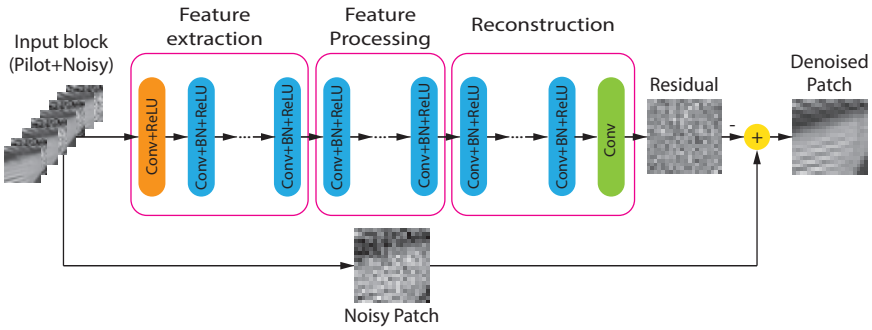


Fig. 3. The architecture of the denoising network

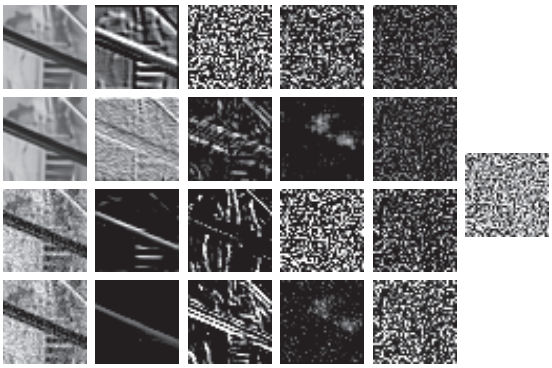


Fig. 4. Feature maps from the denoising network. (a) Patches in an input image (b) the output of the first conv layer (c) the output of the feature extraction stage (at the same time, the input to the feature processing stage) (d) the output of the feature processing stage (at the same time, the input to the reconstruction stage), (e) the input of the last layer, (f) the output residual patch.

where σ_w is the parameter for weighting.

IV. Experiments

A. Training Methodology

The proposed denoising network is implemented using the *Caffe* package [36]. Training a network is identical to finding an optimal mapping function

$$\hat{x}_p = F(\{W_i\}, y_p) \tag{7}$$

where W_i is the weight matrix including the bias for the i-th layer. This is achieved by minimizing a cost function

$$L(\{W_i\}) = \frac{1}{N_{sample}} \sum d(\hat{x}_p, x_p) + \lambda r(\{W_i\}) \tag{8}$$

where N_{sample} is the total number of the training samples, $d(\hat{x}_p, x_p)$ is the distance between estimated result $\hat{x}_p$ and its ground truth x_p, $r(\{W_i\})$ is a regularization term designed to enforce the sparseness, and λ is the weight for the regularization term. Zhao et al [37] proposed several loss functions for neural networks, among which we employ the L1 norm for the distance:

$$d(\hat{x}_p, x_p) = \sum_k |\hat{x}_p[k] - x_p[k]| \tag{9}$$

because of its simplicity for implementation in addition to its promising performance for image restoration. The objective function is minimized using *Adam*, which is known as an efficient stochastic optimization method [36]. In detail, *Adam* solver updates $(W)_i$ by the formula

$$(m_t)_i = \beta_1 (m_{t-1})_i + (1-\beta_1)(\bigtriangledown L(W_t))_i, \tag{10}$$

$$(v_t)_i = \beta_1 (v_{t-1})_i + (1-\beta_1)(\bigtriangledown L(W_t))_i^2, \tag{11}$$

$$(W_{t+1})_i = (W_t)_i - \alpha \frac{\sqrt{1-(\beta_2)^t}}{1-(\beta_1)^t} \frac{(m_t)_i}{\sqrt{(v_t)_i} + \epsilon} \tag{12}$$

where β_1 and β_2 are training parameters, α is the learning rate, and ϵ is a term to avoid zero division. In the proposed algorithm, the parameters are set as: $\lambda = 0.0002, \alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 1e-8$. The initial values of $(W_0)_i$ are set by Xavier initialization [37]. In *Caffe*, the Xavier initialization draws the values from the distribution

$$(W_0)_i \sim N(0, \frac{1}{(N_{in})_i}) \tag{13}$$

where $(N_{in})_i$ is the number of neurons feeding into the layer. The bias of every convolution layer is initialized to a constant value 0.2. We train the BMCNN models for three noise levels: $\sigma = 15, 25$ and 50.

