Comparison of DnCNN and DIDN for Natural Image Denoising

Kumari Pooja

I. Introduction

Image Denoising is a crucial and challenging low-level vision task, which turns out to be an indispensable step in several practical applications like in medical imaging. Generally there are different kinds of noise that can affect an image, based on the source of the noise, but typically the noise caused by poor illumination or high temperature problems occurring during the image acquisition step can be assumed to follow a simple Additive White Gaussian noise model. Thus, this additive white Gaussian noise (AWGN) can be modeled as $y = x + n$, where y is the noisy image, x is the clean image and n is the AWGN.

So the goal of the Image Denoising problem is to retrieve the clean image from the noisy image. There has been a lot of recent work using Deep Neural Network in improving the performance of natural image denoising, but since the convergence of the performance of such models, making any further major advancements has become particularly challenging.

The two architectures discussed in this report, DIDN and DnCNN have made significant improvements to the Image Denoising performance, its computational and memory efficiency, and in the later sections we will also see a comparative study of the two architectures and discuss the pros and cons involved with both the model.

II. Architectures

In the past decade of Image Denoising, several models have been proposed like nonlocal self-similarity (NSS) models, sparse models, gradient models and Markov random field (MRF) models. In particular, the NSS models are popular in state-of-the-art methods such as BM3D, LSSC, NCSR and WNNM.

Despite of their high denoising quality, most of the acknowledged methods suffer from two major drawbacks. Firstly, they involve a complex optimization problem in their testing stage which makes the denoising procedure time consuming, thus sacrificing computational efficiency for higher performance. Secondly, the models in general are non-convex in nature and involves several manually chosen parameters which largely hinders boosting of the denoising performance.

Here in this section we will discuss two architectures namely the DIDN model[1] and the DnCNN model[2] which has significant advantages over the existing models, especially the DIDN model which is even more superior to the DnCNN model. We have also provided a nice comparative study of the two models in the Section III

A. DnCNN

DnCNN is a modified VGG[3] architecture designed specifically for the purpose of image denoising by stripping all the pooling layers and adding batch normalization which largely alleviates the internal co variate shift by incorporating a normalization step and a scale and shift step before the non linearity in each layer.
The DnCNN model has two main features: the residual learning formulation which predicts the noise from noisy images, and batch normalization is incorporated to speed up training as well as boost the denoising performance. By incorporating convolution with ReLU, DnCNN can gradually separate image structure from the noisy observation through the hidden layers.

Now for a given DnCNN with depth D, there are three types of layers:

i $\mathit{Conv} + \mathit{ReLU}$: for the first layer, 64 filters of size $3 \times 3 \times c$ are used to generate 64 feature maps, and rectified linear units (ReLU, max(0,)) are then utilized for non linearity. Here c represents the number of image channels, i.e., $c = 1$ for gray image and $c = 3$ for color image.

ii $\mathit{Conv} + \mathit{BN} + \mathit{ReLU}$: for layers $2 \rightarrow (D - 1)$, 64 filters of size $3 \times 3 \times 64$ are used, and batch normalization is added between convolution and ReLU.

iii Conv : for the final layer, c filters of size $3 \times 3 \times 64$ are used to reconstruct the residual output.

B. DIDN

DIDN stands out from rest of the other architectures including the DnCNN model in the fact that to output the clean image from the noisy image, generic architectures have to extract deep features having the same resolution as the output which results in inefficient consumption of GPU and training time.
DIDN surpasses this issue by proposing a 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. Furthermore, the 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.
The DIDN architecture is a modification of the U-Net architecture[4] which was proposed initially for semantic segmentation. The iterative down-up scaling of feature maps which was used in the U-Net architecture is also adopted in the DIDN architecture and this turns out to be extremely effective in learning an image super resolution task. Hence, DIDN offers a large receptive field and an efficient GPU memory usage by sequential repetition of the contraction and expansion processes.
In the DIDN architecture, a *subpixel*[5] 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.
DIDN consists of four parts: feature extraction, down-up block (DUB), reconstruction, and enhancement.

i ***Initial feature extraction*** : For size of input image $H \times W$, firstly DIDN 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 of stride 2.

ii ***DUB***: Features are extracted using iterative downup scaling through several DUBs. 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.

iii ***Reconstruction***: The reconstruction block consists of nine convolution layers (*Conv*) followed by parametric rectified linear units (*PReLU*) []. More specifically, there are four consecutive residual blocks consisting of $Conv + PReLU + Conv + PReLU$ with additional *Conv* at the end.

iv ***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.

