SinIR: Efficient General Image Manipulation with Single Image Reconstruction
Supplementary Material

Jihyeong Yoo [1] **Qifeng Chen** [1]

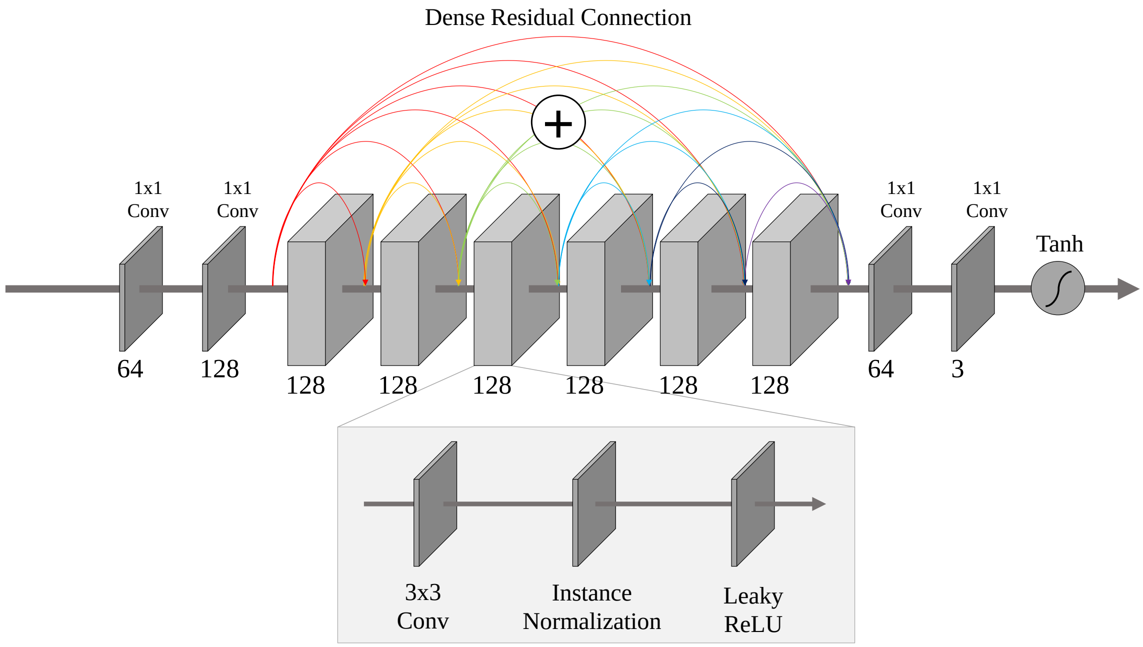

Figure 1. **Network Architecture of SinIR.** The numbers below each layer indicate the number of convolutional kernels. Here we assume that the input and the output are RGB images.

1. Network Architecture

All the networks at every scale use the same architecture described in Figure 1. Each network consists of two 1×1 convolutional layers which map RGB images to feature space, six convolutional blocks which are densely connected (Huang et al., 2017) with residual operation (He et al., 2016) (not concatenation), and two 1×1 convolutional layers which render features to RGB images. Each convolutional block has one 3×3 convolutional layer, an instance normalization layer (Ulyanov et al., 2016) and a LeakyReLU activation layer (negative slope = 0.2) (Maas et al., 2013). We do not use pooling or unpooling inside a network, and thus the inputs and the outputs of each network have the same spatial dimension. Reflection padding is used before 3×3 convolutional layer. Tanh function is used to obtain the final output.

[1]Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. Correspondence to: Qifeng Chen <cqf@ust.hk>.

Proceedings of the 38^{th} *International Conference on Machine Learning*, PMLR 139, 2021.

2. Additional Results

We show more results for all tasks presented in the original paper. The setting described in the paper is used for all results. While SinIR and SinGAN (Shaham et al., 2019) are internal methods trained on a single image, all dedicated methods used for comparisons are external methods trained on large-scale datasets.