B. Training and Test Data

Recent studies [16], [17] show that less than million training samples are sufficient to learn a favorable network. Following these works, we use 400 images from the Berkeley Segmentation dataset(BSDS) [38] for the training. All the images are cropped to the size of 180×180 and data augmentation techniques like flip and rotation are applied. From all the images, training samples are extracted by the procedure in Sec. III-A1. We set the block size as $20 \times 20 \times 4$ and the stride as 20. The total number of the training samples is 259,200.

We test our algorithm on standard images that are widely used for the test of denoising. Fig. 5 shows the 12 images that constitute the test set. The set contains 4 images of size

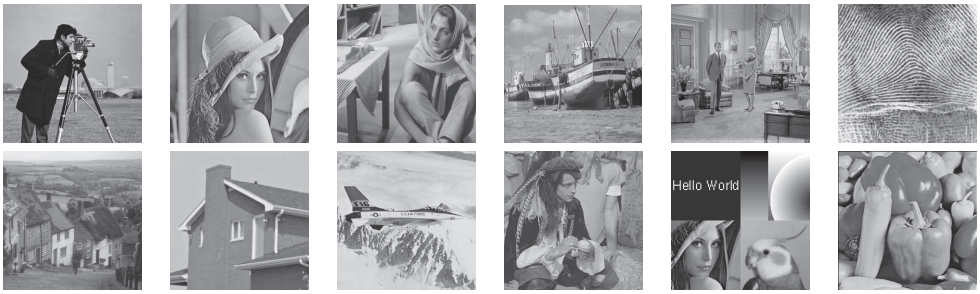


Fig. 5. The 12 test images used in the experiments

256×256 (*Cameraman, House, Peppers* and *Montage*), and 8 images of size 512×512 (*Lena, Barbara, Boat, Fingerprint, Man, Couple, Hill* and *Jetplane*). Note that the test set contains both repetitive patterns and irregularly textured images.

C. Comparision with the State-of-the-Art Methods

In this section, we evaluate the performance of the proposed BMCNN and compare it with the state-of-the-art denoising methods, including NSS based methods(i.e., BM3D [3], NCSR [4], and WNNM [7]), and training based methods (i.e., MLP [2], TNRD [16], and DnCNN [17]). All the experiments are performed on the same machine - Intel 3.4GHz dual core processor, nVidia GTX 780ti GPU and 16GB memory.

1) Quantitave and Qualitative Evaluation: The PSNRs of denoised images are listed in Table I. It can be seen that the proposed BMCNN yields the highest average PSNR for every noise level. It shows that the PSNR is improved by 0.1 0.2dB compared to DnCNN. Especially, there is large performance gain in the case of images with regular and repetitive structure, such as *Barbara* and Fingerprint, which are the images that the NSS based methods perform better than the learning based methods. In this sense, it is believed that adopting the block matching brings the cons of NSS to the learning based metod. Fig. 6 and 7 illustrate the visual results. The NSS based methods tend to blur the complex parts like a stalk of a fruit and the learning based methods often miss details on the repetitive parts such as the stripes of fingerprint. In contrast, the BMCNN recovers clear texture in both types of regions.

2) Running Time: Table II shows the average run-time of the denoising methods for the images of sizes 256×256 and 512×512. For TNRD, DnCNN and BMCNN, the times on GPU are computed. As shown, many conventional NSS based methods including NCSR and WNNM need very long times, which is mainly due to the complex optimization and/or matrix decomposition. On the other hands, since the BM3D consists of simple linear transform and non-linear filtering, it is much faster than the NCSR and WNNM. Since our BMCNN also consists of convolution and simple ReLU function, its computational cost is also less than the WNNM and NCSR. The BMCNN is, however, slower than other learning based approaches for three main reasons. First, our algorithm is a two-step approach that uses another end-to-end denoising algorithm as a preprocessing step. Therefore the computational cost is doubled. Second, the BMCNN contains a block matching step, which is difficult to be implement with GPU. In our algorithm, the block matching step takes almost half of the run-time. Finally, because of the nature of block-matching, the BMCNN is inherently a patch-based denoising algorithm. In order to prevent the artifacts around the boundaries between the denoised patches, the patches are extracted with overlapping. Hence, a pixel is processed multiple times and the overall run-time increases. Although our algorithm is slower for these reasons, our BMCNN is still competitive considering that it is much faster than the conventional NSS based methods and that it provides higher PSNR than others.

