Unsupervised Denoising for Signal-Dependent and Row-Correlated Imaging Noise Supplementary Material

Benjamin Salmon and Alexander Krull
School of Computer, University of Birmingham
brs209@student.bham.ac.uk, a.f.f.krull@bham.ac.uk

https://github.com/krulllab/COSDD

1. Latent Variables Represent Clean Images

The training of the signal decoder assumes that every sampled value of the latent variable $\mathbf{z}$ corresponds one clean image, or signal, $\mathbf{s}$. We denote the signal corresponding to a value of $\mathbf{z}$ by $\mathbf{s}(\mathbf{z})$. Using this relationship, the signal decoder, $f_\nu(\mathbf{z})$, can be trained to estimate

$$f_\nu(\mathbf{z}) \approx \mathbb{E}_{p_\theta(\mathbf{x}|\mathbf{z})}[\mathbf{x}] = \mathbb{E}_{p_\theta(\mathbf{x}|\mathbf{s}(\mathbf{z}))}[\mathbf{x}] = \mathbf{s}(\mathbf{z}). \tag{1}$$

Here, we provide an another way of viewing this deterministic relationship.

If the latent variables of our model truly represent only signals, then the AR decoder, $p_\theta(\mathbf{x}|\mathbf{z})$, must model only the noise generation process. Therefore, different random samples from the AR decoder for the same value of latent variable will be images with the same underlying signal and different random samples of noise. Since the noise is zero-centered, this allows us to produce an estimate of the signal by calculating the mean of many samples from the AR decoder. That is, if $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L$ are L random samples from $p_\theta(\mathbf{x}|\mathbf{z})$,

$$\overline{\mathbf{x}} = \frac{1}{L}\sum_{l=1}^{L} \mathbf{x}_l \approx \mathbb{E}_{p_\theta(\mathbf{x}|\mathbf{z})}[\mathbf{x}] = \mathbf{s}(\mathbf{z}). \tag{2}$$

In this section, we experimentally verify Eq. (1) by estimating the signal underlying a noisy image $\mathbf{x}$ using both techniques; by passing a latent variable sample to the signal decoder and by averaging 10,000 noisy image samples from the AR decoder. If Eq. (1) is true, the two estimates of the signal should be nearly identical for the same value of latent variable.

Figure 1 shows the result of this experiment for two different random samples from the approximate posterior, $\mathbf{z}_1$ and $\mathbf{z}_2$. The two estimates of $\mathbf{s}(\mathbf{z}_1)$ are visually very similar to each other, while exhibiting clear structural differences from the two estimates of $\mathbf{s}(\mathbf{z}_2)$.

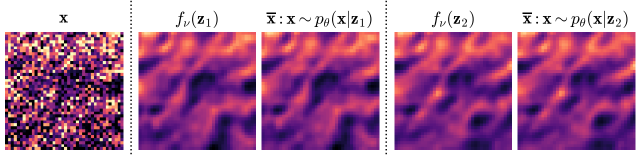

Figure 1. Given a noisy image $\mathbf{x}$, we took two samples from the approximate posterior, $\mathbf{z}_1$ and $\mathbf{z}_2$. For each latent variable sample, we produced one estimate of the signal by passing it through the signal decoder f_ν and one by averaging 10,000 samples from the AR decoder, $p_\theta(\mathbf{x}|\mathbf{z})$.

2. Training and Inference

2.1. Hyperparameters

Both the main VAE and the signal decoder were trained with an Adamax [6] optimizer with a learning rate of 0.002. Both learning rates decreased by a factor of 10 when the validation loss had plateaued for 50 epochs. The models for all datasets were trained for a maximum of 80,000 steps but stopped if validation loss had plateaued for 100 epochs.

For the non-simulated datasets, training images were randomly cropped to a size of 256×256 at each epoch and a batch size of 16 was used, but this was split into 4 virtual batches. For the *FFHQ - Stripe* and *FFHQ - Checkerboard* datasets, training images were kept at their original resolution of 128×128 and a batch size of 64 was used, but split into 16 virtual batches.

2.2. Hardware and Software

The workstation used for this paper's experiments is a 36 core Intel(R) Core(TM) i9-10980XE CPU @ 3.00GHz with 134GB of RAM running Ubuntu 22.04.4, Python 3.11.5 and pytorch-lightning 2.2.1. The GPU used for training is a NVIDIA GeForce RTX 3090 with 24GB of VRAM. For all datasets, training required approximately 20GB of GPU memory with the *large* network and 6GB with the *small* network.

2.3. Times

Training a model for 80,000 steps with our hardware takes approximately 24 hours. After training, denoising 100 images of size 512x512 by randomly sampling 100 denoised estimates takes 13 minutes when using a batch size of 10. Increasing batch size made no improvement as the GPU is at 100% utilization.

2.4. PSNR of Minimum Mean Square Error Estimates

In addition to the quantitative results presented in the paper, the PSNR of the mean of 1, 10 and 1000 random denoised samples can be found in Tab. 1.

3. Architecture

We proposed two network architectures for denoising, one *large* and one *small*. Each model consists of a hierarchical Variational Autoencoder (VAE) [11], an autoregressive decoder [13] and our novel *signal decoder*. In the *large* network, the VAE has 14 levels in its hierarchy. The first 13 levels have 64 latent dimensions each, while the final level has 128 dimensions. The latent variable passed to the decoders is sampled from this final level. At each level on both the bottom-up path and the top-down path is a residual block consisting of two sets of a convolution followed by a batch normalization [4] followed by a Mish activation function [9]. Each residual block is followed by a gated block [12]. Resampling is performed at alternating levels. The *small* network is the same except that it has 6 levels to its hierarchy and half the latent dimensions.

The autoregressive decoder is built with eight layers of conditional PixelCNN blocks as proposed in [13], but we found the performance to be better with a ReLU activation function [1] than with gated units. The convolving kernels in the AR decoder have dimensions $1 \times k$, where k is the kernel size. In the first layer of the decoder, the input is padded with k zeros on its left-hand side, and then a convolution is applied. At all subsequent layers, the input features are padded with $k - 1$ zeroes on the left-hand side. This results in a row-based autoregressive receptive field. For a column-based receptive field, the kernels have dimensions $k \times 1$ and padding is applied to the top of the input. For all of our experiments, $k = 5$. The convolutions in every other layer have dilated kernels [15] and all have 64 filters. The likelihood distribution is a Gaussian mixture model, with 3 components used for all datasets except the *FFHQ* datasets, for which 10 were used, and the *STEM* dataset, for which 5 were used.

The *signal decoder* is a convolutional neural network consisting of four 3×3 convolutions with 128 filters, each followed by a ReLU activation function.

4. Baselines

AP-BSN [7] These models were trained using the code available at https://github.com/wooseoklee4/AP-BSN using hyperparamters detailed in the original publication. We used a stride factor of 5 for all datasets.