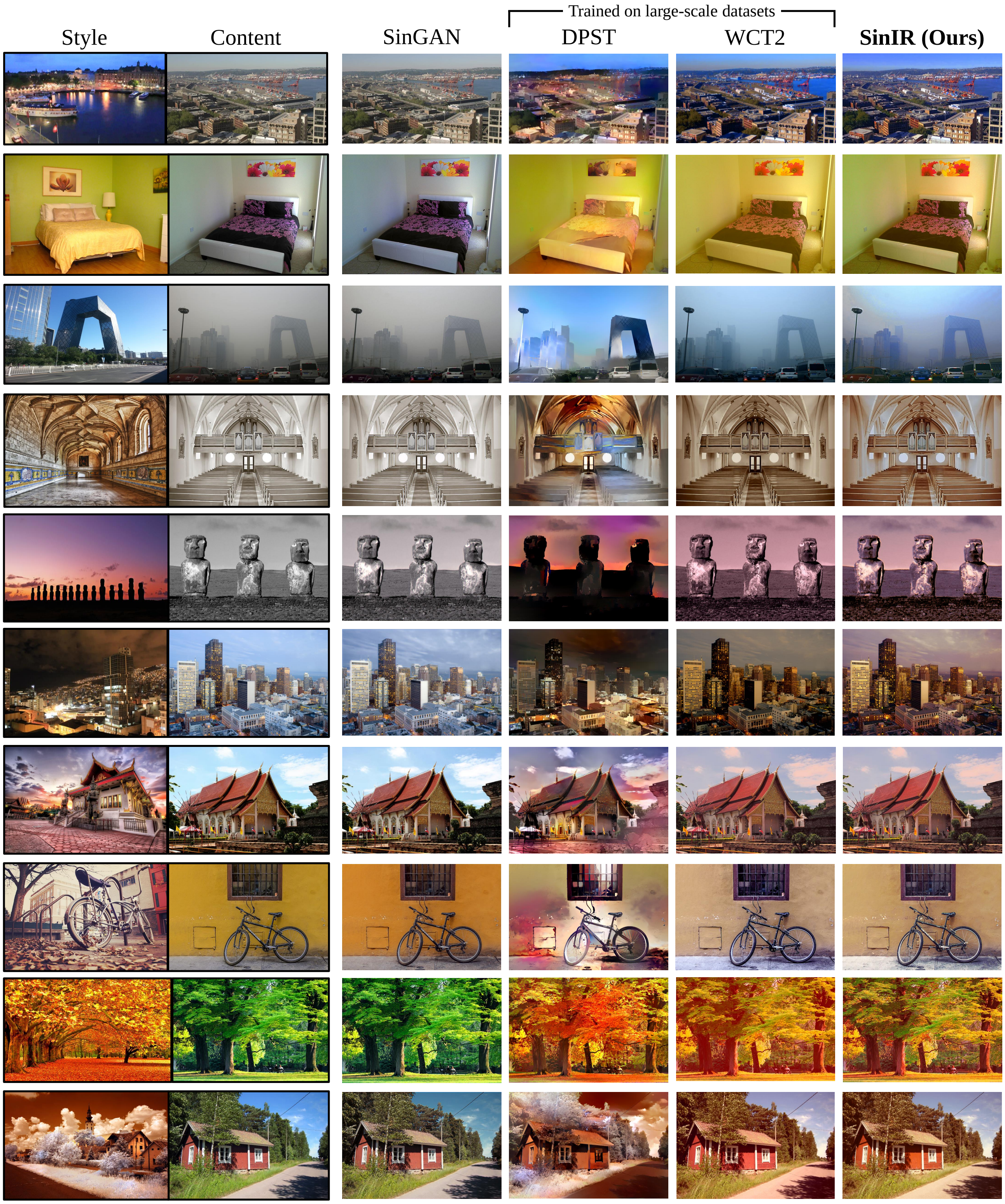

Figure 2. **Photo-realistic Style Transfer.** DPST (Luan et al., 2017) and WCT2 (Yoo et al., 2019) are dedicated methods.