D. Effects of Network Formulation

In this subsection, we modify some settings of the BMCNN to investigate the relations between the settings and performance. All the additional experiments are made with $\sigma = 25$.

1) NSS Prior: In this paper, we take the NSS prior into account by adopting block matching, i.e., by using the aggregated similar patches as the input to the CNN. In order to show the effect of NSS prior, we conduct additional experiments: we train a network that estimates a denoised patch using only two patches as the input, specifically a noisy patch and corresponding pilot patch without further aggregation. We name the network as woBMCNN, and its PSNR results are summarized in Table. III. The result validates that the performance gain is owing to the block matching rather than two-step denoising. Interestingly, the woBMCNN does not performed better than DnCNN, which is used as the preprocessing. Actually, the woBMCNN can be interpreted as a deeper network with similar network formulation and a skip connection [39]. However, since the DnCNN is already a favorable network and the performance with the formulation is saturated, the deeper network can hardly perform better. On the other hands, the BMCNN encodes additional information to the network, which is shown to play an important role.

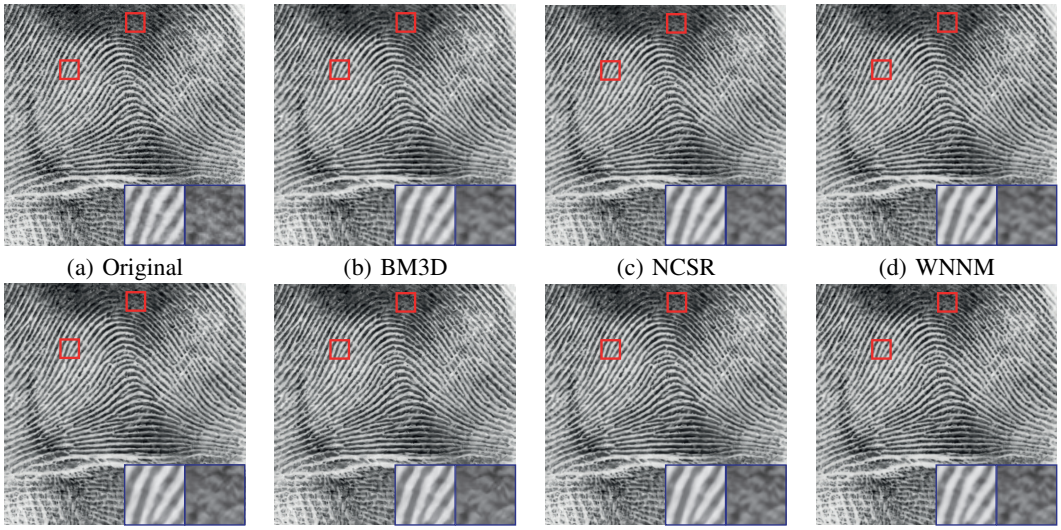
2) Patch Size: In many patch-based algorithms, the patch size is an important parameter that affects the performance. We train three networks with patch size 10×10, 20×20 (base) and 40×40. Table IV shows that 20×20 patch works better than other sizes. Moreover, networks of patch size 10×10 and 40×40 work even worse than its preprocessing.

TABLE I
PSNR OF DIFFERENT DENOISING METHODS. THE BEST RESULTS ARE HIGHLIGHTED IN **BOLD**.

