# NoiseTransfer: Image Noise Generation with Contrastive Embeddings

Seunghwan Lee and Tae Hyun Kim

Dept. of Computer Science, Hanyang University, Seoul, Korea
`{seunghwanlee,taehyunkim}@hanyang.ac.kr`

## 1 Network Configurations

### 1.1 Generator

Fig. 1 shows the architecture of our U-Net based generator. We adopt the spatial feature transform (SFT) layer [6] to efficiently translate features based on the noise embeddings. Moreover, we add randomness in $\beta$ by introducing Gaussian random noise $z$ to generate differently realized noise for each forward. We set the number of feature channels as 64, and we double the number of feature channels at each down-convolution. LeakyRelu function is used for activation between convolution layers.

### 1.2 Discriminator

Fig. 2 shows the architecture of our discriminator. Particularly, our discriminator has three levels of output with different receptive fields to benefit from various levels of contextual information from noisy image. For outputs $D_{gan}(\cdot), m_{noise}(\cdot)$, and $m_{gan}(\cdot)$, the final outputs consist of the set of three levels of each output (*e.g.*, $D_{gan}(\cdot) = \{D_{gan1}(\cdot), D_{gan2(\cdot)}$, and $D_{gan3}(\cdot)\}$), and loss is computed for each set. We adopt additional a two-layer multi-layer perceptron (MLP) to produce $D_{noise}(\cdot)$ from the set of $\{D_{noise1}(\cdot)$, $D_{noise2}(\cdot)$, and $D_{noise3}(\cdot)\}$, and ensure bounded values of noise embeddings by applying hyperbolic tangent function. Instance normalization [5] is utilized in the discriminator, however we derive the scaling and shifting parameters from the noise embeddings for $D_{gan}$. Spectral normalization [4] is used for stable training for both generator and discriminator, and LeakyRelu function is used for activation.

## 2 Limitation

We analyze the limitation of our model. In fact, we do not guarantee that our model generates completely different types of noise that are not included in the training set. We try to transfer and synthesize the unseen noise type during training, salt and pepper noise for example, as shown in Fig. 3. Our model fails to generate a new noisy image with salt and pepper noise due to the mismatch of noise distributions of the training and inference, which is a common problem widely existing in data-driven deep learning methods.

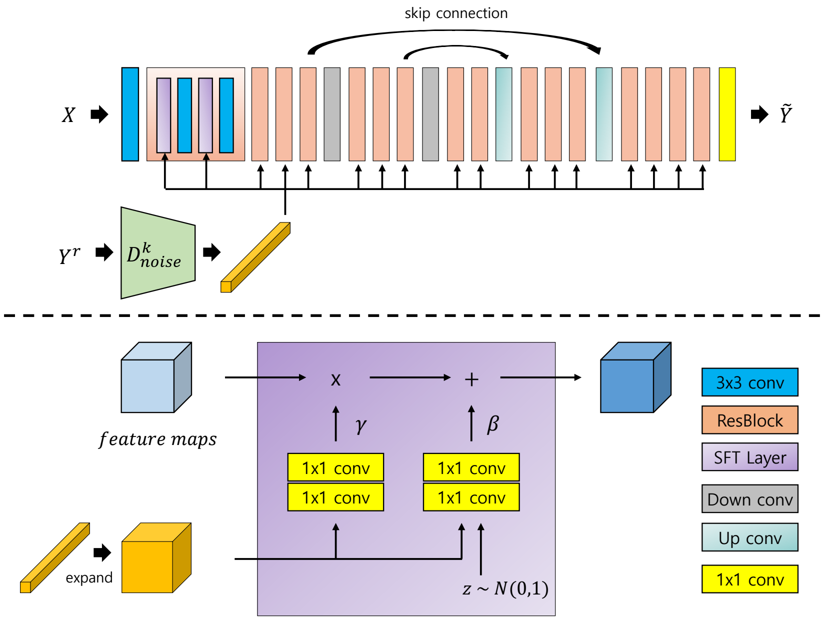

Fig. 1: Architecture of our U-Net based generator. The noise embeddings are expanded to match the shape of input feature maps in SFT layer [6]. Gaussian random noise $z$ is concatenated with the noise embeddings for $\beta$ to generate spatially random noise.

