Diffusion-Based sRGB Real Noise Generation via Prompt-Driven Noise Representation Learning

Supplementary Material

Contents

S. Supplementary Material

S.1. Prompt DiT (P-DiT)

In Fig. 2 and Fig. S1, we introduce P-DiT, which fully leverages the prompt features extracted from the Prompt Encoder $\mathcal{E}$ to synthesize latent codes that align with the embedded information of the input noise characteristics. Our P-DiT is based on DiT [42], a transformer-based CM architecture specifically designed for training diffusion models.

In Fig. S1 (a), the P-DiT consists of a series of B P-DiT blocks through which the noise-added input latent $\mathbf{z}_{t+1}$ is processed. Moreover, for conditioning features, we use the timestep $t+1$, clean images $\boldsymbol{\mathcal{I}}_{\text{Clean}}$, and prompt features $\mathbf{F}_{\text{Local}}$ and $\mathbf{F}_{\text{Global}}$. The timestep enables the model to adjust its denoising predictions based on the noise levels of the diffusion process at each timestep. Precisely, the timestep is embedded by a timestep embedder, composed of sinusoidal embeddings and an MLP block [59]. We also use the clean image to help the model learn the signal-dependent properties of real-world noise and prompt features to learn the camera-dependent properties. This enables the generation of latent codes that capture input noise characteristics without relying on metadata, with these features embedded by a conditional embedder. Specifically, we first downsample these inputs to match the spatial size of the latent codes and then extract shallow features separately using 3×3 convolutional layers, which are concatenated along the channel axis as follows:

$$\mathbf{F}_{\text{Cond}} = \Big[\!\Big[\text{Conv}_{3\times3}\big(\text{PD}(\boldsymbol{\mathcal{I}}_{\text{Clean}})\big), \text{Conv}_{3\times3}\big(\text{PD}(\mathbf{F}_{\text{Local}})\big), \text{Conv}_{3\times3}\big(\text{PD}(\mathbf{F}_{\text{Global}})\big)\Big]\!\Big], \tag{S1}$$

where $\text{PD}(\cdot)$ indicates a pixel downsampling operator. Note that we use $\mathbf{F}_{\text{Global}}$ for all scale levels, and each feature is processed separately. The concatenated feature $\mathbf{F}_{\text{Cond}}$ is then processed with global average pooling and added to the time embeddings.

In Fig. S1 (b), we present the P-DiT block. We employ adaptive layer normalization (AdaLN) as the conditioning mechanism to modulate the statistics of input features. AdaLN consists of two components: layer normalization [43] and adaptive modulation. AdaLN first normalizes the input features to have a mean of zero and a standard deviation of one. Then, the normalized features are modulated using scale and shift parameters derived from the conditioning input.

