Supplementary Material of "Dual Adversarial Network: Toward Real Noise Removal and Noise Generation"

Zongsheng Yue[1,2], Qian Zhao[1], Lei Zhang[2,3], and Deyu Meng[1,4,✉]

[1] Xi'an Jiaotong University, Shaanxi, China
{zsyzam,timmy.zhaoqian}@gmail.com, dymeng@mail.xjtu.edu.cn
[2] Hong Kong Polytechnic University, Hong Kong, China
cslzhang@comp.polyu.edu.hk
[3] DAMO Academy, Alibaba Group, Shenzhen, China
[4] The Macau University of Science and Technology, Macau, China

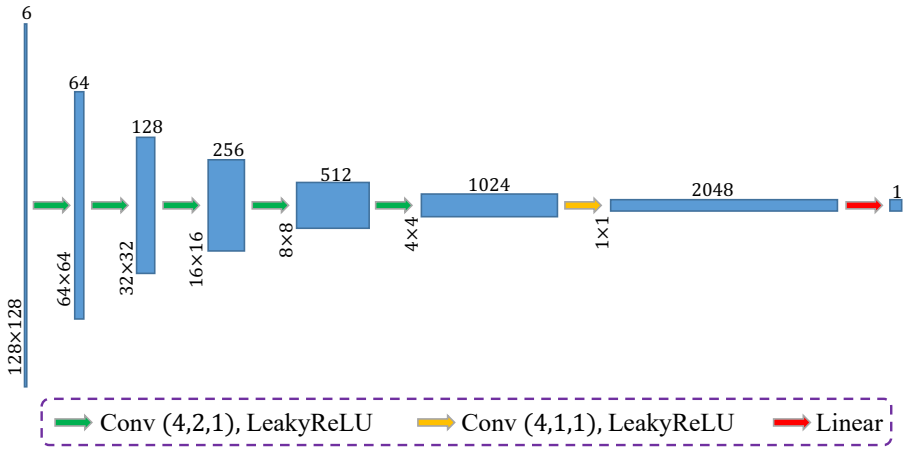

Fig. 1. Illustration of the discriminator in our method. The numbers along with each feature map represent its spatial size and depth. "Conv (k,s,p)" denotes the convolution operation with kernel size $k \times k$, stride s and padding p.

1 Network Architecture

As introduced in the maintext, our proposed method is composed of three parts, including a denoiser (R), a generator (G) and a discriminator (D). In this section, we will describe the detailed architecture for all of them.

1.1 Discriminator

The discriminator, aiming at distinguishing the real example and fake example and guiding the generator to the right direction, plays an important role of

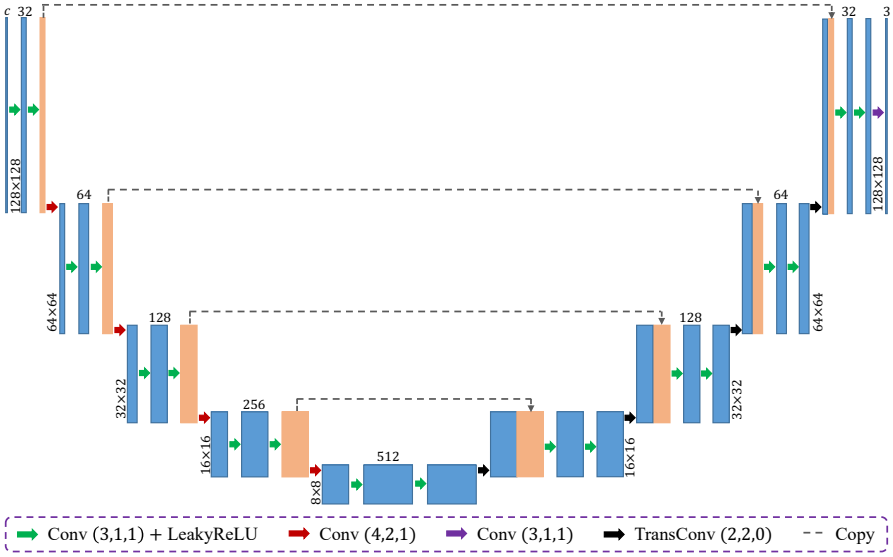

Fig. 2. The UNet backbone for the denoiser R and generator G in our method. The numbers along with each feature map represent its spatial size and depth. "Conv (k,s,p)" denotes the convolution operator with kernel size $k \times k$, stride s and padding p. Similarly, "TransConv (k,s,p)" denotes the transposed convolution operator with kernel size $k \times k$, stride s and padding p. For denoiser R, c=3, and $c = 4$ for generator G.

Gan [1]. In our proposed framework, we adopt the widely used discriminator architecture as in [2, 3], which includes five stride convolution layers to reduce the feature size and one fully connected layer to fuse the extracted features. Fig. 1 shows the detailed configuration. It inputs the concatenated image pairs $(\boldsymbol{x}, \boldsymbol{y})$ with size $128 \times 128 \times 6$ and outputs a scalar.

1.2 Denoiser and Generator

UNet Backbone: For both of R and G, we use UNet [4] architecture due to its fast speed and little GPU usage. It contains one down-path and one up-path and the skip connections between them. The down-path reduce the feature size step by step using stride convolution, while the up-path enlarge the feature size to the origianl image size gradually through transposed convolution. The concrete structure is displayed in Fig. 2. Inspired by [5], the redidual learning strategy is employed for both of R and G, i.e.,

$$R(\boldsymbol{y}) = \boldsymbol{y} - U(\boldsymbol{y}), \tag{1}$$

$$G(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{x} + U([\boldsymbol{x}, \boldsymbol{z}]), \ \boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I}), \tag{2}$$

where $U(\cdot)$ denotes the forword propagation of UNet, $[\cdot, \cdot]$ represents the concatenation operation.

