

A APPENDIX

A.1 ABLATION STUDIES

To evaluate the effectiveness of each component of our method, we conducted the following ablation study on the Kodak24 dataset for synthetic noise, and on the PolyU dataset for real-world noise.

Effect of the test time adaptation. In this section, we design experiments to evaluate the performance enhancement of our proposed test-time adaptation method on a pre-trained Gaussian denoiser across different noise distributions. Additionally, our Noise2Noise-based approach, which does not require pre-training, can also be applied as a single-image denoising method, for which we conduct separate experiments. We utilize a straightforward CNN architecture, comprising five layers with 64 channels of 3×3 convolutions, each followed by a leaky ReLU activation layer. In the final layer, we use a 1×1 convolution. We use ℓ_2 loss and Adam Optimizer to train the network for 3,000 iterations. The learning rate is initialized to 0.001 and will decay by a factor of 2 when it reaches 1500, 2000, and 2500 iterations. Table 1 presents the performance comparisons among the pre-trained Gaussian denoiser, the single-image denoising method without pre-training, and the pre-trained Gaussian denoiser with test-time adaptation across various noise distributions. These three methods are denoted as Pre-trained Model, Without Pre-train, and Pre-train + Adaptation, respectively. To further benchmark the effectiveness of our proposed single-image denoising method without pre-training, we also include results from other data-free deep learning-based methods in the table. By comparing rows 3, 4, 5, and 7, it is evident that our proposed single-image denoising method without pre-training generally outperforms existing single-image denoising approaches in most cases. Furthermore, a comparison between rows 6 and 8 demonstrates that fine-tuning the pre-trained Gaussian denoiser with our proposed test-time adaptation framework significantly boosts denoising performance for both in-distribution and out-of-distribution noise, showcasing the robustness of our method for constructing Noise2Noise training data. Lastly, rows 7 and 8 further confirm that the deep denoising priors embedded in the pre-trained Gaussian denoiser are beneficial for single-image denoising tasks. Overall, our proposed method, which combines pre-training with test-time adaptation, not only achieves the best overall denoising performance but also runs significantly faster than existing single-image denoising methods. It only takes 1 second (0.48 seconds of which is spent on building the pixel bank), while the second-fastest method, ZS-N2N, requires 9 seconds.

Table 1: Average PSNR scores for Gaussian and Poisson denoising on Kodak24, and real-world denoising on PolyU. The best and second results are in **bold** and underlined.

Method	Gaussian			Poisson			Real-world	Time (s)
	$\sigma=10$	$\sigma=25$	$\sigma=50$	$\lambda=50$	$\lambda=25$	$\lambda=10$		
DIP	32.28	27.38	23.95	27.51	25.84	23.81	34.75	59
S2S	29.54	28.39	26.22	28.89	28.31	27.29	35.97	1839
ZS-N2N	<u>33.69</u>	29.07	24.81	29.45	27.49	24.92	35.17	9
Pre-trained model	32.66	<u>29.70</u>	20.94	29.62	24.82	19.94	36.10	-
Without pre-train	33.41	29.65	<u>26.63</u>	<u>29.77</u>	28.15	25.59	36.12	19
Pre-train+Adaptation	34.63	30.75	27.11	30.51	28.89	<u>26.64</u>	36.71	1

Effect of window size W . To explore the impact of window size on the proposed method, we report the algorithm’s performance under different values of $M \in \{24, 32, 40, 48\}$. Table 2 shows the effect of window size on denoising performance. As illustrated in the table, whether for synthetic or real-world noise, the effect of different window sizes on the algorithm’s performance is minimal as long as the window size is sufficiently large. To balance performance and computational complexity, we choose $W = 32$.

Table 2: Denoising PSNR of ablation studies about window size W .