Moreover, to fully leverage the available information from

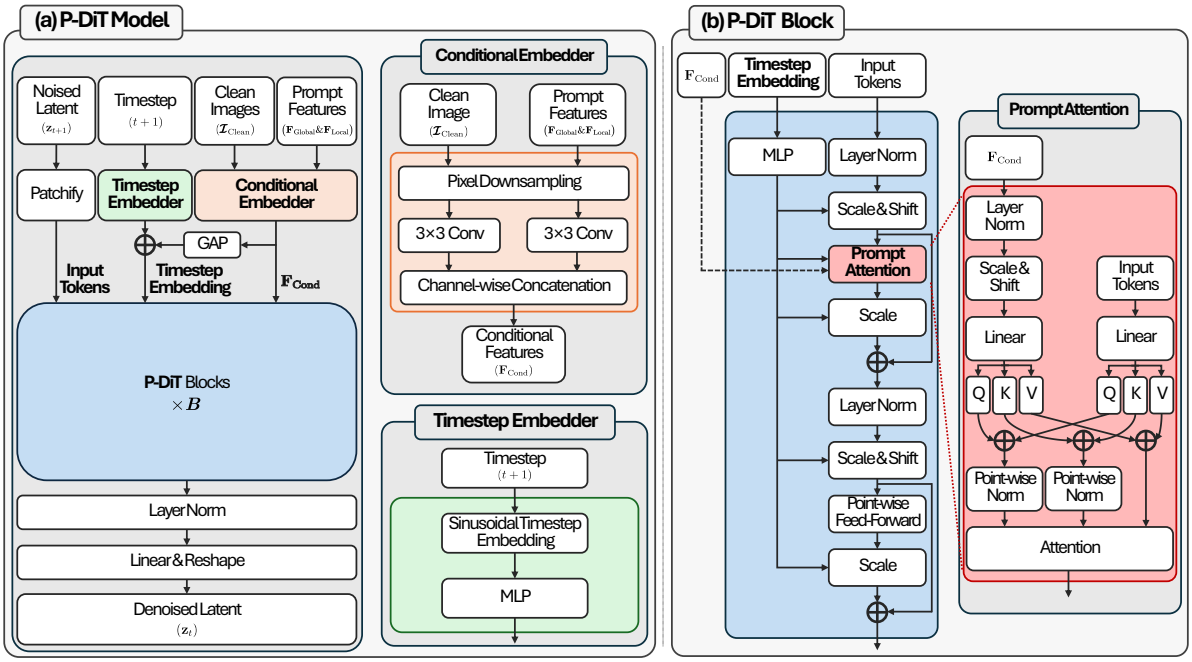

Figure S1. Overview of Prompt DiT (P-DiT). (a) Overall P-DiT structure. (b) P-DiT block.

the conditions and capture locally varying spatial correlations characteristics in prompt features, we further enhance these features through a prompt attention mechanism, referred to as **Prompt Attention**.

Prompt Attention. As illustrated in Fig. S1 (b), we use the conditions to generate the key (K), query (Q), and value (V) within the attention layer by projections, similar to MMDiT [11], enabling the model to effectively capture the spatial information of the prompt features. Specifically, we first modulate the combined conditional features $\mathbf{F}_{\text{Cond}}$ through AdaLN to extract time-dependent information. Then, we generate the key , query, and value features using a single linear layer and combine them with the corresponding features from the input tokens through element-wise addition. Using these conditioned features, we apply cosine attention [20, 37] with point-wise normalization [18] to stabilize the training process [20].

S.2. Training details of P-DiT

S.2.1. CM Parameterization

The CM-based model f_θ aims to approximate the consistency function $f(\cdot,\cdot)$, which satisfies $f(\mathbf{x}_t,\sigma_t)=\mathbf{x}_0$. Therefore, it must adhere to the boundary condition $f(\mathbf{x}_0,\sigma_0)=\mathbf{x}_0$. To ensure this, we follow the parameterization used in EDM [19] and CM [58], defining the model as follows:

$$f_\theta(\mathbf{x}_t,\sigma_t)=c_{\text{skip}}(\sigma_t)\mathbf{x_t}+c_{\text{out}}(\sigma_t)F_\theta(c_{\text{in}}(\sigma_t)\mathbf{x}_t,\sigma_t), \quad \text{(S2)}$$

where F_θ is a free-form neural network, such as P-DiT, and c_{in}, c_{out}, and c_{skip} control the scaling of input, output magnitudes, and the skip connection, respectively. These can be expressed as:

$$c_{\text{in}}(\sigma_t)=\frac{1}{\sqrt{\sigma_{\text{data}}^2+\sigma_t^2}}, \quad c_{\text{skip}}(\sigma_t)=\frac{\sigma_{\text{data}}^2}{(\sigma_t-\sigma_0)^2+\sigma_{\text{data}}^2},$$
$$c_{\text{out}}(\sigma_t)=\frac{\sigma_{\text{data}}(\sigma_t-\sigma_0)}{\sqrt{\sigma_{\text{data}}^2+\sigma_t^2}}. \quad \text{(S3)}$$

These formulations meet the boundary conditions where $c_{\text{skip}}(\sigma_0)=1$ and $c_{\text{out}}(\sigma_0)=0$.

S.2.2. CM Hyperparameter

We detail the hyperparameters for training P-DiT, adopting most from iCT [54] for CM.

Discretization Curriculum. The discretization curriculum is designed to enhance CM training by systematically increasing the number of discretization timesteps T in Eq. (1), improving the quality of generated samples. The discretization curriculum $\mathcal{C}(k)$ is defined as follows:

$$\mathcal{C}(k)=\min(s_0 2^{\frac{k}{K'}},s_1)+1, \quad \text{where } K'=\left\lfloor \frac{K}{\log_2 \frac{s_1}{s_0}+1} \right\rfloor. \quad \text{(S4)}$$

In this context, k ranges over $\{0,1,\ldots,K\}$, where K denotes the total number of training iterations. The parameters s_0 and s_1 refer to the minimum and maximum discretization steps, respectively. While iCT [54] uses $s_0=10$ and $s_1=1280$, we empirically found that setting maximum number of discretization steps s_1 to 160 is sufficient to produce competitive performance.

