Learning to Translate Noise for Robust Image Denoising

Supplementary Material

Inju Ha[*1] Donghun Ryou[*2] Seonguk Seo[3] Bohyung Han[1,2]
[1]ECE & [2]IPAI, Seoul National University [3]Meta
{hij1112, dhryou, bhhan}@snu.ac.kr seonguk@meta.com

1. Proof on Wasserstein Distance

Let P and Q represent two probability distributions over $\mathbb{R}^d$. We use $X \sim P$ and $Y \sim Q$ to denote random variables with the distributions P and Q, respectively. The p-Wasserstein distance between two probability measures P and Q is defined as follows:

$$W_p(P,Q) = \left(\inf_{J \in \mathcal{J}(P,Q)} \int \|x-y\|^p dJ(x,y) \right)^{1/p}, \quad (1)$$

where $\mathcal{J}(P,Q)$ is the set of all joint distributions (or couplings) J on (X,Y) that have marginals P and Q. This formulation describes the minimum cost of transporting mass from distribution P to distribution Q using the coupling J, with the cost measured as the p-th power of the distance between points x and y.

In the Monge formulation, the goal is to find a transport map $T : \mathbb{R}^d \to \mathbb{R}^d$ such that the push-forward of P under T, denoted as $T_{\#}P$, equals Q. This problem can be mathematically formulated as:

$$\inf_T \int |x - T(x)|^p dP(x), \quad (2)$$

where the map T moves the distribution P to Q. However, an optimal map T may not always exist. In such cases, the Kantorovich formulation is used, allowing mass at each point to be split and transported to multiple locations, leading to a coupling-based approach.

For the specific case of $p = 1$, known as the Earth Mover's Distance, the dual formulation of the Wasserstein distance can be expressed as:

$$W_1(P,Q) = \sup_{f \in F} \left(\int f(x) dP(x) - \int f(x) dQ(x) \right), \quad (3)$$

where F represents the set of all Lipschitz continuous functions $f : \mathbb{R}^d \to \mathbb{R}$ such that $|f(y) - f(x)| \leq \|x - y\|$ for all $x, y \in \mathbb{R}^d$. Then, the 1-Wasserstein distance is given by:

$$W_1(P,Q) = \int_0^1 |F^{-1}(z) - G^{-1}(z)| dz, \quad (4)$$

where F^{-1} and G^{-1} denote the quantile functions (inverse CDFs) of P and Q, respectively.

When P and Q are empirical distributions based on the datasets, $X_1, X_2, \ldots, X_n$ and $Y_1, Y_2, \ldots, Y_n$, each of size n, the Wasserstein distance can be computed as a function of the order statistics:

$$W_1(P,Q) = \sum_{i=1}^{n} |X_{(i)} - Y_{(i)}|, \quad (5)$$

where $X_{(i)}$ and $Y_{(i)}$ denote the i-th order statistics of the datasets $X_1, X_2, \ldots, X_n$ and $Y_1, Y_2, \ldots, Y_n$.

In our approach, we utilize (5) to formulate $\mathcal{L}_{\text{spatial}}$ in Eq. (5) and $\mathcal{L}_{\text{freq}}$ in Eq. (10), which are employed during training to explicitly transport the translated noise distribution towards the target Gaussian distribution. We refer to [11] for a detailed discussion on Wasserstein distances and optimal transport.

2. Additional training details

The denoising models are trained for 200K iterations with a batch size of 32, except for Restormer and Xformer, where the batch size is reduced to 4 due to the limitation of computational resources. The noise translation network is trained for 5K iterations with a batch size of 4. The explicit noise translation loss was adopted only after half (2.5k) iterations, as early translated noise exhibits large deviations that destabilize optimization. Denoising network and noise translation network are trained with the AdamW [9] optimizer with an initial learning rate of 10^{-3}, which is reduced to 10^{-7} and 10^{-5} by a cosine annealing schedule, respectively. Each image is randomly cropped to 256×256 for training. All trainings were conducted using two NVIDIA RTX A6000 GPUs.

3. Additional experimental results

3.1. Model-agnostic applicability

To demonstrate the model-agnostic nature of our proposed noise translation framework, we applied our method to distinct architectural paradigms, specifically the transformer-based Restormer [13] and the kernel-based KBNet [15], as noted in the main paper. Integrating our training framework into these baselines yielded consistent performance improvements in real-world denoising tasks, as detailed in Table S.1. These results verify that our approach is not tailored to a specific backbone network but functions as a

[*] indicates equal contribution.

Table S.1. Demonstration of model-agnostic applicability. We compare representative state-of-the-art denoising networks with their counterparts integrated with our proposed framework (+ NTN). The results are presented in terms of PSNR↑ (dB) and SSIM↑. Models marked with an asterisk (*) are evaluated using official out-of-the-box models.

		In-distribution	Out-of-distribution								
Denoising Algo.	Metric	SIDD	Poly	CC	HighISO	iPhone	Huawei	OPPO	Sony	Xiaomi	OOD Avg.
Restormer* [13]	PSNR	40.02	37.66	36.33	38.29	40.13	38.42	39.56	44.19	35.65	38.78
	SSIM	0.9603	0.9793	0.9807	0.9756	0.9734	0.9675	0.9773	**0.9894**	0.9710	0.9768
Restormer + NTN	PSNR	39.22	**38.76**	**37.68**	**40.14**	**41.85**	**39.72**	**40.64**	**44.29**	**36.19**	**39.91**
	SSIM	0.9569	**0.9854**	**0.9866**	**0.9857**	**0.9786**	**0.9766**	**0.9800**	0.9871	**0.9750**	**0.9819**
NAFNet* [4]	PSNR	39.97	37.17	35.69	38.32	40.25	37.73	39.64	43.65	34.99	38.43
	SSIM	0.9600	0.9717	0.9811	0.9788	0.9707	0.9680	0.9786	0.9829	0.9685	0.9750
NAFNet + NTN	PSNR	39.24	**38.72**	**37.84**	**40.00**	**42.08**	**39.83**	**40.55**	**44.34**	**36.17**	**39.94**
	SSIM	0.9570	**0.9855**	**0.9877**	**0.9856**	**0.9812**	**0.9782**	**0.9801**	**0.9875**	**0.9749**	**0.9826**
KBNet* [15]	PSNR	40.35	36.82	35.23	38.09	38.00	35.18	37.80	41.79	34.25	37.15
	SSIM	0.9623	0.9789	0.9810	0.9788	0.9533	0.9462	0.9663	0.9790	0.9644	0.9685
KBNet + NTN	PSNR	39.13	**38.62**	**37.61**	**39.89**	**41.76**	**39.80**	**40.57**	**44.22**	**36.07**	**39.82**
	SSIM	0.9564	**0.9843**	**0.9861**	**0.9849**	**0.9761**	**0.9777**	**0.9798**	**0.9845**	**0.9743**	**0.9810**
XFormer* [14]	PSNR	39.98	37.45	35.95	37.86	39.99	38.35	39.61	43.89	35.54	38.58
	SSIM	0.9603	0.9778	0.9782	0.9741	0.9732	0.9678	0.9779	**0.9892**	0.9706	0.9761
XFormer + NTN	PSNR	39.10	**38.75**	**37.82**	**40.29**	**42.20**	**39.89**	**40.70**	**44.44**	**36.25**	**40.04**
	SSIM	0.9557	**0.9856**	**0.9862**	**0.9860**	**0.9826**	**0.9779**	**0.9807**	0.9886	**0.9751**	**0.9828**

Table S.2. Quantitative comparison of additional state-of-the-art image denoising methods on the SIDD validation set and multiple real-noise benchmarks. We present the results in terms of PSNR↑ (dB) and SSIM↑. Methods marked with an asterisk (*) are evaluated using official out-of-the-box models.