Structured Noise2Void [2] and N2V2 [3] Both these model types were trained using the code available at https://github.com/juglab/n2v, using default hyperparameters found in the example notebooks for SN2V and hyperparameter values found in the original publication of N2V2. Following Broaddus *et al*. [2], SN2V masks should be as small as possible while covering pixels with a noise value that is highly predictive of the noise value in the target pixel. A trial and error test of the mask size for each dataset would be too computationally expensive, so we follow [2] and mask 4 pixels on each side of the target pixel for all datasets except *FFHQ - Checkerboard*. The structured component of noise in the *FFHQ - Checkerboard* dataset can theoretically be predicted by seeing only two pixels in the same column, so entire columns were masked here. The orientation of the pixel mask was determined by looking at the spatial autocorrelation in noise patches for each dataset. The *Mouse Nuclei* dataset is corrupted by unstructured noise, so was denoised with a single pixel mask.

HDN$_{36}$ [10] These models were trained using the code available at https://github.com/juglab/HDN/ using default hyperparameters found in the example notebooks. HDN$_{36}$ requires a pre-trained noise model. We followed Prakash *et al*. [10] and modeled the noise in each dataset using a Gaussian mixture model. The noise model parameters can be estimated from the training data of datasets with available ground truth. For datasets without ground truth, we trained the noise model using denoised images from our method as pseudo-ground truth.

CARE [14] and N2N [8] Both of these model types were trained using the code available at https://github.com/CSBDeep/CSBDeep, using default hyperparameters and setting noisy images as target for N2N.

5. Simulated Data

The Flickr Faces HQ thumbnails dataset [5], with resolution 128×128, was made grayscale by averaging across color channels. For *FFHQ - Stripe*, the ground truth, $\mathbf{s}$, was scaled to have pixel values between 0 and 1, and Poisson noisy images were created as $\mathbf{x} = 0.002 \times \mathcal{P}(\mathbf{s}/0.002)$. Zero-mean Gaussian noise with a standard deviation of 0.02 was then added to these images. Finally, structured noise was created by applying a horizontal Gaussian blur with a standard deviation of 1 to white Gaussian noise with a standard deviation of 0.025 and added on top. For *FFHQ*

Table 1. Our denoiser randomly samples clean images for a given noisy image. A consensus solution can be produced by averaging random samples. The PSNR of the mean of increasing sample sizes is shown below.

	EMCCD			SCM			Simulated	
No. samples	Conv. A	Conv. B	Mouse Actin	Mouse Nuclei*	Actin Conf.	Mito Conf.	FFHQ Stripe	FFHQ Checkerb.
1	36.54	42.33	37.43	42.25	26.81	22.62	32.27	33.03
10	37.39	43.90	39.03	42.91	27.37	23.39	35.24	35.85
100	37.49	44.10	39.23	42.98	27.42	27.50	35.66	36.27
1000	37.50	44.10	39.24	42.99	27.44	27.52	35.74	36.32

- *Checkerboard*, we added noise with inverse signal dependence by sampling Gaussian noise from the distribution $\mathcal{N}(0, 0.15 \times 1/\mathbf{s})$. Then a vertical checkerboard pattern was added by subtracting 0.1 from two pixels and adding 0.1 to the next two pixels along columns. The starting point for the checkerboard was randomly sampled from a uniform distribution. For both *FFHQ* datasets, the final 1000 images were designated as a test set.

6. Additional Qualitative Results

See overleaf for larger denoised images from each dataset.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018. 2

[2] Coleman Broaddus, Alexander Krull, Martin Weigert, Uwe Schmidt, and Gene Myers. Removing structured noise with self-supervised blind-spot networks. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pages 159–163. IEEE, 2020. 2

[3] Eva Höck, Tim-Oliver Buchholz, Anselm Brachmann, Florian Jug, and Alexander Freytag. N2v2–fixing noise2void checkerboard artifacts with modified sampling strategies and a tweaked network architecture. *arXiv preprint arXiv:2211.08512*, 2022. 2

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015. 2

[5] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 2

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[7] Wooseok Lee, Sanghyun Son, and Kyoung Mu Lee. Apbsn: Self-supervised denoising for real-world images via asymmetric pd and blind-spot network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17725–17734, 2022. 2

[8] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2noise: Learning image restoration without clean data. In *International Conference on Machine Learning*, pages 2965–2974. PMLR, 2018. 2

[9] Diganta Misra. Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*, 2019. 2

[10] Mangal Prakash, Mauricio Delbracio, Peyman Milanfar, and Florian Jug. Interpretable unsupervised diversity denoising and artefact removal. In *International Conference on Learning Representations*, 2021. 2

[11] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. *Advances in neural information processing systems*, 29, 2016. 2

[12] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679, 2020. 2

[13] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016. 2

[14] Martin Weigert, Uwe Schmidt, Tobias Boothe, Andreas Müller, Alexandr Dibrov, Akanksha Jain, Benjamin Wilhelm, Deborah Schmidt, Coleman Broaddus, Siân Culley, et al. Content-aware image restoration: pushing the limits of fluorescence microscopy. *Nature methods*, 15(12):1090–1097, 2018. 2