## 3 Visual Results

We provide more experimental results qualitatively. Fig. 4 and Fig. 5 show examples of synthetic noise generation for variable noise levels as stated in the manuscript (Sec. 4.1). Visual comparisons for the real-world noise generation are presented from Fig. 6 to Fig. 8. Fig. 9 and Fig. 10 show visual denoising results on the BSDS500 corrupted synthetically and on the SIDD validation set respectively.

## References

1. Bernardo Henz, Eduardo S. L. Gastal, M.M.O.: Synthesizing camera noise using generative adversarial networks. IEEE Transactions on Visualization and Computer Graphics **27**(3), 2123–2135 (2021). https://doi.org/10.1109/TVCG.2020.3012120
2. Chang, K.C., Wang, R., Lin, H.J., Liu, Y.L., Chen, C.P., Chang, Y.L., Chen, H.T.: Learning camera-aware noise models. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 343–358. Springer (2020)
3. Jang, G., Lee, W., Son, S., Lee, K.M.: C2n: Practical generative noise modeling for real-world denoising. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 2350–2359 (October 2021)

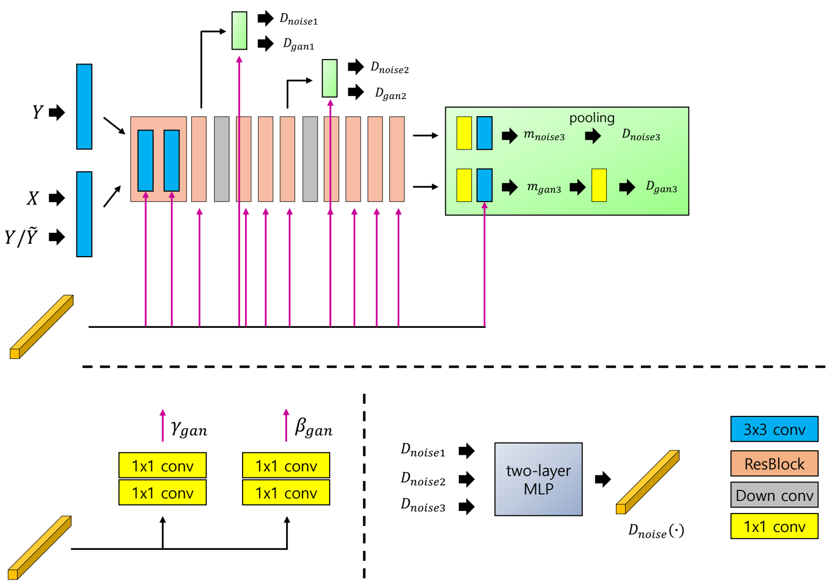

Fig. 2: Architecture of our discriminator. The discriminator has three levels of output with different receptive fields, and convolution weights in residual blocks are shared for $D_{noise}$ and $D_{gan}$. The scaling and shifting parameters for instance normalization in $D_{gan}$ are specially obtained from the noise embeddings, denoted by $\gamma_{gan}$ and $\beta_{gan}$ respectively. The noise embeddings are produced with the set of $\{D_{noise1}, D_{noise2}, \text{and } D_{noise3}\}$ followed by MLP projection.

4. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. In: Proceedings of the International Conference on Learning Representations (ICLR) (2018)
5. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)
6. Wang, X., Yu, K., Dong, C., Loy, C.C.: Recovering realistic texture in image super-resolution by deep spatial feature transform. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
7. Yue, Z., Zhao, Q., Zhang, L., Meng, D.: Dual adversarial network: Toward real-world noise removal and noise generation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 41–58. Springer (2020)
8. Zamir, S.W., Arora, A., Khan, S., Hayat, M., Khan, F.S., Yang, M.H., Shao, L.: Cycleisp: Real image restoration via improved data synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)

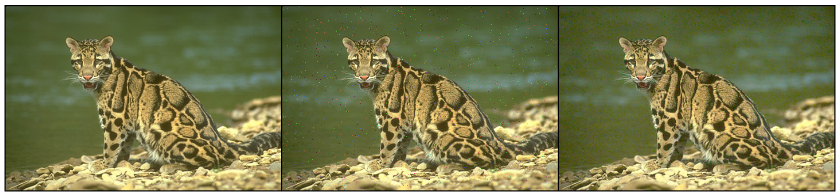

Fig. 3: Failure case of noise generation on the unseen noise type during training. The clean image is corrupted by salt and pepper noise. **Left to Right**: Clean, Noisy, Generated noisy image by our NoiseTransfer.

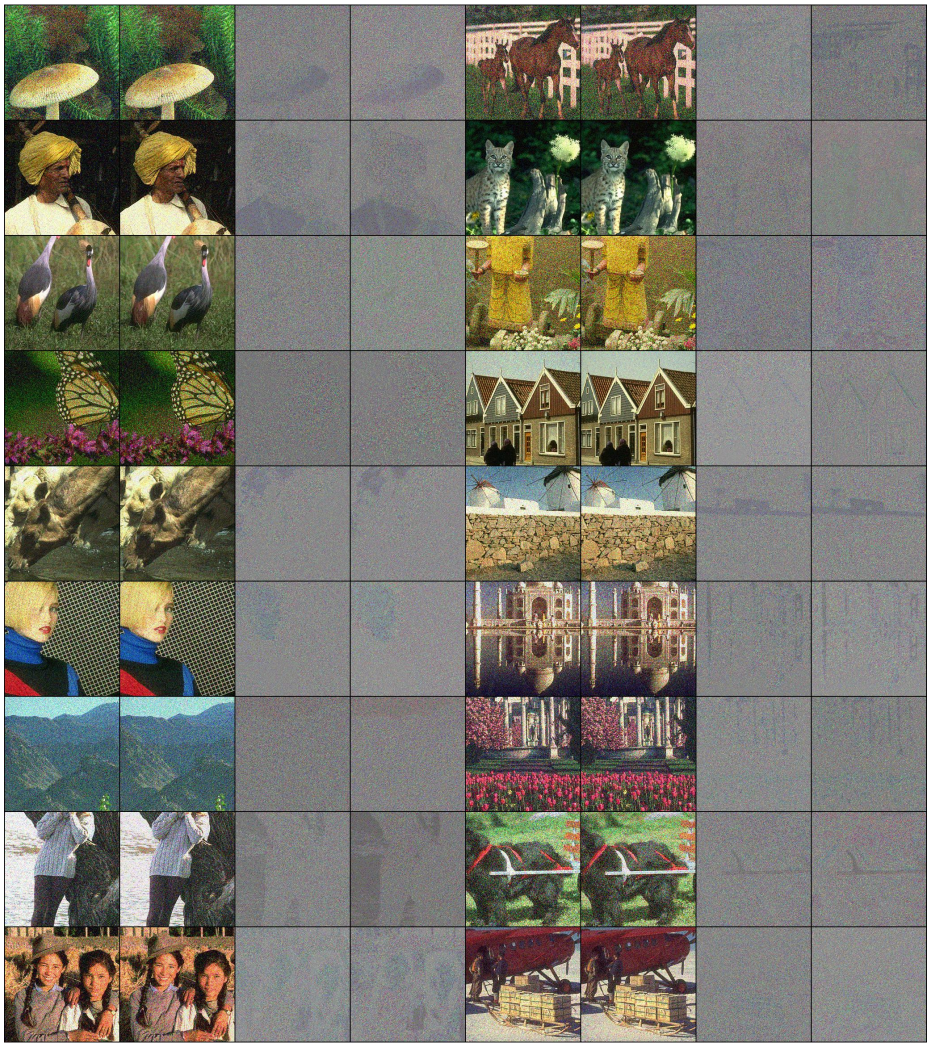

Fig. 4: Examples of synthetic noise generation. Gaussian (first three rows), Poisson (middle three rows), and Poisson-Gaussian noise (last three rows). **Left to Right for each instance**: Ground-truth noisy image, Generated noisy image by our NoiseTransfer, Ground-truth noise, Generated noise by NoiseTransfer (Ours).