Method	BM3D	NCSR	WNNM	MLP	TNRD	DnCNN	BMCNN
$\sigma = 15$							
Cameraman	31.91	32.01	32.17	-	32.18	32.61	**32.73**
Lena	34.22	34.11	34.35	-	34.23	34.59	**34.61**
Barbara	33.07	33.03	**33.56**	-	32.11	32.60	33.08
Boat	32.12	32.04	32.25	-	32.14	32.41	**32.42**
Couple	32.08	31.94	32.13	-	31.89	32.40	**32.41**
Fingerprint	30.30	30.45	**30.56**	-	30.14	30.39	30.41
Hill	31.87	31.90	32.00	-	31.89	**32.13**	32.08
House	35.01	35.04	**35.19**	-	34.63	35.11	35.16
Jetplane	34.09	34.11	34.38	-	34.28	**34.55**	34.53
Man	31.88	31.92	32.07	-	32.18	**32.42**	32.39
Montage	35.11	34.89	35.65	-	35.02	35.52	**35.97**
Peppers	32.68	32.65	32.93	-	32.96	33.21	**33.32**
Average	32.86	32.84	33.10	-	32.82	33.16	**33.26**
$\sigma = 25$							
Cameraman	29.44	29.47	29.64	29.59	29.69	30.11	**30.20**
Lena	32.06	31.95	32.27	32.28	32.05	32.48	**32.53**
Barbara	30.64	30.57	**31.16**	29.51	29.33	29.94	30.58
Boat	29.86	29.68	30.00	29.94	29.89	30.21	**30.25**
Couple	29.69	29.46	29.78	29.72	29.69	30.10	**30.12**
Fingerprint	27.71	27.84	27.96	27.66	27.33	27.64	**28.01**
Hill	29.82	29.68	29.96	29.83	29.77	29.99	**30.00**
House	32.95	32.98	**33.33**	32.66	32.64	33.23	33.32
Jetplane	31.63	31.62	31.89	31.87	31.77	32.06	**32.17**
Man	29.56	29.56	29.73	29.83	29.81	**30.06**	30.06
Montage	32.34	31.84	32.47	32.09	32.27	32.97	**33.47**
Peppers	30.21	29.96	30.45	30.45	30.51	30.80	**30.93**
Average	30.49	30.38	30.72	30.44	30.39	30.80	**30.97**
$\sigma = 50$							
Cameraman	26.18	26.15	26.47	26.37	26.56	26.99	**27.02**
Lena	29.05	28.97	29.32	29.28	28.94	29.42	**29.56**
Barbara	27.08	26.93	**27.70**	25.26	25.69	26.13	26.84
Boat	26.72	26.50	26.89	27.04	26.85	27.17	**27.19**
Couple	26.42	26.19	26.59	26.68	26.48	26.88	**26.91**
Fingerprint	24.55	24.52	**24.79**	24.21	23.70	24.14	24.65
Hill	27.05	26.87	27.12	**27.37**	27.11	27.31	27.33
House	29.70	29.69	30.25	29.82	29.40	30.08	**30.25**
Jetplane	28.31	28.23	28.61	28.56	28.43	28.74	**28.88**
Man	26.73	26.62	26.91	27.05	26.94	**27.18**	27.17
Montage	27.65	27.62	27.97	28.06	28.12	29.03	**29.50**
Peppers	26.69	26.64	26.97	26.71	27.05	27.30	**27.45**
Average	27.18	27.08	27.47	27.20	27.11	27.53	**27.73**

TABLE II
RUN TIME (IN SECONDS) OF VARIOUS DENOISING METHODS OF SIZE 256×256 WITH $\sigma = 25$.

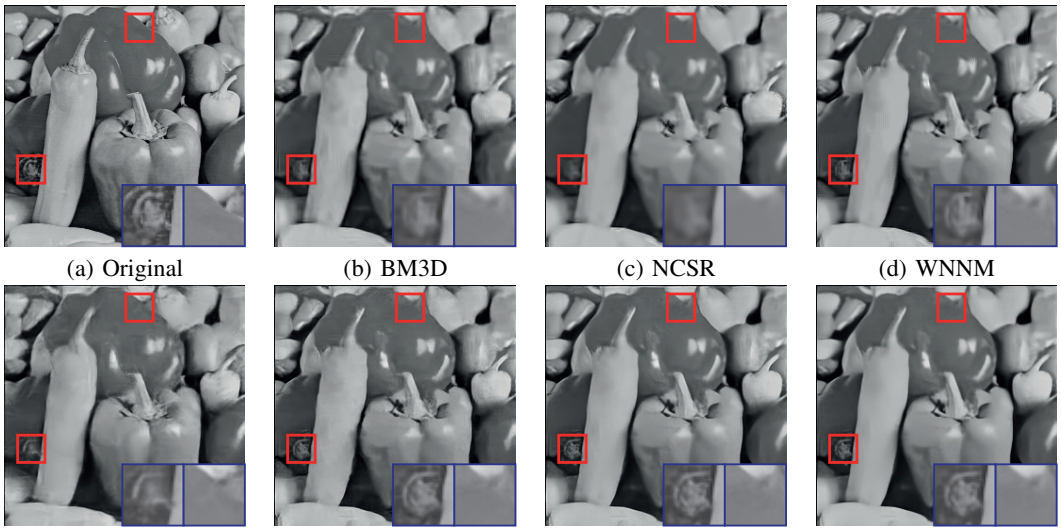
Methods	BM3D	NCSR	WNNM	MLP	TNRD	DnCNN	BMCNN
256×256	0.87	190.3	179.9	2.238	0.038	0.053	2.135
512×512	3.77	847.9	778.1	7.797	0.134	0.203	8.030