	Denoising method	Metric	SIDD	Poly	CC	HighISO	iPhone	Huawei	OPPO	Sony	Xiaomi	Total Avg.
GAN-based	DANet* [12]	PSNR	39.47	37.53	36.16	38.35	40.46	38.40	39.85	44.36	35.57	38.90
		SSIM	0.9570	0.9800	0.9815	0.9770	0.9751	0.9691	0.9789	0.9895	0.9706	0.9754
	C2N + DIDN* [6]	PSNR	35.36	37.60	36.92	38.82	40.48	38.56	40.08	**44.63**	35.44	38.65
		SSIM	0.9321	0.9781	0.9813	0.9765	0.9745	0.9679	0.9793	**0.9899**	0.9684	0.9720
	PNGAN + MIRNet* [2]	PSNR	**39.98**	37.41	36.10	38.24	39.93	38.02	39.56	43.15	35.24	38.62
		SSIM	**0.959**	0.9783	0.9810	0.9763	0.9711	0.9678	0.9782	0.9860	0.9690	0.9741
Others	PDDenoising* [16]	PSNR	33.95	37.10	35.82	37.09	39.13	37.69	38.75	43.60	35.03	37.57
		SSIM	0.8986	0.9716	0.9742	0.9628	0.9611	0.9592	0.9707	0.9866	0.9626	0.9608
	CLIPDenoising* [5]	PSNR	34.79	37.54	36.30	38.01	40.09	38.74	39.56	42.94	35.50	38.16
		SSIM	0.8982	0.9794	0.9809	0.9771	0.9685	0.9715	0.9769	0.9824	0.9707	0.9672
	MaskDenoising* [3]	PSNR	28.66	34.56	33.87	34.61	36.54	34.89	35.30	37.89	33.46	34.42
		SSIM	0.7127	0.9553	0.9703	0.9649	0.9273	0.9586	0.9593	0.9354	0.9531	0.9263
	IDF* [8]	PSNR	31.73	37.22	36.14	37.50	40.36	37.77	38.64	42.58	34.95	37.43
		SSIM	0.8011	0.9790	0.9809	0.9759	0.9776	0.9649	0.9728	0.9848	0.9617	0.9554
Ours	**NAFNet + NTN**	PSNR	39.24	**38.72**	**37.84**	**40.00**	**42.08**	**39.83**	**40.55**	44.34	**36.17**	**39.86**
		SSIM	0.9570	**0.9855**	**0.9877**	**0.9856**	**0.9812**	**0.9782**	**0.9801**	0.9875	**0.9749**	**0.9797**

highly compatible framework capable of boosting the performance of diverse denoising architectures.

3.2. Comparison with additional denoising methods

Table S.2 presents quantitative comparisons between additional image denoising methods and our approach. Generation-based methods such as DANet [12], C2N [6], and PNGAN [2] utilize the SIDD dataset for training, focusing on synthetic-to-real noise generation. PDDenoising [16] applies pixel-shuffle down-sampling to denoise complex real noise with a simple Gaussian denoiser. However, pixel-shuffling alters image semantics, and optimal shuffling parameters vary per image, making optimization challenging and yielding poor performance. MaskDenoising [3] and IDF [8] are trained solely on Gaussian noise ($\sigma = 15$), leading to notably poor performance on real-world noise datasets. CLIPDenoising [5] leverages the CLIP encoder and incorporates additional training on synthetic noise generated using Poisson-Gaussian models for sRGB denoising. However, none of these methods outperform our approach across most benchmarks, underscoring the efficacy of our noise translation framework in handling real-world noise scenarios.

3.3. Comparison with generalization methods

Table S.3 showcases results from our reproduction of the AFM [10] and LAN [7] methods using their officially released code, in combination with NAFNet, to enable a direct comparison with our proposed method. Applying LAN to other denoising architectures from our experiments, such as Restormer [13], KBNet [15], and Xformer [14], was infeasible due to excessive memory demands of LAN when processing images with resolutions exceeding 512 × 512. Additionally, our reproduction experiments revealed

Table S.3. Quantitative comparisons with state-of-the-art generalization methods on the SIDD validation set and multiple real-noise benchmarks. We present the results in terms of PSNR↑ (dB) and SSIM↑. Methods marked with a dagger (†) indicate evaluations performed on reproduced models under standardized conditions.

Baseline	Generalization	Metric	In-distribution	Out-of-distribution								
			SIDD	Poly	CC	HighISO	iPhone	Huawei	OPPO	Sony	Xiaomi	OOD Avg.
NAFNet [4]	None	PSNR	39.97	37.17	35.69	38.32	40.25	37.73	39.64	43.65	34.99	38.43
		SSIM	0.9600	0.9717	0.9811	0.9788	0.9707	0.9680	0.9786	0.9829	0.9685	0.9750
	+ AFM† [10]	PSNR	39.83	37.28	36.31	38.40	39.68	38.21	39.84	43.31	35.39	38.55
		SSIM	0.9592	0.9696	0.9815	0.9783	0.9545	0.9683	0.9788	0.9637	0.9709	0.9707
	+ LAN† [7]	PSNR	39.88	37.17	35.59	38.21	40.48	37.53	39.42	43.66	34.83	38.36
		SSIM	0.9594	0.9792	0.9808	0.9787	0.9766	0.9680	0.9773	**0.9883**	0.9667	0.9770
	+ NTN	PSNR	39.24	**38.72**	**37.84**	**40.00**	**42.08**	**39.83**	**40.55**	**44.34**	**36.17**	**39.94**
		SSIM	0.9570	**0.9855**	**0.9877**	**0.9856**	**0.9812**	**0.9782**	**0.9801**	0.9875	**0.9749**	**0.9826**

Table S.4. Expanded table of Table 2: Gaussian injection and explicit noise translation loss.

	Metric	In-distribution	Out-of-distribution								
		SIDD	Poly	CC	HighISO	iPhone	Huawei	OPPO	Sony	Xiaomi	OOD Avg.
w/o GIBlock	PSNR	39.35	38.32	37.25	39.22	40.80	39.24	39.75	43.86	35.74	39.27
	SSIM	0.9573	0.9820	0.9864	0.9794	0.9700	0.9727	0.9745	0.9857	0.9683	0.9774
w/o Explicit Loss	PSNR	39.05	38.54	37.58	39.79	41.53	39.68	40.40	43.89	36.00	39.61
	SSIM	0.9556	0.9835	0.9866	0.9844	0.9737	0.9773	0.9790	0.9827	0.9737	0.9801
w/o adaptive gating ($\mathcal{L}_{\text{explicit}}$)	PSNR	39.33	38.59	37.10	39.35	41.69	39.56	40.27	**44.34**	35.95	39.61
	SSIM	0.9572	0.9846	0.9864	0.9821	0.9793	0.9753	0.9782	**0.9888**	0.9720	0.9808
Full (**Ours**)	PSNR	39.24	**38.72**	**37.84**	**40.00**	**42.08**	**39.83**	**40.55**	**44.34**	**36.17**	**39.94**
	SSIM	0.9570	**0.9855**	**0.9877**	**0.9856**	**0.9812**	**0.9782**	**0.9801**	0.9875	**0.9749**	**0.9826**

that LAN's adaptation methodology not only requires substantial computational resources but also fails to improve generalization performance at resolutions exceeding 256 × 256. In contrast, our noise translation framework consistently delivers robust and superior generalization performance across diverse out-of-distribution benchmarks, outperforming both AFM and LAN.