Window Size	24	32	40	48
PSNR ($\sigma = 25$)	30.81	30.84	30.84	30.84
PSNR ($\sigma = 50$)	27.00	27.05	27.12	27.05
PSNR ($\lambda = 25$)	28.68	28.71	28.71	28.73
PSNR (real world)	36.70	36.71	36.71	36.71

Effect of patch size k . To evaluate the impact of different patch sizes on the proposed method, we report the algorithm’s performance under different values of $k \in 5, 7, 9$. Table 3 shows the performance of the algorithm for each value of k . From the table, it can be observed that the performance significantly improves when increasing k from 5 to 7, while further increasing k results in marginal performance gains. This is because smaller image blocks may lead to poor block similarity due to noise interference, whereas larger image blocks make it harder to find potentially similar blocks. Considering the trade-off between performance and computational complexity ($k = 7$, 0.48s vs. $k = 9$, 0.88s), we set $k = 7$ for all cases in this paper.

Table 3: Denoising PSNR of ablation studies about patch size k .

Patch Size	5	7	9
PSNR ($\sigma = 25$)	30.46	30.75	30.78
PSNR ($\sigma = 50$)	26.13	27.11	27.18
PSNR ($\lambda = 25$)	28.32	28.89	28.93
PSNR (real world)	36.55	36.71	36.74

Effect of the number of non-local patches M . To evaluate the impact of different non-local patches on the proposed method, we report the algorithm’s performance under different values of $M \in 12, 16, 20$. Table 4 illustrates the correlation between network performance and M . It can be observed that the algorithm performs best when $M = 16$. Therefore, we choose $M = 16$ in this paper.

Table 4: Denoising PSNR of ablation studies about non-local patches M .

Non-local Patch	12	16	20
PSNR ($\sigma = 25$)	30.74	30.75	30.73
PSNR ($\sigma = 50$)	26.08	27.11	27.07
PSNR ($\lambda = 25$)	28.86	28.89	28.88
PSNR (real world)	36.69	36.71	36.71

Effect of the number of similar pixels p . Table 5 illustrates the correlation between network performance and the number of similar pixels, p . It can be observed that as p increases, the denoising performance of the network initially improves but then declines. This phenomenon can be attributed to the fact that increasing p enhances the network’s learning and generalization capabilities by expanding the training sample pool. On the other hand, excessively high p values reduce the similarity between pixels, leading to a decrease in the similarity of clean content in pseudo-samples used for fine-tuning, which negatively impacts network performance. Based on these observations, we select $p = 20$ in our study.

Table 5: Denoising PSNR of ablation studies about similar pixels p .

Similar Pixel	2	4	6	8	10	15	20	25	30
PSNR ($\sigma = 25$)	28.21	29.74	30.26	30.43	30.56	30.63	30.75	30.75	30.65
PSNR ($\sigma = 50$)	22.26	24.78	25.84	26.35	26.69	27.05	27.11	27.11	26.82
PSNR ($\lambda = 25$)	25.30	27.34	28.19	28.49	28.76	28.86	28.89	28.87	28.55
PSNR (real world)	36.78	36.79	36.79	36.79	36.78	36.75	36.71	36.67	36.62

Effect of Adaptation Iterations for Each Image. In this section, we investigate the impact of different adaptive iterations on each image. As shown in Table 6, we compare the TTA performance of our method using different iteration counts for each image. The test time includes the time to construct the pixel bank. When the number of iterations is small, the TTA Denoising model is unable to effectively learn how to remove noise from the images. Conversely, when the iteration count is too high, the performance improvements of TTA Denoising diminish. Additionally, for lower noise levels, the TTA Denoising model converges more quickly, while for higher noise levels (e.g., $\sigma = 50, \lambda = 10$), the network requires more iterations to achieve better performance. Therefore, for higher noise levels (e.g., $\sigma = 50, \lambda = 10$), we set the iteration count to 100, whereas for other cases, we set it to 10.

Table 6: Average PSNR scores for Gaussian and Poisson denoising on Kodak24 and real-world noise on PloyU. The best results are in **bold**.