C. Training Configuration

While training the DIDN and DnCNN models, we followed the standard configurations recommended in the reference papers [1] & [2].
The Additive Gaussian Noise level was chosen to have random sigma σ in the range $[5, 50]$ to train the model, For training, we have used BSD500 dataset[6] of colored image $(c = 3)$ and the CBSD68 dataset[6] for testing purposes. Data augmentation process is followed same for both the architecture having random rotation and flip. Other than these common configurations, we used the following model specific configurations in our comparative study.

1) DnCNN: For training the DnCNN model we used batch size of 32 for mini-batch training the DnCNN model, with an epoch of 12 (due to time limit) with Adam optimiser (lr = 10^{-4}) and learning decay rate of 0.1. We trained the model on patches of size $50 \times 50 \times 3$ with 25% overlap and used the L2 loss function defined as follows:

$$L2(\Theta) = \frac{1}{2N} \sum_{i=1}^{N} \parallel \mathcal{R}(y_i; \Theta) - (y_i - x_i) \parallel_F^2$$

where, $\mathcal{R}(y_i; \Theta)$ is the residual mapping $\mathcal{R}(y; \Theta) = y - x$, i.e, the noise itself.

2) DIDN: For training the DIDN model, we used batch size of 16 for mini-batch learning, with an epoch of 12 with Adam optimiser (lr = 10^{-4}) and learning rate decay rate of 0.5. We trained the model on patches of size $64 \times 64 \times 3$ with no overlap and we used the L1 loss function defined as follows:

$$L1(\Theta) = \frac{1}{N} \sum_{i=1}^{N} \mid \mathcal{F}(x_i; \Theta) - y_i \mid$$

where, $\mathcal{F}(x_i; \Theta)$ is the predicted clean image.

III. Results and Discussions

In this section, we have discussed the objective and subjective comparison of DnCNN and DIDN model.

Generally, comparison comes with several common parameters and employ one or two changes. However, in deep learning architectures, keeping several parameters standardized bring best out of that model. So, we first start with standard architecture configuration explained in the papers for our comparison with some common configuration. Following are the contexts which are chosen to do comparison of these two models keeping the computing power and time limitations.

- Naturally, noise present in the image are not known that's why we have selected the noise level randomly from $\sigma = [5, 50]$ for generating noisy images.
- As discussed in [II-C], data preparation has been done only by changing the patch size and overlap between two patches to get better result according to the papers but we followed the same augmentation procedure for both architectures and trained on BSD500 [data reference]dataset.
- Training dataset is prepared with augmentation but validation batches are constructed only using original patches and corresponding generated noisy patches for fair learning.
- Patches are normalized between $[0-1]$ as it generally speeds up the learning and faster convergence and keep all values on common scale.
- Tuned parameters are kept same as mentioned in the corresponding papers in order to get better result and not experimented more due to time limit.
- Due to limited computing power, standard library is used for data augmentation which is efficient in terms of CPU memory usage. Moreover, batch size set to be lower value due to the same reason.

A. Results

As we can ses in Fig. 1 & Fig. 2, DIDN performs better than DnCNN for denoising even they are trained on same set random noise level. However, DIDN also smoothened the edges present of the images. Moreover, DnCNN could not denoise fully over the patch. In Table I, we have calculated the average PSNR and SSIM over the whole set of CBSD68 image dataset. The patches have no overlap while testing for the both network architectures. As compared to DnCNN, DIDN has shown better denoising capabilities.

B. Advantages and Disadvantages of DIDN and DnCNN

Following are the advantages and disadvantages discovered while experimenting both the architectures.

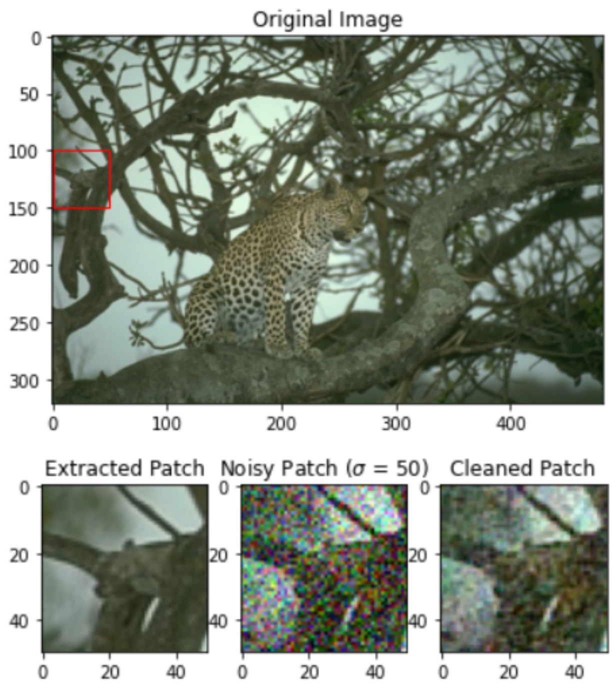

Fig. 1. DnCNN prediction for noise level 50 (PSNR (dB) / SSIM: 21.64 / 0.6490)