3.4. Expanded Table: Gaussian Injection Block and Explicit Noise Translation Loss

Table S.4 provides the complete results for the OOD benchmarks in Table 2 of the main paper. These results further demonstrate the effectiveness of the Gaussian injection block and explicit noise translation loss within the proposed framework.

3.5. Expanded Table: Adaptability of the Noise Translation Network

Table S.5 extends the results of Table 4 in the main paper, analyzing various combinations of denoising networks used during the training and inference phases within our framework. In this context, $\mathcal{D}_{\theta^*}$ for testing refers to the pretrained denoising network utilized during inference, while $\mathcal{D}_{\theta^*}$ for training $\mathcal{T}_\phi$ denotes the pretrained denoising network employed for training the noise translation network. For instance, when $\mathcal{D}_{\theta^*}$ for testing is NAFNet and $\mathcal{D}_{\theta^*}$ for training $\mathcal{T}_\phi$ is KBNet, the noise translation network is trained to map real-world noise to align with KBNet's prior but is evaluated with NAFNet during inference. Notably, when $\mathcal{D}_{\theta^*}$ for testing and $\mathcal{D}_{\theta^*}$ for training $\mathcal{T}_\phi$ correspond to different pretrained denoising models, our framework maintains high performance by effectively translating noise distributions to align with the priors of the inference model. This adaptability implies that as new denoising architectures are introduced, they can be seamlessly incorporated into our framework, potentially achieving even greater performance improvements without the need to retrain the noise translation network. Such flexibility ensures the practical applicability of our noise translation framework in real-world scenarios.

3.6. Comparison with naïve combination of real and synthetic dataset for training

We compare our method with the naive approach of combining Gaussian noise dataset with the SIDD dataset for training. The results are presented in Table S.6. The Gaussian noise dataset used in this comparison is identical to the one which is used for our method. The results demonstrate that naively combining real-noise and Gaussian noise datasets during training does not yield significant improvements in generalization performance. This highlights the superiority and efficacy of our method in enhancing denoising performance across diverse noise distributions.

3.7. Incorporating a pure Gaussian pretrained denoising network

We conduct an experiment on the officially published denoising model, Restormer [13], which was trained for blind gaussian noise removal. By leveraging it in our framework, we trained the corresponding noise translation network with the proposed methodology to evaluate the overall performance. The results are provided in Table S.7, which demon-

Table S.5. Expanded table of Table 4: Adaptability of the noise translation network.

$\mathcal{D}_{\theta^*}$ for testing	$\mathcal{D}_{\theta^*}$ for training $\mathcal{T}_\phi$	Metric	In-distribution	Out-of-distribution								
			SIDD	Poly	CC	HighISO	iPhone	Huawei	OPPO	Sony	Xiaomi	OOD Avg.
Restormer	Restormer	PSNR	39.22	38.76	37.68	40.14	41.85	39.72	40.64	44.29	36.19	39.91
		SSIM	0.9569	0.9854	0.9866	0.9857	0.9786	0.9766	0.9800	0.9871	0.9750	0.9819
	NAFNet	PSNR	39.16	38.65	37.34	39.98	41.92	39.73	40.56	43.95	36.19	39.79
		SSIM	0.9564	0.9851	0.9861	0.9854	0.9811	0.9770	0.9800	0.9871	0.9750	0.9821
	KBNet	PSNR	39.21	38.53	37.05	39.88	41.93	39.65	40.53	43.81	36.14	39.69
		SSIM	0.9567	0.9848	0.9851	0.9850	0.9817	0.9766	0.9799	0.9873	0.9743	0.9819
	Xformer	PSNR	39.12	38.73	37.80	40.11	41.65	39.64	40.42	44.11	36.18	39.83
		SSIM	0.9559	0.9850	0.9867	0.9852	0.9774	0.9763	0.9795	0.9866	0.9746	0.9814
NAFNet	Restormer	PSNR	39.16	38.61	38.00	40.07	41.95	39.71	40.41	44.05	36.22	39.88
		SSIM	0.9565	0.9846	0.9877	0.9856	0.9800	0.9775	0.9793	0.9858	0.9748	0.9819
	NAFNet	PSNR	39.24	38.72	37.84	40.00	42.08	39.83	40.55	44.34	36.17	39.94
		SSIM	0.9570	0.9855	0.9877	0.9856	0.9812	0.9782	0.9801	0.9875	0.9749	0.9826
	KBNet	PSNR	39.17	38.67	37.54	39.83	41.95	39.73	40.45	44.18	36.11	39.81
		SSIM	0.9565	0.9853	0.9870	0.9849	0.9814	0.9777	0.9798	0.9875	0.9741	0.9822
	Xformer	PSNR	39.06	38.66	38.04	40.07	41.93	39.72	40.46	44.11	36.17	39.89
		SSIM	0.9560	0.9848	0.9877	0.9852	0.9792	0.9773	0.9794	0.9857	0.9743	0.9817
KBNet	Restormer	PSNR	39.14	38.59	37.79	40.14	41.77	39.67	40.45	44.02	36.22	39.83
		SSIM	0.9564	0.9843	0.9863	0.9855	0.9776	0.9773	0.9798	0.9850	0.9751	0.9814
	NAFNet	PSNR	39.12	38.62	37.67	39.95	41.88	39.73	40.51	44.16	36.16	39.84
		SSIM	0.9563	0.9849	0.9863	0.9853	0.9797	0.9776	0.9800	0.9861	0.9747	0.9818
	KBNet	PSNR	39.13	38.62	37.61	39.89	41.76	39.80	40.57	44.22	36.07	39.82
		SSIM	0.9564	0.9843	0.9861	0.9849	0.9761	0.9777	0.9798	0.9845	0.9743	0.9810
	Xformer	PSNR	39.08	38.64	37.74	40.08	41.76	39.69	40.48	44.09	36.18	39.83
		SSIM	0.9560	0.9845	0.9862	0.9851	0.9776	0.9772	0.9797	0.9849	0.9747	0.9812
Xformer	Restormer	PSNR	39.09	38.60	37.81	40.25	42.12	39.72	40.51	44.10	36.28	39.92
		SSIM	0.9558	0.9851	0.9859	0.9861	0.9831	0.9776	0.9801	0.9885	0.9753	0.9827
	NAFNet	PSNR	39.17	38.62	37.73	40.18	42.29	39.75	40.56	44.27	36.19	39.95
		SSIM	0.9563	0.9856	0.9860	0.9860	0.9851	0.9775	0.9806	0.9896	0.9750	0.9832
	KBNet	PSNR	39.10	38.51	37.43	39.98	42.12	39.69	40.47	43.98	36.15	39.79
		SSIM	0.9557	0.9851	0.9854	0.9852	0.9844	0.9774	0.9801	0.9889	0.9743	0.9826
	Xformer	PSNR	39.10	38.75	37.82	40.29	42.20	39.89	40.70	44.44	36.25	40.04
		SSIM	0.9557	0.9856	0.9862	0.9860	0.9826	0.9779	0.9807	0.9886	0.9751	0.9828

Table S.6. Quantitative comparisons of real-world denoising performance across various training set configurations. Methods marked with an asterisk (*) are evaluated using official out-of-the-box models.