[15] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 2

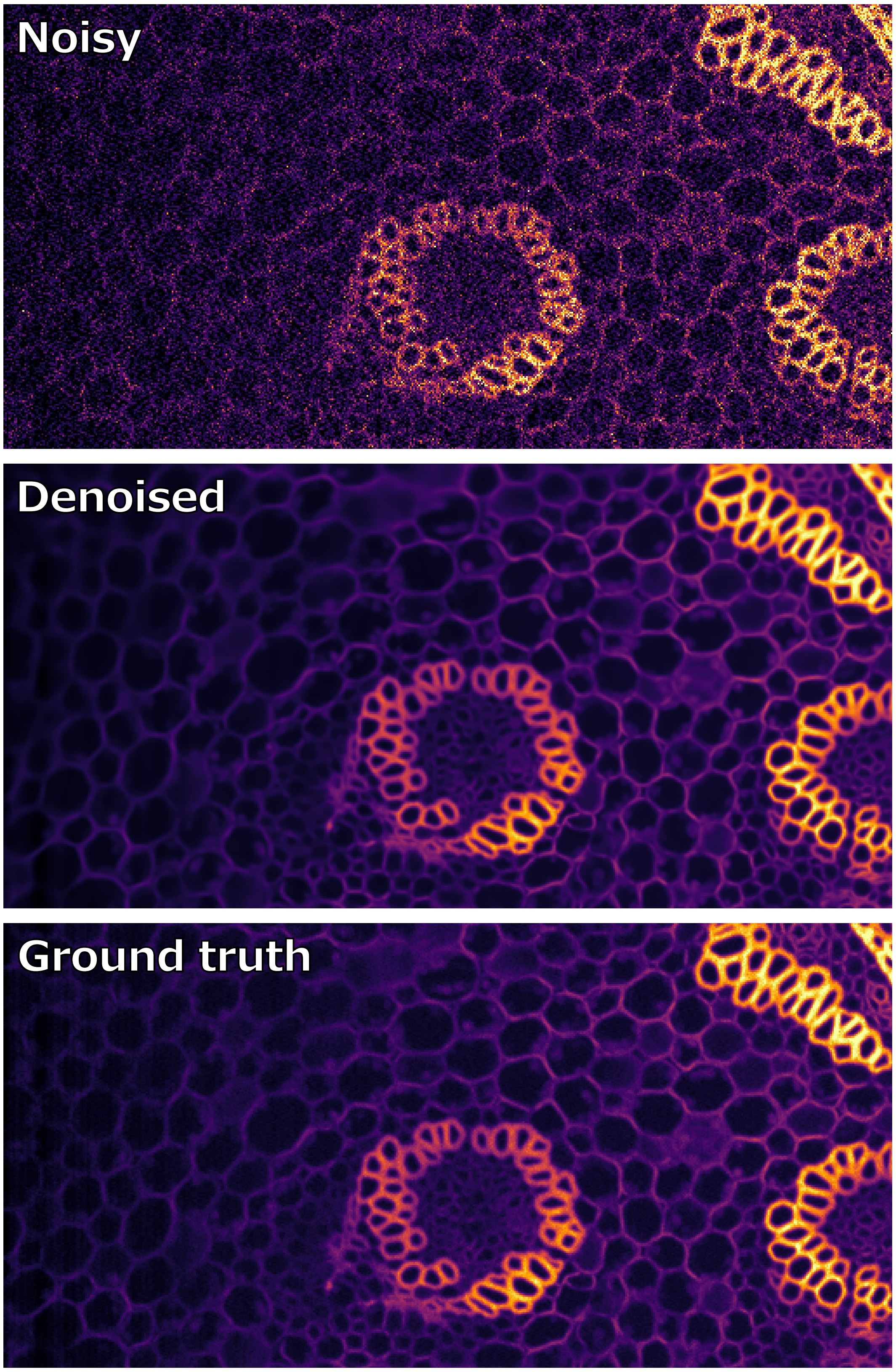

Figure 2. *Convallaria A*

