Supplementary Material for:
Masked Image Training for Generalizable Deep Image Denoising

Appendix

A. Details of the Test Noise

We evaluate the generalization performance of the models on six different synthetic noise types to evaluate the generalization performance on the noise out of the training set:
(1) **Speckle noise** is a kind of noise that can occur during the acquisition of medical images or tomography images. We use different variances σ^2 to obtain different levels of noise. The *imnoise* function in MATLAB is used for generating Speckle noise. We add multiplicative noise according to the equation $J = I + n * I$, where n is uniformly distributed random noise with mean 0 and variance σ^2, J is the noisy image.
(2) **Poisson noise** is a kind of signal-dependent noise that occurs during the acquisition of digital images. We amplified the noise using different scaling factor α using the equation $J = I + n * \alpha$, where we generate Poisson noise n first, then multiply it by a scaling factor α.
(3) **Spatially-correlated noise** indicates additive Gaussian noise filtered with an average kernel of size 3×3. Different levels indicate different standard deviations σ for the used Gaussian noise. This is to synthesize the complex artifact after denoising using a flawed algorithm.
(4) **Salt & pepper noise**. Different noise levels represent different noise densities, denoted by d. The *imnoise* function in MATLAB is used for generating Salt & pepper noise. This noise can appear during image acquisition as a result of camera imaging pipeline errors.
(5) **Image signal processing (ISP) noise**. Modern digital cameras aim to produce visually pleasing and accurate images that match human perception. The raw sensor data captured by the camera cannot directly produce a usable image, and several post-processing stages are required to convert its linear intensities into the final image [3]. As the original raw image contains noise, the post-processed image exhibits more complex noise. Since there are no adequate real noisy and noise-free image pairs, many denoising algorithms perform poorly on real data due to the gap between synthetic and real noise. In our experiments, we use the default parameter settings of [3] to synthesize ISP noise on RGB images.

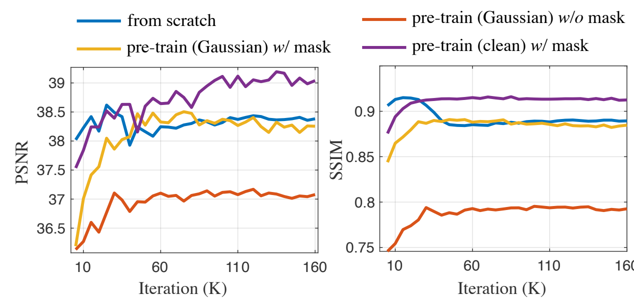

Figure 1. Training curve of different methods validated using our SIDD testset.

(6) **Mixture noise** is obtained by mixing the above different types of noise with different levels. We consider the real-world case where the image suffers from multiple degradations. The order of noise adding is Gaussian noise (variances σ_g^2), speckle noise (variances σ_{s1}^2), Poisson noise (scale α), Salt & pepper noise (density d), speckle noise (variances σ_{s2}^2). Since speckle noise is a multiplicative noise, it will have different effects when used in different positions. It will be multiplied by the noise already existing in the image to obtain complex noise degradation. There are 4 levels:

1. $\sigma_g^2 = 0.003$, $\sigma_{s1}^2 = 0.003$, $\alpha = 1$, $d = 0.002$, $\sigma_{s2}^2 = 0.003$;

2. $\sigma_g^2 = 0.004$, $\sigma_{s1}^2 = 0.004$, $\alpha = 1$, $d = 0.002$, $\sigma_{s2}^2 = 0.004$;

3. $\sigma_g^2 = 0.006$, $\sigma_{s1}^2 = 0.006$, $\alpha = 1$, $d = 0.003$, $\sigma_{s2}^2 = 0.006$;

4. $\sigma_g^2 = 0.008$, $\sigma_{s1}^2 = 0.008$, $\alpha = 1$, $d = 0.004$, $\sigma_{s2}^2 = 0.008$;

The noise patterns produced by these four settings are completely different from existing studies.

We also include two real noise types in this work: the Smartphone Image Denoising Dataset (SIDD) [1] and Monte Carlo (MC) rendered image noise [5].

ID	Pre-train	SIDD Fine-tune	Masked Traning	PSNR	SSIM	LPIPS
1	Gaus. 15			32.11	0.6606	0.5434
2	Gaus. 15		✓	33.01	0.6999	0.4626
3	None	✓		38.36	0.8879	0.3555
4	Gaus. 15	✓		37.08	0.7920	0.3622
5	Gaus. 15	✓	✓	38.15	0.8822	0.3237
6	Clean	✓	✓	**39.11**	**0.9135**	**0.2614**

Table 1. Masked pre-training for limited paired data. Our method of pre-training on clean images by masked training first and then fine-tuning on target limited dataset yields the best results.

B. Additional Comparisons

Methods for Comparison. We compare our method with several classical methods: DnCNN [14], RIDNet [2], RNAN [16], SwinIR [10], Restormer [13], Dropout [8]. Among them, Dropout [8] was proposed to improve the generalization ability and relieve the overfitting problem. Following [8], we apply the dropout layer with a dropout probability of 0.7 before the output convolutional layer of the baseline model.

Masked Training as Pre-training. In many real-world scenarios, we can only access very limited image pairs for training. It is not enough to adequately train a denoising network because the network can easily overfit the training data. The performance of the network will be limited if it is trained only on limited data. The pre-training and fine-tuning paradigm may be helpful in this case. One approach is to train the network on the synthetic data first and then fine-tune it on the target data [14], but the performance may also be unsatisfactory because of the gap between the pre-train data and the target data. In this paragraph, we will introduce a practical approach that uses the masked training method for pre-training. We first pre-train the model on clean images with the masked training strategy, and then fine-tune the model on the limited real training samples with the mask. This allows the model to obtain generalization ability even when trained on extremely limited training data. Pre-training on clean images enables the network to learn the content representation of natural images and thus benefits the fine-tuning of target noise. To conduct such experiments, we use images from the SIDD dataset [1]. SIDD contains real noisy images with high-quality clean references. Due to different lighting and different cameras, the noise of the image is also different. It is consistent with the complex noise situation in the real world. In order to simulate a scenario with extremely limited training samples, the training set only contains two 4K noisy – clean image pairs from SIDD. We also selected one image from each of the ten scenes, for a total of ten images as a test set. Table 1 shows the experiment settings and results. For experiment 3, we directly train the model on the limited training samples. For experiment 4 and 5, we first pre-train the models using Gaussian noise with $\sigma = 15$ and then fine-tune them on target noise. While for experiment 6, we pre-trained the model on clean (noise-free) images with the proposed masked training strategy, and then fine-tuned it on the target training samples. The model pre-trained on clean images using the proposed masked training achieves the best results. This demonstrates the potential of our approach as a new low-level pre-training method. In addition, our method pre-trained on noisy images is not as effective as pre-trained on clean images, which illustrates that our method benefits from learning information about the image's distribution. Visual results are shown in Figure 2. Our method preserves the most texture detail. Figure 1 shows the training curves for different experiments. The numerical performance of the model pre-trained on Gaussian noise and fine-tuned without masking (red line) is generally low and does not increase with training. For the model trained from scratch directly on SIDD (blue line), its PSNR starts to fluctuate at the beginning of training and does not improve any further. Its SSIM even drops with training. This indicates a severe overfitting problem. In contrast, the method using the proposed masked training (purple and yellow lines) can continue to improve the performance during the training process. This indicates that the model has not yet had an overfitting problem. The method pre-trained with clean images (purple line) performs better.

