Image Denoising and the Generative Accumulation of Photons Supplementary Material

Alexander Krull[1,2], Hector Basevi[2,3], Benjamin Salmon[1], Andre Zeug[4], Franziska Müller[4], Samuel Tonks[1], Leela Muppala[1], and Aleš Leonardis[1]

[1]School of Computer Science, University of Birmingham, UK
[2]Centre of Membrane Proteins and Receptors, Universities of Nottingham and Birmingham, UK
[3]Metabolism and Systems Research, University of Birmingham, UK
[4]Medizinische Hochschule Hannover, Germany

1. Code and Data Availability

The code and datasets will be made publicly available here: `https://github.com/krulllab/GAP`.

2. Comparing DDPM versus our Generative Model

In Figure 1, we qualitatively compare our results against a denoising diffusion model (DDPM) [1]. While DDPM models achieve impressive sample quality, they are, unlike our method not able to learn the generation of denoised images when trained on noisy data.

3. Additional Details on the Training Procedure

During training, we must ensure that our CNN is able to produce high quality predictions for a range of photon counts. To achieve this, we randomly pick a value for p for each training patch (see section 3.5 in the main paper). To achieve a good coverage over different noise levels/photon counts we use a concept we call pseudo-PSNR, explained in section 3.1. We sample uniformly from a range of pseudo-PSNR numbers and then compute the corresponding value for p accordingly. The process is described in section 3.2

3.1. The Pseudo-PSNR value

When imaging a static sample the resulting PSNR number of the recorded image depends on the amount of light that was allowed to hit the detector. We can expect a longer exposure time or stronger light intensity to produce a cleaner image with a higher PSNR value than an image recorded with a shorter exposure or reduced light intensity, in which the detector was allowed to collect fewer photons. We define the intensity of an image as the average as the average number of photons per pixel

$$\gamma = \frac{|\mathbf{x}|}{n}. \tag{1}$$

To compute the PSNR value of a noisy image $\mathbf{x}$ one generally requires the correctly scaled version of the normalised clean ground truth image $\mathbf{s}$. We can compute the correctly scaled signal as $\bar{\mathbf{s}} = \gamma n \mathbf{s}$, which is then directly comparable to the noisy image $\mathbf{x}$. The equation used for this is

$$\text{PSNR}(\mathbf{x}, \bar{\mathbf{s}}) = 10 \log_{10} \frac{\bar{\mathbf{s}}_{\text{max}}^2}{\text{MSE}}, \tag{2}$$

where $\bar{\mathbf{s}}_{\text{max}}$ is the maximum value of the absolute signal $\bar{\mathbf{s}}$, and MSE is the mean squared error between $\mathbf{x}$ and $\bar{\mathbf{s}}$.

The idea of pseudo-PSNR is to directly compute the PSNR value we might expect for a shot noise corrupted image of a certain intensity, without requiring us to compare a noisy and clean image. We define the pseudo-PSNR value for intensity γ