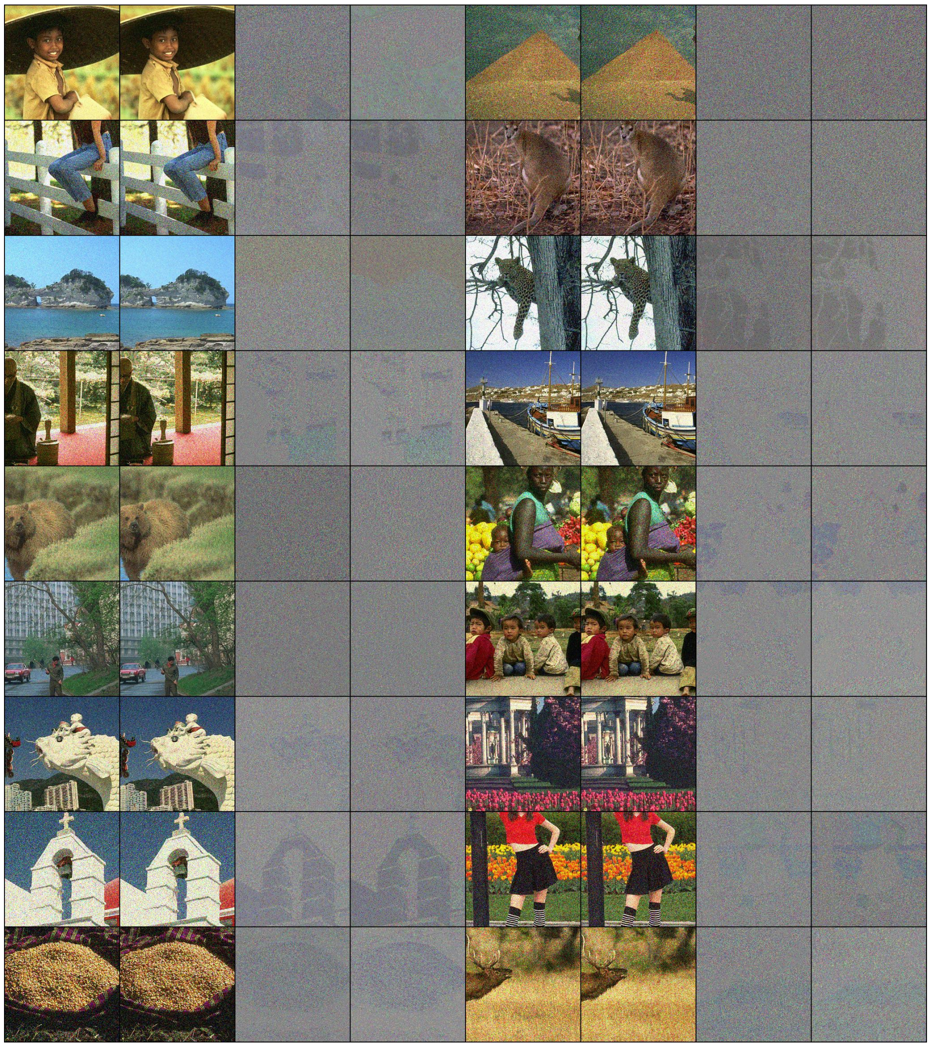

Fig. 5: Examples of synthetic noise generation. Gaussian (first three rows), Poisson (middle three rows), and Poisson-Gaussian noise (last three rows). **Left to Right for each instance**: Ground-truth noisy image, Generated noisy image by our NoiseTransfer, Ground-truth noise, Generated noise by NoiseTransfer (Ours).