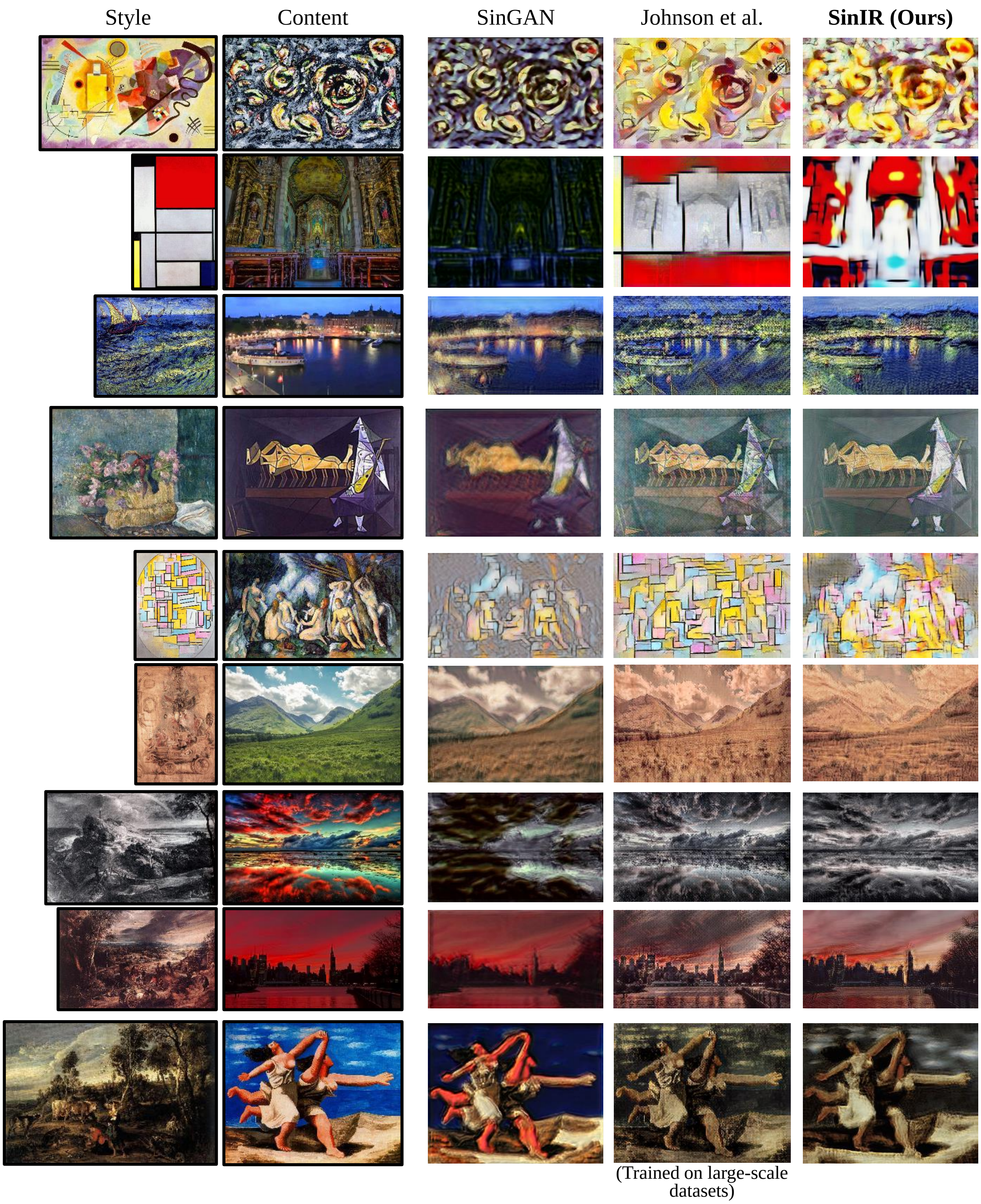

Figure 3. **Artistic Style Transfer.** (Johnson et al., 2016) is a dedicated method.