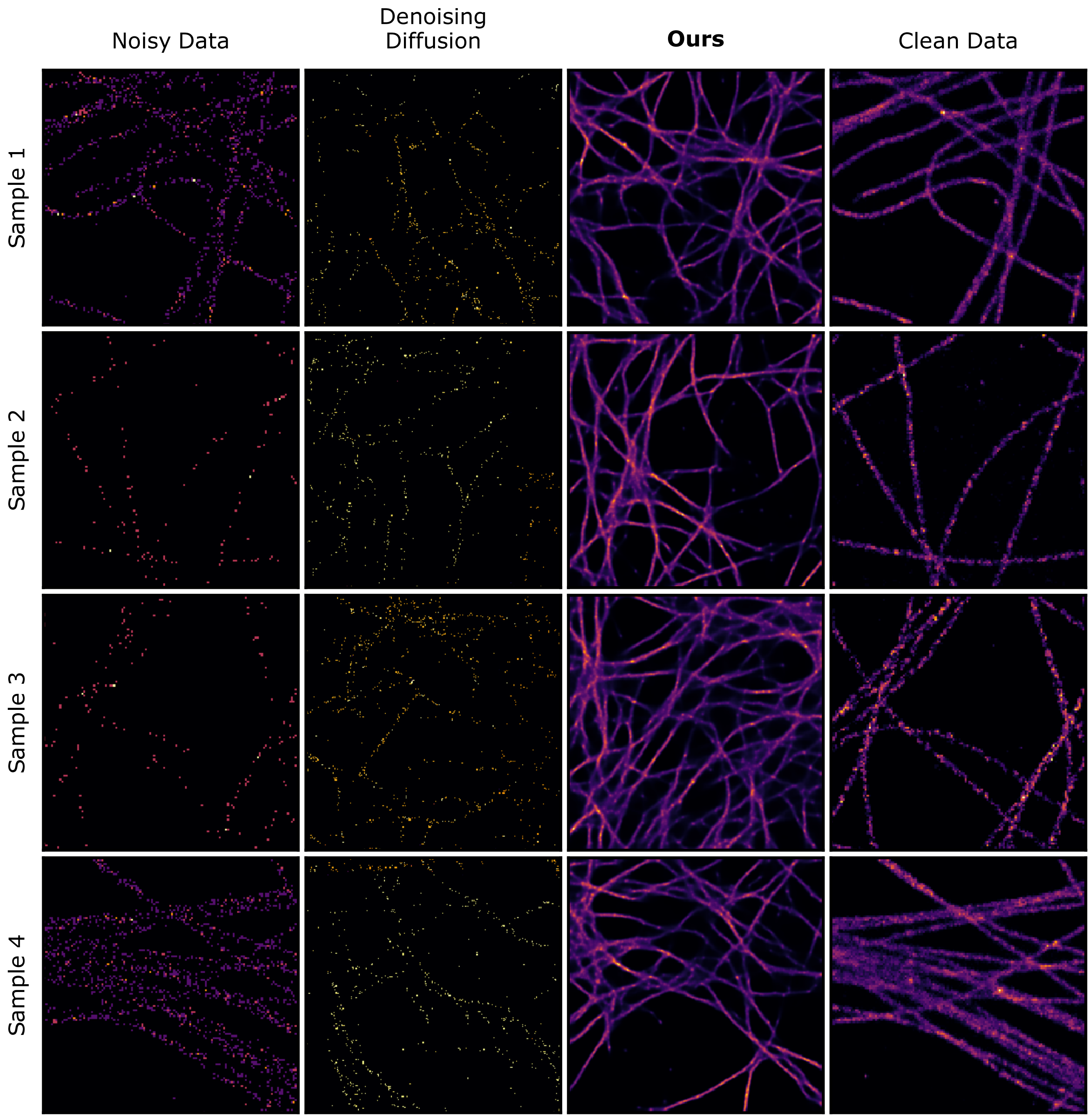

Figure 1. **DDPM vs GAP: Randomly generated samples from DDPM and GAP compared to paired clean and noisy samples.** DDPM [1] and our method were trained on 256x256 patches sampled from the low-SNR MT-SM dataset, each input was obtained via data augmentation strategies that included random cropping, vertical and horizontal flipping, and random transposing. DDPM was trained with default settings and a batch size of 4 on a single NVIDIA GTX1080Ti for 40,000 iterations.

as the PSNR value we would expect for the shot noise corrupted version of a flat signal $\mathbf{s}$, with all pixel values being $s_i = \frac{1}{n}$. Based on the shape of the Poisson distribution, we should expect for such an image $\mathrm{MSE} = \gamma$ and $\bar{\mathrm{s}}_{\max} = \gamma$. Based on Eq 2, we cacalulate the pseudo PSNR as

$$\begin{aligned}\text{PSNR}_{\text{PS}}(\gamma) &= 10\log_{10}\frac{\bar{\mathbf{s}}^2_{\text{max}}}{\text{MSE}} \\ &= 10\log_{10}\frac{\gamma^2}{\gamma} \\ &= 10\log_{10}\gamma\end{aligned} \tag{3}$$

We can invert Eq. 3 to compute the corresponding intensity γ for a given pseudo PSNR value as

$$\gamma = 10^{\frac{\text{PSNR}_{\text{PS}}}{10}}. \tag{4}$$

3.2. Training Pair Sampling

Before we can split out traiing patches into input and target using a binomial distribution (see section 3.5 in the main paper), we have to determine the success probability parameter p sed in the distribution. To achieve this, for each training patch, we first sample from a uniform distribution over pseudo PSNR values between a predefined minimum and maximum.