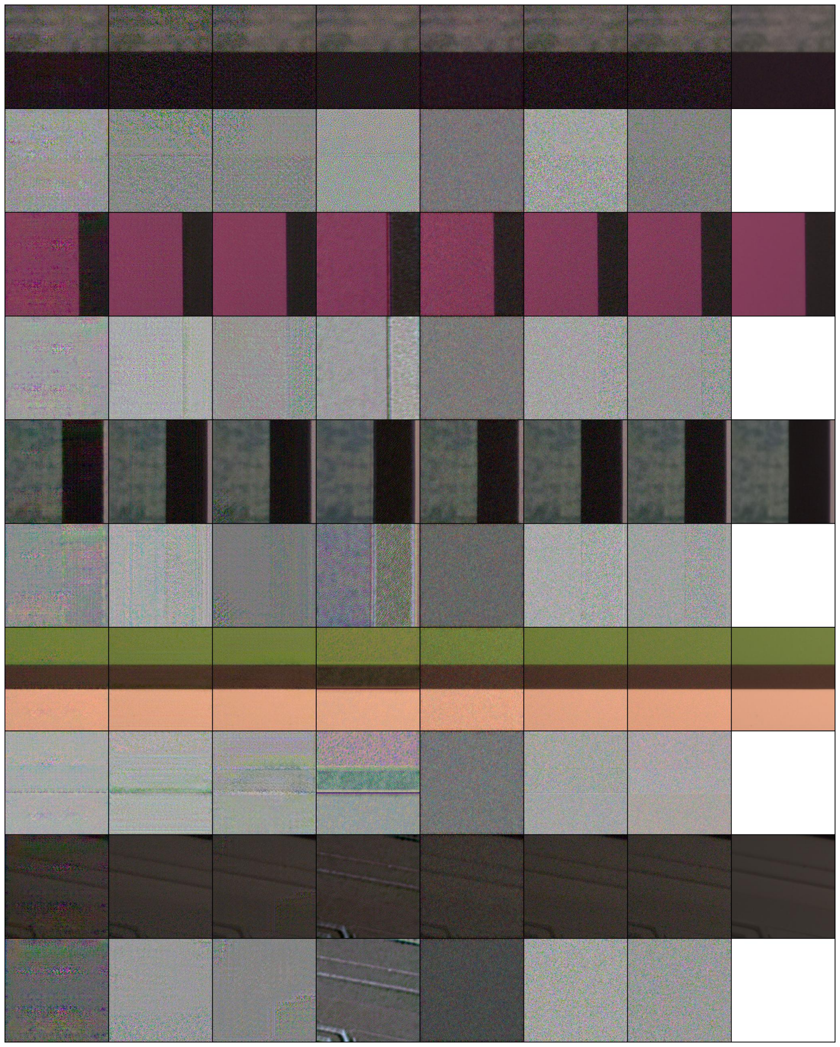

Fig. 6: Visual results of noise generation on the SIDD validation set. The corresponding noise is displayed below for each noisy image. **Left to Right**: CA-NoiseGAN [2], DANet [7], GDANet [7], NoiseGAN [1], C2N [3], NoiseTransfer (Ours), Noisy, Clean.