Denoising network	Trained noise	Metric	SIDD	Poly	CC	HighISO	iPhone	Huawei	OPPO	Sony	Xiaomi	Total Avg.
Restormer [13]	SIDD*	PSNR	40.02	37.66	36.33	38.29	40.13	38.42	39.56	44.19	35.65	38.92
		SSIM	0.9603	0.9793	0.9807	0.9756	0.9734	0.9675	0.9773	**0.9894**	0.9710	0.9749
	SIDD + Gaussian	PSNR	39.67	37.61	35.52	38.08	40.38	38.51	39.83	43.72	35.54	38.76
		SSIM	0.9586	0.9800	0.9771	0.9765	0.9748	0.9690	0.9795	0.9865	0.9714	0.9748
	Ours	PSNR	39.22	**38.76**	**37.68**	**40.14**	**41.85**	**39.72**	**40.64**	**44.29**	**36.19**	**39.83**
		SSIM	0.9569	**0.9854**	**0.9866**	**0.9857**	**0.9786**	**0.9766**	**0.9800**	0.9871	**0.9750**	**0.9791**
NAFNet [4]	SIDD*	PSNR	39.97	37.17	35.69	38.32	40.25	37.73	39.64	43.65	34.99	38.60
		SSIM	0.9600	0.9717	0.9811	0.9788	0.9707	0.9680	0.9786	0.9829	0.9685	0.9734
	SIDD + Gaussian	PSNR	39.61	37.93	36.10	38.20	40.66	37.92	39.66	43.49	35.65	38.80
		SSIM	0.9582	0.9809	0.9797	0.9788	0.9779	0.9704	0.9815	**0.9886**	0.9716	0.9764
	Ours	PSNR	39.24	**38.72**	**37.84**	**40.00**	**42.08**	**39.83**	**40.55**	**44.34**	**36.17**	**39.86**
		SSIM	0.9570	**0.9855**	**0.9877**	**0.9856**	**0.9812**	**0.9782**	**0.9801**	0.9875	**0.9749**	**0.9797**

strate that even when using a pure Gaussian pretrained out-of-the-box model within our framework, the real-world denoising performance undergoes significant improvement. This enhancement can be attributed to our noise translation network, which effectively adapts the real-world noise into a form closer to the Gaussian noise that the pretrained model is adept at handling. Note that, in our pretraining approach, the denoising network is trained on Gaussian noise augmented with the same real-world noise used to train the noise translation network. This distinction in pretraining strategies explains the observed performance gap between the two results.

3.8. Effects of hyperparameters

Table S.8 presents the ablative results of our hyperparameters, including pretraining noise level (σ), noise injection level ($\tilde{\sigma}$), explicit loss weight (α), and spatial frequency ratio (β). The pretraining level part of Table S.8 shows the ablation results for the Gaussian noise levels added to create noisy input during denoising network pretraining. As

Table S.7. Quantitative results of applying noise translation framework to denoising model pre-trained on pure synthetic Gaussian noise.

Denoising methods	Metric	SIDD	Poly	CC	HighISO	iPhone	Huawei	OPPO	Sony	Xiaomi	Total Avg.
Restormer-Gaussian [13]	PSNR	23.99	35.97	33.41	35.31	38.02	36.94	37.84	42.90	34.49	35.43
	SSIM	0.4990	0.9585	0.9563	0.9464	0.9448	0.9502	0.9629	0.9831	0.9554	0.9063
+ Ours	PSNR	**39.08**	**37.88**	**35.75**	**39.33**	**41.21**	**38.44**	**39.96**	**43.18**	**35.63**	**38.94**
	SSIM	**0.9542**	**0.9832**	**0.9800**	**0.9827**	**0.9823**	**0.9719**	**0.9790**	**0.9835**	**0.9720**	**0.9765**

Table S.8. Sensitivity results on various hyperparameters in our framework.

Pretraining Level			Noise Injection Level			Explicit Loss Weight			Spatial-Frequency Ratio		
σ	SIDD	OOD Avg.	$\tilde{\sigma}$	SIDD	OOD Avg.	α	SIDD	OOD Avg.	β	SIDD	OOD Avg.
5	39.37	38.96	0	39.43	39.37	0	39.07	39.74	0	39.13	39.84
10	39.29	39.64	1	39.38	39.54	0.001	39.08	39.76	0.0005	39.16	39.88
15	39.24	**39.94**	5	39.09	39.78	0.005	39.11	39.80	0.001	39.18	39.91
20	38.88	39.53	15	39.15	39.86	0.01	39.14	39.85	0.002	**39.24**	**39.94**
25	38.73	39.21	50	39.15	39.90	0.05	**39.24**	**39.94**	0.005	39.17	39.94
			100	**39.24**	**39.94**	0.1	39.19	39.92	0.01	39.09	39.77
			200	39.19	39.93	0.5	39.13	39.07	0.02	39.00	38.97

the noise level increased, the performance on in-distribution (ID) consistently decreased. For out-of-distribution (OOD), the performance improved until noise level of 15, beyond which it began to degrade, due to oversmoothing effects caused by learning to handle strong noise. The noise injection level part presents the ablation results for Gaussian noise injection levels. Increasing the noise level led to a decline in ID performance, while OOD performance improved up to a certain point. The explicit loss weight and spatial-frequency ratio parts show the ablation results for the α and β values in Eq. (11) and (10), respectively. Overall, the ablation experiments determined the optimal hyperparameters as follows: a pretraining noise level $\sigma = 15$, a Gaussian noise injection level $\tilde{\sigma} = 100$, $\alpha = 5 \times 10^{-2}$, and $\beta = 2 \times 10^{-3}$.

3.9. Training Stability

To evaluate the stability of training our noise translation network, we conducted five independent training runs using different random seeds and measured the PSNR metrics across multiple datasets. The standard deviations for each dataset were as follows: SIDD (0.023), Poly (0.007), CC (0.021), HighISO (0.009), iPhone (0.008), Huawei (0.010), OPPO (0.016), Sony (0.022), and Xiaomi (0.020). The average standard deviation across all datasets was 0.013, confirming the robustness and stability of training with our framework.

4. Qualitative Analysis

4.1. Additional OOD datasets and SIDD dataset

Qualitative results on all datasets in the main paper are shown in Figures S.1, S.2, S.3 and S.4. Our method significantly surpasses the PSNR scores of other denoising models on all OOD datasets (iPhone, Huawei, OPPO, Sony, Xiaomi). For the in-distribution SIDD, our method produces clean results without generating unnecessary zipper artifacts.

4.2. NIND (Natural Image Noise Dataset)

The NIND [1] dataset comprises high-resolution images captured at various ISO levels. High ISO images are noisier due to increased sensor sensitivity, while low ISO images, though cleaner, may require more optimal settings or equipment to capture effectively. Figure S.5 and Figure S.6 illustrate the qualitative results of our method applied to high ISO images, demonstrating its ability to effectively remove noise while preserving fine image details. The denoised outputs are visually clean and free from artifacts, achieving a quality comparable to images captured at low ISO levels.

4.3. Real-world smartphone image noise and denoising results

Finally, we present the denoised results of images captured using our Galaxy S22+ smartphone, as shown in Figure S.7.

