Deep Iterative Down-Up CNN for Image Denoising

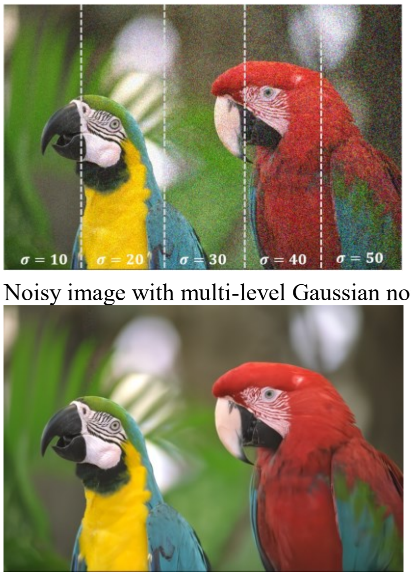
Songhyun Yu, Bunjun Park, and Jechang Jeong
Department of Electronics and Computer Engineering, Hanyang University, Seoul, Korea
fkdlzmftld@gmail.com, kkbbbj@gmail.com, jjeong@hanyang.ac.kr

Abstract

Networks using down-scaling and up-scaling of feature maps have been studied extensively in low-level vision research owing to efficient GPU memory usage and their capacity to yield large receptive fields. In this paper, we propose a deep iterative down-up convolutional neural network (DIDN) for image denoising, which repeatedly decreases and increases the resolution of the feature maps. The basic structure of the network is inspired by U-Net which was originally developed for semantic segmentation. We modify the down-scaling and up-scaling layers for image denoising task. Conventional denoising networks are trained to work with a single-level noise, or alternatively use noise information as inputs to address multi-level noise with a single model. Conversely, because the efficient memory usage of our network enables it to handle multiple parameters, it is capable of processing a wide range of noise levels with a single model without requiring noise-information inputs as a work-around. Consequently, our DIDN exhibits state-of-the-art performance using the benchmark dataset and also demonstrates its superiority in the NTIRE 2019 real image denoising challenge.

1. Introduction

Image denoising is a representative low-level vision task that restores a clean image, x, from noisy image, y. There are many types of noise that can be generated in an image, but the noise caused by poor illumination or high temperature problems occurring during the image acquisition step can be assumed to follow a Gaussian distribution. When noise n conforms to a Gaussian distribution, additive white Gaussian noise (AWGN) is modeled as $y = x + n$. During the past decades, many studies have been conducted with the aim of reducing Gaussian noise in images [2–6]. The performance of natural image denoising has reportedly converged, making any further major performance improvements significantly more challenging [1].



(a) Noisy image with multi-level Gaussian noise

(b) Restored image using DIDN (PSNR: 34.86 dB)

Figure 1: Our single DIDN handles multi-level noise without requiring any noise information to be input.

Burger et al. [7] have achieved denoising performance similar to that of BM3D [5] using a plain neural network, and many algorithms have been developed to significantly improve the performance of image denoising using convolutional neural network (CNN) [8–14]. However, existing methods have some limitations in their network architectures and training methods. First, the requirement for the output of a network employed in an image denoising task to be a denoised image means that deep features used in that network need to have the same resolutions as the output, which consumes a lot of GPU memory and training time. These GPU and training-time costs account for the fact that existing plain CNN-based denoising architectures are limited in their depth, number of parameters, and receptive field. Performance improvements can be achieved using a hierarchical network structure that changes the resolution of the feature maps, as then the receptive field can be much larger for the same GPU memory cost [14]. Second, existing networks either require a model per noise level to be processed, or rely on noise-information input to facilitate process multi-level noise with a single model.

However, this solution to handling multi-level noise is impractical, as in practical applications noise information is not readily available to the network, and noise levels vary widely depending on the situation.

Liu et al. [14] developed a network for image restoration by combining the wavelet transform and inverse wavelet transform approaches with a U-Net [24] structure, achieving a good trade-off between the computational complexity and the receptive field. However, as the wavelet transform consists of a convolution with specific weights and a sub-sampling process, it is considered a special case of the convolution layer; using this approach may limit performance compared to the performance that can be achieved using a trainable convolution layer.

To address these limitations, we embed down-scaling into the contracting process of the network with a trainable convolution layer with a stride of 2, and embed up-scaling in the expanding process using a subpixel layer [15], which was proposed in image super-resolution work for the up-scaling of features. As the subpixel layer performs up-scaling only by rearranging features and not by performing any convolution operation, loss during the up-scaling process can be reduced if a sufficient number of features are provided before up-scaling. We also test the proposed network's ability to cope with unknown noise levels and develop efficient training methods to improve the multi-level noise model to offer greater practical utility. Our network's efficient GPU memory usage enables it to consider enough parameters to handle multiple-level noise with a single model. Experimental results demonstrate that our multi-level noise model shows similar performance to a single-level noise model without requiring the input of noise information (Figure 1). We also use the weight-averaging method [16] for image denoising to reduce the bias caused by weight selection and improve the performance without increasing the model's parameters or computational complexity. Although the effect of this weight-averaging strategy has been validated in image classification work, we pioneer its successful application in image denoising tasks. In summary, the contributions of this work are as follows:

- A novel CNN architecture iteratively contracting and expanding features with very large receptive field.
- Modification of the down- and up-scaling process used by U-Net for image denoising task.
- Application of the weight-averaging technique to Gaussian noise image denoising, resulting in a more generalized and performance-enhanced model without the additional parameters.
- An efficient method to train a single model such that it can handle multi-level noise (unknown noise) without noise information inputs.
- State-of-the-art Gaussian image denoising performance.