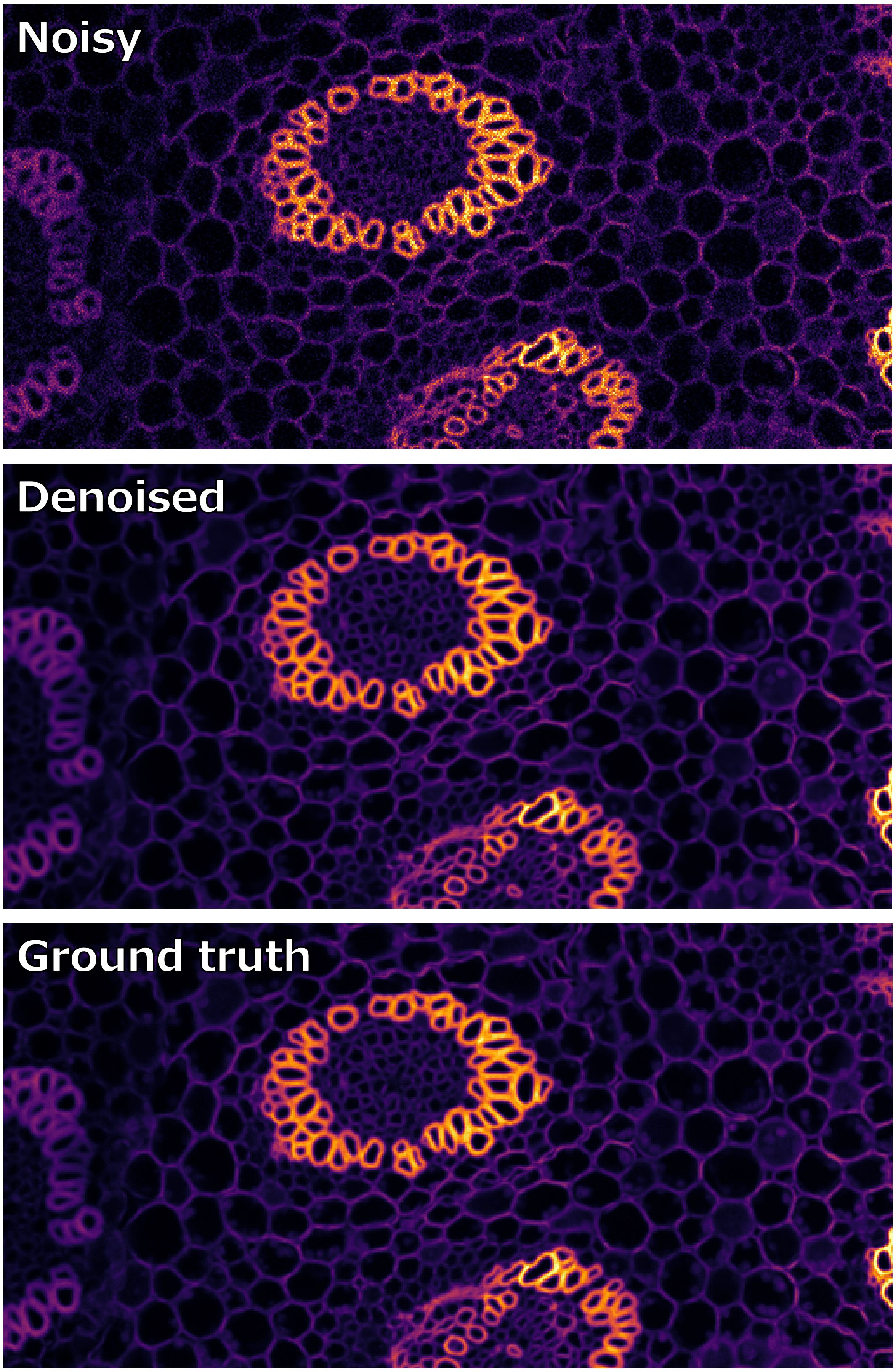

Figure 3. *Convallaria B*

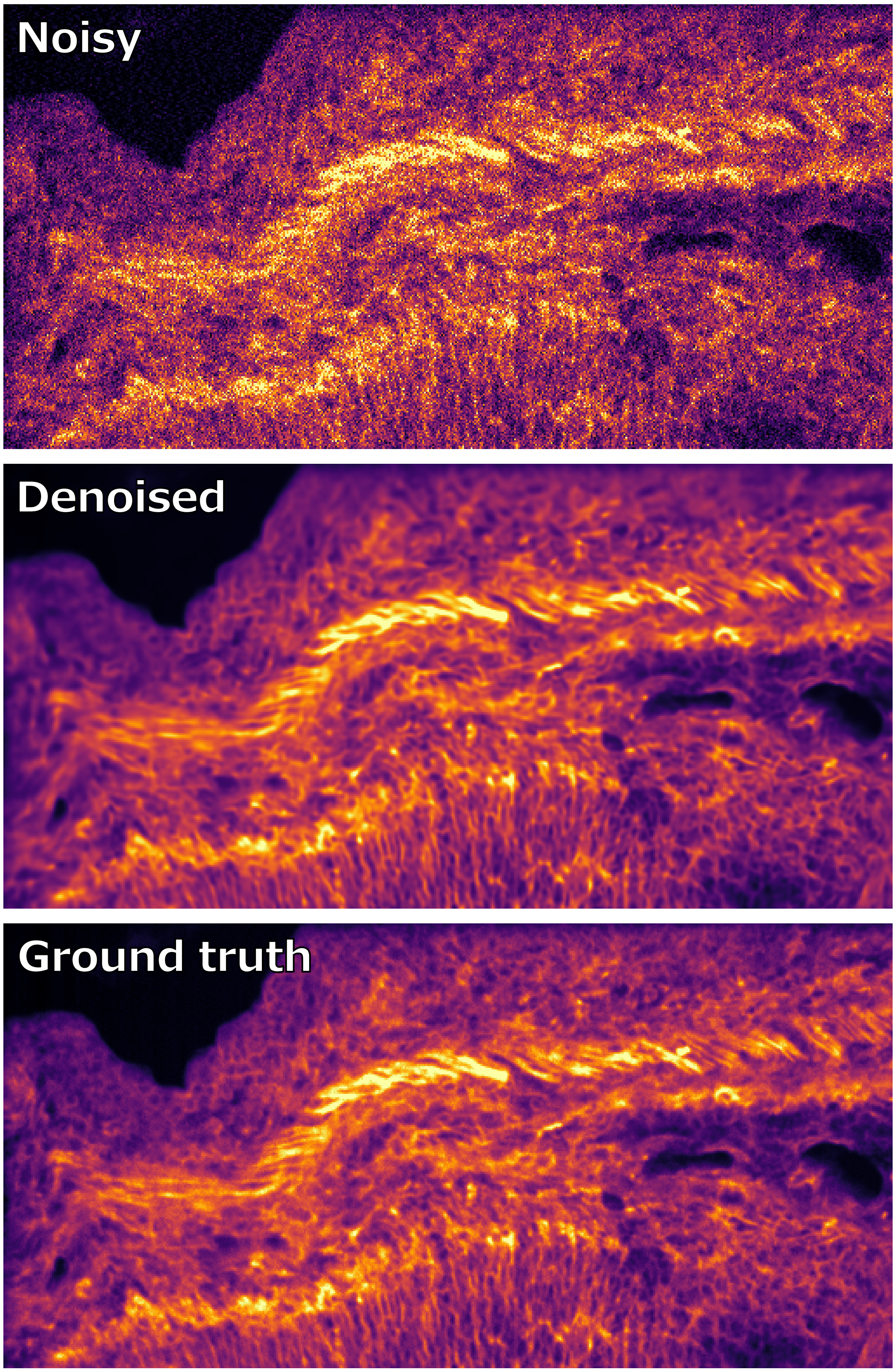

Figure 4. *Mouse Actin*