5. Limitations

Although our approach provides strong robustness to diverse real-world noise, it may appear to compromise performance on in-distribution (ID) datasets. As discussed in Figure 8, this effect is largely due to the pronounced overfitting of conventional denoising models, which tend to reconstruct dataset-specific artifacts rather than genuine image content.

Moreover, our evaluation primarily relies on PSNR and SSIM computed against the provided ground-truth images. While these metrics are standard practice, they inherently struggle to capture perceptual realism and may not align well with human judgments. Although our method consistently improves these metrics, such quantitative measures do not fully represent perceptual quality, which remains subjective and can vary across viewers. Future work includes exploring alternative evaluation protocols or perceptually aligned metrics that better account for realism, semantic consistency, and human preference.

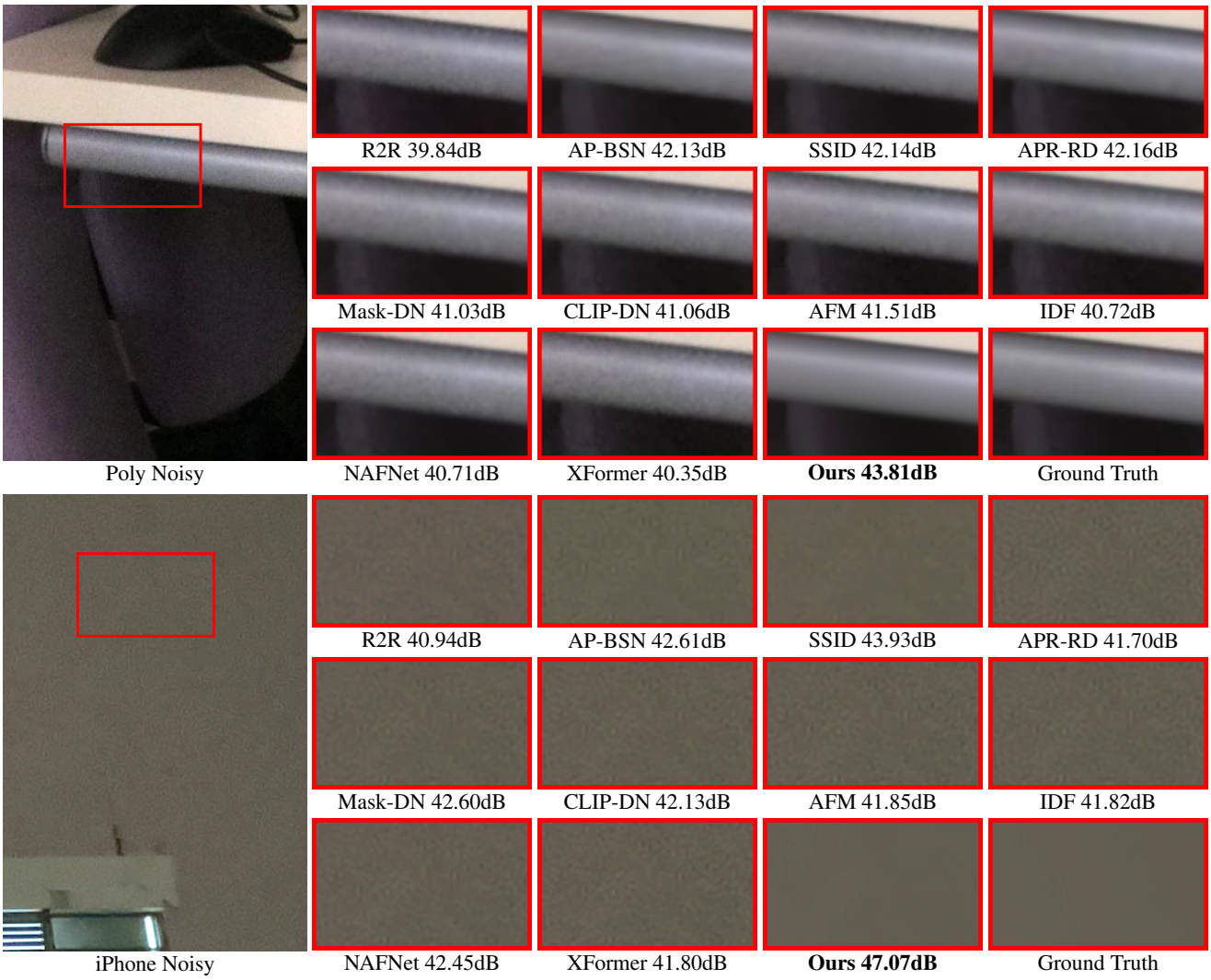


Figure S.1. Comparison between the qualitative results of various denoising networks including ours (noise translation network with pretrained NAFNet), on the out-of-distribution (OOD) datasets (Poly and iPhone). Our result displays cleaner outputs compared to other state-of-the-art networks that are directly trained from a real-noise dataset. Zoom in for better comparison.