(a) Original (b) BM3D (c) NCSR (d) WNNM

(e) MLP (f) TNRD (g) DnCNN (h) BMCNN

Fig. 6. Denoising result of the *Fingerprint* image with $\sigma = 25$.



(a) Original (b) BM3D (c) NCSR (d) WNNM

(e) MLP (f) TNRD (g) DnCNN (h) BMCNN

Fig. 7. Denoising result of the *Peppers* image with $\sigma = 50$.

Burger et al. [2] revealed that a larger patch contains more information, and thus the neural network can learn more accurate objective function with larger training patches. On the contrary, we cannot train the mapping function reasonably with small patches. But the large patch degrades the block matching performance, because it becomes more difficult to find well matched patches as the patch size increases. Fig. 8 shows the block matching result and the error for various patch sizes. For a 10×10 patch and a 20×20 patch, the block-matching finds almost the same patches and the error is very small. For a 40×40 patch, on the other hands, decent portion of the patch does not fit well and the error becomes so big. Conventional NSS based algorithms including [3] and [5] also prefer small patches whose sizes are around 10×10 for these reasons. To conclude, 20×20 is the proper patch size that satisfies both CNN and NSS prior.

3) Stride: Since the proposed method is patch-based, its performance depends on the stride value to divide the input images into the patches. With a small stride, each pixel appears in many patches, which means that every pixel is processed

TABLE III
PSNR RESULTS OF BMCNN, WOBMCNN AND THEIR BASE PREPROCESSING-DNCNN.

	BMCNN	woBMCNN	DnCNN
Cameraman	30.20	30.08	30.11
Lena	32.53	32.45	32.48
Barbara	30.58	29.91	29.94
Boat	30.25	30.17	30.21
Couple	30.12	30.09	30.10
Fingerprint	28.01	27.61	27.64
Hill	30.00	29.96	29.99
House	33.32	33.20	33.23
Jetplane	32.17	32.02	32.06
Man	30.06	30.04	30.06
Montage	33.47	33.02	32.97
Peppers	30.93	30.81	30.80
Average	30.97	30.78	30.80

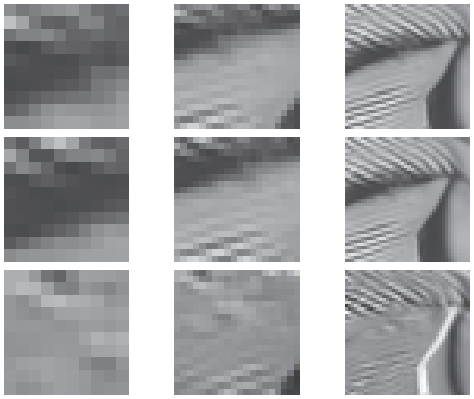
TABLE IV
PSNR RESULTS OF BMCNN WITH VARIOUS PATCH SIZES.

	10×10	20×20	40×40
Cameraman	28.82	30.20	30.00
Lena	30.42	32.53	32.39
Barbara	29.05	30.58	29.72
Boat	29.01	30.25	30.07
Couple	28.89	30.12	29.97
Fingerprint	27.24	28.01	27.51
Hill	28.83	30.00	29.89
House	30.85	33.32	33.11
Jetplane	30.20	32.17	31.96
Man	28.84	30.06	29.96
Montage	30.47	33.47	32.72
Peppers	29.31	30.93	30.67
Average	29.33	30.97	30.66

multiple times. It definitely increases the computational costs but has the possiblity of performance improvement. We test our algorithm with various stride values and the results are summarized in the Table V. From the result, we determine that a stride value around the half of the patch size shows reasonable performance for both the run time and the PSNR.