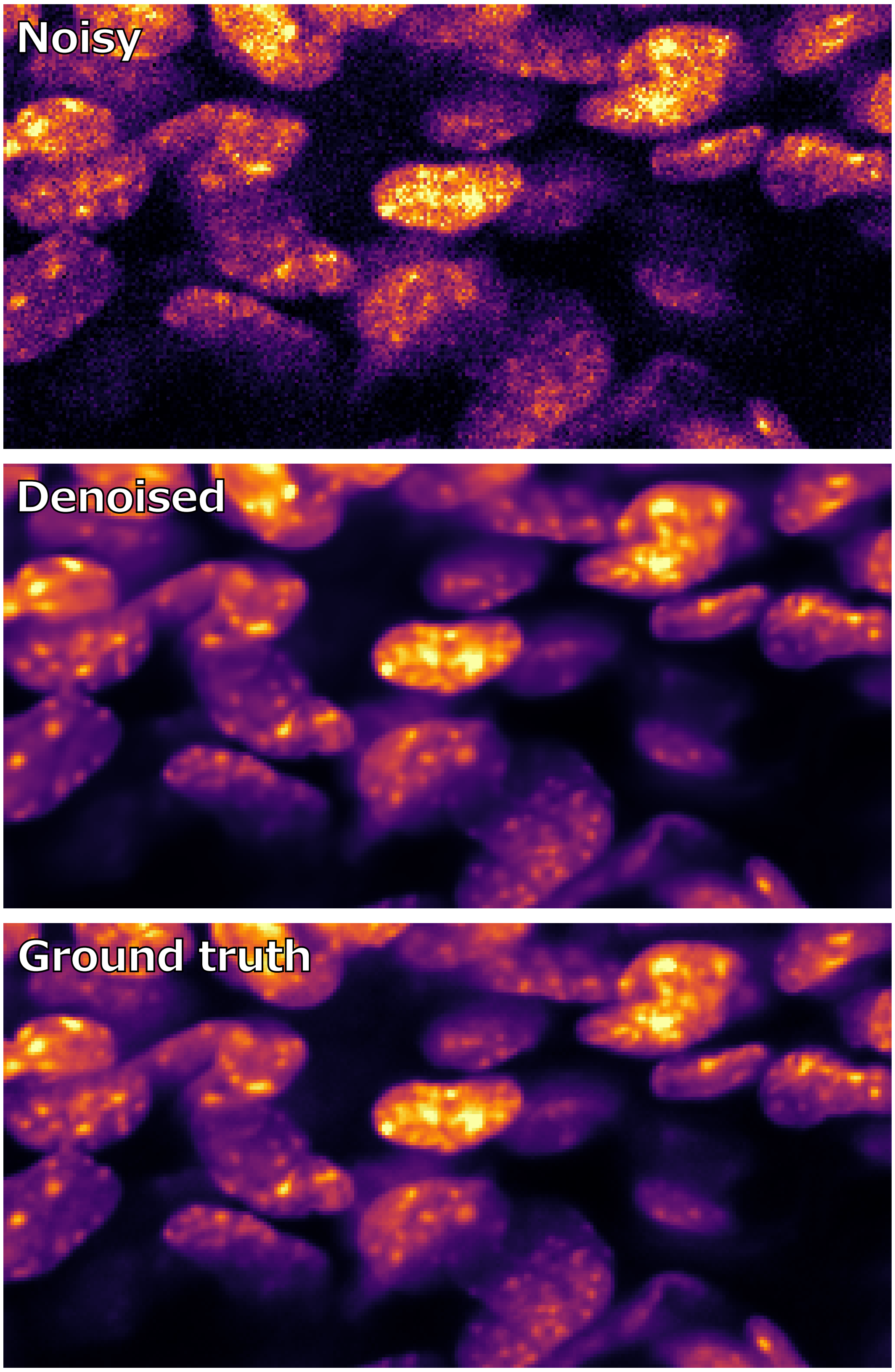

Figure 5. *Mouse Nuclei*

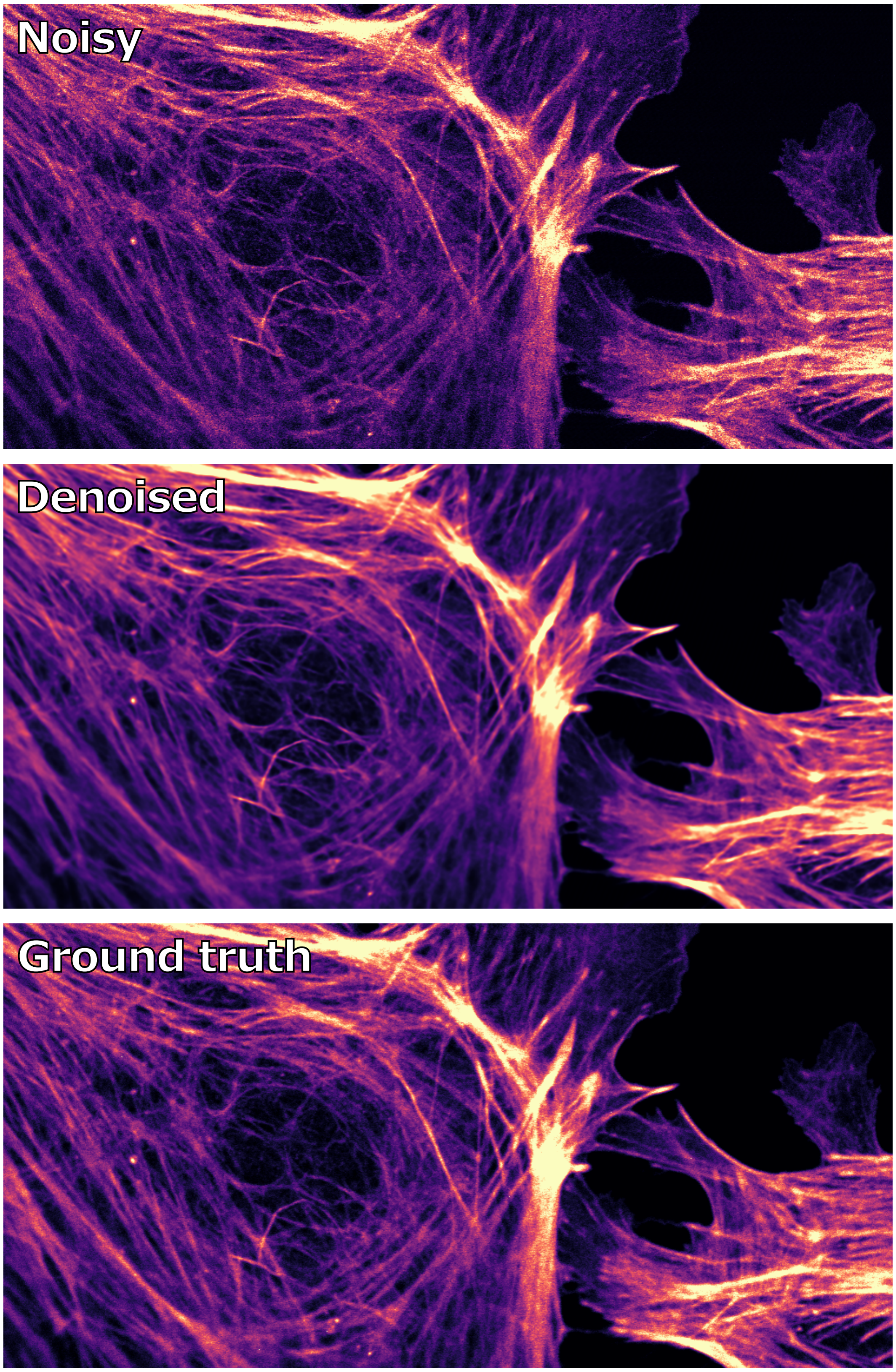

Figure 6. *Actin Confocal*