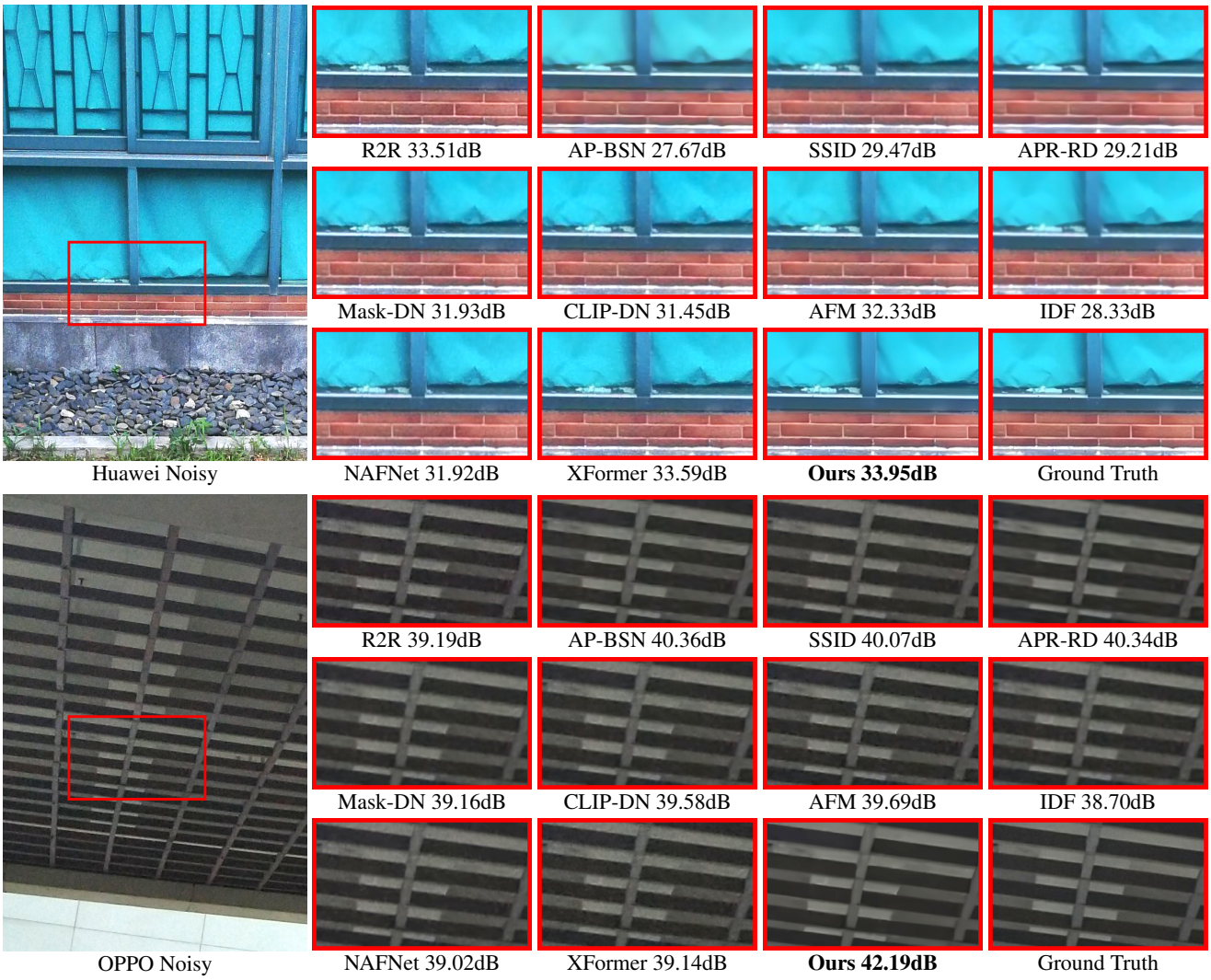


Figure S.2. Comparison between the qualitative results of various denoising networks including ours (noise translation network with pretrained NAFNet), on the out-of-distribution (OOD) datasets (Huawei and OPPO). Our result displays cleaner outputs compared to other state-of-the-art networks that are directly trained from a real-noise dataset. Zoom in for better comparison.

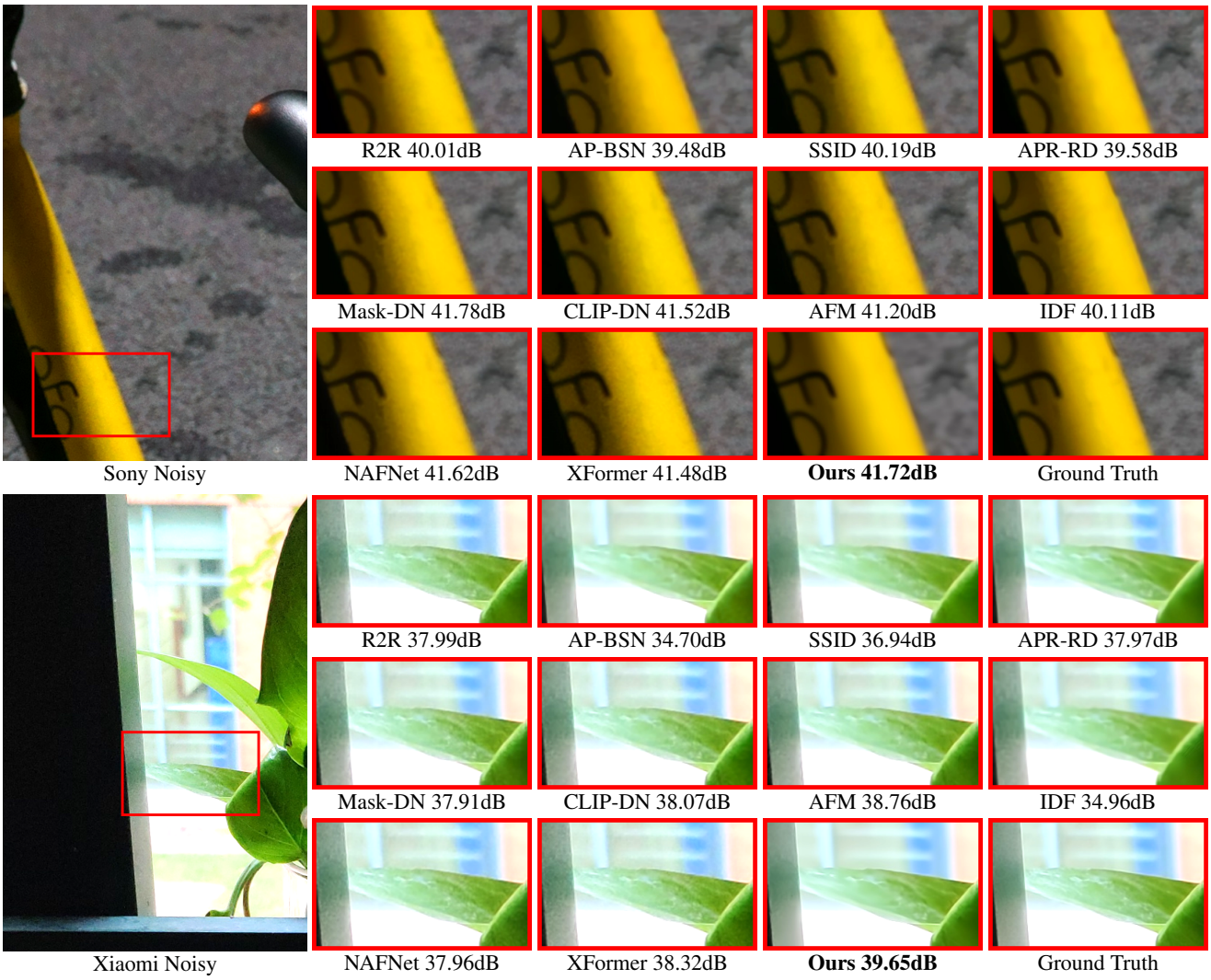


Figure S.3. Comparison between the qualitative results of various denoising networks including ours (noise translation network with pretrained NAFNet), on the out-of-distribution (OOD) datasets (Sony and Xiaomi). Our result displays cleaner outputs compared to other state-of-the-art networks that are directly trained from a real-noise dataset. Zoom in for better comparison.