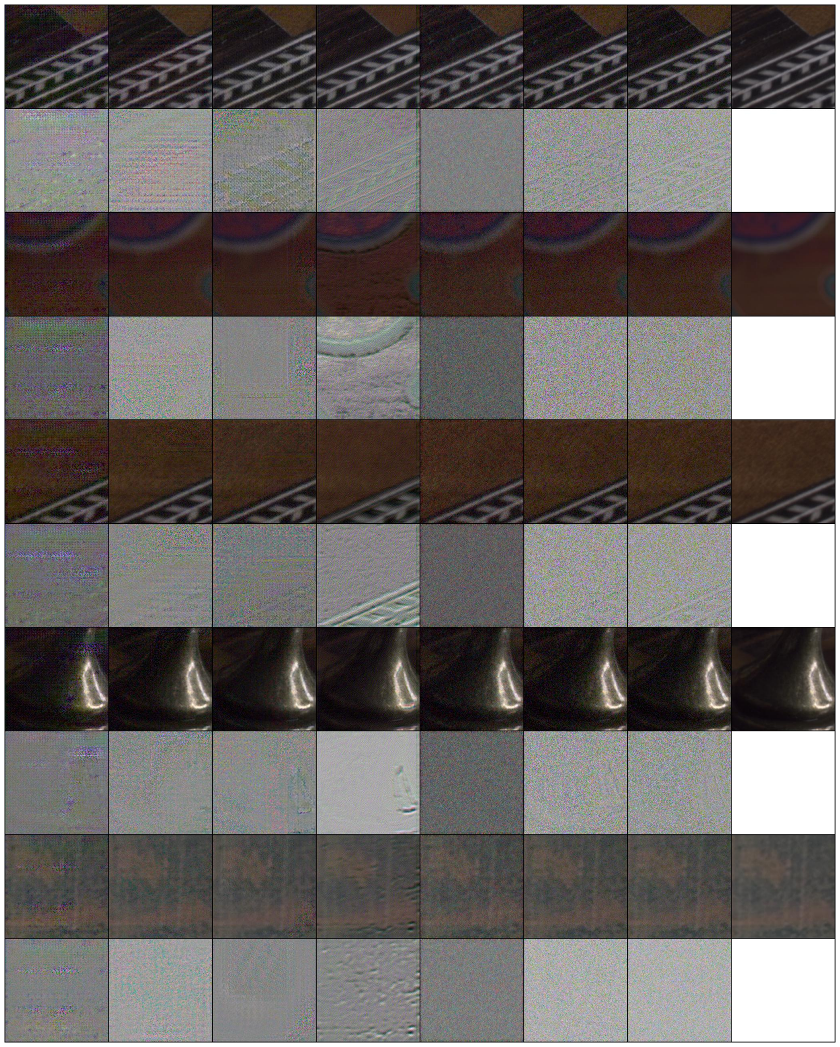

Fig. 7: Visual results of noise generation on the SIDD validation set. The corresponding noise is displayed below for each noisy image. **Left to Right**: CA-NoiseGAN [2], DANet [7], GDANet [7], NoiseGAN [1], C2N [3], NoiseTransfer (Ours), Noisy, Clean.