Iterations	Gaussian			Poisson			Real-world	Time (s)
	$\sigma=10$	$\sigma=25$	$\sigma=50$	$\lambda=50$	$\lambda=25$	$\lambda=10$		
0	32.66	29.70	20.94	29.62	24.82	19.94	36.10	0.00
1	33.41	30.50	22.80	30.42	26.51	22.01	36.29	0.50
2	34.21	30.86	24.59	30.81	27.62	23.85	36.39	0.51
5	34.78	30.82	25.71	30.75	28.56	25.21	36.50	0.55
10	34.75	30.75	26.39	30.51	28.89	25.82	36.71	0.62
20	34.47	30.43	26.65	30.39	28.85	26.10	36.88	0.74
50	34.13	30.35	26.94	30.35	28.98	26.49	36.90	1.12
100	33.82	30.27	27.11	30.26	28.99	26.64	36.88	1.79

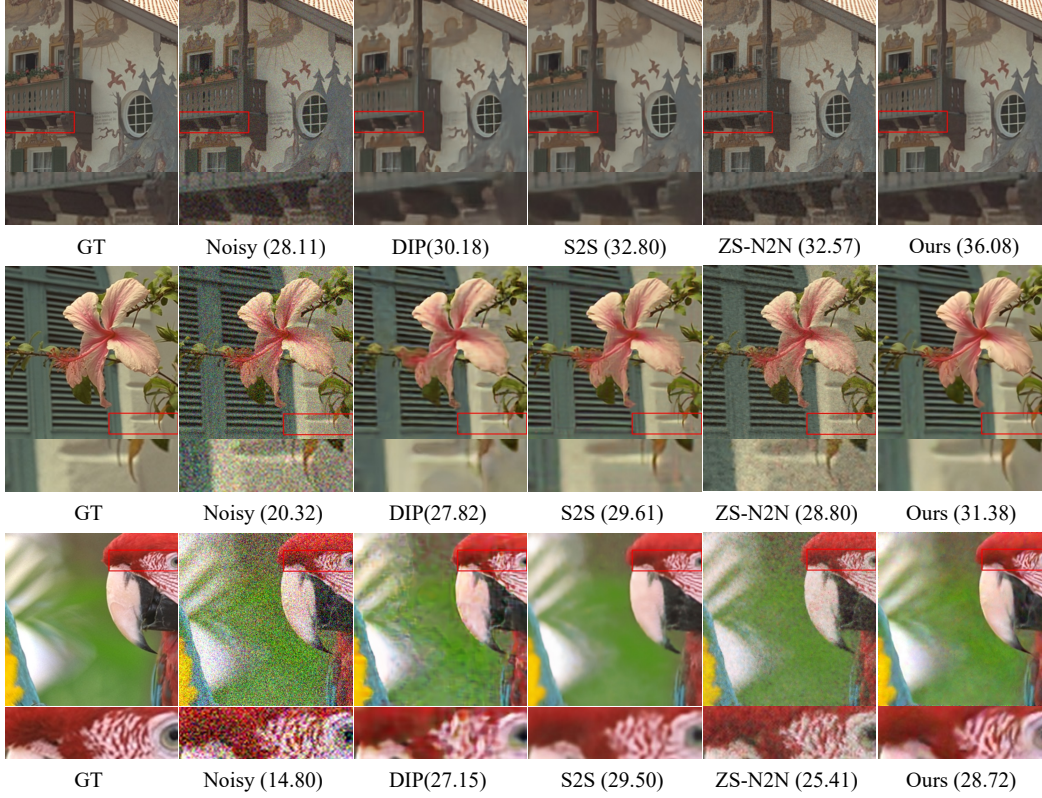


Figure 1: Gaussian denoising on Kodak24 images. The numbers in parentheses are PSNR scores (dB). Upper row: $\sigma = 10$, middle row: $\sigma = 25$, lower row; $\sigma = 50$.

A.2 MORE VISUALIZATION RESULTS

In this part, we show more visualization comparison results of different denoising methods on test images of the Kodak24, MacMaster18, and KVASIR20 datasets. Figures 1, 4, 2, 3, 5, and 6 show the denoising performance of different comparison methods. From these figures, it can be seen that while the S2S method achieves higher PSNR scores in some denoised images, it often produces overly smooth images. On the other hand, ZS-N2N performs poorly at higher noise levels, with the denoised images still containing a significant amount of noise. Visually, the images denoised by our method appear more pleasant.