Noise Schedule. The noise schedule plays a crucial role in determining the sampling of noise levels during the CM

training, significantly influencing the quality of the generated samples. To define the noise schedule, we first discretize the noise level as follows: $\sigma_{\text{min}} = \sigma_0 < \sigma_1 < \cdots < \sigma_T = \sigma_{\text{max}}$ where $\sigma_{\text{min}} = 0.002,\ \sigma_{\text{max}} = 80$. As in [19, 54, 58], we set σ_t as:

$$\sigma_t = \left(\sigma_{\text{min}}^{1/\tau} + \frac{t-1}{\mathcal{C}(k)-1}\left(\sigma_{\text{max}}^{1/\tau} - \sigma_{\text{min}}^{1/\tau}\right)\right)^{\tau}, \tag{S5}$$

where $t \in \{1, 2, \ldots, \mathcal{C}(k)\}$. We use $\tau = 7$ and τ controls for the step length between noise levels σ_t and σ_{t+1}. As τ increases, the step length at lower noise levels decreases, allowing the model to better capture high-frequency details.

Additionally, we utilize a lognormal distribution for noise level sampling, which reduces the emphasis on higher noise levels and mitigates the accumulation of errors in CT loss at lower noise levels. The noise sampling schedule is defined as:

$$\sigma_t, \text{ where } t \sim p(t), \text{ and}$$
$$p(t) \propto \text{erf}\left(\frac{\log(\sigma_{t+1}) - P_{\text{mean}}}{\sqrt{2}P_{\text{std}}}\right) - \text{erf}\left(\frac{\log(\sigma_t) - P_{\text{mean}}}{\sqrt{2}P_{\text{std}}}\right), \tag{S6}$$

where erf indicates error function, and $P_{\text{mean}}, P_{\text{std}}$ determine the shape of log distribution. We choose $P_{\text{mean}} = -1.1,\ P_{\text{std}} = 2.0$, following iCT [54].

Loss Function. In Eq. (4), we use the pseudo-Huber loss [54] as the distance function $d(\cdot)$. The pseudo-Huber loss transitions between $\mathcal{L}_1$ and $\mathcal{L}_2$ metrics and is more robust to outliers than the $\mathcal{L}_2$ metric. The pseudo-Huber loss is defined as:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\|\mathbf{x} - \mathbf{y}\|_2^2 + c^2} - c, \tag{S7}$$

where $c = 0.00054\sqrt{m}$, and m indicates data dimensionality.

Loss Weighting. The weighting function $\lambda(\cdot)$ in Eq. (4) modulates the significance of CT losses across varying noise levels during training. Following [54], we set the weighting function $\lambda(\cdot)$ as follows:

$$\lambda(\sigma_t) = \frac{1}{\sigma_{t+1} - \sigma_t}. \tag{S8}$$

By assigning lower weights to higher noise levels, the weighting function ensures that the model focuses on learning from lower noise levels, where the data is more distinct and informative. This strategy improves sample quality by reducing the influence of errors associated with higher noise levels, thereby enhancing the overall performance of consistency models.

S.2.3. Latent Code Normalization

The formulation of EDM [19] assumes that the mean and standard deviation of the training data are zero and σ_{data}, respectively, as stated in Eq. (S2). Following the approach in [20], we also normalize the encoded latent codes using precomputed statistics from the training data. Specifically, we first calculate the channel-wise mean and standard deviation of the latent codes from the training dataset. Then, we subtract the input latent code by the precomputed mean to achieve a mean of zero, and divide it by the precomputed standard deviation, followed by multiplying by σ_{data}, to set the standard deviation to σ_{data}. When the latent codes are generated, we reverse this procedure before transforming them back to the image space via the Decoder $\mathcal{D}$.