Quantitative Comparison. In Figure 4, we present the complete test curves including the LPIPS results on different noise types and levels. Our method demonstrates a slower performance degradation compared to other models, indicating a better generalization ability, especially when dealing with more severe noise types. We provide full numerical results in Table 2, Table 4, Table 3, and Table 5, where we evaluate our method on four benchmark datasets, namely CBSD68 [11], Kodak24 [6], McMaster [15], and Urban100 [7]. Our method outperforms other state-of-the-art models significantly across all noise types. Particularly, we obtain a significant lead in LPIPS performance, suggesting that our results have better human visual perceptual quality.

Additional Visual Results. Figure 5 shows more visual comparisons. The model's performance without masked training is significantly limited over the various noise types. Our model still effectively removes noise when dealing with a variety of noise outside the training set.

C. Additional Analyses of CKA

In the main text, in order to investigate how masked training differs from normal training strategy, we utilize

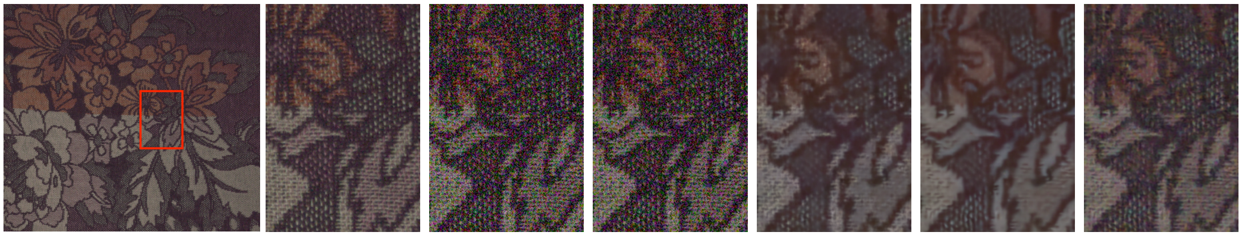

Figure 2. Visual comparison of different methods on real smartphone noise dataset SIDD [1]. "SwinIR" is trained on Gaussian noise, $\sigma = 15$. "from scratch" is trained directly on the target two SIDD training samples. "pre-train *w/o* mask" is pre-trained on Gaussian noise, $\sigma = 15$, and fine-tuned without mask. "pre-train *w/* mask" is pre-trained on clean images and fine-tuned by masked training.

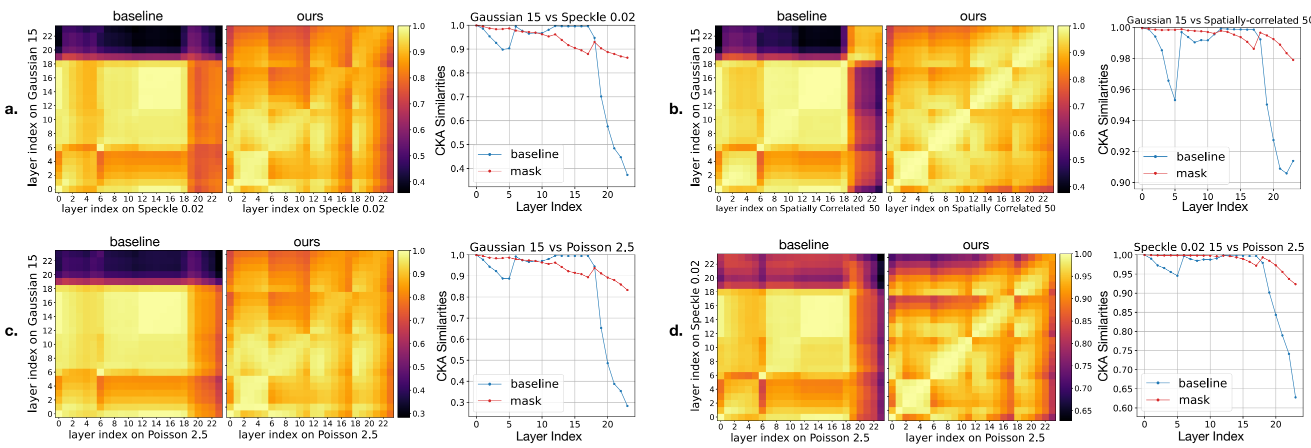

Figure 3. CKA similarity to analyze the representation similarity of network layers.