2. Related Work

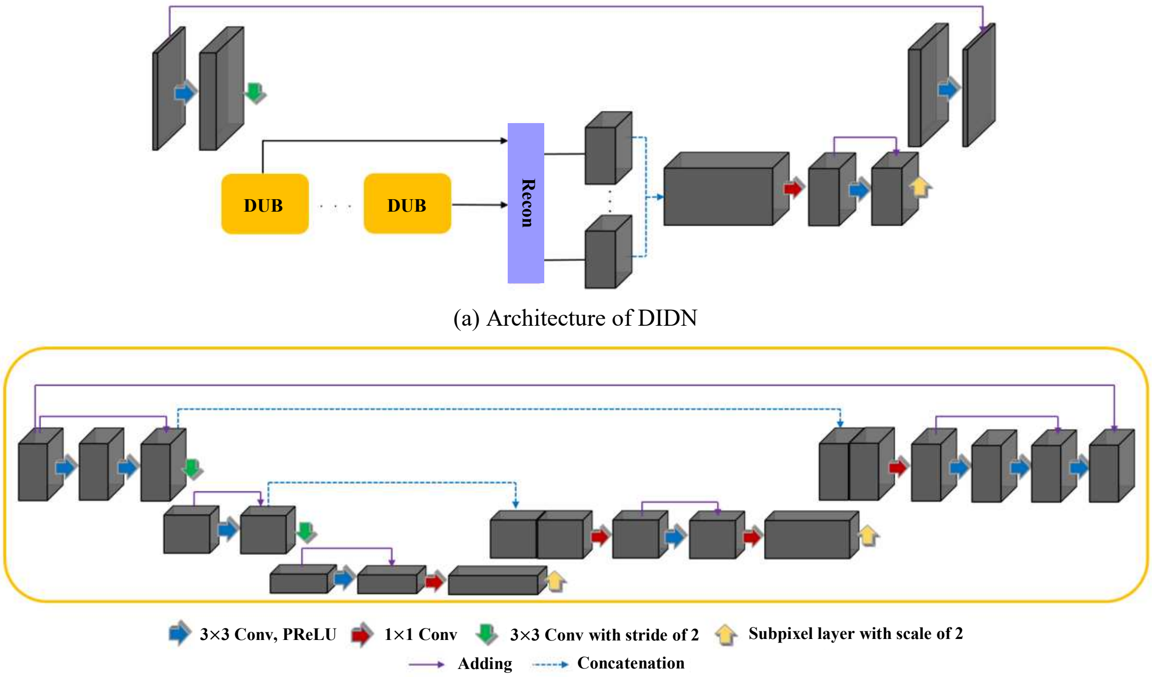
2.1. Deep learning based image denoising

The development of deep learning has facilitated a large performance improvement in image denoising. Jain et al. [18] were the first to use a simple CNN with five layers for image denoising, but did not achieve significant performance improvement over conventional methods. An auto-encoder-based denoising model was proposed in [19] but the performance fell short of that possible using BM3D [5]. Burger et al. [7] trained a multi-layer perceptron to learn a mapping from a noisy image patch to a clean image, achieving similar performance to BM3D. DnCNN [8] achieved a significant performance improvement over conventional methods in image denoising using convolution, batch normalization [35], and ReLU as a basic structure, and successfully trained deep networks by utilizing global residual learning [21]. Zhang et al. [13] proposed a CNN with seven layers, considering the trade-off between computational cost and accuracy. This succeeded in increasing the receptive field using dilated convolution [20]. Tai et al. [9] proposed the MemNet, which offers a large receptive field while maintaining a small number of parameters by using a recursive CNN structure and dense skip connections [22]. Finally, Zhang et al. [11] proposed a residual dense network (RDN) that uses both residual learning and dense connection as its basic structure, maximizing feature reuse and achieving a significant improvement in the performance of Gaussian noise image denoising.

All the studies acknowledged above are subject to some limitations: they assume known noise, and require the training of a specific model for each noise level to be considered. As such, they cannot handle unknown noise or multi-level noise with a single model. To address this problem, FFDNet [10] was designed to use a noise map as an input to the network, resulting in a single model which can process multi-level noise (Gaussian noise σ from 0 to 75). The UDN [12] trains on various noise levels using the noise level as input to the trainable projection unit of the network (with noise σ from 0 to 29 and 30 to 55 for each model). However, these methods still require a noise level as input when testing, and as the results vary depending on the input noise information, they are difficult to apply to unknown noise data.

2.2. Deep networks using down-up scaling

To maintain the depth and computational complexity of the network while increasing the receptive field, Zhang et al. [13] used dilated convolution, but this approach suffers from the gridding artifact because it sub-samples the features sparsely [23]. In order to achieve a better trade-off between the receptive field and the computational cost,



(a) Architecture of DIDN

(b) Structure of DUB

Figure 2: The architecture of the proposed DIDN.

some studies have been conducted into the use of down- and up-scaling of the feature maps. U-Net [24] was proposed in semantic segmentation. In U-Net, max-pooling is used to reduce the resolution of feature maps by half and simultaneously increase the number of feature maps by a factor of two to reduce data loss. Up-sampling followed by 2×2 convolution are used for up-scaling the features. Liu et al. [14] used the wavelet transform and the inverse wavelet transform approaches for the down- and up-scaling of feature maps, respectively, to increase the receptive field. As the wavelet transform approach consists of a convolution and sub-sampling, it can be seen as a special case of a convolution layer. Shi et al. [15] proposed a sub-pixel convolution layer for up-scaling of low resolution features at super-resolution, which greatly reduces the computational complexity compared to that of the deconvolution layer. FFDNet implemented reversible down- and up-scaling using paired sub-sampling and subpixel convolution, resulting in high GPU memory efficiency and an increased receptive field. DBPN [25] learned the up-scaling process by iteratively up- and down-scaling feature maps using the deconvolution layer and a convolution layer with a stride of 2 in the image super-resolution work, and achieved state-of-the-art performance.