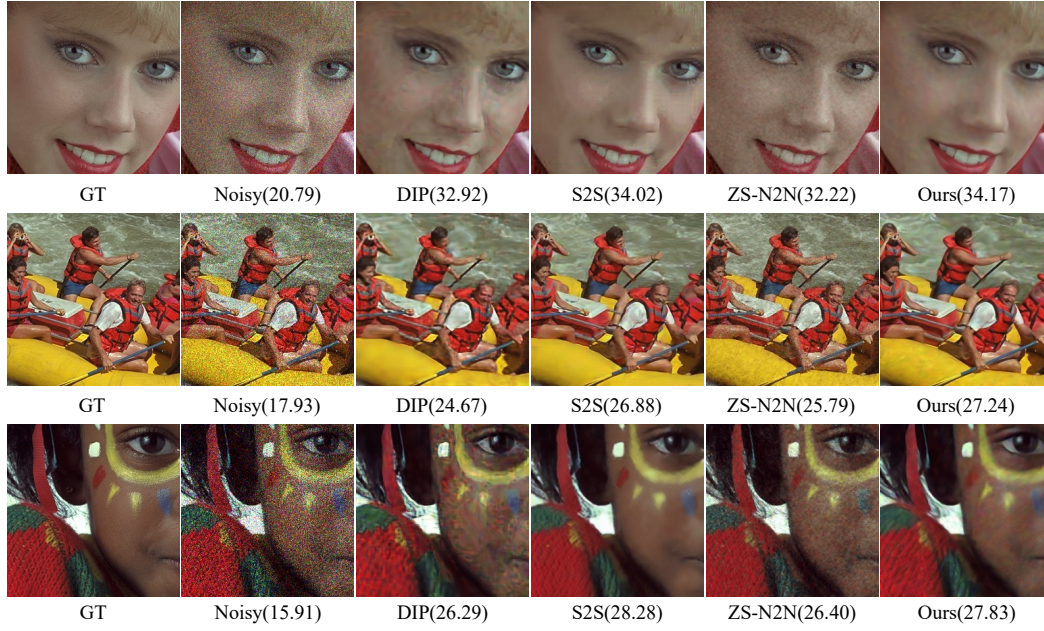


Figure 2: Poisson denoising on Kodak24 images. The numbers in parentheses are PSNR scores (dB). Upper row: $\lambda = 50$, middle row: $\lambda = 25$, lower row; $\lambda = 10$.