The goal is to set p, so that the average photon number (intensity) of the resulting input image corresponds to the drawn pseudo PSNR value. We compute the corresponding intensity γ using Eq. 4 from the randomly determined pseudo PSNR value. Then, we compute the corresponding success probability for the binomial distribution as

$$p = \frac{\gamma}{|\mathbf{x}|/n}, \tag{5}$$

such that the input image photon count after the split will correspond to the drawn pseudo PSNR. The result is then clipped to values below 0.99 to guaranty that at least 1% of photons is on average assigned to the target image.

We use the following intervals to sample the pseudo PSNR values: $[-40 : -5]$ for NPC-SM, $[-40, -10]$ for MT-SM, $[-40, -10]$ for Conv-PC, and $[-40, 20]$ for Neuro-PC.

4. Details on Photon Sampling Procedure

Here, we want to discuss the details of the photon sampling procedure. Depending on whether we perform image generation or diverity denoising, we initialise the the process with an empty image or noisy image $\mathbf{x}_0$. We then apply our trained CNN to compute the probability distribution over the possible next photon position

$$\bar{\mathbf{s}}_t = f(\mathbf{x}_t; \theta). \tag{6}$$

Because of our softmax output layer it is guaranteed each pixel value $\bar{s}_{t,i} \geq 0$ and that

$$\sum_i^n \bar{s}_{t,i} = 1. \tag{7}$$

We then sample a set of new photons represented by the image $\mathbf{x}_t^{\text{new}}$, where each pixel value $\mathbf{x}_{t,i}^{\text{new}}$ holds the number of photons that will be added at location i. Each pixel value $x_{t,i}^{\text{new}}$ is drawn from a Poisson distribution with mean

$$\lambda_{t,i} = n\bar{s}_{t,i}\alpha_t, \tag{8}$$

where α_t controls how many photons will on average be sampled in total in $\mathbf{x}_t^{\text{new}}$. We set as

$$\alpha_t = \max(\beta\sum_i^n x_{t,i}, 1), \tag{9}$$

where the parameter β controls the rate at which the photon number increases on average. In our experiments, we set $\beta = 10\%$. The maximum operation ensures that the number of photons is increasing from the beginning even when starting with an empty image $\mathbf{x}_0$. Finally, we compute the next photon count image as

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{x}_t^{\text{new}} \tag{10}$$

and repeat the process.

5. Detailed Description of Datasets

Here, we want to give additional information about the counting datasets we recorded.

5.1. Conv-PC dataset

Data were recorded from a convallaria majalis rhizome section sample slide at the same system (Leica TCS SP8 TPE DIVE with FALCON and the HC PL IRAPO 25x dipping objective). Here we used 850 nm excitation at 1% laser power, , HyD-RLD detector, emission range of 650 – 700 nm (Note to Alex: Ch1: 600nm - 650nm, Ch2: 650nm - 700nm), pixel size 0.6 x 0.6 µm, a 8 MHz resonant scanner and 4x averaging. A time series of 8192 frames was recorded and later binned 4 times. A subset of this data was taken for the study. Although the xy-drift of the dataset was below 2 pixels, we used the drift-corrected mean of the complete dataset as ground truth.

5.2. Neuro-PC dataset

Data were obtained form 11–14 week old male Mice (C57BL/6J background) stereotactically injected with AAV-hGFAP-5-HT4R-eGFP and AAV-hGFAP-tdTomato to the CA1 region of the hippocampus 3 weeks prior to experiment. Data where recorded from acute slices of the mouse hippocampus region at a Leica TCS SP8 TPE DIVE with FALCON, using following acquisition settings: HC PL IRAPO 25x dipping objective, excitation 920 nm at 15-30%, HyD-RLD detector, emission 490 – 560 nm (eGFP), and 560 – 650 nm (tdTomato), voxel size (0.1 x 0.1 x 0.5 µm), 4x averaging, scan speed 600 Hz.

6. Network Architecture

We use a modified UNet [6] architecture, with skip connections and residual blocks. Each residual donw-block and up-block consists of 3 3×3 convolutions with RELU activation functions after the second convolutions and at the end of the block. We use max-pooling for down-sampling and transposed convolutions for up-sampling.