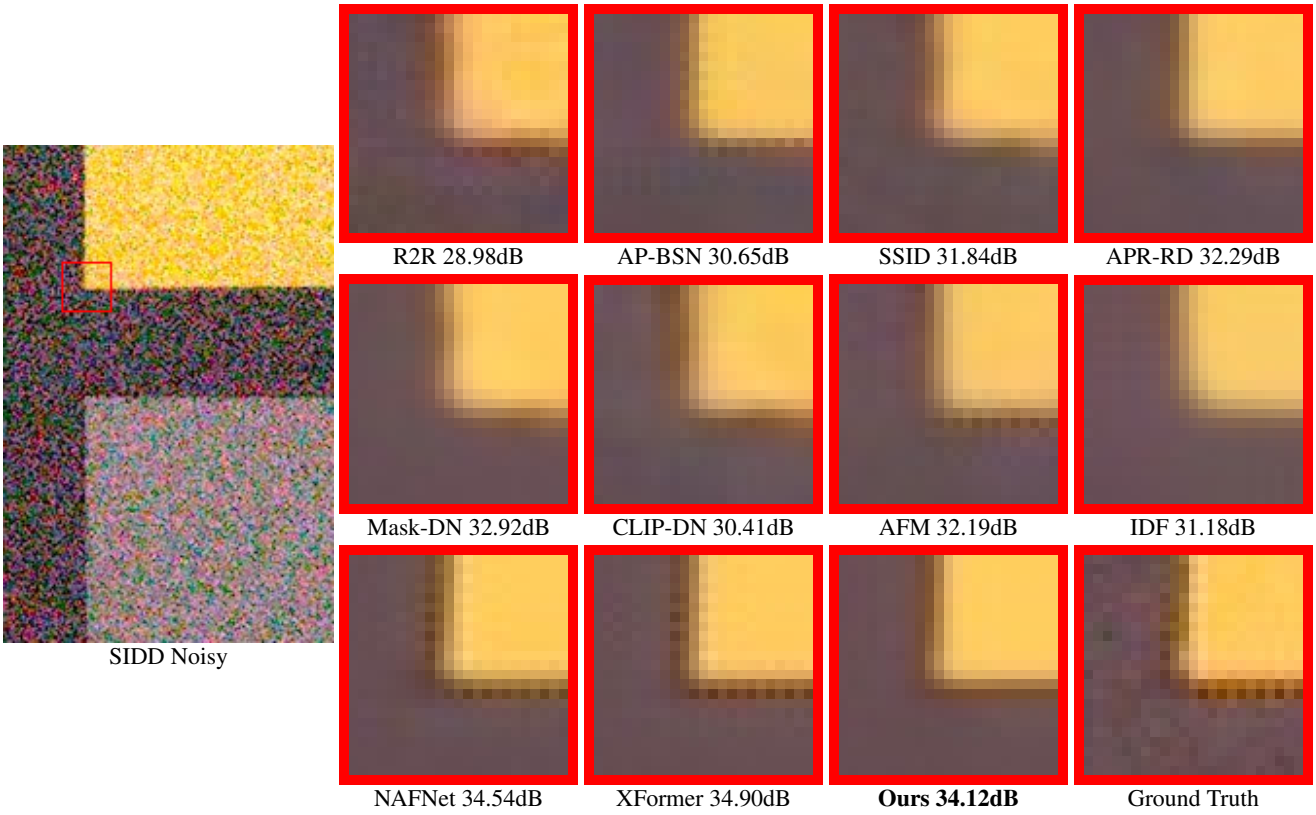


Figure S.4. Comparison between the qualitative results of various denoising networks including ours (noise translation network with pretrained NAFNet), on the SIDD dataset. Our result displays competitive visual quality with real noise.

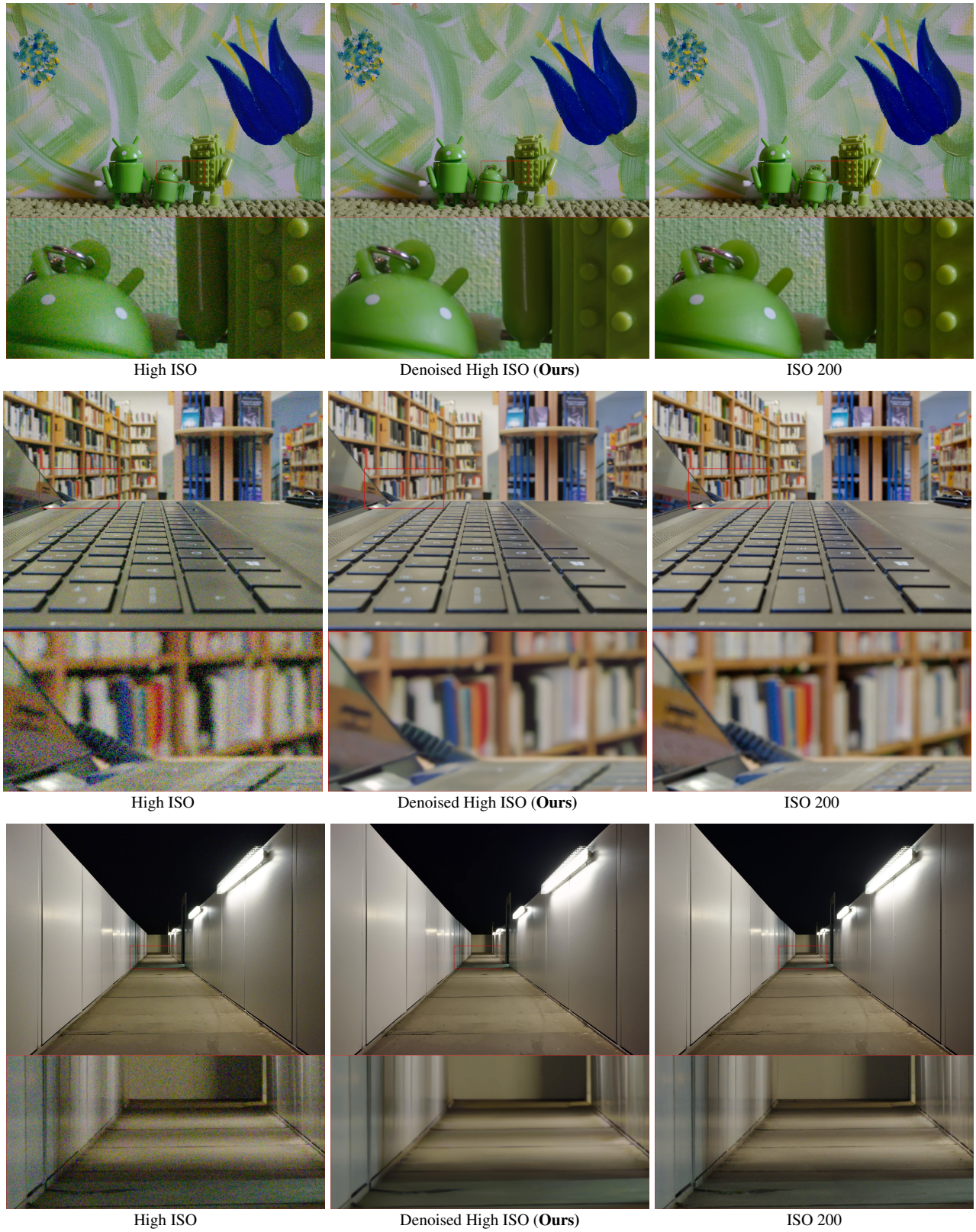


Figure S.5. Qualitative results of our noise translation network with pretrained NAFNet applied to high ISO images from the NIND dataset. Our method effectively removes noise while preserving fine details, producing outputs comparable to low ISO (ISO 200) images. Zoom in for a closer inspection of the visual quality.