S.2.4. P-DiT Hyperparamters

Following the approach in [54, 58], we update P-DiT parameters using an exponential moving average with a decay rate of 0.9999 to stabilize the training process. P-DiT's model hyperparameters are based on DiT-S [42], except for the number of blocks ($B = 8$), to improve efficiency. The input noised latent is tokenized with a patch size of 1, allowing for finer noise information to be embedded in the latent code. To mitigate overfitting, we apply a dropout rate of 0.1 to the pointwise feed-forward layer and add minor noise and apply downsampling operation with a factor of 2 to the conditional clean images [58].

S.3. Model Size and Inference Speed of PNG

Our PNG consists of two sub-models, PAE with 14.9M parameters and P-DiT with 29.1M, yielding roughly 44M parameters in total. This total is comparable to NeCA-W [12], whose camera-specific model for each of the five manufacturers in the SIDD dataset contains 40.5M parameters. To train both PAE and P-DiT, we utilize four NVIDIA A6000 GPUs. Following the approach in [19, 20, 54, 58], we train P-DiT using mixed-precision, which reduces both training time and memory consumption.

To evaluate the computational complexity of our two-stage framework, we compare the inference time at different image resolutions with two other SOTA models, NeCA-W [12] and NAFlow [21]. Specifically, we measure the number of images that each model can process per second on a single A6000 GPU. As shown in Tab. S1, with a resolution of 256×256, PNG can generate 57 images per second, which is about 4.38 times faster than NAFlow. At a 512×512 resolution, PNG generates 21 images per second,

Table S1. Inference speed comparison between NeCA-W, NAFlow, and PNG.

Resolution	Inference Time (image / second)		
	NeCA-W	NAFlow	**Ours**
256×256	150	13	**57**
512×512	38	8	**21**
1024×1024	9	4	**5**

Table S2. Effect of conditioning features on different components in P-DiT. The best results are shown in **bold**.

Conditioning		Generation	
Timestep	Attention	KLD↓	AKLD↓
✗	✗	0.5661	0.4132
✓	✗	0.0287	0.1291
✓	✓	**0.0261**	**0.1108**

Table S3. Effect of GPB and LPB on SIDD validation noisy image reconstruction. The best results are shown in **bold**.

Combination		Reconstruction	
GPB	LPB	PSNR↑	SSIM↑
✗	✗	37.58	0.9800
✓	✗	46.30	0.9982
✓	✓	**46.54**	**0.9983**

2.625 times faster than NAFlow. At the highest resolution of 1024×1024, PNG still performs 1.25 times faster than NAFlow, producing 5 images per second, demonstrating its practicality at high image resolutions. Even though NeCA-W shows the highest speed at all resolutions, the inference time difference ratio between NeCA-W and PNG decreases as the resolution increases, from 2.63 to 1.8. Hence, our method demonstrates strong performance while maintaining reasonable computational complexity at high resolutions compared to recent SOTA models.

S.4. Additional Ablation Studies

S.4.1. Effect of Conditioning Features in P-DiT

P-DiT generates latent codes that embed noise information through conditional features. Specifically, in Fig. S1 (a), we first add the conditional feature $\mathbf{F}_{\text{Cond}}$ to the timestep embedding, and in Fig. S1 (b), we condition the conditional feature through prompt attention.

In Tab. S2, we investigate the effect of conditional features on two different components: the timestep embedding and attention. P-DiT without any conditional features fails to capture the target noise information, resulting in inferior performance in terms of KLD and AKLD. The model that conditions the conditional features on both components demonstrates superior performance compared to the model that only conditions them on the timestep embedding. This highlights that utilizing spatial information through the proposed prompt attention allows the model to fully exploit the conditional features.

Table S4. KLD score depending on different number of blocks B.

# Blocks	KLD↓
$B = 4$	0.0350
$B = 6$	0.0345
$B = 8$	**0.0261**

S.4.2. Effect of GPB and LPB on Noisy Image Reconstruction

In Tab. S3, we also assess the influence of both prompt blocks on noisy image reconstruction. Incorporating GPB and LPB enhances the fidelity of the reconstructed images, as the prompt features encompass meaningful information about noise characteristics at different scales (see Fig. 3). Therefore, it is essential to utilize both prompt features to enhance the quality of the reconstructed images. In particular, we pass the latent directly from the Prompt Encoder $\mathcal{E}$ to the Decoder $\mathcal{D}$, without using P-DiT for this experiment.