Table 1. The PSNR and SSIM results of different methods on the Nam benchmark. The best results are highlighted in bold.

Metrics	Methods							
	CBM3D	WNNM	DnCNN	CBDNet	RIDNet	VDN	GDANet	GDANet$_+$
PSNR↑	35.36	35.33	35.68	39.20	39.33	38.66	**39.91**	39.79
SSIM↑	0.8708	0.8812	0.8811	0.9676	0.9623	0.9613	**0.9693**	0.9689

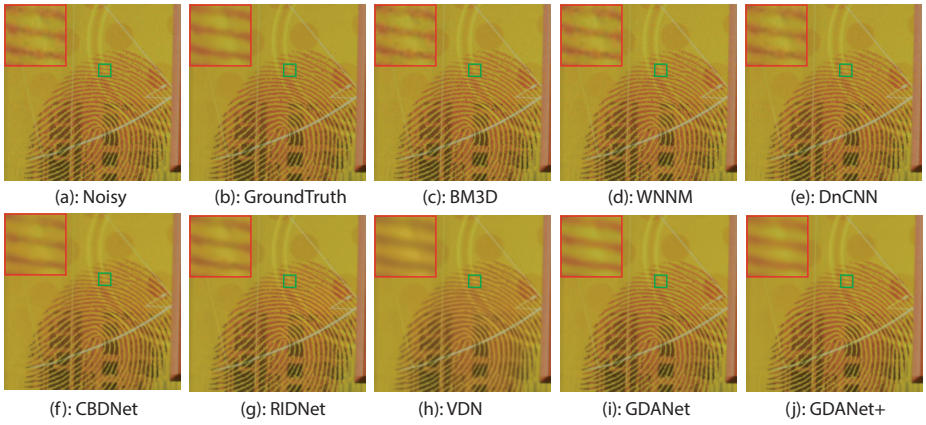

Fig. 3. One typical denoising example of Nam benchmark by different methods.

2 Additional Experimental Results

2.1 Results on Nam Benchmark

This benchmark contains 11 real static scenes and the corresponding noise-free images, which are obtained by averaging 500 noisy images of the same scenes. We cropped these images into 512×512 patches, and randomly selected 100 of them for the purpose of evaluation. The quantitative PSNR and SSIM results are given in Table 1. It is easy to see that our proposed GDANet performs better than the other compared methods. Note that VDN does not achieve good performance since the noisy images in this benchmark are JPEG compressed, which is not considered in VDN. For easy comparison, we also display one typical denoised example by different methods in Fig. 3, and the better visual performance of our methods can be observed.

Different from the results on DND benchmark (see the maintext), GDANet performs more stably than GDANet$_+$ as shown in Table 1, especially on SSIM metric. That's because the noise types simulated by the generator G, which are mainly determined by the training data set, does not match well with that contained in the testing set. Therefore, GDANet is suggested to be used in such general real-world denoising task with uncertain noise types, while DANet$_+$ is more suitable in the scenario that provides similar training and testing data sets.

Table 2. Running time (in seconds) of different methods for denoising images with size 256×256, 512×512 and 1024×1024.

Methods	Device	Image size		
		256	512	1024
CBM3D	CPU	3.115	12.964	53.601
DnCNN	GPU	0.011	0.038	0.142
VDN	GPU	0.011	0.030	0.115
DANet	GPU	0.007	0.014	0.055

Table 3. Running time (in seconds) of different methods for generating noisy images with size 256×256, 512×512 and 1024×1024.

Methods	Device	Image size		
		256	512	1024
CBDNet	CPU	0.059	0.249	1.017
ULRD	CPU	0.068	0.266	1.502
GRDN	GPU	0.149	0.617	2.313
DANet	GPU	0.008	0.017	0.069

2.2 Running Time

In this part, we compare the running time of different methods for both the noise remove and noise generation tasks. The evaluation was performed on a computer with 6-cores Inter(R) Core(TM) i7-8700K CPU @ 3.3GHz and an Nvidia GTX 1080Ti GPU.

Table 2 lists the running time of CBM3D, DnCNN, VDN and DANet for denoising task. Benifiting from the GPU computation, DnCNN, VDN and DANet are much faster than CBM3D. Even though both implemented on GPU, our proposed DANet is still about two times faster than DnCNN due to the simple UNet [4] architecture. As for the noise generation task, the running time of different methods are listed in Table 3. Firstly, DANet is much faster than all the other methods, expecially GRDN, which also uses deep neural network as generator. Secondly, CBDNet and ULRD both unfolded the in-camera processing pipelines, and had fewer computation burden than deep neural network. The lower speed of them is mainly limited by the computing power of CPU. No mattter considering denoising or generation task, DANet is very competitive for real applications.

References

1. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems 27. pp. 2672–2680 (2014)
2. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved training of wasserstein gans. In: NIPS'17 Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 5769–5779 (2017)
3. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. In: ICLR 2016 : International Conference on Learning Representations 2016 (2016)
4. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 234–241 (2015)
5. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. IEEE Transactions on Image Processing **26**(7), 3142–3155 (2017)