3. Proposed Network

In this section, we introduce the overall architecture and properties of the proposed network. We discuss the down- and up-scaling (down-up scaling) strategy used in the contraction and expansion steps in the network, and the network configurations required for efficient training. In the following subsections, we compare the results of the single-noise model with those of the multi-noise model, and explain the training strategies employed to train the multi-level noise model efficiently.

3.1. Network architecture

The U-Net [24] was initially proposed for semantic segmentation. U-Net consists of two paths: a contraction path that reduces the size of deep features, and an expansion path that increases the size of these features. The core principle of U-Net is to reduce the resolution of the features to increase the receptive field, and then reuse the features through concatenation of matching resolution levels to minimize information loss caused by down-up scaling. The efficiency of U-Net's U-shaped structure is verified in image denoising [14] and ascribed to its large receptive field, high GPU memory efficiency, and low computational cost. DBPN [25] demonstrates that iterative down-up scaling of feature maps is effective for learning an image super-resolution task. However, this method increases the overall size of the feature maps, decreasing the receptive field and increasing the computational complexity.

Drawing on [24] and [25], we propose an iterative down-up scaling network termed the deep iterative down-up network (DIDN), offering a large receptive field and efficient GPU memory usage by sequential repetition of the contraction and expansion processes. Figure 2 shows the overall architecture of the proposed network. Here, the gray blocks represent feature maps, and the feature maps construct a hierarchical structure comprised of four distinct resolution levels. DIDN consists of four parts: feature extraction, down-up block (DUB), reconstruction, and enhancement.

Initial feature extraction: When the size of the input image is H×W, DIDN first extracts N features using a 3×3 convolution on the input image, and extracts the features of $\frac{H}{2} \times \frac{W}{2} \times 2N$ size through the convolution layer using a stride of 2.

DUB: The extracted features subjected to iterative down-up scaling through several DUBs. In the DUB, contraction and expansion are performed by two down-scaling and up-scaling processes. A 3×3 convolution layer with a stride of 2 and a subpixel layer are used in down- and up-scaling, respectively. In the down-scaling process, the size of the feature maps is decreased by half in the horizontal and vertical directions, and the number of the features is doubled. In the up-scaling process, because the number of input features is reduced by a quarter through the subpixel layer, the number of feature maps is increased through the 1×1 convolution layer before the subpixel layer to maintain information density. As in U-Net [24], features of the same resolution level are concatenated to increase the reuse of these features in the hierarchical structure. The features at the beginning and the end of the block are linked by skip connection [21].

Reconstruction: Inspired by MemNet [9], we place a common reconstruction block after the last DUB to take advantage of all the local output. The outputs of all the DUBs form the inputs to the reconstruction block, and all the outputs of the reconstruction block are concatenated to go through the enhancement stage. The reconstruction block consists of nine convolution layers (Conv) followed by parametric rectified linear units (PReLU) [38]. More specifically, there are four consecutive residual blocks consisting of 'Conv + PReLU + Conv + PReLU' with additional Conv at the end.

Enhancement: Finally, through the 1×1 convolution, the number of output feature maps in the reconstruction block is decreased, and up-scaling is performed at the subpixel layer to generate the final denoised image.

3.2. Down-up process

Other approaches to select the upscaling layer do exist, such as using a deconvolution layer and up-sampling followed by a convolution layer. However, these upscaling layers contain interpolation or padding processes which can include degradation in the feature maps. As image denoising is a low-level vision task in which it is important to enhance the pixel-level accuracy, in DIDN we adopt a subpixel convolution layer as an up-scaling operator. The subpixel convolution layer requires neither interpolation nor a padding process, but instead allows the network to propagate detail information directly from low resolution to higher resolution, an advantageous method for upscaling features in image denoising. There are also other options in the down-scaling layer such as max-pooling and sub-sampling, but we chose to adopt a trainable convolution layer to improve the performance.

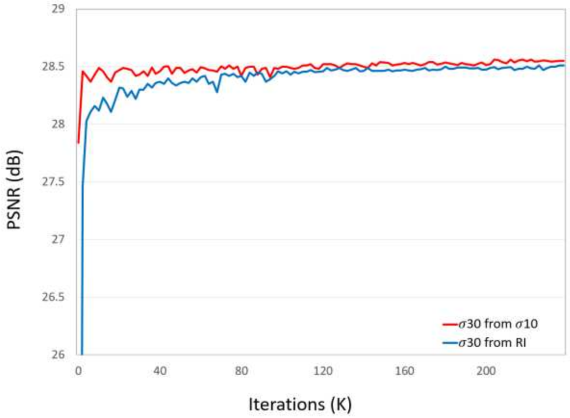


Figure 3: Effect of pre-trained weight initialization on grayscale BSD68 dataset with noise level 30.

Methods	Noise level		
	10	30	50
Single_RI	35.16	29.90	27.85
Single_PI	**35.18**	30.02	27.91
Multi_RI	35.14	30.01	27.94
Multi_PI_σ50	35.14	30.02	27.94
Multi_PI_ σ10	35.15	**30.03**	**27.95**

Table 1: PSNR (dB) comparison of training and weight initialization methods using the grayscale Kodak24 dataset. Best performance at each noise level is bolded.