Since we are training a single network to handle a range of different noise levels and photon counts in its input, normalzing the input is not trivial. To avoid normalization, we use a sinusoidal frequency encoding [7] applied to each pixel value at the input of our network. We use 10 different sinosoids with frequencies at different powers of 10.

7. Hyper Parameters

Since our datasets have differing sizes, we define one training epoch as 500 training steps. We use the first 90% of images in each dataset as training data and the last 10% as validation set. We use the ADAM optimizer [3]. We use an initial learning rate of $1\mathrm{e}{-4}$ and reduce the learning rate using the pytorch *ReduceLROnPlateau* scheduler with a patience of 10 by a factor of 2.

8. Details on the Loss Function

Here we show that the loss function from Eq. 13 in the main paper, which uses images target images $\mathbf{x}^k_{\text{tar}}$ with multiple photons is equivalent to using single photons represented by one-hot-encoding images. We can write the loss as

$$\begin{aligned} L(\theta) &= -\sum_{k=1}^{m}\sum_{i=1}^{n} \ln f_i(\mathbf{x}^k_{\text{inp}};\theta)x^k_{\text{tar},i} \\ &= -\sum_{k=1}^{m}\sum_{i=1}^{n} \ln f_i(\mathbf{x}^k_{\text{inp}};\theta)\sum_{t=1}^{T} x^{k,t}_{\text{tar},i} \end{aligned}, \tag{11}$$

where $\sum_{t=1}^{T} x^{k,t}_{\text{tar},i}$ is the one-hot-encoding photon image for photon t from $\mathbf{x}^k_{\text{tar}}$. Note that the order of photons does not matter here. We can then continue to write

$$\begin{aligned} L(\theta) &= -\sum_{k=1}^{m}\sum_{i=1}^{n} \ln f_i(\mathbf{x}^k_{\text{inp}};\theta)\sum_{t=1}^{T} x^{k,t}_{\text{tar},i} \\ &= -\sum_{k=1}^{m}\sum_{t=1}^{T}\sum_{i=1}^{n} \ln f_i(\mathbf{x}^k_{\text{inp}};\theta)x^{k,t}_{\text{tar},i} \end{aligned}. \tag{12}$$

In this formulation it becomes clear that using target images with multiple photons is equivalent to replicating each input image T times and using it together with each of the corresponding single-photon target images. This corresponds to the same training data distribution as randomly sampling single photon targets.

9. Additional Qualitative Results

Here, we show random uncurated samples for all datasets in Figure 2

10. Image Generation on Natural image datasets

To show the potential of our method, we show uncurated random outputs of our generative model when applied on natural image datasets in Figure 3. To account for the greater complexity of these datasets, we trained 8 expert networks , each specialised on a sub-range of pseudo PSNR values. Each network is scaled up to 7 levels (instead of 6) and starting with 32 feature (instead of 28) channels When generating images we switch between these expert networks as the image gains more and more photons. Apart from this the approach is the same as for the microscopy data.

11. Denoising with added Gaussian Noise

To test the limitations of our method we use the BSD68 dataset [5] with simulated shot noise at peak photon level 20 (see [4]) and additionally at various levels of additive white Gaussian noise. Results for this experiment can be found in Table 1. We observe htat PSNR drops substantially when larger amounts of Gaussian noise are added. Note the the model never saw Gaussian noise during training and that Gaussian noise goes against the assumtions mate by our method.

Table 1. We show the PSNR achieved by our method when addition different levels of Gaussian noise to the data. Experiment was done on the BSD68 [5] dataset with Poisson noise at peak photon level 20 (see [4]) and the respectively additional Gaussian noise.

Standard deviation in photons	0	0.25	0.5	1	2	4
PSNR	27.01	26.99	26.91	26.28	22.56	16.54

12. Runtime

The model requires 0.012 seconds to process a 481×321 pixel image on an Nvidia GA102 [GeForce RTX 3090]. We believe this can be improved by processing multiple images in parallel.

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 33:6840–6851, 2020.

[2] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, pages 4401–4410, 2019.

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 3rd international conference on learning representations, iclr 2015. In *ICLR*, 2015.