the centered kernel alignment (CKA) [4, 12] to analyze the differences between network representations obtained from those two training methods. In detail, we calculate the representations of two layers $\mathbf{X} \in \mathbb{R}^{m \times p_1}$ and $\mathbf{Y} \in \mathbb{R}^{m \times p_2}$ on the same m data points, with p_1 and p_2 neurons respectively. Gram matrices $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$ and $\mathbf{L} = \mathbf{Y}\mathbf{Y}^\top$ are used to compute CKA:

$$\mathrm{CKA}(\mathbf{K}, \mathbf{L}) = \frac{\mathrm{HSIC}(\mathbf{K}, \mathbf{L})}{\sqrt{\mathrm{HSIC}(\mathbf{K}, \mathbf{K})\mathrm{HSIC}(\mathbf{L}, \mathbf{L})}}$$

where HSIC is the Hilbert-Schmidt independence criterion [9]. Given the centering matrix $\mathbf{H} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$, and centered Gram matrices $\mathbf{K}' = \mathbf{HKH}$ and $\mathbf{L}' = \mathbf{HLH}$, we have $\mathrm{HSIC}(\mathbf{K}, \mathbf{L}) = \mathrm{vec}\left(\mathbf{K}'\right) \cdot \mathrm{vec}\left(\mathbf{L}'\right) / (m-1)^2$. More CKA results are shown in Figure 3. We first compare the correlation of the features between different noise types. For the baseline model, the correlation between the features of Gaussian noise and other different noises at the deep level is relatively low (a, b, c). Besides, the feature correlation between the noise outside the training set is also low (d). The model using the proposed masked training is able to have a high correlation in all cases. Figure 3 (a) shows the cross-model comparison between baseline and masked training models. We find that a significant difference between the two is that the features of the deeper layers of the baseline model have low correlations with all layers of our model. This indicates that these two training methods have inconsistent learning patterns for features, especially for the deeper layers. To explore how the model performs on different noise, Figure 3 (b) shows the cross-noise comparison between in-distribution noise and out-of-distribution noise (Gaussian and Poisson noise). For the baseline model, there is a low correlation between the different noise in the deep layers. It shows that the network processes these two types of noise differently for the deep layers. The other types of noise share a similar phenomenon. We suggest that this is because the baseline approach makes the deep layer of the model focus on overfitting the patterns of the training set, which leads to the poor generalization of the deep layers to handle different noise. In our model, the correlation between adjacent layers in our model is high. The proposed masked training forces the network to learn the distribution of the images themselves, which is similar to different types of noise. This allows our method to have a stronger generalization capability.

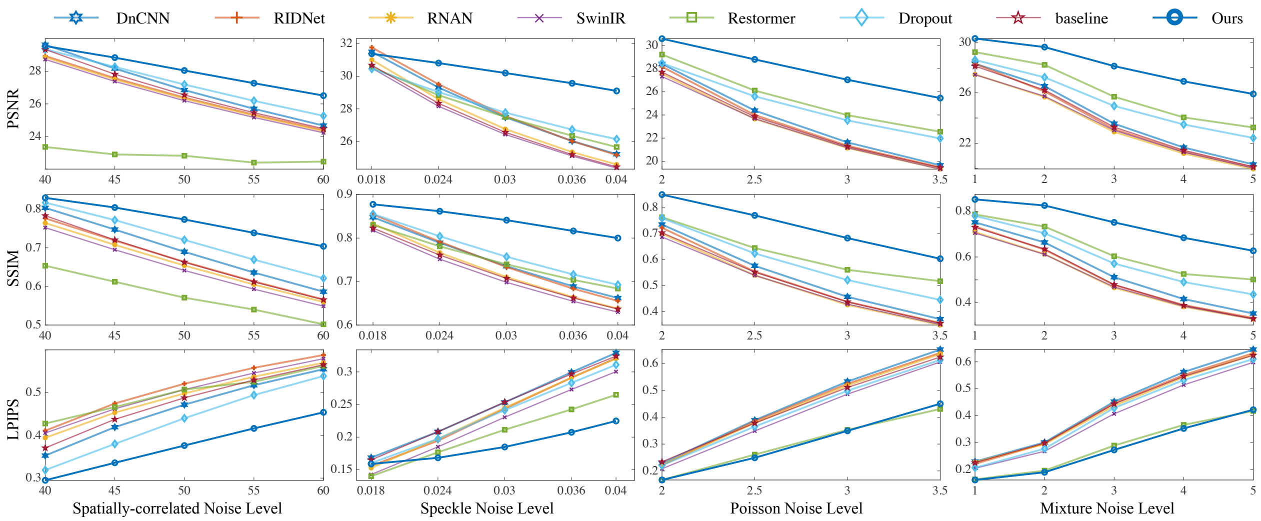

Figure 4. Performance comparisons on four noise types with different levels on the Kodak24 dataset [6]. All models are trained only on Gaussian noise. Our masked training approach demonstrates good generalization performance across different noise types.

References

[1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A high-quality denoising dataset for smartphone cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1692–1700, 2018.

[2] Saeed Anwar and Nick Barnes. Real image denoising with feature attention. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3155–3164, 2019.

[3] Tim Brooks, Ben Mildenhall, Tianfan Xue, Jiawen Chen, Dillon Sharlet, and Jonathan T Barron. Unprocessing images for learned raw denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11036–11045, 2019.

[4] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13:795–828, 2012.

[5] Arthur Firmino, Jeppe Revall Frisvad, and Henrik Wann Jensen. Progressive denoising of monte carlo rendered images. In *Computer Graphics Forum*, volume 41, pages 1–11. Wiley Online Library, 2022.

[6] Rich Franzen. Kodak lossless true color image suite. source: http://r0k.us/graphics/kodak/, 1999.

[7] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, 2015.

[8] Xiangtao Kong, Xina Liu, Jinjin Gu, Yu Qiao, and Chao Dong. Reflash dropout in image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6002–6012, 2022.

[9] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[10] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *CVPR*, 2021.

[11] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423. IEEE, 2001.

[12] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34:12116–12128, 2021.

[13] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5728–5739, 2022.

[14] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7):3142–3155, 2017.

[15] Lei Zhang, Xiaolin Wu, Antoni Buades, and Xin Li. Color demosaicking by local directional interpolation and nonlocal adaptive thresholding. *Journal of Electronic imaging*, 20(2):023016, 2011.

[16] Yulun Zhang, Kunpeng Li, Kai Li, Bineng Zhong, and Yun Fu. Residual non-local attention networks for image restoration. *arXiv preprint arXiv:1903.10082*, 2019.