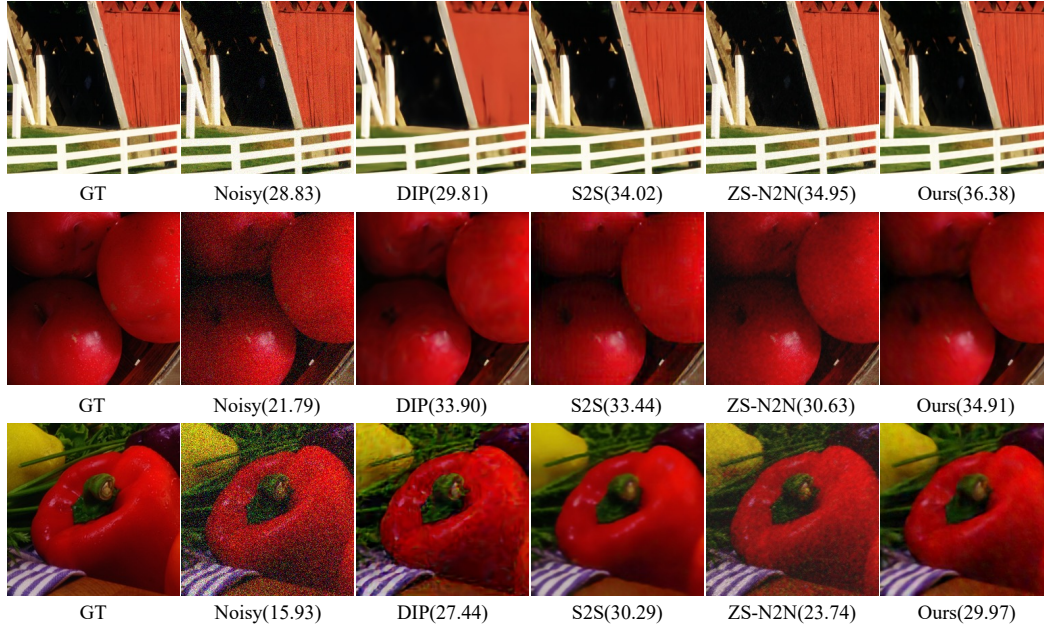


Figure 3: Gaussian denoising on MacMaster18 images. The numbers in parentheses are PSNR scores (dB). Upper row: $\sigma = 10$, middle row: $\sigma = 25$, lower row; $\sigma = 50$.

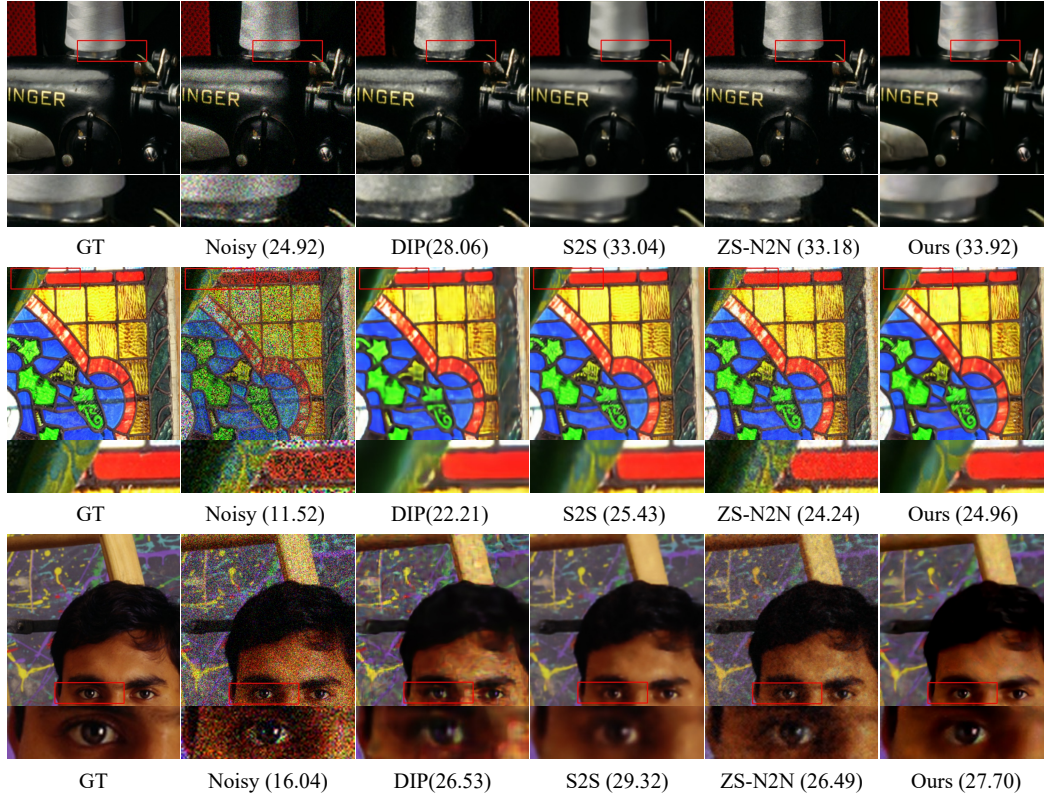


Figure 4: Poisson denoising on McMaster18 images. The numbers in parentheses are PSNR scores (dB). Upper row: $\lambda = 50$, middle row: $\lambda = 25$, lower row; $\lambda = 10$.