[4] Hao Liang, Rui Liu, Zhongyuan Wang, Jiayi Ma, and Xin Tian. Variational bayesian deep network for blind poisson denoising. *Pattern Recognition*, 143:109810, 2023.

[5] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *CVPR*, volume 2, pages 416–423. IEEE, 2001.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[7] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 7537–7547, 2020.

[8] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

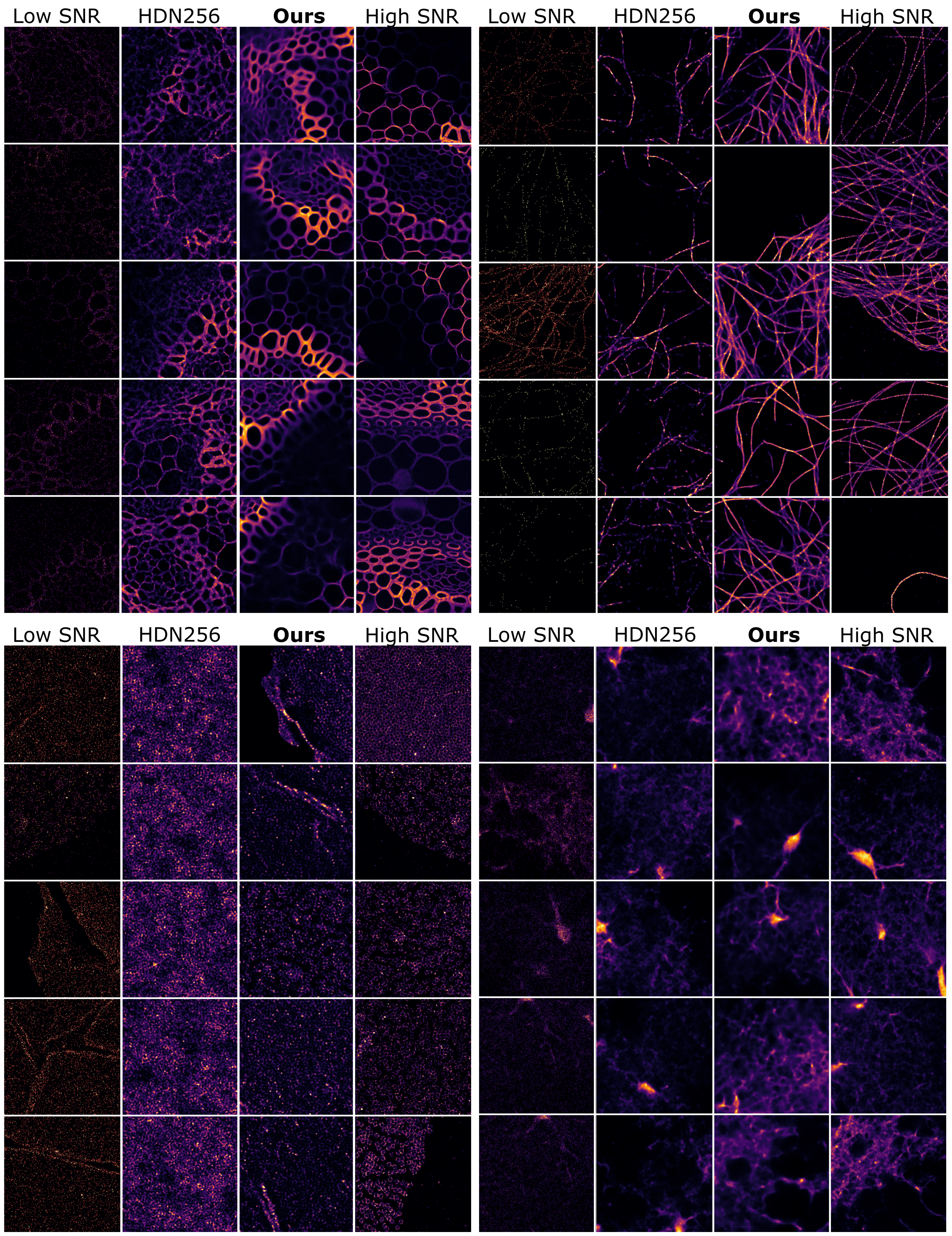

Figure 2. **Comparison of sample quality:** We show randomly selected generated samples for all datasets (top-left to bottom-right: *Conv-PC, MT-SM, NPC-SM, Neuro-PC*) compared to randomly cropped real high- and low-SNR patches.