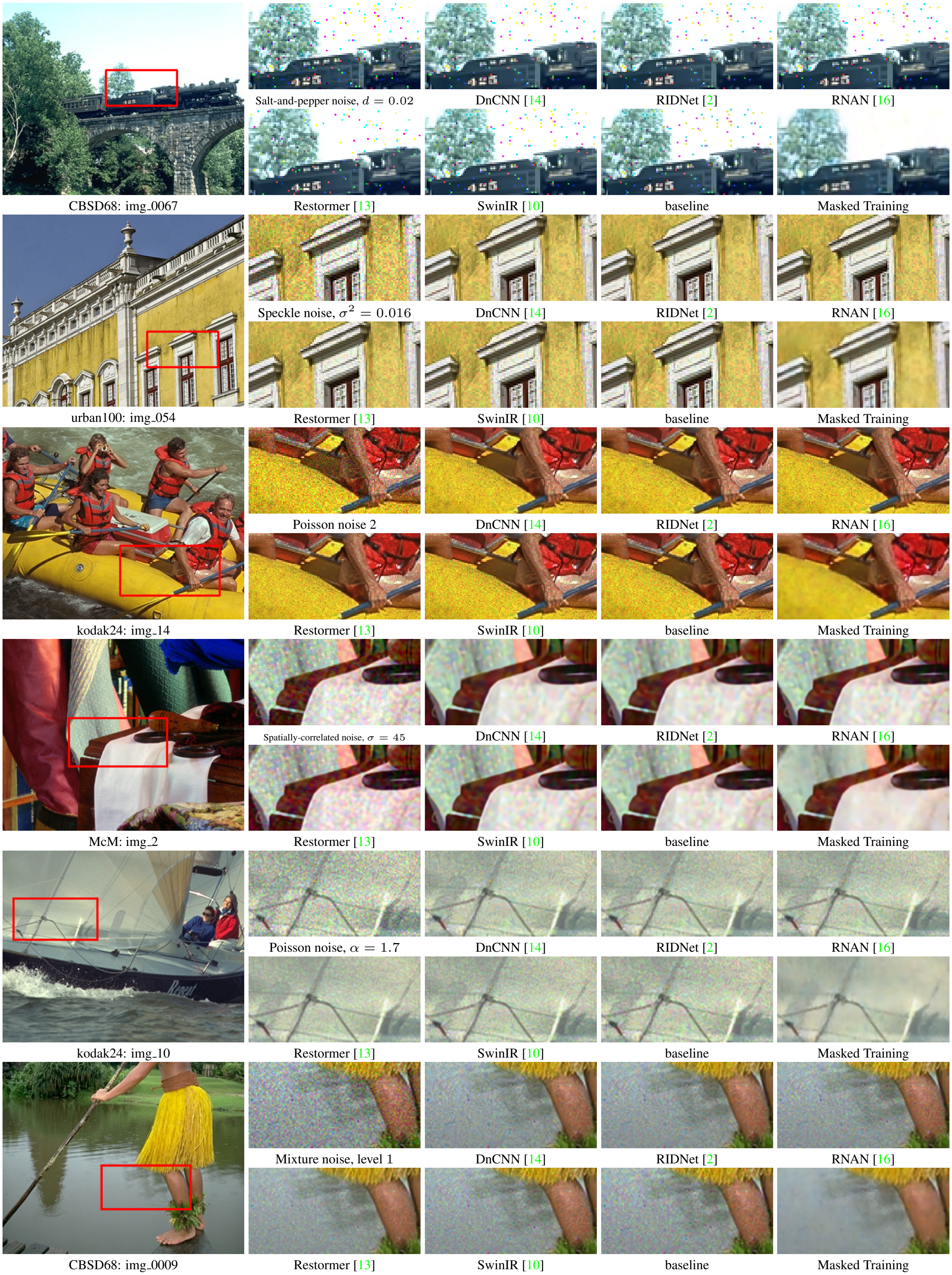

Figure 5. Visual comparison.