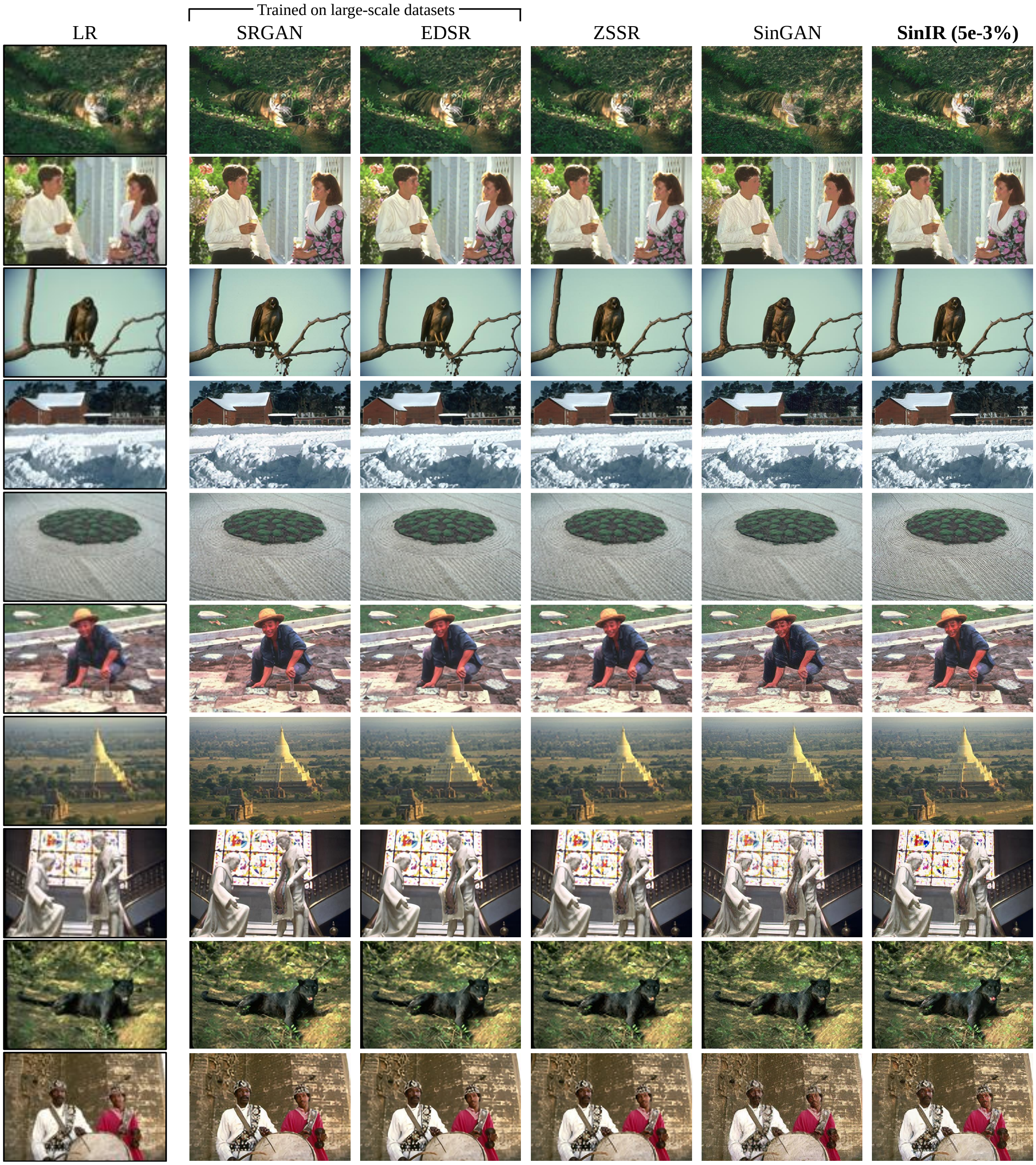

Figure 4. **Super-resolution.** SRGAN (Ledig et al., 2017), EDSR (Lim et al., 2017), ZSSR (Shocher et al., 2018) are dedicated methods. For simplicity, only results from SinIR with 5e-3% of random pixel shuffling are shown. (Please see super-resolution section in the original paper for details.)