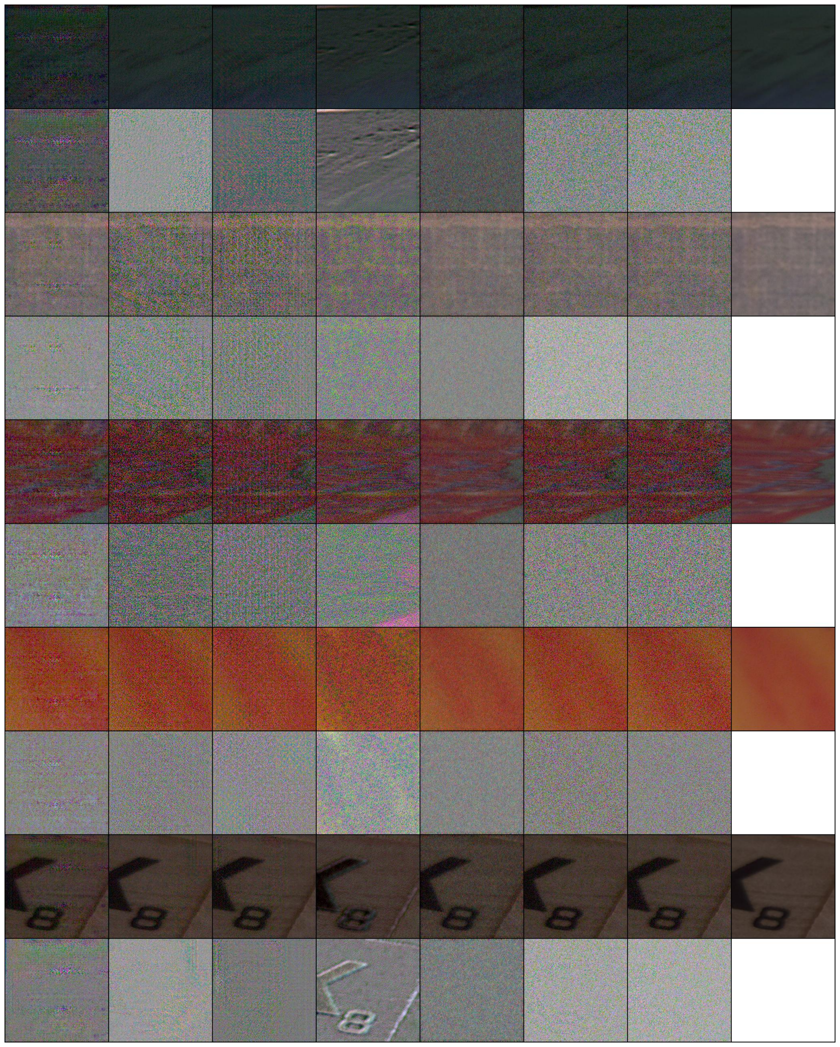

Fig. 8: Visual results of noise generation on the SIDD validation set. The corresponding noise is displayed below for each noisy image. **Left to Right**: CA-NoiseGAN [2], DANet [7], GDANet [7], NoiseGAN [1], C2N [3], NoiseTransfer (Ours), Noisy, Clean.