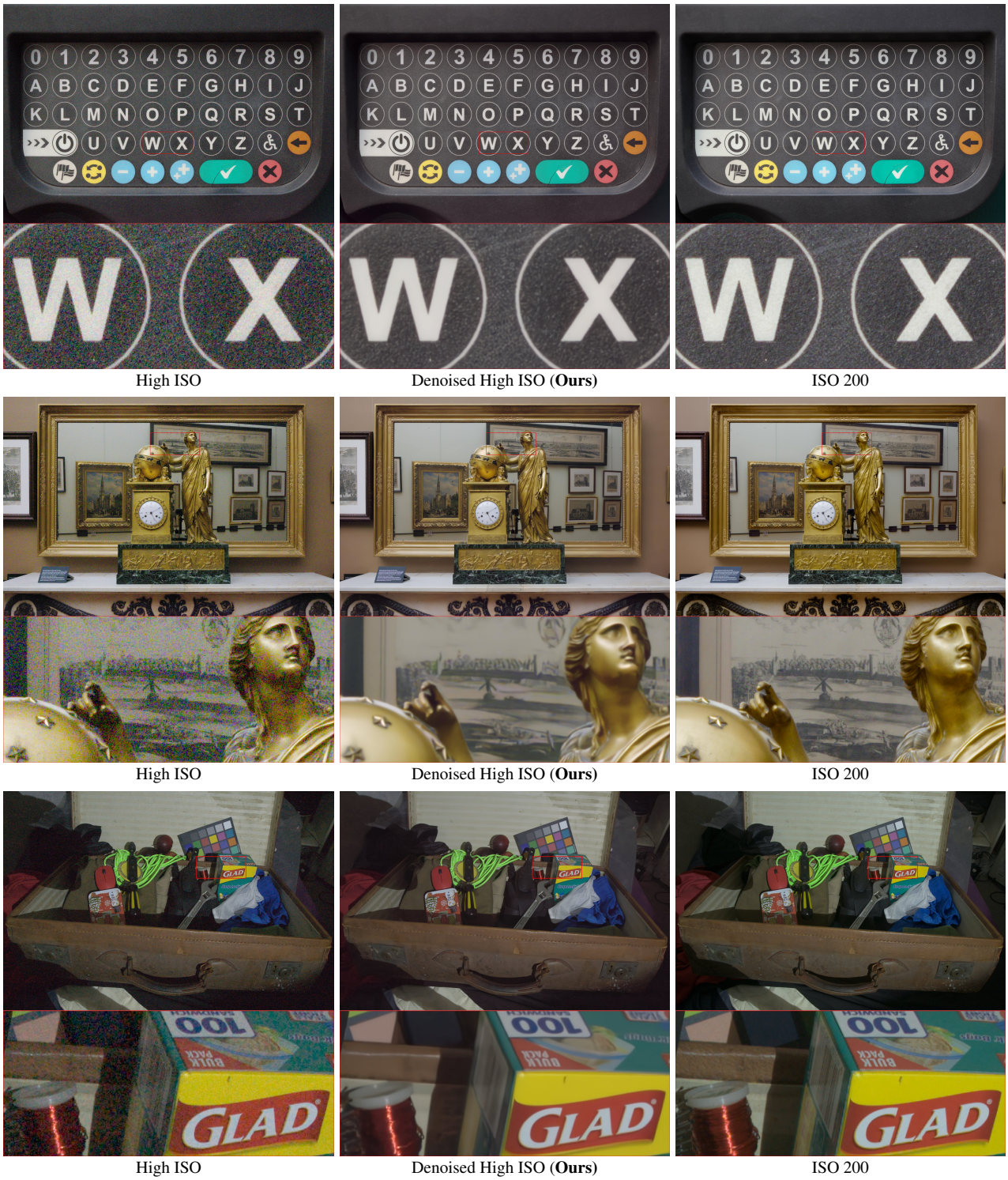
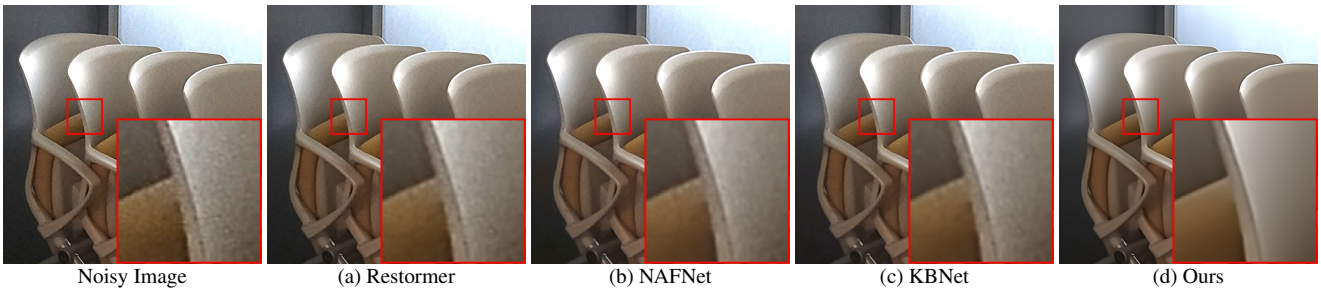


Figure S.6. Qualitative results of our noise translation network with pretrained NAFNet applied to high ISO images from the NIND dataset. Our method effectively removes noise while preserving fine details, producing outputs comparable to low ISO (ISO 200) images. Zoom in for a closer inspection of the visual quality.



Noisy Image (a) Restormer (b) NAFNet (c) KBNet (d) Ours

Noisy Image (e) Restormer (f) NAFNet (g) KBNet (h) Ours

Figure S.7. Comparison between the qualitative results of various denoising networks including ours (noise translation network with pretrained NAFNet) on images captured by Galaxy S22+ smartphone. Zoom in for a closer inspection of the visual quality.

References

[1] Benoit Brummer and Christophe De Vleeschouwer. Natural image noise dataset. In *CVPRW*, 2019. 5

[2] Yuanhao Cai, Xiaowan Hu, Haoqian Wang, Yulun Zhang, Hanspeter Pfister, and Donglai Wei. Learning to generate realistic noisy images via pixel-level noise-aware adversarial training. In *NeurIPS*, 2021. 2

[3] Haoyu Chen, Jinjin Gu, Yihao Liu, Salma Abdel Magid, Chao Dong, Qiong Wang, Hanspeter Pfister, and Lei Zhu. Masked image training for generalizable deep image denoising. In *CVPR*, 2023. 2

[4] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. In *ECCV*, 2022. 2, 3, 4

[5] Jun Cheng, Dong Liang, and Shan Tan. Transfer clip for generalizable image denoising. In *CVPR*, 2024. 2

[6] Geonwoon Jang, Wooseok Lee, Sanghyun Son, and Kyoung Mu Lee. C2n: Practical generative noise modeling for real-world denoising. In *ICCV*, 2021. 2

[7] Changjin Kim, Tae Hyun Kim, and Sungyong Baik. Lan: Learning to adapt noise for image denoising. In *CVPR*, 2024. 2, 3

[8] Dongjin Kim, Jaekyun Ko, Muhammad Kashif Ali, and Tae Hyun Kim. Idf: Iterative dynamic filtering networks for generalizable image denoising. In *ICCV*, 2025. 2

[9] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 1

[10] Donghun Ryou, Inju Ha, Hyewon Yoo, Dongwan Kim, and Bohyung Han. Robust image denoising through adversarial frequency mixup. In *CVPR*, 2024. 2, 3

[11] C. Villani and American Mathematical Society. *Topics in Optimal Transportation*. American Mathematical Society, 2003. 1

[12] Zongsheng Yue, Qian Zhao, Lei Zhang, and Deyu Meng. Dual adversarial network: Toward real-world noise removal and noise generation. In *ECCV*, 2020. 2

[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 *CVPR*, 2022. 1, 2, 3, 4, 5

[14] Jiale Zhang, Yulun Zhang, Jinjin Gu, Jiahua Dong, Linghe Kong, and Xiaokang Yang. Xformer: Hybrid x-shaped transformer for image denoising. In *ICLR*, 2024. 2

[15] Yi Zhang, Dasong Li, Xiaoyu Shi, Dailan He, Kangning Song, Xiaogang Wang, Honwei Qin, and Hongsheng Li. Kbnet: Kernel basis network for image restoration. *arXiv preprint arXiv:2303.02881*, 2023. 1, 2

[16] Yuqian Zhou, Jianbo Jiao, Haibin Huang, Yang Wang, Jue Wang, Honghui Shi, and Thomas Huang. When awgn-based denoiser meets real noises. In *AAAI*, 2020. 2