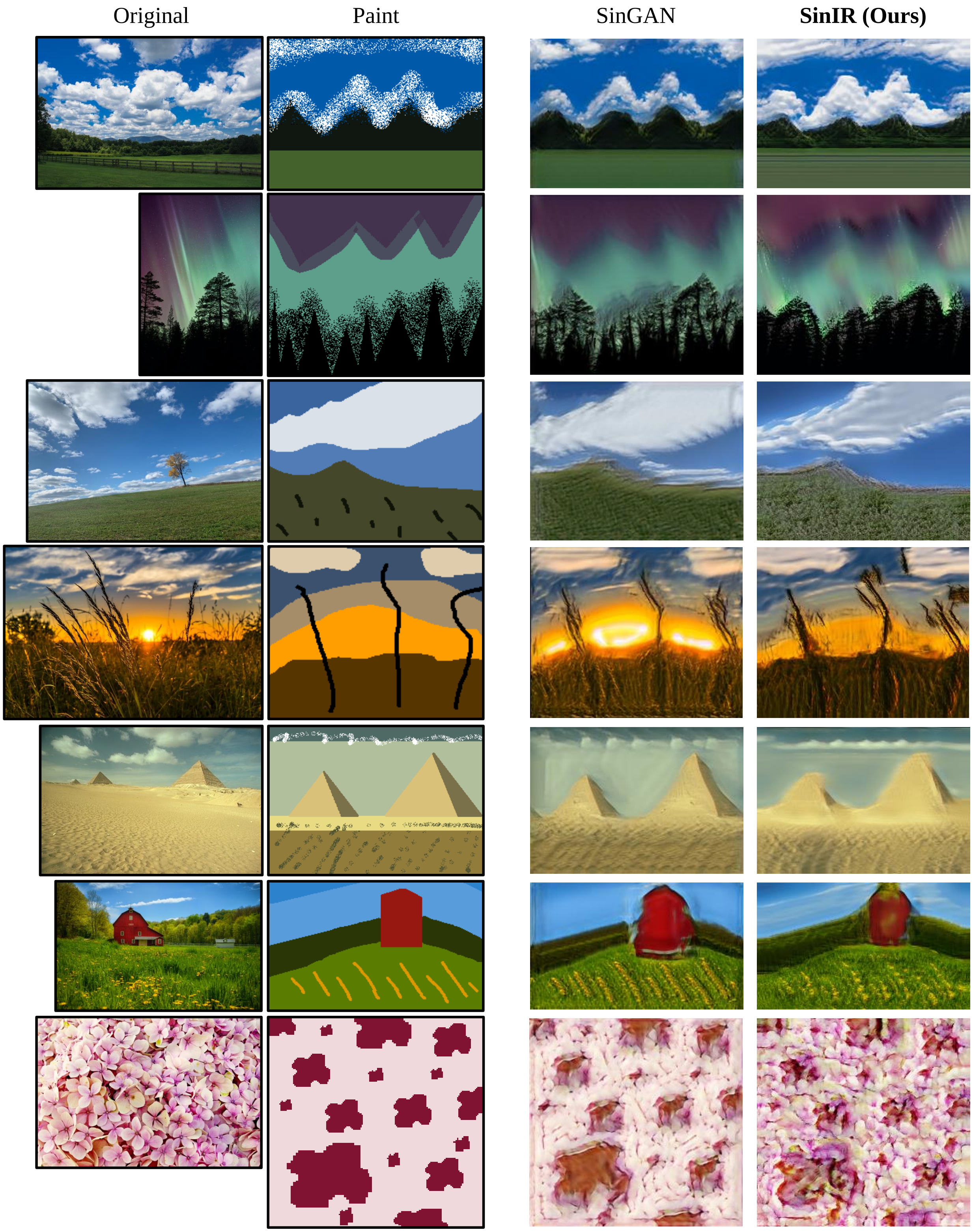

Figure 5. **Paint-to-Image.**

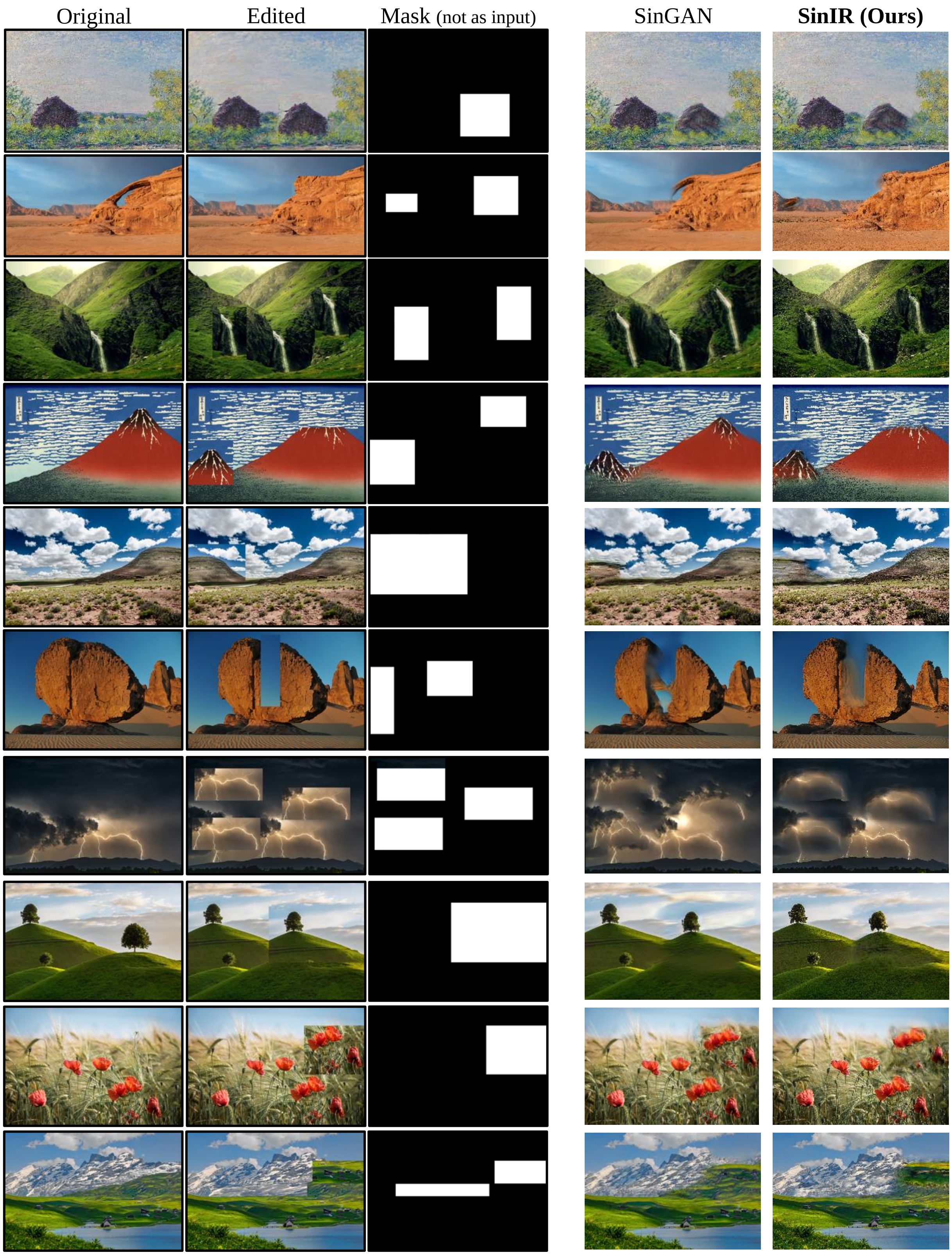

Figure 6. **Editing.**

Figure 7. **Harmonization.** Deep Painterly Harmonization (Luan et al., 2018) is a dedicated method.