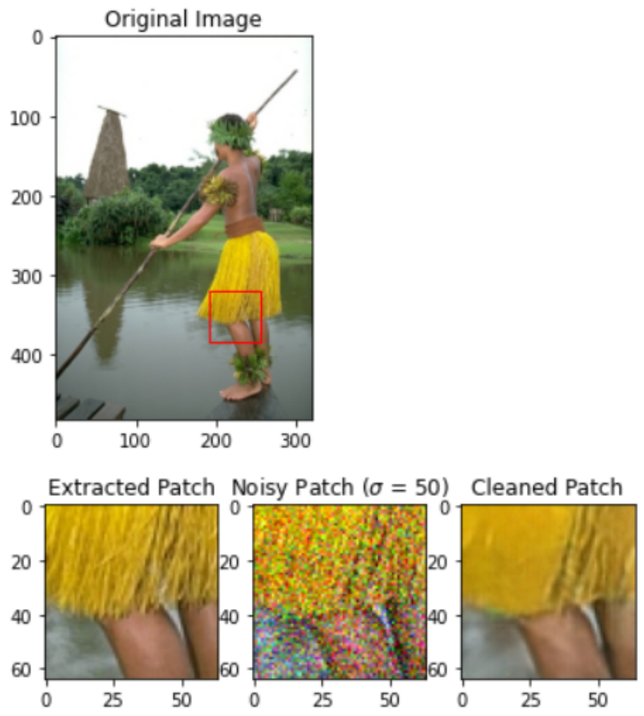

Fig. 2. DIDN prediction for noise level 50 (PSNR (dB) / SSIM: 25.27 / 0.7642)

Methods	CBSD68				
	Noise Level				
	$\sigma = 10$	$\sigma = 20$	$\sigma = 30$	$\sigma = 40$	$\sigma = 50$
DnCNN	32.77 / 0.9464	30.27 / 0.9221	26.65 / 0.8249	23.76 / 0.7003	21.37 / 0.5808
DIDN	35.92 / 0.9737	32.50 / 0.9434	30.52 / 0.9139	29.12 / 0.8864	28.01 / 0.8601

TABLE I
PSNR (dB) / SSIM comparison of DnCNN and DIDN

- In terms of efficient use of GPU resources, DIDN is better than DnCNN. Moreover, DIDN keeps a balance between CPU and GPU usage while training and in contrast DnCNN has more CPU memory usage.
- DIDN takes more training time than DnCNN (1 hr for 12 epochs for DnCNN and 4 hours for DIDN)and due to complex architecture, DIDN trained weights is huge (1 GB in our case). However, DnCNN just save weights with 8 MB results easy handling.
- DIDN denoises better than DnCNN.
- DnCNN takes 30ms to predict 1 patch of size $50 \times 50 \times 3$ from CBSD68 while DIDN takes 80ms to predict 1 patch of size $64 \times 64 \times 3$).

C. *Difficulties Encountered*

Due to time bound, making both model runnable with sufficient modularity and reasonable unit testing took major time than experimenting on different sets of parameters for the comparison. Due to computing power availability, all parameters of both architectures are not followed and made it onto manageable form. DnCNN is not trained properly within time (*i.e.* validation results are not stable with the increasing epochs) due to multiple changes and debugging.

IV. Future Prospects

Due to time limit and computing power constraints, several possibility of comparison could not be explored. It would be interesting to exam how the networks behave on same patch size and strides for both the architectures with overlap and no overlap. As DIDN is advanced complex architecture, we can also test the DIDN prediction by training on one specific noise level and test on different noise level rather than training on mix set of noise level.

References

[1] Songhyun Yu, Bumjun Park, and Jechang Jeong. Deep iterative down-up CNN for image denoising. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops 2019-June (2019), pp. 2095-2103. issn: 21607516. doi: 10.1109/CVPRW.2019.00262.

[2] Kai Zhang et al. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising". In: IEEE Transactions on Image Processing 26.7 (2017), pp. 3142-3155. ISSN: 10577149. doi:10.1109/TIP.2017.2662206.

[3] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, in Proc. Int. Conf. Learn. Represent., 2015, pp. 114.

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

[5] W. Shi, J. Caballero, F. Huszar, 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.

[6] Pablo Arbelaez et al. Contour Detection and Hierarchical Image Segmentation. In: IEEE Trans. Pattern Anal. Mach. Intell. 33 (2011), pp. 898-916.