Speckle noise	$\sigma^2 = 0.02$			$\sigma^2 = 0.024$			$\sigma^2 = 0.03$			$\sigma^2 = 0.04$		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	30.74	0.8281	0.1806	29.31	0.7891	0.2082	27.49	0.7353	0.2533	25.22	0.6620	0.3292
RIDNet [2]	31.01	0.8337	0.1665	29.51	0.7916	0.1944	27.57	0.7331	0.2436	25.17	0.6554	0.3212
RNAN [16]	30.15	0.8101	0.1660	28.59	0.7662	0.1972	26.76	0.7101	0.2449	24.59	0.6377	0.3203
SwinIR [10]	29.64	0.7939	0.1555	28.16	0.7514	0.1851	26.43	0.6981	0.2305	24.37	0.6298	0.3004
Restormer [13]	29.95	0.8135	0.1521	28.84	0.7810	0.1767	27.50	0.7395	0.2113	25.66	0.6839	0.2649
Dropout [8]	29.97	0.8382	0.1709	29.03	0.8041	0.1974	27.77	0.7570	0.2413	26.14	0.6925	0.3110
baseline	29.84	0.8016	0.1778	28.34	0.7608	0.2082	26.56	0.7071	0.2536	24.44	0.6367	0.3242
Ours	**31.22**	**0.8739**	**0.1594**	**30.81**	**0.8617**	**0.1683**	**30.20**	**0.8412**	**0.1849**	**29.10**	**0.8000**	**0.2248**
Poisson noise	$\alpha = 2$			$\alpha = 2.5$			$\alpha = 3$			$\alpha = 3.5$		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	28.41	0.7359	0.2284	24.38	0.5767	0.3887	21.63	0.4571	0.5330	19.65	0.3711	0.6521
RIDNet [2]	28.17	0.7231	0.2215	24.00	0.5546	0.3849	21.34	0.4379	0.5246	19.48	0.3567	0.6397
RNAN [16]	27.55	0.7000	0.2231	23.66	0.5402	0.3783	21.14	0.4263	0.5184	19.33	0.3486	0.6355
SwinIR [10]	27.32	0.6877	0.2081	23.68	0.5398	0.3487	21.17	0.4294	0.4860	19.32	0.3506	0.6059
Restormer [13]	29.22	0.7639	0.1662	26.11	0.6452	0.2608	23.98	0.5613	0.3530	22.55	0.5174	0.4306
Dropout [8]	28.47	0.7601	0.2209	25.61	0.6245	0.3652	23.53	0.5218	0.4986	21.97	0.4454	0.6136
baseline	27.70	0.7040	0.2339	23.85	0.5524	0.3782	21.27	0.4377	0.5109	19.45	0.3550	0.6241
Ours	**30.59**	**0.8510**	**0.1662**	**28.80**	**0.7709**	**0.2488**	**27.04**	**0.6834**	**0.3493**	**25.46**	**0.6039**	**0.4502**
Spatially-correlated	$\sigma = 40$			$\sigma = 45$			$\sigma = 50$			$\sigma = 55$		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	29.63	0.8036	0.3527	28.17	0.7474	0.4192	26.85	0.6898	0.4718	25.70	0.6360	0.5173
RIDNet [2]	28.94	0.7766	0.4109	27.58	0.7189	0.4746	26.39	0.6637	0.5208	25.34	0.6131	0.5580
RNAN [16]	28.86	0.7644	0.3943	27.50	0.7078	0.4532	26.32	0.6542	0.4980	25.28	0.6050	0.5373
SwinIR [10]	28.73	0.7524	0.4056	27.38	0.6951	0.4620	26.20	0.6414	0.5070	25.17	0.5930	0.5458
Restormer [13]	23.42	0.6533	0.4412	23.06	0.6109	0.4783	22.82	0.5709	0.5072	22.59	0.5353	0.5356
Dropout [8]	29.35	0.8173	0.3188	28.27	0.7719	0.3800	27.19	0.7206	0.4400	26.19	0.6694	0.4943
baseline	29.34	0.7834	0.3706	27.82	0.7205	0.4375	26.55	0.6628	0.4878	25.46	0.6118	0.5295
Ours	**29.55**	**0.8296**	**0.2949**	**28.84**	**0.8045**	**0.3358**	**28.05**	**0.7735**	**0.3762**	**27.27**	**0.7388**	**0.4163**
Salt & pepper	$d = 0.002$			$d = 0.004$			$d = 0.008$			$d = 0.012$		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	24.75	0.6785	0.3639	21.15	0.4952	0.5626	17.55	0.2993	0.8196	15.47	0.2066	0.9779
RIDNet [2]	25.19	0.6769	0.3617	21.38	0.4934	0.5498	17.65	0.2969	0.8029	15.60	0.2066	0.9598
RNAN [16]	23.59	0.6416	0.3829	20.42	0.4639	0.5599	17.21	0.2850	0.8048	15.31	0.2006	0.9644
SwinIR [10]	23.42	0.6329	0.3873	20.21	0.4511	0.5710	17.00	0.2688	0.8103	15.14	0.1875	0.9614
Restormer [13]	23.81	0.6384	0.3919	20.99	0.4831	0.5551	19.79	0.3878	0.6512	19.25	0.3257	0.7574
Dropout [8]	27.44	0.7180	0.3041	24.36	0.5557	0.4898	21.01	0.3790	0.7415	19.03	0.2902	0.9047
baseline	25.36	0.6510	0.3694	21.93	0.4747	0.5642	18.42	0.2939	0.8153	16.46	0.2106	0.9656
Ours	**30.52**	**0.8477**	**0.1768**	**28.48**	**0.7681**	**0.2786**	**25.01**	**0.5958**	**0.5039**	**22.48**	**0.4622**	**0.6979**
Mixture noise	**level 1**			**level 2**			**level 3**			**level 4**		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	28.31	0.7514	0.2299	26.53	0.6636	0.3011	23.55	0.5117	0.4522	21.66	0.4162	0.5622
RIDNet [2]	28.13	0.7335	0.2215	26.11	0.6320	0.2971	23.13	0.4776	0.4461	21.34	0.3899	0.5514
RNAN [16]	27.46	0.7090	0.2280	25.67	0.6126	0.2948	22.90	0.4657	0.4369	21.19	0.3826	0.5431
SwinIR [10]	27.44	0.7049	0.2051	25.73	0.6113	0.2682	23.03	0.4689	0.4073	21.29	0.3847	0.5145
Restormer [13]	29.23	0.7859	0.1639	28.22	0.7330	0.1965	25.69	0.6034	0.2894	24.05	0.5257	0.3662
Dropout [8]	28.61	0.7797	0.2071	27.23	0.7039	0.2777	24.96	0.5715	0.4290	23.49	0.4906	0.5324
baseline	28.12	0.7295	0.2259	26.22	0.6346	0.2985	23.28	0.4795	0.4441	21.44	0.3885	0.5463
Ours	**30.31**	**0.8518**	**0.1617**	**29.63**	**0.8251**	**0.1903**	**28.12**	**0.7513**	**0.2732**	**26.91**	**0.6841**	**0.3530**

Table 2. Quantitative comparison on Kodak24 [6].