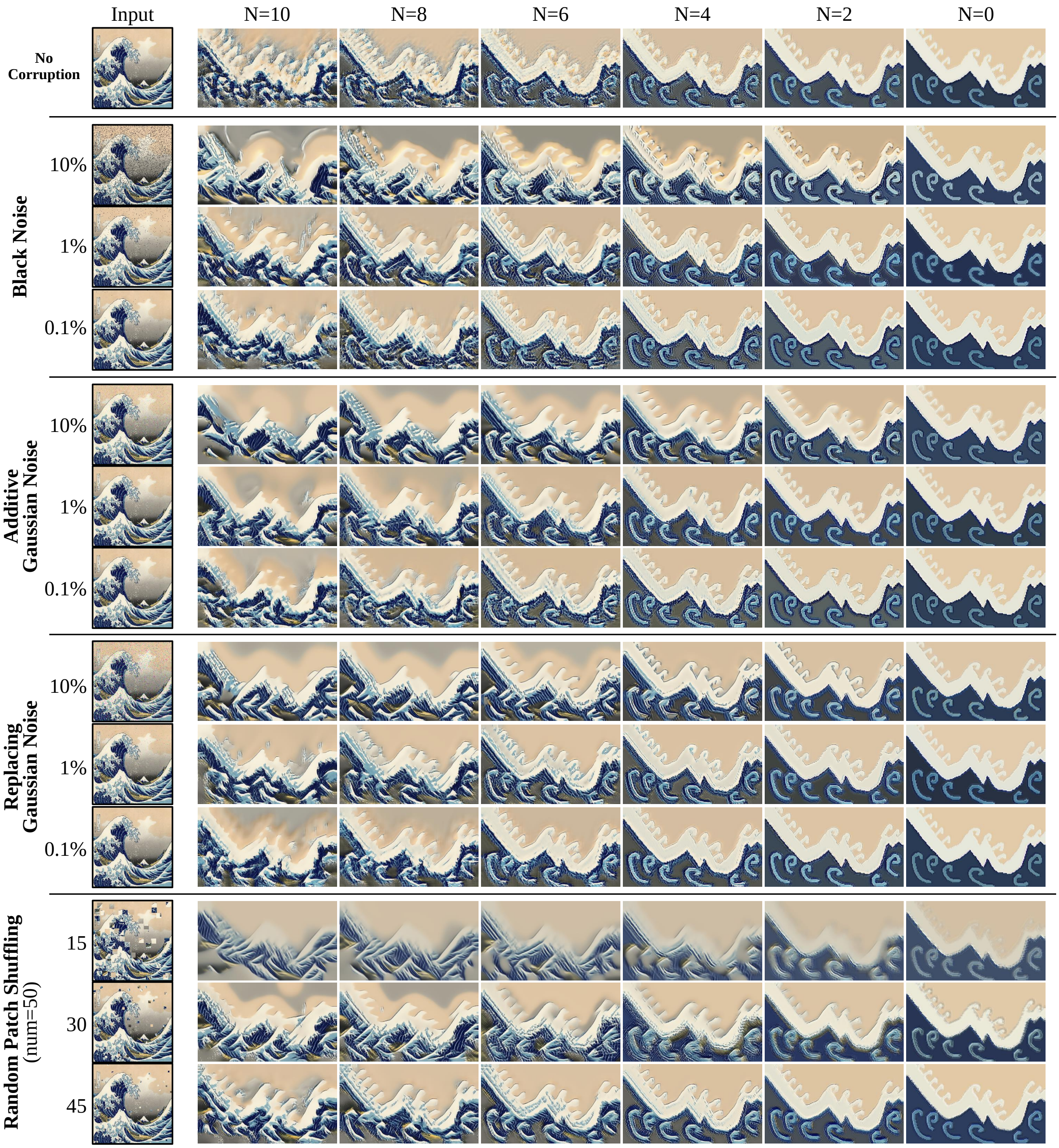

Figure 8. **Effect of different corrupting noise.** We explore 4 types of corruption. The numbers at the top indicate inference starting scales. Please see Section 2.1 of the original paper. The numbers on the left-hand side of inputs indicate the intensity of corruption. For random patch shuffling, they are the ratio of (a longer side of the original image) to (a side of a patch). For example, in case of the 30, (a longer side of the original image) / 30 = (a side of a patch). For other corruption schemes, the numbers are the percentage of randomly sampled pixels to be corrupted. Please see Section 3 for details.