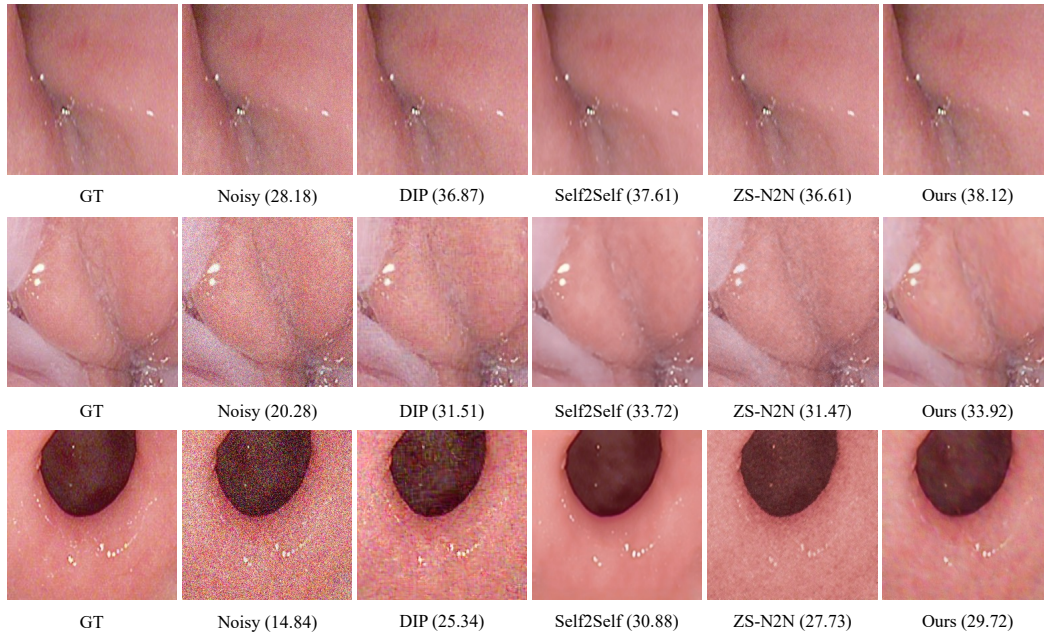


Figure 5: Gaussian denoising on KVASIR20 images. The numbers in parentheses are PSNR scores (dB). Upper row: $\sigma = 10$, middle row: $\sigma = 25$, lower row; $\sigma = 50$.

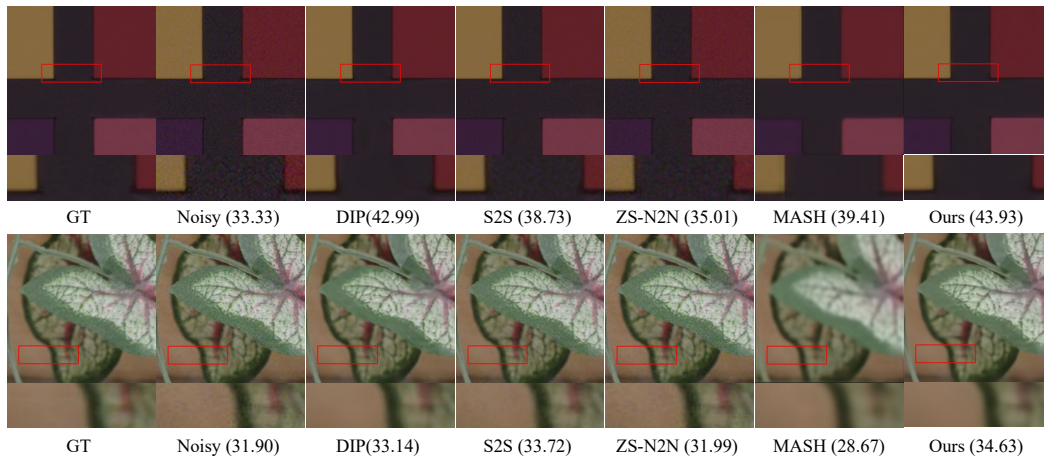


Figure 6: Visual comparison of our method against other competing methods on samples from the real-world datasets. Top: SIDD dataset. Bottom: PolyU dataset. The numbers in parentheses are PSNR scores (dB).