Speckle noise	$\sigma^2 = 0.02$			$\sigma^2 = 0.024$			$\sigma^2 = 0.03$			$\sigma^2 = 0.04$		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	30.67	0.8254	0.1506	29.24	0.7927	0.1840	27.54	0.7551	0.2269	25.49	0.7095	0.2856
RIDNet [2]	30.77	0.8261	0.1444	29.31	0.7934	0.1757	27.58	0.7551	0.2168	25.49	0.7081	0.2750
RNAN [16]	29.77	0.8066	0.1492	28.32	0.7745	0.1814	26.67	0.7377	0.2224	24.75	0.6932	0.2796
SwinIR [10]	29.17	0.7947	0.1258	27.83	0.7660	0.1524	26.30	0.7322	0.1893	24.46	0.6909	0.2412
Restormer [13]	28.89	0.8005	0.1300	27.95	0.7790	0.1515	26.81	0.7523	0.1807	25.30	0.7173	0.2213
Dropout [8]	28.64	0.8153	0.1416	27.85	0.7852	0.1688	26.89	0.7501	0.2032	25.64	0.7062	0.2525
baseline	28.86	0.7283	0.1353	27.61	0.7014	0.1593	26.15	0.6679	0.1938	24.38	0.6251	0.2437
Ours	**30.33**	**0.8157**	**0.1130**	**30.01**	**0.8016**	**0.1238**	**29.53**	**0.7800**	**0.1412**	**28.66**	**0.7463**	**0.1761**
Poisson noise	$\alpha = 2$			$\alpha = 2.5$			$\alpha = 3$			$\alpha = 3.5$		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	29.13	0.7771	0.1772	25.40	0.6740	0.2915	22.78	0.5910	0.3972	20.86	0.5261	0.4846
RIDNet [2]	29.00	0.7706	0.1681	25.17	0.6636	0.2838	22.59	0.5836	0.3877	20.76	0.5227	0.4730
RNAN [16]	28.13	0.7488	0.1760	24.58	0.6476	0.2897	22.18	0.5710	0.3916	20.44	0.5119	0.4765
SwinIR [10]	27.85	0.7419	0.1468	24.48	0.6459	0.2472	22.12	0.5710	0.3419	20.35	0.5122	0.4229
Restormer [13]	28.74	0.7765	0.1310	25.78	0.6936	0.2082	23.57	0.6296	0.2778	21.94	0.5792	0.3342
Dropout [8]	27.74	0.7699	0.1649	25.56	0.6751	0.2645	23.84	0.5986	0.3558	22.47	0.5377	0.4355
baseline	27.89	0.7024	0.1557	24.51	0.6025	0.2522	22.19	0.5361	0.3427	20.49	0.4761	0.4207
Ours	**30.01**	**0.8016**	**0.1120**	**28.67**	**0.7439**	**0.1683**	**27.23**	**0.6876**	**0.2329**	**25.99**	**0.6347**	**0.2976**
Spatially-correlated	$\sigma = 40$			$\sigma = 45$			$\sigma = 50$			$\sigma = 55$		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	29.92	0.8159	0.2221	28.59	0.7672	0.2718	27.35	0.7160	0.3197	26.23	0.6665	0.3654
RIDNet [2]	29.36	0.7958	0.2608	28.06	0.7433	0.3146	26.90	0.6910	0.3624	25.85	0.6426	0.4056
RNAN [16]	29.16	0.7792	0.2542	27.85	0.7257	0.3053	26.70	0.6751	0.3514	25.68	0.6286	0.3941
SwinIR [10]	29.10	0.7710	0.2498	27.77	0.7165	0.3005	26.61	0.6658	0.3446	25.59	0.6193	0.3876
Restormer [13]	24.46	0.6408	0.2867	23.90	0.6043	0.3217	23.48	0.5723	0.3542	23.18	0.5431	0.3874
Dropout [8]	28.15	0.7946	0.2123	27.32	0.7542	0.2562	26.47	0.7097	0.3021	25.65	0.6649	0.3493
baseline	29.43	0.7731	0.2365	28.05	0.7191	0.289	26.61	0.6532	0.3513	25.82	0.6223	0.3770
Ours	**28.96**	**0.7996**	**0.1952**	**28.36**	**0.7779**	**0.2216**	**27.65**	**0.7529**	**0.2507**	**27.01**	**0.7251**	**0.2827**
Salt & pepper	$d = 0.002$			$d = 0.004$			$d = 0.008$			$d = 0.012$		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	23.53	0.6675	0.3607	20.13	0.4878	0.5403	16.72	0.2966	0.7748	14.73	0.2057	0.9320
RIDNet [2]	24.01	0.6639	0.3581	20.48	0.4864	0.5288	16.93	0.2960	0.7584	14.92	0.2065	0.9131
RNAN [16]	22.62	0.6428	0.3731	19.54	0.4651	0.5374	16.43	0.2854	0.7626	14.59	0.2007	0.9193
SwinIR [10]	22.68	0.6391	0.3580	19.50	0.4581	0.5226	16.32	0.2749	0.7379	14.47	0.1914	0.8889
Restormer [13]	23.04	0.6398	0.3667	20.10	0.4829	0.5207	18.64	0.3555	0.6163	18.34	0.3156	0.6797
Dropout [8]	25.83	0.6771	0.3082	23.04	0.5197	0.4693	19.89	0.3536	0.6918	17.96	0.2709	0.8487
baseline	24.06	0.6224	0.3485	20.87	0.4630	0.5183	17.69	0.2959	0.7378	15.86	0.2156	0.8867
Ours	**29.51**	**0.7929**	**0.1504**	**27.45**	**0.7117**	**0.2476**	**24.03**	**0.5508**	**0.4350**	**21.59**	**0.4313**	**0.5968**
Mixture noise	**level 1**			**level 2**			**level 3**			**level 4**		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	28.41	0.7627	0.1869	26.88	0.6989	0.2406	24.16	0.5781	0.3564	22.33	0.4877	0.4447
RIDNet [2]	28.38	0.7509	0.1781	26.65	0.6811	0.2337	23.82	0.5558	0.3479	22.03	0.4659	0.4335
RNAN [16]	27.52	0.7285	0.1886	25.99	0.6616	0.2414	23.42	0.5412	0.3510	21.75	0.4533	0.4351
SwinIR [10]	27.57	0.7271	0.1601	26.07	0.6619	0.2050	23.56	0.5453	0.3059	21.86	0.4557	0.3869
Restormer [13]	28.59	0.7674	0.1410	27.53	0.7210	0.1703	25.29	0.6263	0.2462	23.71	0.5578	0.2991
Dropout [8]	27.47	0.7515	0.1694	26.41	0.6924	0.2190	24.58	0.5856	0.3255	23.27	0.5086	0.4079
baseline	28.05	0.7472	0.1665	26.40	0.6810	0.2148	23.70	0.5418	0.3229	21.91	0.4397	0.4061
Ours	**29.91**	**0.8267**	**0.1094**	**29.44**	**0.8111**	**0.1312**	**28.24**	**0.7570**	**0.1870**	**27.15**	**0.7018**	**0.2452**

Table 3. Quantitative comparison on McMaster [15].

