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].

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 $\sigma$ 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 $\sigma$ 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,
some studies have been conducted into the use of down- and up-scaling of the feature maps. U-Net \cite{ronneberger2015u} 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 \(2 \times 2\) convolution are used for up-scaling the features. Liu et al. \cite{liu2015wavelet} 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. \cite{shi2016real} 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 \cite{zhang2018image} 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.

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 \cite{ronneberger2015u} 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-scaling. The efficiency of U-Net’s U-shaped structure is verified in image denoising \cite{liu2015wavelet} and ascribed to its large receptive field, high GPU memory efficiency, and low computational cost. DBPN \cite{zhang2018image} 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 \cite{ronneberger2015u} and \cite{zhang2018image}, 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

Figure 2: The architecture of the proposed DIDN.
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 \( H \times W \), DIDN first extracts \( N \) features using a \( 3 \times 3 \) 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 \( 3 \times 3 \) 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 \( 1 \times 1 \) 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 \( 1 \times 1 \) 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.

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

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

<table>
<thead>
<tr>
<th>Methods</th>
<th>Noise level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td>Single_RI</td>
<td>35.16</td>
</tr>
<tr>
<td>Single_PI</td>
<td><strong>35.18</strong></td>
</tr>
<tr>
<td>Multi_RI</td>
<td>35.14</td>
</tr>
<tr>
<td>Multi_PI_(\sigma,50)</td>
<td>35.14</td>
</tr>
<tr>
<td>Multi_PI_(\sigma,10)</td>
<td>35.15</td>
</tr>
</tbody>
</table>
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.

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

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Methods</th>
<th>Noise level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>BSD68</td>
<td>Model 1</td>
<td>33.970</td>
</tr>
<tr>
<td></td>
<td>Model 2</td>
<td>33.976</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>33.978</td>
</tr>
<tr>
<td></td>
<td>Model1 + Model2</td>
<td>35.152</td>
</tr>
<tr>
<td>Kodak24</td>
<td>Model 2</td>
<td>35.153</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>35.163</td>
</tr>
</tbody>
</table>

Table 2: Effect of weight averaging on grayscale image denoising. Best performance at each noise level is bolded.
are required to train a single model using the GeForce GTX 1080Ti. Drawing on [32], we use Ω-loss to train the model as follows:

\[ l_\Omega(\theta) = \frac{1}{N} \sum_{i=1}^{N} |F(x_i; \theta) - y_i|, \]

where \( N \) is a batch size, \( F(\cdot) \) is the network function with learnable parameter \( \theta \), 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 the weight values at the end of each cycle. The cosine period periodically changing the learning rate and then averaging lists the effects of using snapshot-ensemble on the image denoising.

<table>
<thead>
<tr>
<th>Methods</th>
<th>BSD68</th>
<th>Kodak24</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Noise level</td>
<td>Noise level</td>
</tr>
<tr>
<td>Noisy</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
</tr>
</tbody>
</table>

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

Table 3: PSNR (dB) / SSIM comparison of methods on grayscale 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.

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.

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

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]...
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