3.3. Multiple noise levels

In EDSR [17], fast convergence and improved results are obtained by using the weights of the pre-trained model, at a lower scale, as the initial values of the weights for learning the model at a higher scale. The authors conclude that the different scales are interrelated in the super-resolution task.

In Gaussian noise image denoising, the degree of degradation varies depending on the noise levels, but as the noise properties are the same, we can extend this weight initialization strategy to image denoising. Figure 3 shows the result of training at a noise level of 30 on grayscale image denoising. The blue line represents the use of random initialization (RI), whereas the red line represents the use of pre-trained initialization (PI) weights at a noise level of 10. Using the pre-trained model, converges occurs much faster and ultimately performance is improved even further, indicating that Gaussian noise is also interrelated between

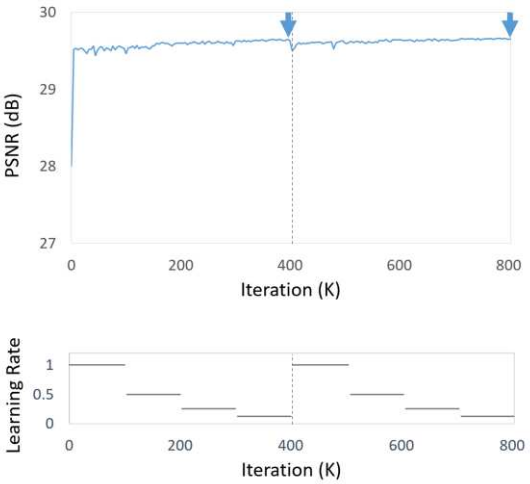


Figure 4: Weight averaging points. Upper and lower tindicate validation PSNR (dB) and learning rate, respectively, during training.

adjacent noise-levels.

From this property, the possibility arises of addressing a wide range of noise levels in a single model. The multi-level noise model, capable of handling noise levels ranging from 5 to 50, is trained in three ways: 1) using RI, 2) using PI from a noise level of 10, and 3) using PI from a noise level of 50. Table 1 shows the results of the experiment on the Kodak24 [37] dataset. In Table 1, the first model is indicated as 'Multi_RI'; the second model as 'Multi_PI_ σ50'; and, the third model as 'Multi_PI_ σ10'. For single-level noise model, the model using PI is indicated as 'Single_PI' and the model using RI is indicated as 'Single_RI' in Table 1. For the single-level noise model using PI, the model for noise level 10 is initialized from the weights pre-trained at noise level 5; the model for noise level 30 is initialized from the weights pre-trained at noise level 10; and, the model for noise level 50 is initialized from the weights pre-trained at noise level 30.

In the single-level noise model, using PI increases the average peak signal-to-noise ratio (PSNR) by 0.02, 0.12, and 0.06 dB at noise levels of 10, 30, and 50, respectively, compared to the results with RI. Comparing the RI results produced by the single-level noise model with those produced by the multi-level noise model, it is found that the multi-level noise model outperforms the single-level noise model at high noise levels, but it exhibits inferior performance at low noise levels. This tendency is consistent with the results of VDSR [34], where a multi-scale model shows a lower PSNR than a single-scale model at a low scale in image super-resolution. Using the PI from noise level 10 improves the DIDN's performance at low noise levels and also increases the PSNR at higher noise levels. Consequently, our multi-level noise model outperforms single-level models at almost all noise levels. Furthermore, because noise information in the real-world is often unknown, the multi-level noise model is much more practically applicable than any single-level model.

Dataset	Methods	Noise level		
		10	30	50
BSD68	Model 1	33.970	28.578	26.461
	Model 2	33.976	28.577	26.459
	Model1 + Model2	**33.978**	**28.583**	**26.467**
Kodak24	Model 1	35.152	30.026	27.947
	Model 2	35.153	30.028	27.950
	Model1 + Model2	**35.163**	**30.037**	**27.959**

Table 2: Effect of weight averaging on grayscale image denoising. Best performance at each noise level is bolded.

Owing to DIDN's efficient GPU memory usage, it can accommodate deeper layers, more features, and a sufficiently large number of parameters (190M for DIDN using 6 DUBs) to enable a single model to be trained to process a wide range of noise levels without requiring any noise information inputs. We can conclude that the proposed multi-level noise model successfully performs the functions of both noise estimation and noise reduction within a single model.

4. Experimental Results

This section details the dataset, training steps, and ensemble strategy used in the experiment. Thereafter, objective and subjective comparisons with the results of state-of-the-art studies are drawn. Moreover, we present the results of our DIDN's performance in the NTIRE 2019 real image denoising challenge [40].

4.1. Dataset

To train our model, we use the DIV2K dataset [29], composed of 800 training images and 100 validation images, each with a resolution of 2K. For the comparison, BSD68 [30] and Kodak24 [37] are used as grayscale and color versions of the test dataset, respectively. Noisy images are generated by adding Gaussian noise at a specific noise level to the clean images from the datasets.