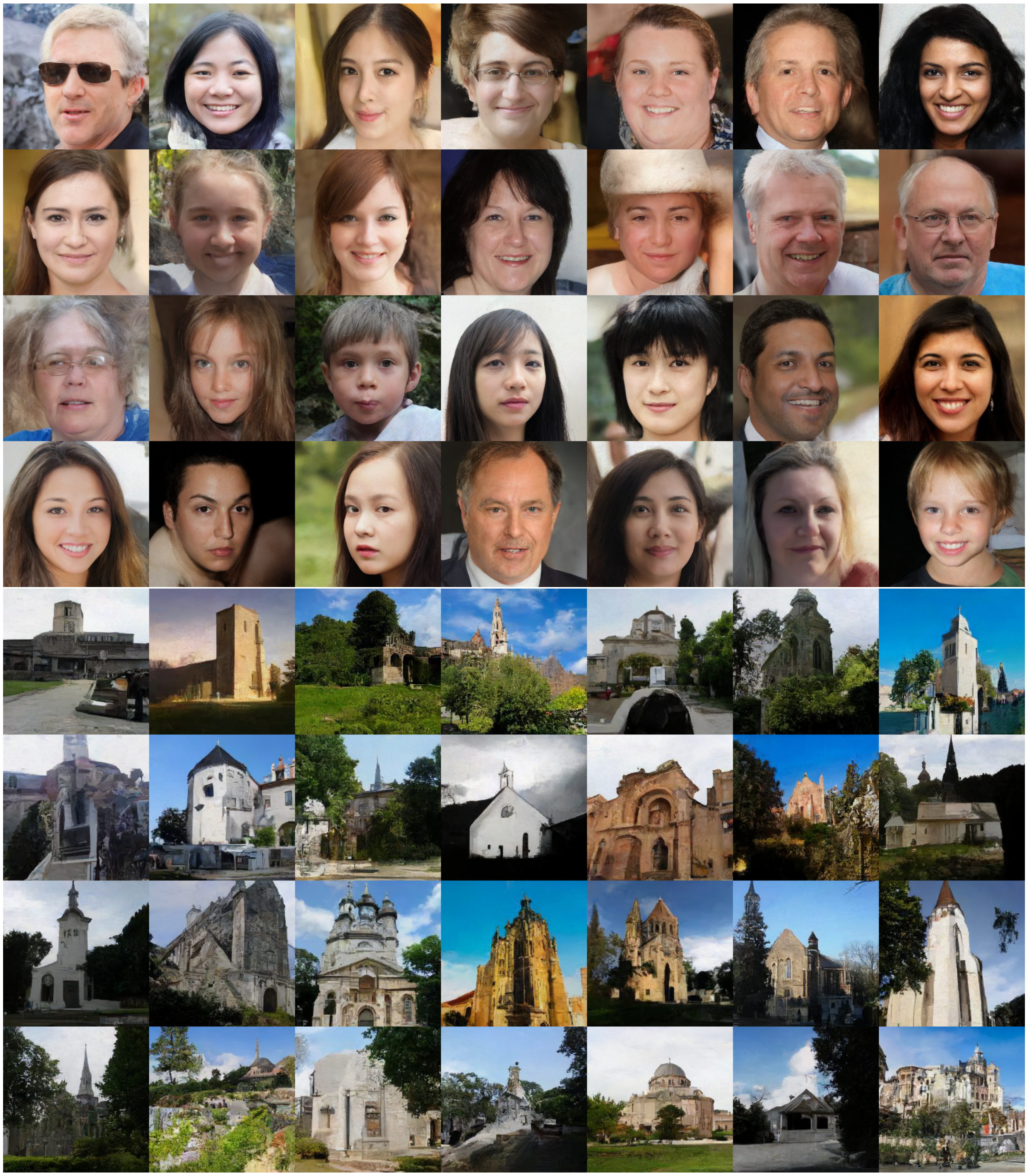
Figure 3. **Randomly selected sample images generated by our model.** Results of our method when trained on the 256×256 pixel versions of FFHQ dataset [2] and the LSUN-churches dataset [8].