3. Effect of Different Corrupting Scheme

Although we use only *random pixel shuffling* in the original paper, alternative corrupting schemes can be considered. Figure 8 illustrates the effect of different corruption. Here we explore 4 types of corruption. From Figure 8, we can see that applying mild corruption to the input of SinIR generally gives better results regardless of corruption schemes, compared to those of no corruption (the first row in Figure 8). Also, consistent with the findings discussed in Section 2.1.2 of the original paper, regardless of the corrupting schemes, when the intensity of the corruption becomes high, the results become smoothed and vice versa.

Black Noise *Force randomly sampled pixels to be completely black.* This corrupting scheme is originally used in denoising autoencoder (Vincent et al., 2008) with MNIST dataset (Bengio et al., 2006; Lecun et al., 1998). Compared to other corrupting schemes, this corruption sometimes produces unnatural textures (*e.g.*, floating objects with black edges in Figure 8). This is probably because we are using natural images, not the MNIST dataset. Considering that natural images often include more complex structures, simply *turning off* pixels ignores such visual properties and may prevent learning better relationship between adjacent pixels.

Additive Gaussian Noise *Add gaussian noise to randomly sampled pixels.* For gaussian noise, we set mean and variance to 0 and 0.5 with the pixel value of [-1, 1]. The corrupted pixel values are clipped at -1 and 1. Compared to *Black Noise*, this corrupting scheme generally produces better results that are close to random pixel shuffling that is used in the original paper. A possible reason is that now we are corrupting the input based on its original pixel values.

Replacing Gaussian Noise *Replace randomly sampled pixels with gaussian noise.* For gaussian noise, we set mean and variance to 0 and 0.5 with the pixel value of [-1, 1]. The corrupted pixel values are clipped at -1 and 1. The results are similar to *Additive Gaussian*, but it sometimes produces unnatural objects as *Black Noise* does. It is probably because both of them corrupt pixels not based on its original values.

Random Patch Shuffling *Shuffle randomly sampled patches.* We shuffle 50 patches for Figure 8. As we directly use internal patches, this scheme produces well-textured results (*e.g.* wave bubbles). These results are closest to those of *random pixel shuffling* used in the original paper.

References

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. Greedy layer-wise training of deep networks. In *NeurIPS*, 2006.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *CVPR*, 2017.

Johnson, J., Alahi, A., and Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 1998.

Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A. P., Tejani, A., Totz, J., Wang, Z., and Shi, W. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

Lim, B., Son, S., Kim, H., Nah, S., and Lee, K. M. Enhanced deep residual networks for single image super-resolution. In *CVPR*, 2017.

Luan, F., Paris, S., Shechtman, E., and Bala, K. Deep photo style transfer. In *CVPR*, 2017.

Luan, F., Paris, S., Shechtman, E., and Bala, K. Deep painterly harmonization. *Comput. Graph. Forum*, 37(4): 95–106, 2018.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

Shaham, T. R., Dekel, T., and Michaeli, T. Singan: Learning a generative model from a single natural image. In *ICCV*, 2019.

Shocher, A., Cohen, N., and Irani, M. ”zero-shot” super-resolution using deep internal learning. In *CVPR*, 2018.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. S. Instance normalization: The missing ingredient for fast stylization. *arXive: 1607.08022*, 2016.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.

Yoo, J., Uh, Y., Chun, S., Kang, B., and Ha, J. Photorealistic style transfer via wavelet transforms. In *ICCV*, 2019.