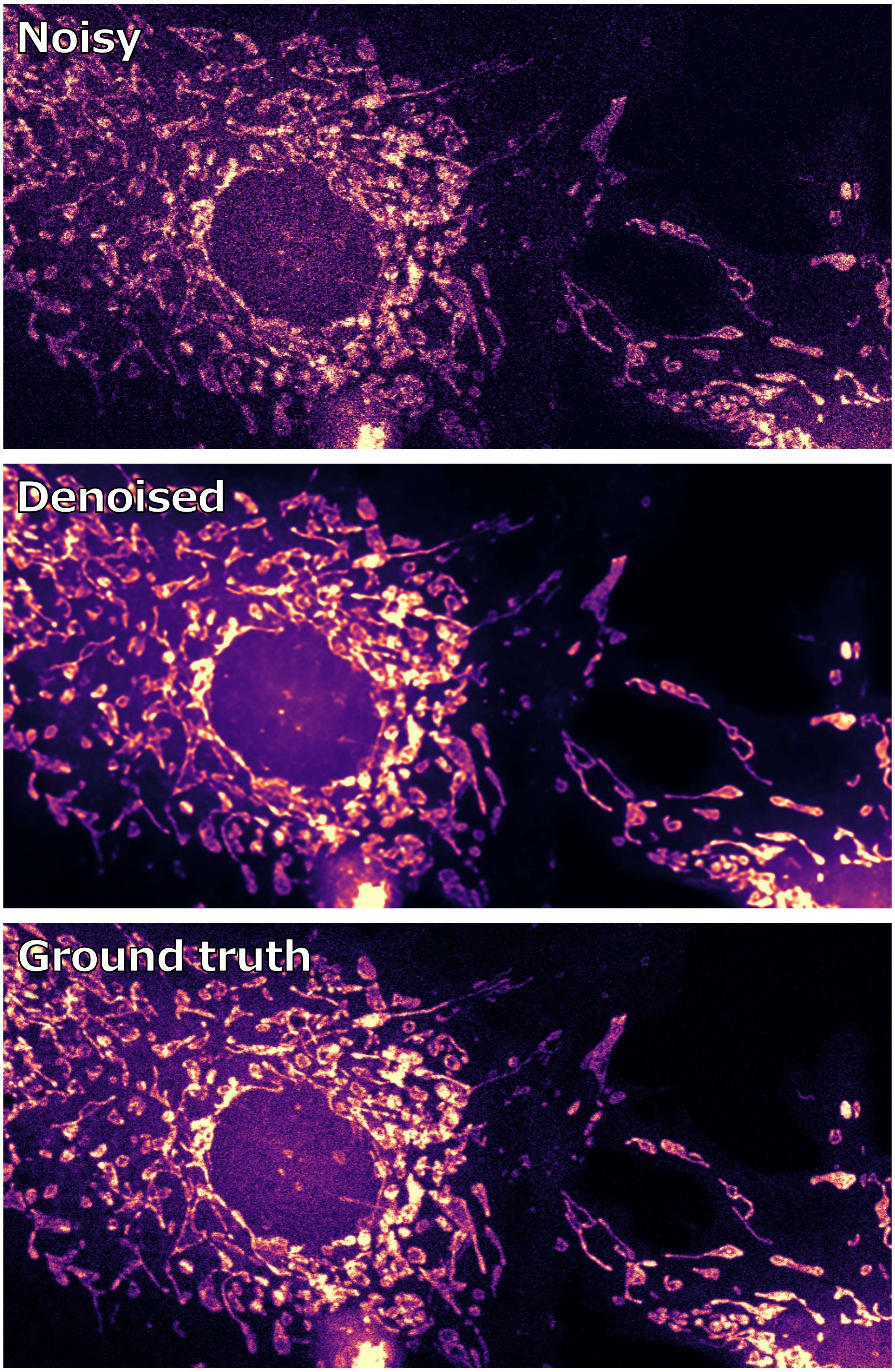

Figure 7. *Mito Confocal*

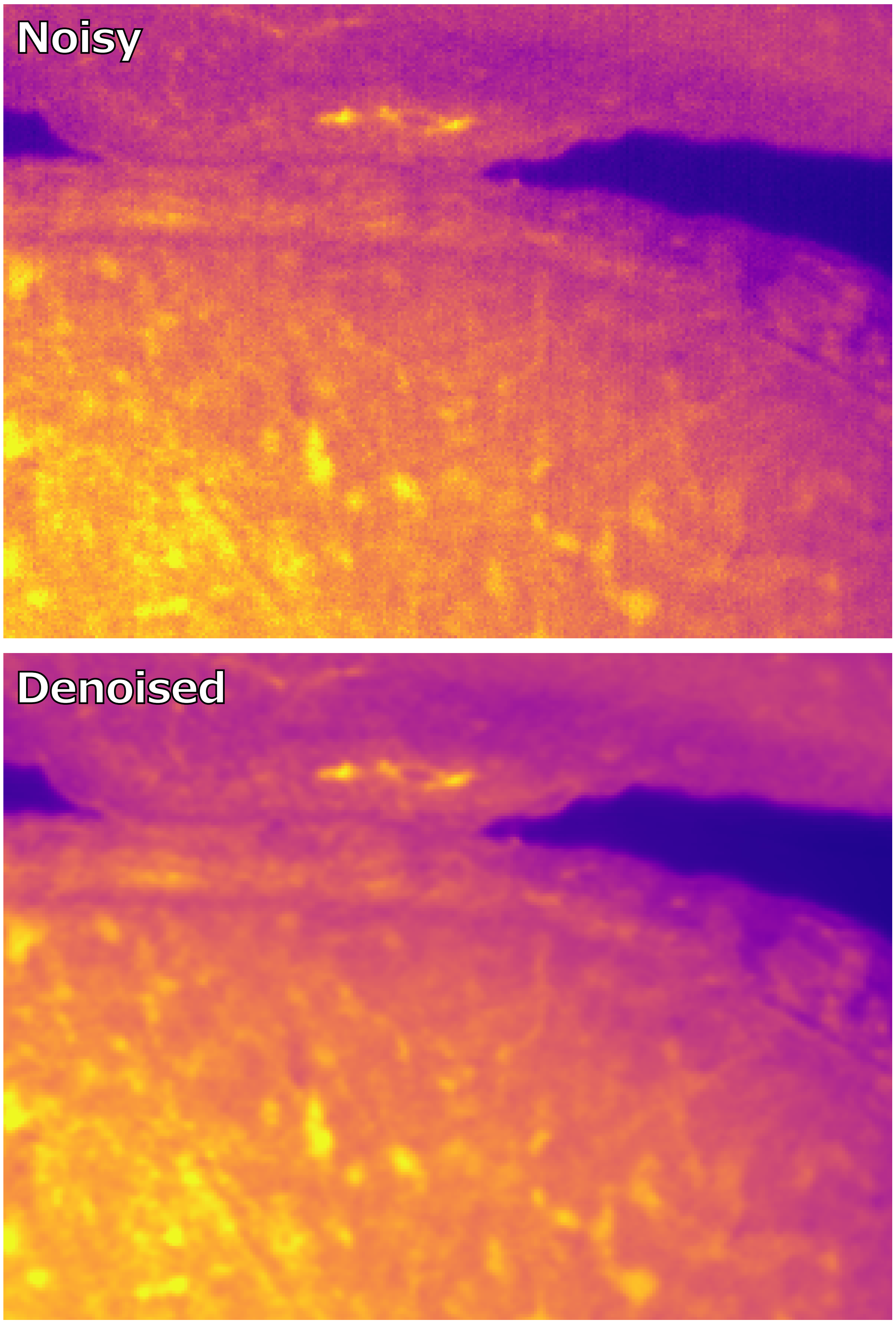

Figure 8. *Embryo*