4) Pilot Signal: We also conduct an experiment to see how the different preprocessing method (other than DnCNN in the previous experiments) affect the the overall performance. For the experiment, we employ BM3D [3] due to its NSS based nature and reasonable run time. Table VI shows the denoising performance with different preprocessing methods and two interesting characteristics can be found.

- The performance on the individual image depends on the preprocessing method. BMCNN-BM3D shows better performance on *Barbara, Fingerprint* and *House*, where NSS based WNNM performed better than the CNN based DnCNN.
- However, the overall performance shows negligible difference. It implies the overall performance of the denoising network depends on the network formulation, not the preprocessing.



(a) 10×10 Error : 0.0515 (b) 20×20 Error : 0.0672 (c) 40×40 Error : 0.1813

Fig. 8. The illustration of block matching result for various patch sizes. The first row shows reference patches, the second row shows the 3rd-similar patches to the references and the third row shows the difference of the first and the second row. The error is defined as the average value of the difference.

TABLE V
PSNR AND RUN TIME RESULTS OF BMCNN WITH VARIOUS STRIDE.

	5	10	15	20
Cameraman	30.21	30.20	30.20	30.18
Lena	32.53	32.53	32.52	32.51
Barbara	30.58	30.58	30.56	30.49
Boat	30.25	30.25	30.25	30.24
Couple	30.12	30.12	30.12	30.11
Fingerprint	28.03	28.01	28.01	27.97
Hill	30.00	30.00	30.00	30.00
House	33.32	33.32	33.32	33.30
Jetplane	32.17	32.17	32.16	32.16
Man	30.06	30.06	30.05	30.05
Montage	33.49	33.47	33.45	33.40
Peppers	30.94	30.93	30.93	30.92
Average PSNR	30.98	30.97	30.96	30.94
Average time(256×256)	4.271	2.151	1.765	1.603
Average time(512×512)	16.79	8.101	6.492	5.777

V. CONCLUSION

In this paper, we have proposed a framework that combines two dominant approaches in up-to-date image denoising algorithms, i.e., the NSS prior based methods and CNN based methods. We train a network that estimates a noise component from a group of similar patches. Unlike the conventional NSS based methods, our denoiser is trained in a data-driven manner to learn an optimal mapping function and thus achieves better performance. Our BMCNN also shows better performance than the existing CNN based method especially in the case of images with regular structure, because the BMCNN considers NSS in addition to the local characteristics.

ACKNOWLEDGMENT

This research was supported in part by Samsung Electronics Co., Ltd., and in part by Projects for Research and Development of Police science and Technology under Center for Research and Development of Police science and Technology and

TABLE VI
PSNR OF BMCNN RESULTS WITH TWO DIFFERENT PREPROCESSING.

	BMCNN-DnCNN	BMCNN-BM3D
Cameraman	**30.20**	30.03
Lena	**32.53**	32.49
Barbara	30.58	**31.23**
Boat	**30.25**	30.16
Couple	**30.12**	30.02
Fingerprint	28.02	**28.06**
Hill	30.00	**30.05**
House	33.32	**33.43**
Jetplane	**32.17**	32.14
Man	**30.06**	29.94
Montage	**33.47**	33.31
Peppers	**30.93**	30.78
Average	**30.97**	30.97

Korean National Police Agency (PA-C000001) and supported by the Brain Korea 21 Plus Project in 2018.

REFERENCES

[1] A. Buades, B. Coll, and J. -M. Morel, "A non-local algorithm for image denoising," in *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2005.

[2] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with bm3d?" in *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2012.

[3] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative ltering," *IEEE Trans. Image process.*, vol. 16, no. 8, pp. 2080-2095, 2007.

[4] W. Dong, L. Zhang, G. Shi, and X. Li, "Nonlocally centralized sparse representation for image restoration," *IEEE Trans. Image process.*, vol. 22, no. 4, pp. 1620-1630, 2013.

[5] W. Dong, G. Shi, and X. Li, "Nonlocal image restoration with bilateral variance estimation: a low-rank approach," *IEEE Trans. Image process.*, vol. 22, no. 2, pp. 700-711, 2013.