4.2. Training / Implementation details

For training, all images are split into 64×64 patches. One batch consists of 16 randomly selected patch pairs of training data, and 36K iterations constitute one epoch. All patches are augmented using random rotation and flip. The Adam optimizer [39] with an initial learning rate of 10^{-4} is used for training, and the learning rate is halved every 3 epochs. In total, a single model is trained over 12 epochs (approximately 400K iterations), during which process the learning rate is decreased 3 times. Approximately 3 days

Methods	BSD68					Kodak24				
	Noise level					Noise level				
	10	20	30	40	50	10	20	30	40	50
Noisy	28.26 / 0.7092	22.35 / 0.4687	18.97 / 0.3349	16.64 / 0.2526	14.91 / 0.1982	28.23 / 0.6574	22.28 / 0.4013	18.87 / 0.2731	16.52 / 0.2001	14.79 / 0.1542
BM3D [5]	33.32 / 0.9158	29.61 / 0.8337	27.75 / 0.7731	26.46 / 0.7242	25.60 / 0.6858	34.39 / 0.9127	30.93 / 0.8405	29.12 / 0.7877	27.84 / 0.7452	26.98 / 0.7140
DnCNN [8]	33.88 / 0.9270	30.27 / 0.8563	28.36 / 0.7999	27.11 / 0.7541	26.23 / 0.7189	34.90 / 0.9223	31.47 / 0.8576	29.62 / 0.8071	28.37 / 0.7666	27.49 / 0.7368
IRCNN [13]	33.74 / 0.9262	30.16 / 0.8562	28.26 / 0.7989	27.08 / 0.7548	26.19 / 0.7171	34.76 / 0.9215	31.38 / 0.8576	29.52 / 0.8056	28.37 / 0.7676	27.45 / 0.7342
FFDNet [10]	33.76 / 0.9266	30.23 / 0.8576	28.39 / 0.8032	27.18 / 0.7597	26.29 / 0.7245	34.81 / 0.9226	31.47 / 0.8603	29.69 / 0.8123	28.51 / 0.7741	27.62 / 0.7437
DIDN	33.98 / 0.9284	30.44 / 0.8614	28.58 / 0.8075	27.37 / 0.7655	26.47 / 0.7310	35.16 / 0.9263	31.83 / 0.8677	30.04 / 0.8222	28.84 / 0.7856	27.96 / 0.7562
DIDN+	34.01 / 0.9286	30.47 / 0.8618	28.61 / 0.8081	27.40 / 0.7663	26.50 / 0.7320	35.20 / 0.9267	31.87 / 0.8683	30.08 / 0.8230	28.88 / 0.7867	28.01 / 0.7576

Table 3: PSNR (dB) / SSIM comparison of methods on gray-scale image denoising. Dataset BSD68 and Kodak24 are used for noise levels 10, 20, 30, 40, and 50. Best and second best performances are highlighted in red and blue, respectively.

Methods	CBSD68					Kodak24				
	Noise level					Noise level				
	10	20	30	40	50	10	20	30	40	50
Noisy	28.30 / 0.7114	22.40 / 0.4707	19.03 / 0.3363	16.72 / 0.2539	15.00 / 0.1993	28.24 / 0.6598	22.31 / 0.4030	18.93 / 0.2744	16.60 / 0.2011	14.87 / 0.1549
CBM3D [36]	35.89 / 0.9507	31.89 / 0.8935	29.72 / 0.8432	28.08 / 0.7888	27.36 / 0.7622	36.57 / 0.9425	32.92 / 0.8901	30.89 / 0.8452	29.17 / 0.7937	28.62 / 0.7765
DnCNN [8]	36.12 / 0.9536	32.37 / 0.9050	30.32 / 0.8611	28.95 / 0.8223	27.92 / 0.7882	36.58 / 0.9446	33.20 / 0.8984	31.28 / 0.8579	29.95 / 0.8225	28.94 / 0.7915
IRCNN [13]	36.06 / 0.9533	32.27 / 0.9045	30.22 / 0.8607	28.85 / 0.8222	27.86 / 0.7889	36.70 / 0.9448	33.19 / 0.8984	31.24 / 0.8581	29.91 / 0.8229	28.92 / 0.7939
FFDNet [10]	36.14 / 0.9540	32.34 / 0.9045	30.31 / 0.8603	28.96 / 0.8217	27.96 / 0.7881	36.80 / 0.9462	33.32 / 0.9000	31.39 / 0.8596	30.08 / 0.8248	29.10 / 0.7949
DIDN	36.48 / 0.9565	32.73 / 0.9108	30.71 / 0.8706	29.36 / 0.8348	28.35 / 0.8041	37.32 / 0.9500	33.88 / 0.9083	31.97 / 0.8724	30.68 / 0.8418	29.72 / 0.8156
DIDN+	36.52 / 0. 9567	32.77 / 0.9114	30.75 / 0.8714	29.40 / 0.8359	28.40 / 0.8054	37.37 / 0.9503	33.94 / 0.9090	32.03 / 0.8734	30.75 / 0.8431	29.80 / 0.8173

Table 4: PSNR (dB) / SSIM comparison of methods on color image denoising. Dataset CBSD68 and Kodak24 are used for noise levels 10, 20, 30, 40, and 50. Best and second best performances are highlighted in red and blue, respectively.

Methods	BSD68			Kodak24		
	Noise level			Noise level		
	15	25	50	15	25	50
Noisy	24.79	20.48	14.91	24.74	20.39	14.79
BM3D [5]	31.08	28.57	25.60	32.30	29.92	26.98
MWCNN [14]	31.86	29.41	26.54	33.14	30.81	28.02
DIDN	31.85	29.39	26.47	33.16	30.81	27.96
DIDN+	31.88	29.42	26.50	33.20	30.85	28.01

Table 5: PSNR (dB) comparison with the latest high-performance methods on grayscale image denoising.