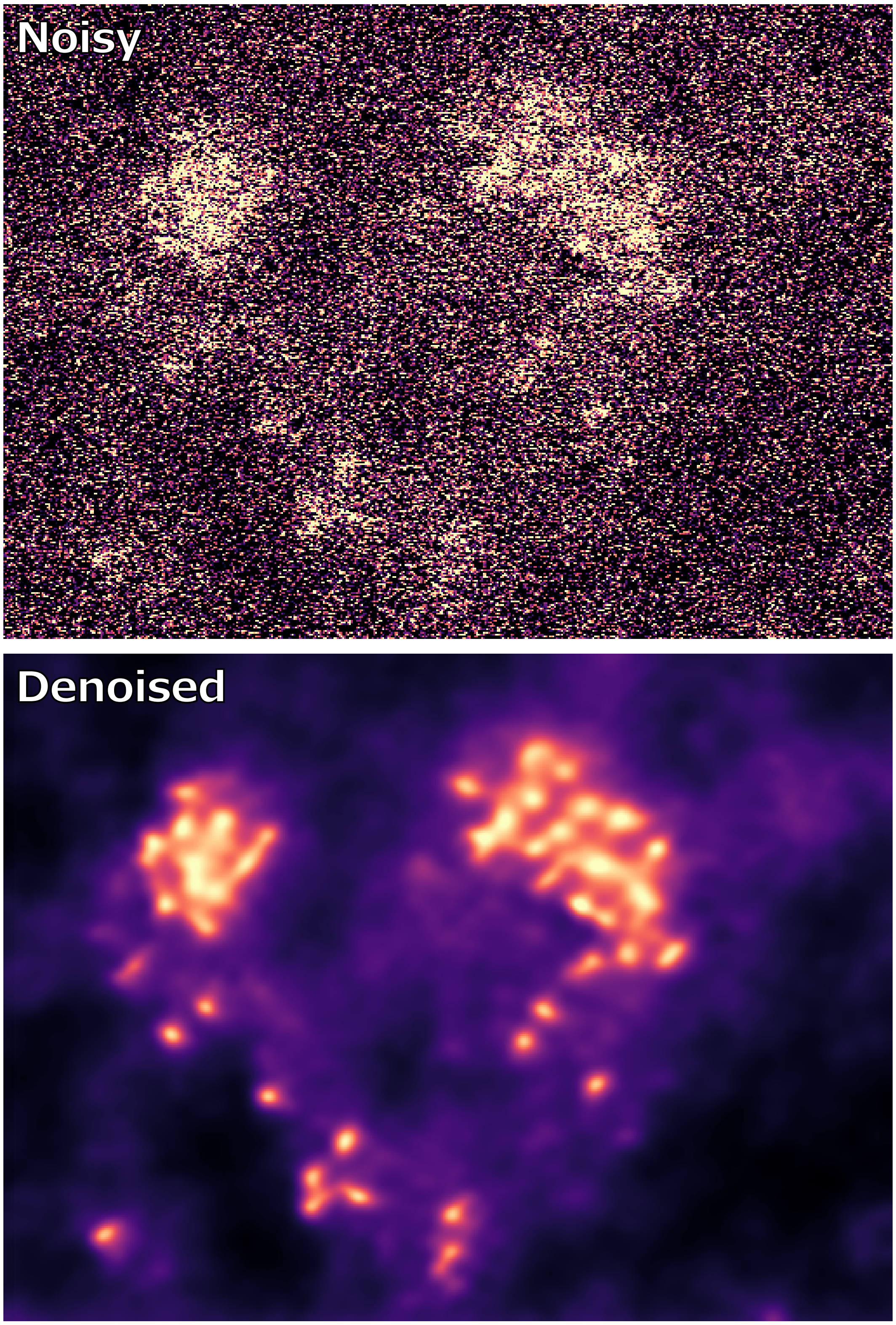

Figure 9. *STEM*

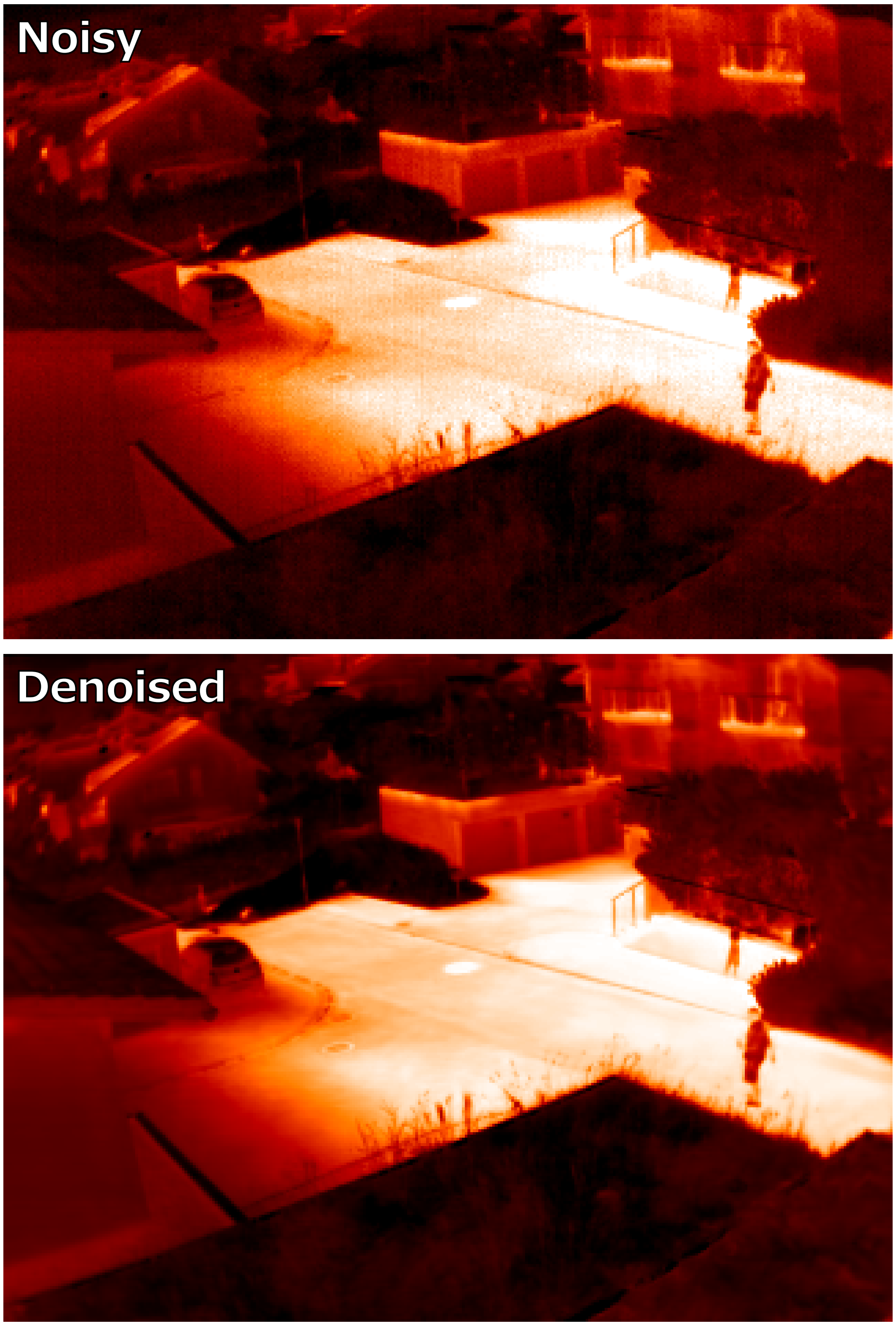

Figure 10. *IR*