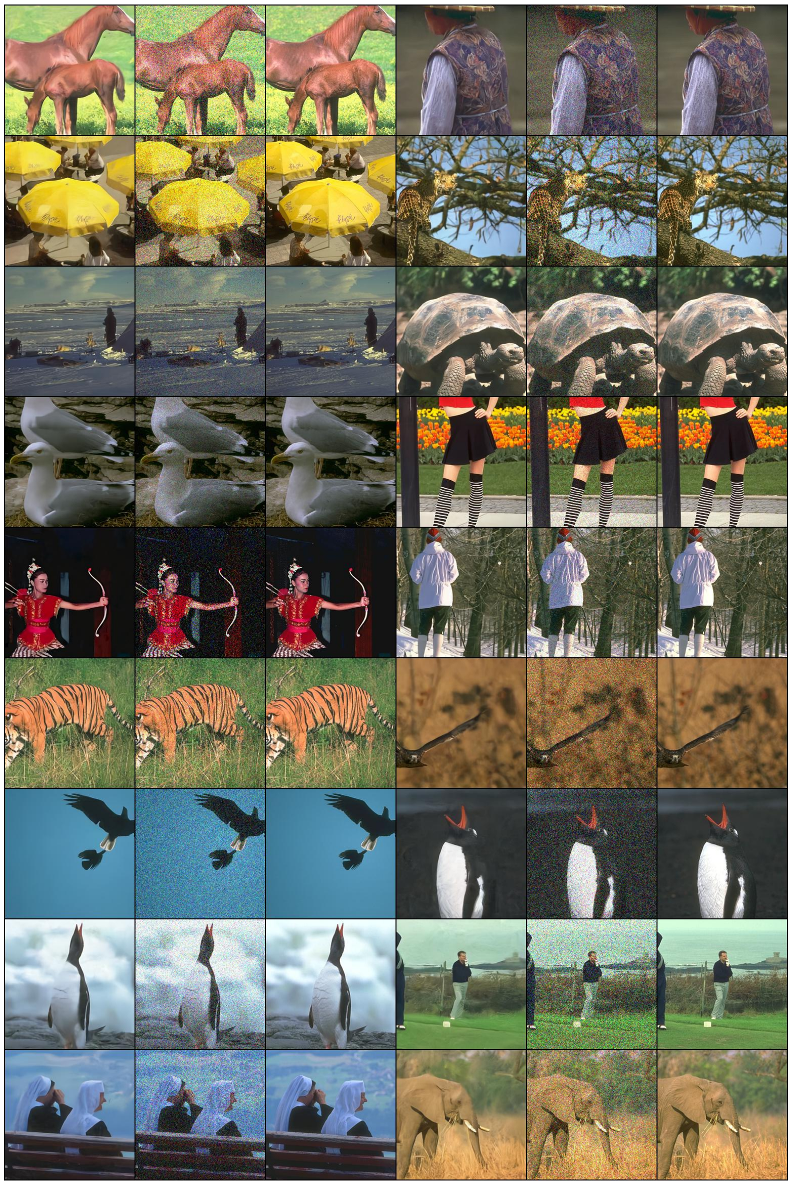

Fig. 9: Visual results for synthetic noise removal on the BSDS500. Gaussian (first three rows), Poisson (middle three rows), and Poisson-Gaussian noise (last three rows). **Left to Right for each instance**: RIDNet results trained by NoiseTransfer (Ours), and Noisy and Clean image.

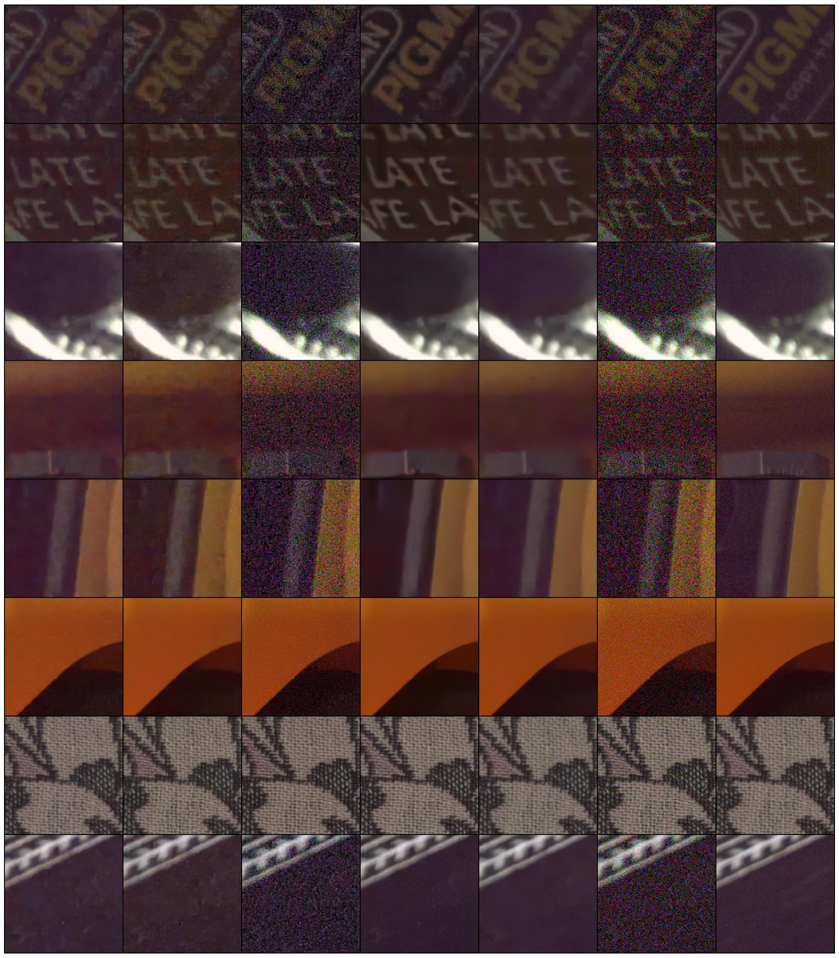

Fig. 10: Visual results for real noise removal on the SIDD validation set. **Left to Right**: RIDNet results trained by DANet [7], GDANet [7], C2N [3], CycleISP [8], and NoiseTransfer (Ours), and Noisy and Clean image.