Methods	CBSD68			Kodak24		
	Noise level			Noise level		
	10	30	50	10	30	50
Noisy	28.30	19.03	15.00	28.24	18.93	14.87
CBM3D [36]	35.89	29.72	27.36	36.57	30.89	28.62
RDN [11]	36.47	30.67	28.31	37.31	31.94	29.66
DIDN	36.48	30.71	28.35	37.32	31.97	29.72
DIDN+	36.52	30.75	28.40	37.37	32.03	29.80

Table 6: PSNR (dB) comparison with the latest high-performance methods on color image denoising.

are required to train a single model using the GeForce GTX 1080Ti. Drawing on [32], we use $l1$-loss to train the model as follows:

$$l_1(\theta) = \frac{1}{N}\sum_{i=1}^{N} |F(x_i;\theta) - y_i|, \tag{1}$$

where N is a batch size, $F(\cdot)$ is the network function with learnable parameter θ, and x_i and y_i denote patch pairs of the noisy image and ground truth in the training data.
Our DIDN has 6 DUBs and extracts 128 initial feature maps. All convolution layers have a kernel size of 3×3 or 1×1. For the Gaussian noise image denoising, two DIDNs are trained, one for grayscale image denoising and the other for color image denoising.

4.3. Ensemble strategy

In machine learning, the ensemble technique is a method to improve generalization and performance by reducing the bias and variance of a single model. Three ensemble methods, namely snapshot ensemble [16], self-ensemble [26], and model ensemble, are tested on the DIDN.

Snapshot ensemble is a method to train a model by periodically changing the learning rate and then averaging the weight values at the end of each cycle. The cosine learning rate is used in [16], but we use the Adam optimizer, as explained in Section 4.2, halving the learning rate every 3 epochs. We train a model for two cycles and average their weights of the end of each cycle as shown in Figure 4. Table 2 lists the effects of using snapshot-ensemble on the Kodak24 dataset at noise levels of 10, 30, and 50. The highest performances during the two learning rate cycles are similar, and the performance is slightly improved when two weights are averaged, which is effective because it does not require additional parameters or additional computation for testing. As in [17], the self-ensemble generates eight inputs by the rotation and flip of one input, producing a total of eight outputs through the same model. The outputs are then inverse-transformed and averaged to create the final output. Tables 3, 4, 5, and 6 list the results for the self-ensemble. Self-ensemble results are marked as 'DIDN+' in

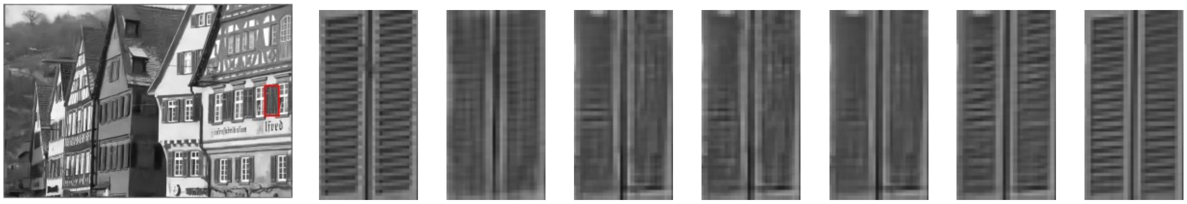


Image ‘8’ in Kodak24

Ground truth (PSNR (dB)/SSIM) | BM3D [5] (23.90 / 0.7292) | DnCNN [8] (24.26 / 0.7447) | IRCNN [13] (24.38 / 0.7412) | FFDNet [10] (24.53 / 0.7476) | MWCNN [14] (25.15 / 0.7815) | DIDN+ (25.41 / 0.7904)

Figure 5: Grayscale image denoising results at noise level 50.

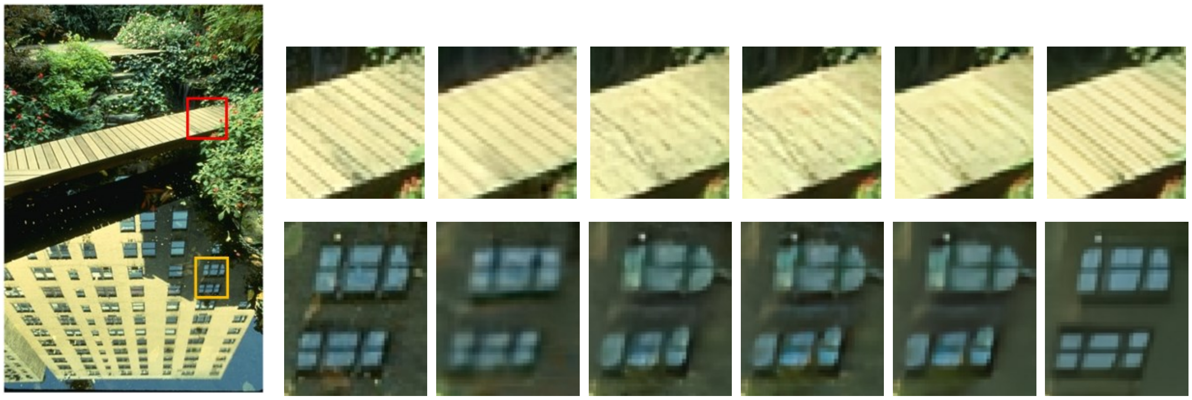


Image ‘148026’ in CBSD68

Ground truth (PSNR (dB) / SSIM) | CBM3D [36] (24.40 / 0.8003) | DnCNN [8] (25.14 / 0.8298) | IRCNN [13] (25.04 / 0.8271) | FFDNet [14] (25.07 / 0.8302) | DIDN+ (25.45 / 0.8435)

Figure 6: Color image denoising results at noise level 50.

the tables. Compared to DIDN, self-ensemble shows a significant improvement in PSNR without an increase in either the number of model parameters or the training time.