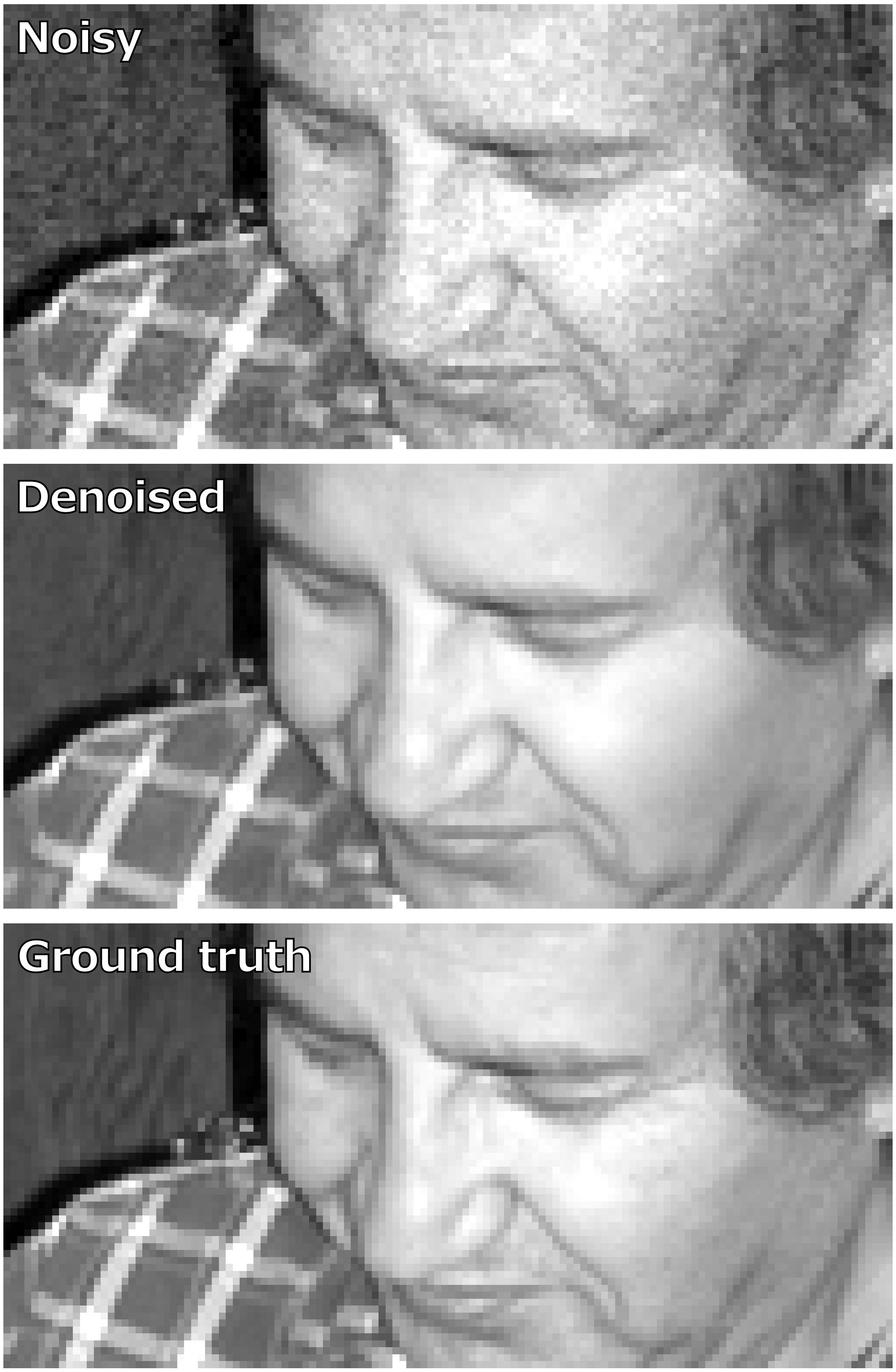

Figure 11. *FFHQ - Stripe*

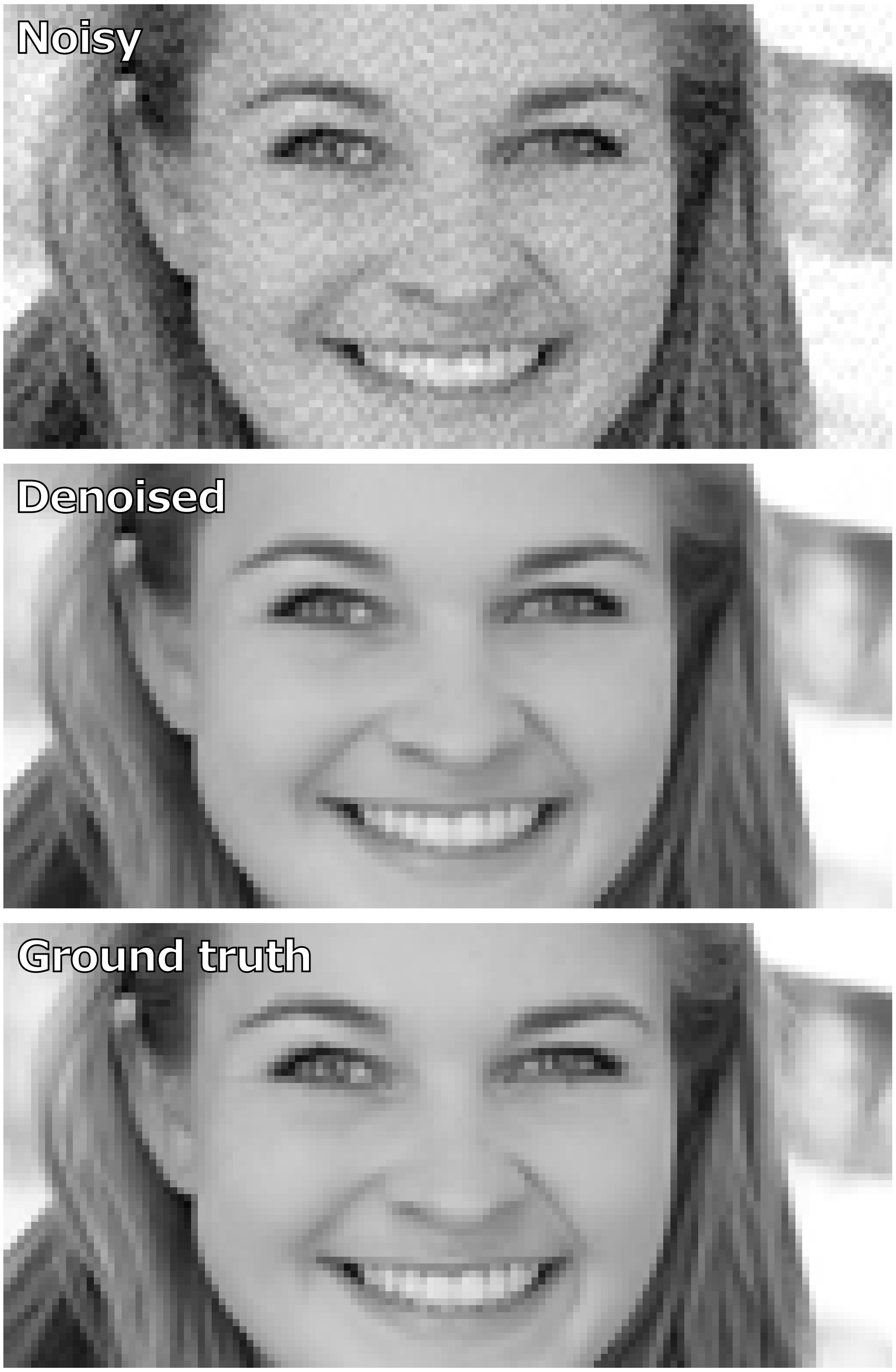

Figure 12. *FFHQ - Checkerboard*