S.4.3. Effect of Number of P-DiT Blocks

As shown in Tab. S4, we measure the KLD scores across different numbers of blocks B in the P-DiT to evaluate their impact on noise generation, with B representing the hierarchical levels used in our P-DiT. The results indicate that increasing the number of blocks improves the quality of noise generation. Even with half the number of blocks, P-DiT yet achieves a moderate KLD score, with only a slight difference from the final version (B=8), suggesting an opportunity to optimize the trade-off between model complexity and performance.

S.5. Further Analysis of the Generalization Performance

As shown in Tab. 3, denoising networks trained with our synthesized dataset exhibit strong robustness to unseen noise. We further conduct experiments to assess the generalization capability of our approach on external datasets. The results demonstrate that expanding the training dataset through synthetic image generation helps mitigate overfitting to the training distribution and leads to improved performance on external dataset.

For the experiments, we select 15,000 non-overlapping patches from the SIDD training set to generate synthetic noisy images to train the DnCNN network. This setup ensures that each noisy sample is unique, providing a controlled environment to systematically analyze the impact of the size of the dataset on generalization.

In Tab. S5, we organize the training data into three distinct groups: ×1, ×2, and ×4, where each factor indicates the number of synthetic noisy samples generated per clean image using PNG. Importantly, performance on the in-distribution dataset (SIDD+) remains consistent across

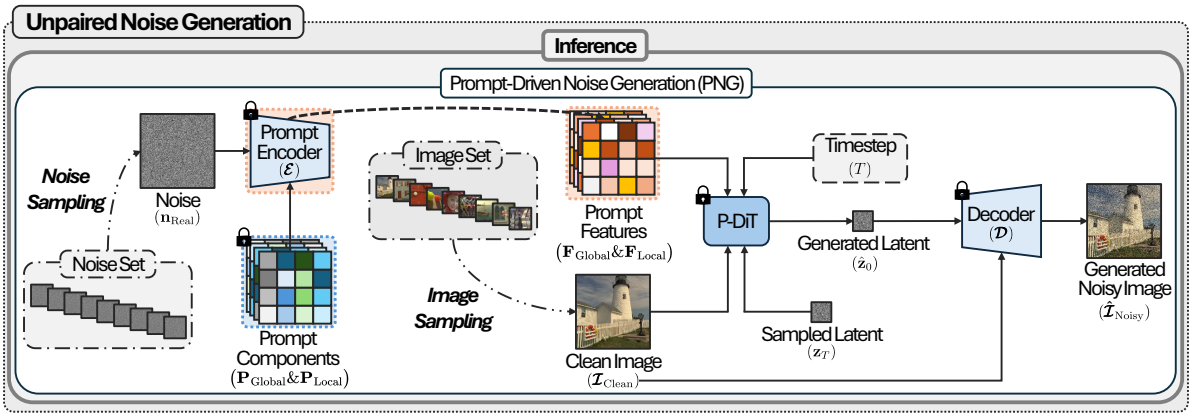

Figure S2. Overview of unpaired noise generation process.

Table S5. Quantitative results on denoising generalization performance depending on the size of the synthesized dataset using PNG. The multiplication sign (×) indicates the scaling applied to the original number of patches. The best and second-best results are denoted as **bold** and underline, respectively.

# Samples	PolyU		Nam		SIDD Validation		SIDD+		Average	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
×1	37.40	**0.9569**	37.29	0.9578	36.55	0.8887	35.72	0.8997	36.74	0.9258
×2	37.61	0.9546	37.92	**0.9580**	36.95	0.8838	35.87	**0.9078**	37.09	0.9260
×4	**37.72**	0.9542	**37.94**	0.9578	**37.27**	**0.8955**	**36.00**	0.9022	**37.23**	**0.9274**

all groups. However, the denoising network's robustness on external datasets improves as the number of training samples increases, with the ×4 group achieving the highest average performance. These findings suggest that increasing noise diversity through synthetic data generation effectively reduces overfitting and highlights the efficacy of the PNG approach.