Model ensemble is a method of training several models for the same task and averaging their output values. If the model ensemble is additionally applied to the self-ensemble models, the number of parameters, the training time, and the testing time are all doubled. However, in our experiment, there is no significant increase in PSNR and when the performance difference between the models is large, the performance of the model ensemble is sometimes worse than that of the best model among them.

Our DIDN adopts and combines the snapshot ensemble and self-ensemble strategies.

4.4. Comparisons with the state-of-the-art

The proposed network is compared with DnCNN [8], IRCNN [13], FFDNet [10], MWCNN [14], and RDN [11]. All these methods except RDN are tested using their publicly accessible code. The results of RDN are drawn from the cited paper.

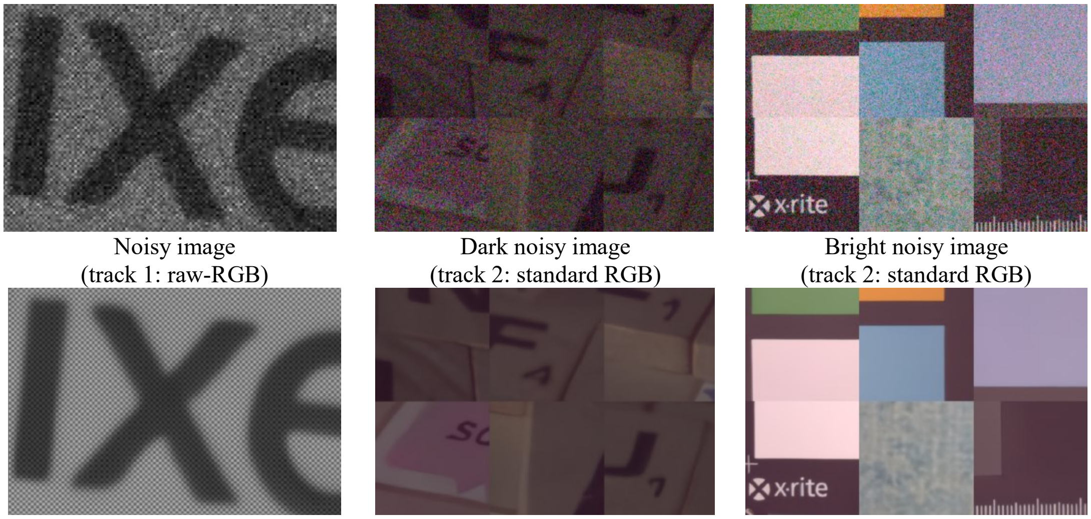
Table 3 gives the peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) in grayscale image denoising. The Kodak24 and BSD68 datasets are used in the experiments, and grayscale images are generated using the 'rgb2gray' function in MATLAB. The best result for each noise level is given in red, and the second-best in blue. In most cases our DIDN shows the best objective scores. Although all methods except FFDNet are trained for a single-level noise and our DIDN is trained for multiple noise levels, DIDN outperforms single-level noise models. Table 4 gives the performances in color image denoising. Similar to its grayscale results, the proposed DIDN successfully learns to process a wide range of noise levels in color image denoising. Tables 5 and 6 compare DIDN with the latest high-performance models in image denoising. DIDN shows higher PSNR values than MWCNN except at a noise level of 50, and improved performance at all noise levels compared to RDN. Note that MWCNN and RDN are trained for a single-level noise, and MWCNN uses more than 5,000 training images while DIDN is a multi-level noise model and trained using only 900 training images.

Methods	DnCNN [8]	RDN [11]	DIDN
Parameters	558K	22M	165M
MACs	2G	90G	70G

Table 7: Parameter number and MAC comparison on color image denoising. MACs are calculated for 64×64 image.

Figures 5 and 6 show the denoised images of the DIDN and conventional methods applied to grayscale and color images. Compared to conventional methods, the proposed network reduces noise and preserves the detail information of the image, resulting in visually pleasant images.

Table 7 compares the number of parameters and multiply-accumulate operations (MACs) with DnCNN [8]



Noisy image
(track 1: raw-RGB)

Dark noisy image
(track 2: standard RGB)

Bright noisy image
(track 2: standard RGB)

DIDN+
(50.52 dB / 0.9956)

DIDN+
(34.38 dB / 0.9857)

DIDN+
(42.95 dB / 0.9988)

Figure 7: Our results at NTIRE 2019 real image denoising challenge.

Rank	Track 1: Raw RGB				Track 2: Standard RGB			
	Team	Model	PSNR	SSIM	Team	Model	PSNR	SSIM
1	1st	1st	52.114	0.9969	1st	1st	39.932	0.9736
2	**Eraser**	**DIDN**	**52.107**	**0.9969**	Eraser	DHDN	39.883	0.9731
3	Eraser	DHDN	52.092	0.9968	**Eraser**	**DIDN**	**39.818**	**0.9730**
4	4th	4th	51.947	0.9967	4th	4th	39.675	0.9726
5	5th	5th	51.939	0.9967	5th	5th	39.611	0.9726

Table 8: NTIRE 2019 real image denoising challenge results for two tracks [40].

and RDN [11]. DIDN has more parameters, but has fewer MACs than RDN. Owing to the hierarchical structure of DIDN, it can have more trainable parameters than models with plain CNN structure while maintaining the computational complexity and memory usage low.