Speckle noise	$\sigma^2 = 0.02$			$\sigma^2 = 0.024$			$\sigma^2 = 0.03$			$\sigma^2 = 0.04$		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	29.90	0.8380	0.1699	28.57	0.8044	0.1982	26.90	0.7610	0.2374	24.84	0.7035	0.2996
RIDNet [2]	30.11	0.8404	0.1597	28.75	0.8044	0.1884	27.03	0.7590	0.2305	24.87	0.6999	0.2927
RNAN [16]	29.36	0.8228	0.1593	27.95	0.7883	0.1872	26.28	0.7451	0.2276	24.28	0.6870	0.2893
SwinIR [10]	28.89	0.8101	0.1602	27.55	0.7774	0.1867	25.98	0.7362	0.2251	24.07	0.6810	0.2849
Restormer [13]	29.16	0.8279	0.1518	28.13	0.8015	0.1742	26.84	0.7667	0.2049	25.17	0.7202	0.2523
Dropout [8]	29.13	0.8447	0.1684	28.28	0.8171	0.1953	27.16	0.7804	0.2347	25.69	0.7311	0.2936
baseline	29.11	0.8122	0.1794	27.75	0.7801	0.2077	26.15	0.7393	0.2465	24.19	0.6837	0.3050
Ours	**30.46**	**0.8777**	**0.1435**	**30.08**	**0.8697**	**0.1511**	**29.49**	**0.8502**	**0.1691**	**28.53**	**0.8169**	**0.2060**
Poisson noise	$\alpha = 2$			$\alpha = 2.5$			$\alpha = 3$			$\alpha = 3.5$		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	28.13	0.7790	0.1957	24.40	0.6417	0.3284	21.77	0.5295	0.4524	19.83	0.4446	0.5639
RIDNet [2]	28.00	0.7705	0.1878	24.08	0.6199	0.3237	21.50	0.5082	0.4459	19.67	0.4279	0.5542
RNAN [16]	27.38	0.7505	0.1902	23.73	0.6081	0.3201	21.29	0.5003	0.4405	19.51	0.4220	0.5498
SwinIR [10]	27.12	0.7392	0.1849	23.69	0.6049	0.3094	21.27	0.4992	0.4282	19.46	0.4200	0.5393
Restormer [13]	28.68	0.7973	0.1506	25.67	0.6951	0.2361	23.54	0.6167	0.3139	22.25	0.5598	0.3831
Dropout [8]	28.03	0.7953	0.1975	25.42	0.6823	0.3220	23.45	0.5901	0.4366	21.94	0.5182	0.5418
baseline	27.55	0.7517	0.2085	23.92	0.6173	0.3346	21.42	0.5087	0.4510	19.63	0.4259	0.5572
Ours	**30.01**	**0.8656**	**0.1390**	**28.48**	**0.8053**	**0.2072**	**26.84**	**0.7318**	**0.2974**	**25.33**	**0.6616**	**0.3937**
Spatially-correlated	$\sigma = 40$			$\sigma = 45$			$\sigma = 50$			$\sigma = 55$		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	29.38	0.8304	0.2819	28.02	0.7839	0.3379	26.78	0.7349	0.3864	25.68	0.6880	0.4290
RIDNet [2]	28.74	0.8092	0.3306	27.45	0.7603	0.3865	26.32	0.7122	0.4300	25.31	0.6670	0.4672
RNAN [16]	28.68	0.7983	0.3192	27.39	0.7499	0.3703	26.25	0.7029	0.4122	25.25	0.6591	0.4500
SwinIR [10]	28.56	0.7883	0.3353	27.26	0.7389	0.3853	26.13	0.6918	0.4298	25.13	0.6484	0.4664
Restormer [13]	24.54	0.7076	0.3661	24.17	0.6689	0.4007	23.70	0.6320	0.4348	23.35	0.5978	0.4640
Dropout [8]	28.89	0.8383	0.2580	27.89	0.7999	0.3109	26.90	0.7563	0.3656	25.96	0.7123	0.4135
baseline	29.11	0.8109	0.3071	27.69	0.7578	0.3658	26.48	0.7078	0.4147	25.42	0.6625	0.4537
Ours	**29.08**	**0.8445**	**0.2431**	**28.43**	**0.8242**	**0.2765**	**27.71**	**0.7985**	**0.3127**	**27.03**	**0.7719**	**0.3476**
Salt & pepper	$d = 0.002$			$d = 0.004$			$d = 0.008$			$d = 0.012$		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	24.39	0.7102	0.3205	20.88	0.5423	0.5032	17.33	0.3499	0.7615	15.27	0.2510	0.9304
RIDNet [2]	24.83	0.7065	0.3165	21.12	0.5400	0.4912	17.44	0.3470	0.7459	15.41	0.2510	0.9096
RNAN [16]	23.32	0.6768	0.3312	20.19	0.5127	0.4970	16.99	0.3343	0.7464	15.12	0.2443	0.9133
SwinIR [10]	23.21	0.6724	0.3416	20.04	0.5035	0.5123	16.84	0.3206	0.7541	14.97	0.2320	0.9190
Restormer [13]	23.58	0.6779	0.3429	20.77	0.5292	0.5016	19.13	0.4143	0.6322	18.37	0.3500	0.7409
Dropout [8]	26.92	0.7433	0.2739	23.97	0.5999	0.4380	20.70	0.4330	0.6832	18.75	0.3431	0.8508
baseline	25.09	0.6879	0.3289	21.71	0.5261	0.5088	18.25	0.3480	0.7621	16.30	0.2594	0.9216
Ours	**29.96**	**0.8558**	**0.1512**	**28.01**	**0.7893**	**0.2295**	**24.69**	**0.6391**	**0.4408**	**22.23**	**0.5174**	**0.6331**
Mixture noise	**level 1**			**level 2**			**level 3**			**level 4**		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	27.91	0.7876	0.1955	26.28	0.7151	0.2561	23.52	0.5791	0.3825	21.70	0.4867	0.4833
RIDNet [2]	27.80	0.7740	0.1888	25.97	0.6885	0.2510	23.14	0.5463	0.3777	21.38	0.4589	0.4752
RNAN [16]	27.16	0.7543	0.1946	25.52	0.6718	0.2515	22.89	0.5366	0.3711	21.22	0.4532	0.4683
SwinIR [10]	27.10	0.7477	0.1827	25.51	0.6668	0.2378	22.96	0.5363	0.3563	21.29	0.4523	0.4533
Restormer [13]	28.54	0.8091	0.1493	27.50	0.7625	0.1796	25.17	0.6509	0.2599	23.52	0.5729	0.3270
Dropout [8]	28.01	0.8076	0.1841	26.78	0.7455	0.2455	24.70	0.6296	0.3722	23.29	0.5532	0.4672
baseline	27.81	0.7717	0.2022	26.06	0.6916	0.2659	23.27	0.5476	0.3927	21.48	0.4563	0.4886
Ours	**29.74**	**0.8672**	**0.1342**	**29.14**	**0.8466**	**0.1551**	**27.80**	**0.7900**	**0.2231**	**26.62**	**0.7305**	**0.2964**

Table 4. Quantitative comparison on CBSD68 [11].