[6] A. Foi, V. Katkovnik, and K. Egiazarian, "Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images," *IEEE Trans. Image process.*, vol. 16, no. 5, pp. 1395-1411, 2007.

[7] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted nuclear norm minimization with application to image denoising," in *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2014.

[8] O. G. Guleryuz, "Weighted overcomplete denoising," in *Proc. of Asilomar Conference on Signals, Systems and Computer*, 2003, pp. 1992-1996.

[9] V. Jain and S. Seung, "Natural image denoising with convolutional networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[10] X. Jia, X. Feng, and W. Wang, "Adaptive regularizer learning for low rank approximation with application to image denoising," in *Proc. of International Conference on Image Processing (ICIP)*, 2016.

[11] B. Wang, T. Lu, and Z. Xiong, "Adaptive boosting for image denoising: Beyond low-rank representation and sparse coding," in *Proc. of the International Conference on Pattern Recognition (ICPR)*, 2016.

[12] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Nonlocal sparse models for image restoration," in *Proc. of the International Conference on Computer Vision (ICCV)*, 2009.

[13] M. Nejati, S. Samavi, S. R. Soroushmehr, and K. Najarian, "Lowrank regularized collaborative ltering for image denoising," in *Proc. of International Conference on Image Processing (ICIP)*, 2015.

[14] X. Zha, Zhiyuan ad Liu, Z. Zhou, X. Huang, J. Shi, Z. Shang, L. Tang, Y. Bai, Q. Wnag, and X. Zhang, "Image denoising via group sparsity residual constraint," in *Proc. of International Conference on Acoustics, Speech and Signal Processing(ICASSP)*, 2017.

[15] U. Schmidt and S. Roth, "Shrinkage elds for effective image restoration," in *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2014.

[16] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A exible framework for fast and effective image restoration," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2016.

[17] 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. Image process.*, 2017.

[18] H. C. Burger, C. Schuler, and S. Harmeling, "Learning how to combine internal and external denoising methods," in *Proc. of German Conference on Pattern Recognition*, 2013.

[19] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image process.*, vol. 15, no. 12, pp. 3736-3745, 2006.

[20] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM review*, vol. 52, no. 3, pp. 471-501, 2010.

[21] Y.LeCun,L.Bottou,Y.Bengio,andP.Haffner,"Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

[22] C. J. Schuler, H. Christopher Burger, S. Harmeling, and B. Scholkopf, "A machine learning approach for non-blind image deconvolution," in *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2013.

[23] S. Geman and C. Grafgne, "Markov random eld image models and their applications to computer vision," in *Proc. of the International Congress of Mathematicians*, 1986.

[24] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. of European Conference on Computer Vision (ECCV)*, 2014.

[25] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Image process.*, vol. 19, no. 11, pp. 2861-2873, 2010.

[26] C. Dong, Y. Deng, C. Change Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *Proc. of the International Conference on Computer Vision (ICCV)*, 2015.

[27] J. Kim, J. K. Lee, and K. M. Lee, "Deeply-recursive convolutional network for image super-resolution," in *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2016.

[28] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks,"in *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2016.

[29] Y. Bengio,P. Simard, and P. Frasconi, "Learninglong-term dependencies with gradient descent is difcult," *IEEE Trans. Neural networks*, vol. 5, no. 2, pp. 157-166, 1994.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for imagerecognition," in *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2016.

[31] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. of International Conference on Machine Learning (ICML)*, 2015.

[32] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Proc. of Asian Conference on Computer Vision(ACCV)*, 2014.

[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv*:1409.1556, 2014.

[34] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems(NIPS)*, 2016.

[35] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proc. of the International Conference on Computer Vision(ICCV)*, 2015.

[36] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. of the ACM international conference on Multimedia (ACM MM)*, 2014.

[37] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Trans. Computational Imaging*, 2016.

[38] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference for Learning Representations*, 2015.

[39] X. Glorot and Y. Bengio, "Understanding the difculty of training deep feedforward neural networks." in *Aistats*, vol. 9, 2010, pp. 249-256.

[40] P. Arbelaez, C. Fowlkes, and D. Martin, "The berkeley segmentation dataset and benchmark," *http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds*, 2007.

[41] X. Mao, C. Shen, and Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Advances in Neural Information Processing Systems(NIPS)*, 2016.