4.5. NTIRE 2019 Real Image Denoising Challenge

This work was originally developed for participation in the NTIRE 2019 real image denoising challenge [40]. This challenge incorporates two tracks: track 1 for projects focusing on removing noise from the raw-RGB images that are not demosaiced and that have specific color patterns, and track 2 for projects focusing on removing noise from standard RGB images.

The purpose of the challenge is to remove real world noise from images. In the SIDD data set [33], 320 image pairs (noisy and clean images) with resolutions of 4 or 5K are used for training. Because the dataset includes images with various noise levels, dynamic ranges, and brightnesses, the model is designed to have enough parameters to estimate and process a variety of cases.

We trained a DIDN with 10 DUBs in track 1 and 8 DUBs in track 2, and our results were ranked second and third in the challenge, respectively. This ranking proves that DIDN provides superior performance in handling real world noise. Table 8 compares the challenge results, and Figure 7 shows the resulting images from the validation set. DIDN successfully removes noise in images with various brightnesses and noise compositions using a single model. Moreover, in raw-RGB denoising, DIDN not only reduces noise, but also restores the original color patterns well.

5. Conclusion

In this paper, we have proposed a deep learning algorithm for single-image denoising. We modified the U-Net to be suitable for image denoising and used this as a base module. By sequencing the modules, we constructed a network structure that repeatedly down- and up-samples deep feature maps. This memory-efficient structure yields a large receptive field and enables the model to include a sufficiently large number of parameters to achieve improved performance.

To address multi-level and real world noise, we successfully developed a single model with the capacity to process Gaussian noise on levels ranging from 5 to 50. Furthermore, we presented a weight initialization method and applied ensemble techniques to efficiently train a multi-level noise model to enhance denoising performance. Experimental results demonstrate that our multi-level noise model surpasses the performance of existing single-level noise models and multi-level noise models in objective and subjective evaluation, and does so without requiring noise information inputs. Our proposed network has already demonstrated its superiority in real-world denoising tasks, achieving second and third place in tracks 1 and 2, respectively, at the NTIRE 2019 real image denoising challenge.

References

[1] A. Levin and N. Nadler. Natural image denoising: Optimality and inherent bounds. In *CVPR 2011*.

[2] M. Elad and M. Aharo. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 15(12):3736–3745, 2006.

[3] K. Dabov, A. Foi, and K. Egiazarian. Video denoising by sparse 3D transform-domain collaborative filtering. In *ESPC 2007*.

[4] M. K. Mihcak, I. Kozintsev, K. Ramchandran, and P. Moulin. Low-complexity image denoising based on statistical modeling of wavelet coefficients. *IEEE Signal Processing Letters*, 6(12):300–303, 1999.

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

[6] A. Buades, B. Coll, and J. M. Morel. A non-local algorithm for image denoising. In *CVPR 2005*.

[7] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with BM3D?. In *CVPR 2012*.

[8] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7): 3142–3155, 2017.

[9] Y. Tai, J. Yang, X. Liu, and C. Xu. Memnet: A persistent memory network for image restoration. In *ICCV 2017*.

[10] K. Zhang, W. Zuo, and L. Zhang. FFDNet: Toward a fast and flexible solution for CNN-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018.

[11] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu. Residual dense network for image restoration. *arXiv preprint arXiv:1812.10477*, 2018

[12] S. Lefkimmiatis. Universal denoising networks: a novel CNN architecture for image denoising. In *CVPR 2018*.

[13] K. Zhang, W. Zuo, S. Gu, and L. Zhang. Learning deep CNN denoiser prior for image restoration. In *CVPR 2017*.

[14] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo. Multi-level wavelet-CNN for image restoration. In *CVPRW 2018*.

[15] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, and D. Rueckert. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR 2016*.

[16] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

[17] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, K. Enhanced deep residual networks for single image super-resolution. In *CVPRW 2017*.

[18] V. Jain and S. Seung. Natural image denoising with convolutional networks. In *NIPS 2009*.

[19] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *NIPS 2012*.

[20] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

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

[22] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR 2017*.

[23] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell. Understanding convolution for semantic segmentation. In *WACV 2018*.

[24] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI 2015*.

[25] M. Haris, G. Shakhnarovich, and N. Ukita. Deep back-projection networks for super-resolution. In *CVPR 2018*.

[26] R. Timofte, R. Rothe, and L. Van Gool. Seven ways to improve example-based single image super resolution. In *CVPR 2016*.

[27] T. Garipov, P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *NIPS 2018*.

[28] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[29] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, L. Zhang, et al. NTIRE 2017 challenge on single image super-resolution: Methods and results. In *CVPRW 2017*.

[30] 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*.

[31] K. Dabov, A. Foi, V. Katkovnik, and K. O. Egiazarian. Color image denoising via sparse 3D collaborative filtering with grouping constraint in luminance-chrominance space. In *ICIP 2007*.

[32] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, 2017.

[33] A. Abdelhamed, S. Lin, and M. S. Brown. A high-quality denoising dataset for smartphone cameras. In *CVPR 2018*.

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

[35] S. Ioffe, S and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[36] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Color image denoising via sparse 3d collaborative filtering with grouping constraint in luminance-chrominance space. In *ICIP 2007*.

[37] R. Franzen. Kodak lossless true color image suite. source: http://r0k.us/graphics/kodak.

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

[39] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv*:1412.6980, 2014.

[40] A. Abdelhamed, R. Timofte, M. S. Brown, et al. NTIRE 2019 challenge on real image denoising: Methods and results. In *CVPRW 2019*.