Speckle noise	$\sigma^2 = 0.02$			$\sigma^2 = 0.024$			$\sigma^2 = 0.03$			$\sigma^2 = 0.04$		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	28.66	0.8207	0.1456	27.28	0.7880	0.1745	25.64	0.7478	0.2138	23.67	0.6962	0.2716
RIDNet [2]	28.73	0.8218	0.1386	27.31	0.7874	0.1683	25.63	0.7457	0.2086	23.63	0.6933	0.2662
RNAN [16]	27.99	0.8047	0.1414	26.60	0.7726	0.1697	25.01	0.7333	0.2085	23.14	0.6826	0.2652
SwinIR [10]	27.50	0.7931	0.1408	26.19	0.7626	0.1683	24.68	0.7256	0.2059	22.88	0.6772	0.2609
Restormer [13]	28.22	0.8100	0.1370	27.17	0.7851	0.1578	25.86	0.7529	0.1874	24.15	0.7106	0.2302
Dropout [8]	27.69	0.8258	0.1516	26.83	0.7981	0.1797	25.78	0.7639	0.2167	24.42	0.7200	0.2693
baseline	27.66	0.7916	0.1611	26.33	0.7617	0.1877	24.80	0.7242	0.2241	22.98	0.6753	0.2772
Ours	**28.97**	**0.8771**	**0.1062**	**28.60**	**0.8642**	**0.1180**	**28.04**	**0.8421**	**0.1421**	**27.12**	**0.8055**	**0.1832**

Poisson noise	$\alpha = 2$			$\alpha = 2.5$			$\alpha = 3$			$\alpha = 3.5$		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	27.72	0.7814	0.1656	24.06	0.6682	0.2738	21.52	0.5807	0.3740	19.65	0.5128	0.4638
RIDNet [2]	27.51	0.7728	0.1600	23.75	0.6536	0.2697	21.27	0.5675	0.3686	19.51	0.5025	0.4561
RNAN [16]	26.88	0.7550	0.1634	23.37	0.6428	0.2682	21.02	0.5593	0.3662	19.30	0.4953	0.4544
SwinIR [10]	26.59	0.7451	0.1586	23.27	0.6392	0.2575	20.95	0.5575	0.3533	19.21	0.4929	0.4426
Restormer [13]	28.39	0.7964	0.1326	25.34	0.7049	0.2043	22.89	0.6266	0.2802	21.25	0.5684	0.3524
Dropout [8]	27.19	0.7928	0.1722	24.82	0.6989	0.2706	22.98	0.6269	0.3607	21.55	0.5698	0.4437
baseline	26.94	0.7511	0.1790	23.45	0.6425	0.2788	21.09	0.5593	0.3712	19.40	0.4936	0.4556
Ours	**28.72**	**0.8710**	**0.1051**	**27.48**	**0.8142**	**0.1668**	**26.04**	**0.7446**	**0.2464**	**24.71**	**0.6845**	**0.3232**

Spatially-correlated	$\sigma = 40$			$\sigma = 45$			$\sigma = 50$			$\sigma = 55$		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	29.87	0.8526	0.1912	28.50	0.8110	0.2371	27.23	0.7677	0.2795	26.09	0.7258	0.3173
RIDNet [2]	29.24	0.8364	0.2216	27.89	0.7908	0.2702	26.68	0.7464	0.3116	25.62	0.7051	0.3464
RNAN [16]	29.07	0.8203	0.2248	27.72	0.7767	0.2674	26.54	0.7351	0.3052	25.50	0.6961	0.3385
SwinIR [10]	28.99	0.8116	0.2360	27.64	0.7678	0.2769	26.46	0.7265	0.3131	25.43	0.6882	0.3455
Restormer [13]	26.38	0.7360	0.2593	25.56	0.7011	0.2902	24.77	0.6686	0.3189	24.06	0.6384	0.3455
Dropout [8]	28.68	0.8529	0.1797	27.78	0.8191	0.2204	26.86	0.7808	0.2635	25.96	0.7411	0.3046
baseline	29.58	0.8440	0.2092	28.11	0.7950	0.2567	26.84	0.7492	0.2974	25.74	0.7076	0.3323
Ours	**28.06**	**0.8586**	**0.1720**	**27.55**	**0.8410**	**0.1976**	**26.98**	**0.8196**	**0.2266**	**26.40**	**0.7951**	**0.2562**

Salt & pepper	$d = 0.002$			$d = 0.004$			$d = 0.008$			$d = 0.012$		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	24.01	0.7372	0.2643	20.55	0.5828	0.4143	17.05	0.4029	0.6335	15.01	0.3062	0.7973
RIDNet [2]	24.56	0.7372	0.2613	20.88	0.5835	0.4062	17.20	0.4023	0.6220	15.16	0.3072	0.7824
RNAN [16]	23.01	0.7132	0.2744	19.87	0.5582	0.4137	16.71	0.3892	0.6223	14.86	0.2999	0.7840
SwinIR [10]	22.90	0.7075	0.2823	19.74	0.5507	0.4215	16.56	0.3790	0.6231	14.71	0.2910	0.7773
Restormer [13]	23.42	0.7145	0.2799	20.53	0.5772	0.4086	18.65	0.4571	0.5308	17.81	0.3967	0.6311
Dropout [8]	26.33	0.7591	0.2326	23.48	0.6279	0.3647	20.29	0.4781	0.5635	18.35	0.3943	0.7181
baseline	24.92	0.7224	0.2667	21.56	0.5752	0.4130	18.11	0.4103	0.6263	16.15	0.3225	0.7840
Ours	**28.58**	**0.8655**	**0.1158**	**26.93**	**0.8074**	**0.1850**	**24.01**	**0.6780**	**0.3530**	**21.75**	**0.5652**	**0.5140**

Mixture noise	**level 1**			**level 2**			**level 3**			**level 4**		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
DnCNN [14]	27.62	0.7842	0.1656	26.08	0.7221	0.2120	23.41	0.6112	0.3116	21.64	0.5332	0.3907
RIDNet [2]	27.51	0.7725	0.1592	25.75	0.7011	0.2076	23.01	0.5844	0.3080	21.31	0.5099	0.3851
RNAN [16]	26.85	0.7535	0.1651	25.28	0.6866	0.2092	22.75	0.5759	0.3046	21.13	0.5041	0.3813
SwinIR [10]	26.79	0.7475	0.1566	25.26	0.6816	0.1973	22.81	0.5751	0.2878	21.19	0.5040	0.3634
Restormer [13]	28.45	0.8085	0.1269	27.39	0.7665	0.1517	25.03	0.6716	0.2171	23.26	0.5984	0.2749
Dropout [8]	27.22	0.7976	0.1608	26.11	0.7431	0.2077	24.22	0.6484	0.3035	22.91	0.5849	0.3770
baseline	27.47	0.7795	0.1718	25.79	0.7136	0.2191	23.12	0.5931	0.3170	21.38	0.5131	0.3925
Ours	**28.57**	**0.8749**	**0.0995**	**28.08**	**0.8566**	**0.1186**	**26.97**	**0.8053**	**0.1747**	**25.97**	**0.7516**	**0.2337**

Table 5. Quantitative comparison on Urban100 [7].