S.6. Unpaired Noise Generation

In Tab. 1, we use paired noisy image $\mathcal{I}_{\text{Noisy}}$ and clean image $\mathcal{I}_{\text{Clean}}$ from test datasets during inference to compute $\mathbf{n}_{\text{Real}} = \mathcal{I}_{\text{Noisy}} - \mathcal{I}_{\text{Clean}}$, following the experimental setup used in previous works [2, 12, 21, 28]. To broaden PNG's applicability to cases where only some noisy-clean pairs are available for training alongside extra clean-only images, we also evaluate it in an unpaired setting. In this scenario, as illustrated in Fig. S2, noise images $\mathbf{n}_{\text{Real}}$ are randomly sampled from the training dataset (*e.g.* SIDD) and used to synthesize noisy images $\hat{\mathcal{I}}_{\text{Noisy}}$ conditioned on new, unpaired clean images $\mathcal{I}_{\text{Clean}}$ that are randomly sampled from the external dataset (*e.g.* ImageNet [9]), during the inference.

S.6.1. Generated Noise Quality Assessment.

Experiment Settings. In Tab. S6, we evaluate noise quality under the same unpaired setting as described above, using NAFlow, NeCA-W, and PNG. All models are trained on the paired SIDD training dataset, and NeCA-W and NAFlow are evaluated using their official weights as described in Sec. 4.2 of the main manuscript. To evaluate how well noise generation networks model each specific noise domains with SIDD validation, we categorize results by different five smartphone types. Notably, in this table, we use metadata only for categorization; however, in real-world applications, our method can synthesize a dataset following the trained noise distribution without relying on any metadata.

Noise Quality Results. Tab. S6 shows that, while the unpaired setting introduces a slight performance drop compared to the paired setting, PNG still achieves comparable or superior noise quality metrics (KLD and AKLD) compared to other baseline methods such as NeCA-W and NAFlow. Furthermore, PNG achieves even superior performance compared to NeCA-W in the paired setting (KLD: 0.0342, AKLD: 0.1436), despite utilizing limited noise information. This flexibility underscores the adaptability of PNG, which synthesizes realistic noisy images without requiring paired clean–noisy images.

This makes PNG also suitable for scenarios where paired datasets are unavailable. Additionally, in the paired setting, PNG demonstrates strong performance without the need for explicitly defined metadata such as ISO or camera type. These results further support the practicality of PNG across a wide range of use cases, regardless of whether paired or unpaired data are available.

S.6.2. Denoising Quality Assessment.

Experiment Settings. To further assess the quality of generated noise, we train the DnCNN model only using noisy

Table S6. Quantitative results for the noise generation performance on SIDD validation set under unpaired settings. The results are reported using KLD↓ and AKLD↓, with the best and second-best results highlighted in **bold** and underline, respectively. † indicates that the model is evaluated under an unpaired setting.

Methods	G4		GP		IP		N6		S6		Average	
	KLD↓	AKLD↓	KLD↓	AKLD↓	KLD↓	AKLD↓	KLD↓	AKLD↓	KLD↓	AKLD↓	KLD↓	AKLD↓
NeCA-W†	0.0513	0.1765	0.0487	0.1819	0.0258	0.1609	0.0603	0.1496	0.0581	0.2205	0.0489	0.1779
NAFlow†	0.4482	1.6918	0.2260	1.0135	0.3267	1.1612	0.2200	0.8725	0.1462	0.5456	0.2734	1.0569
Ours†	**0.0350**	**0.1567**	**0.0312**	**0.1553**	**0.0222**	**0.1392**	**0.0291**	**0.1229**	**0.0204**	**0.1326**	**0.0276**	**0.1413**

Table S7. Quantitative results on denoising generalization performance trained with synthetic noisy-clean pairs generated in an unpaired manner. The best and second-best results are denoted as **bold** and underline, respectively. † indicates that noise is generated under an unpaired setting.

Methods	PolyU		Nam		SIDD Validation		SIDD+		Average	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
NeCA-W†	37.39	0.9555	37.13	0.9495	32.86	0.7490	33.92	0.8342	35.33	0.8720
NAFlow†	32.40	0.9157	31.73	0.9381	30.89	0.7776	31.60	0.7953	31.66	0.8567
Ours†	**38.00**	**0.9618**	**38.37**	**0.9633**	**32.93**	**0.7803**	**34.25**	**0.8557**	**35.89**	**0.8903**

images generated via an unpaired approach. To ensure a rigorous assessment, we randomly select noise from 15,000 non-overlapping patches from the SIDD training set and randomly paired these with clean images from the ImageNet [9] validation set, which contains 50,000 images from various scenes. Subsequently, 50,000 synthetic noisy-clean image pairs are generated from the ImageNet dataset.

Denoising Results. As shown in Tab. S7, the DnCNN model trained on our method outperforms SOTA methods, including NeCA-W and NAFlow, across all real-world datasets. These results confirm the superior performance of unpaired noise generation with PNG. Specifically, these denoising results are consistent with the unpaired noise generation results in Table S6, where NAFlow produces the lowest denoising performance, which can be attributed to its lower-quality unpaired noise generation. NeCA-W achieves the second-best denoising results, but our method surpasses NeCA-W, with average improvements of 0.56 in PSNR and 0.0183 in SSIM. In summary, compared to recent SOTA methods such as NeCA-W and NAFlow, our approach is capable of generating high-quality noise in both paired and unpaired scenarios, demonstrating its practicality for real-world applications.

S.7. Qualitative Results of Noise Generation

In Fig. S3, we provide additional visualizations of generated noise, comparing our method (PNG) with other approaches: C2N [17], NeCA-W [12], and NAFlow [21].

S.8. Qualitative Results of Denoising Performance

In Fig. S4, we provide additional visualizations of denoising results, where the denoising network is trained on synthetic datasets. We compare our method (PNG) with other approaches, including C2N, NeCA-W, and NAFlow.

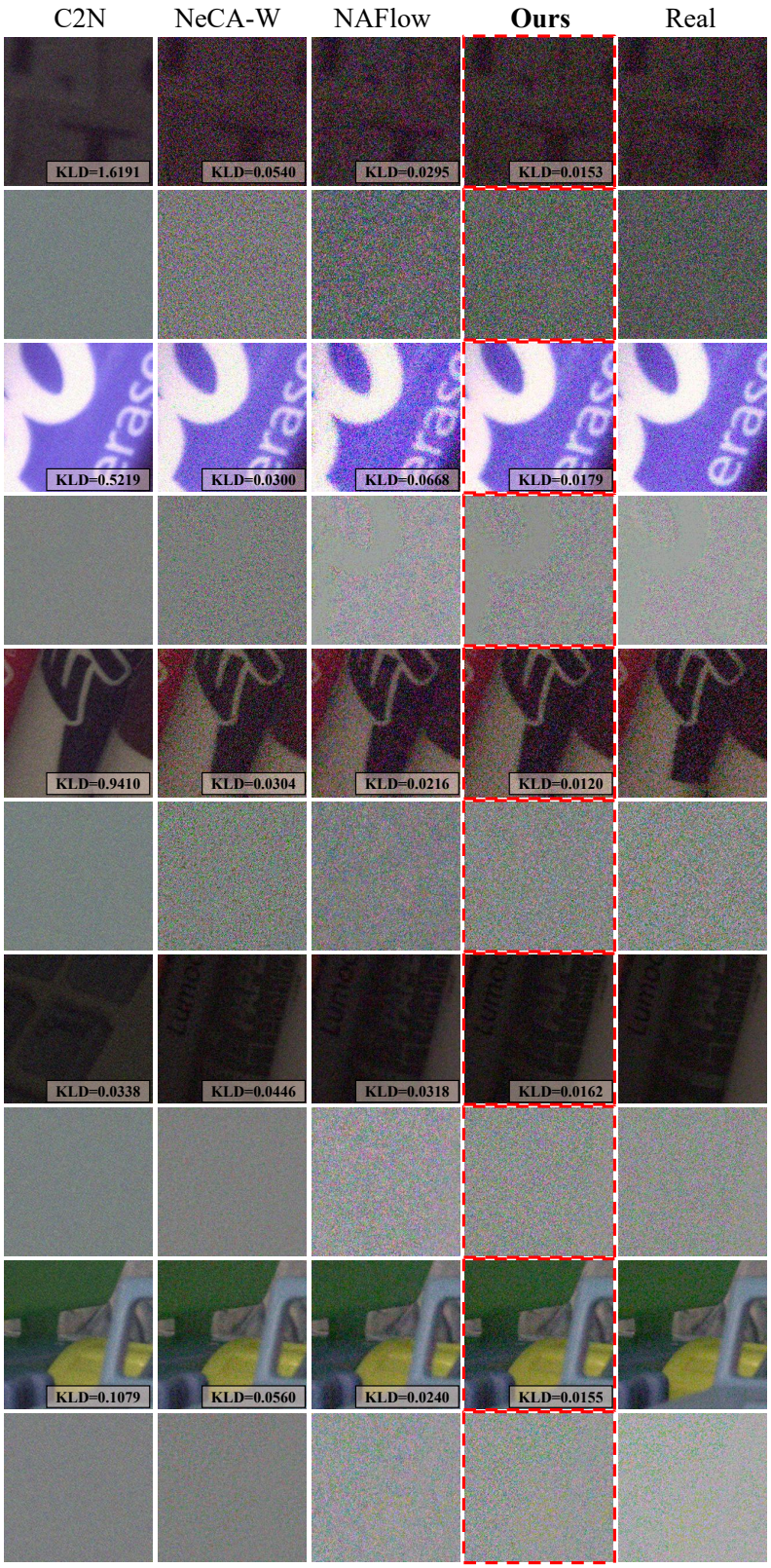

Figure S3. Visualization of synthetic noisy images on the SIDD validation set. From left to right: C2N, NeCA-W, NAFlow, Ours (PNG), and real noisy images.

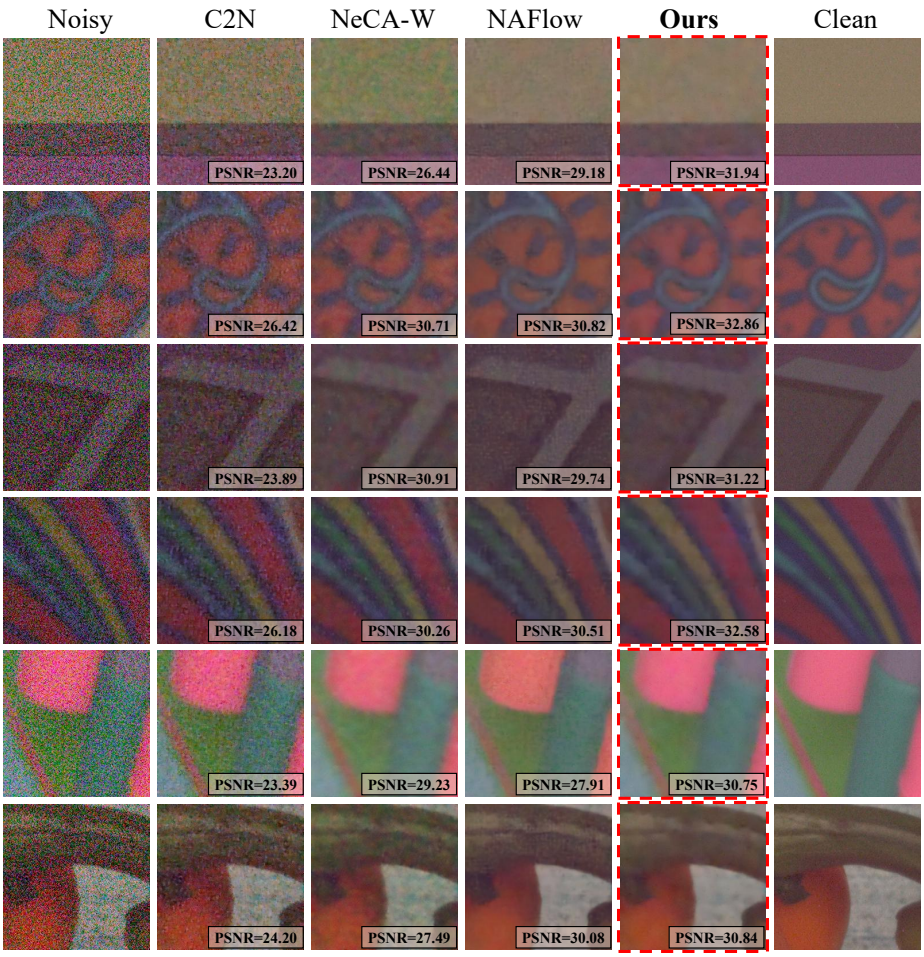

Figure S4. Visual comparison on denoising results with PSNR↑ on SIDD validation set from DnCNN trained on each method.