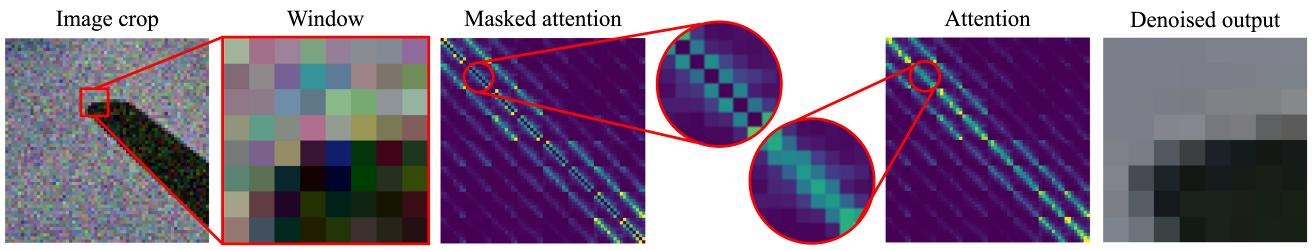

Figure 7. Example of masked (training, Eq. (5)) and unmasked (inference, Eq. (6)) window attention on an 8×8 window. The window contains 64 pixels, thus the maps are of size 64×64. The mask is applied additively before the softmax operation (Sec. 4.3).

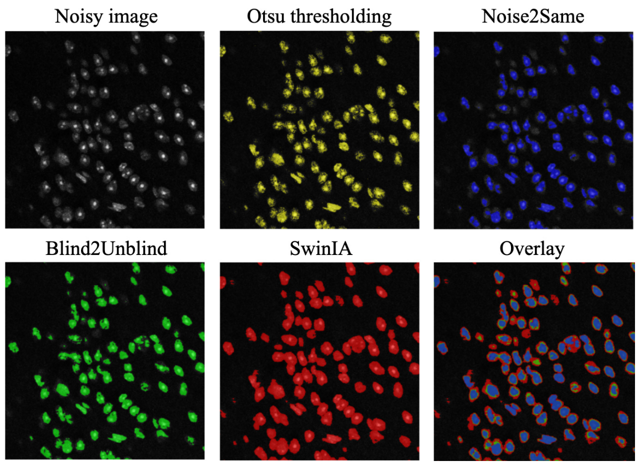

Figure 8. Binary masks produced by Otsu [25] and k-means clustering on model feature maps on FMD Confocal Mice [42].

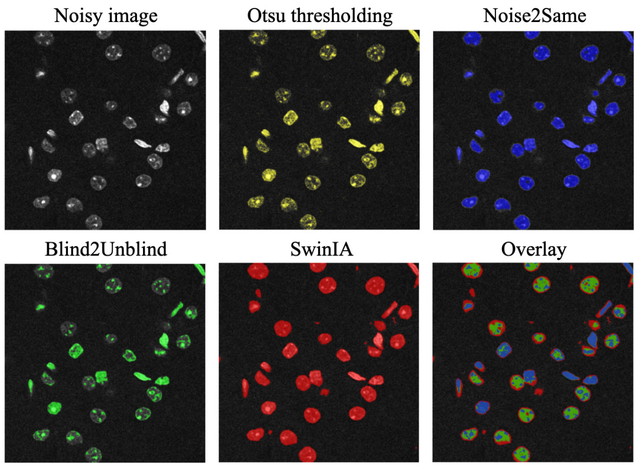

Figure 9. Binary masks produced by Otsu [25] and k-means clustering on model feature maps on FMD Two-Photon Mice [42].

A. Attention mask justification

Opposed to the multitude of methods based on pixel masking, our approach does not utilize random masks. More importantly, we do not mask pixels at all. Pixel mask, unlike our attention mask, has many disadvantages: it implies tuning hyperparameters, increased training time for multiple forward passes, or slower convergence. Furthermore, masked pixels may appear in the receptive field of their neighbors in training, thus affecting the denoising.

The attention is a parameter-free matrix, a dot product between trained embedding projections. Therefore, our attention mask neither hides any trainable parameters nor alters the input but simply cancels attention of each pixel to itself during training (1.6% of the full matrix). It enables our model to learn meaningful embeddings through neighboring pixel interaction without collapsing to identity. During inference, the attention scores on diagonal will remain high despite the masking during training, because they reflect pixel self-similarity. Therefore, it is only natural to unhide the main diagonal to let the model propagate the signal directly from each pixel. Fig. 7 illustrates this unblinding by comparing attention maps.

B. Embeddings analysis

We further investigate the features extracted by SwinIA in the last transformer block. In Figs. 8 and 9, we continue the comparison of binary segmentations performed on the original image with Otsu thresholding [25] and on the final feature maps of the models with k-means clustering into two clusters. We also include the clusterings of feature maps produced by Noise2Same [37] and Blind2Unblind [35]. For better comparison, we include the overlay of model embedding clusterings in these figures.

Compared over two microscopy datasets, clusters on the embeddings produced by SwinIA are more complete, sharp, and close to the shape of the cells than those of the counterparts. SwinIA also follows image semantics rather than raw pixel values in its embeddings: on Two-Photon Mice, every highlighted object is a cell, while for Blind2Unblind [35] clusters represent either full objects or their subcomponents depending on their brightness (note that the result is similar to Otsu intensity thresholding). It is worth noting that there is a high similarity between the embeddings of the pixels that are far from each other and never participate in the same attention window. This confirms that the training of

Dataset	Noise2Same		Blind2Unblind		SwinIA	
	TT (h)	AIT (ms)	TT (h)	AIT (ms)	TT (h)	AIT (ms)
Synthetic (sRGB)	4	26	—	35	10	416
Synthetic (grayscale)	2	12	—	—	10	239
ImageNet	1	14	25	206	10	554
HànZì	1	5	6	6	10	29
Microscopy	1.5	20	—	32	4	415

Table 6. Comparison of training time (TT) in hours, and average inference time (AIT) on the test set in milliseconds of Noise2Same [37], Blind2Unblind [35], and SwinIA (ours) on various datasets. For Blind2Unblind [35], we report the results of the experiments that we ran or re-evaluated ourselves, the code and the weights for experiments with synthetic grayscale noise are missing from the official repository.

Criterion	Noise2Same	Blind2Unblind	SwinIR	SwinIA
Number of trainable parameters	5.564M	1.100M	4.610M	3.966M
FLOPs/image $1 \times 64 \times 64$ (training)	10.002G	19.639G	18.978G	15.890G
FLOPs/image $1 \times 64 \times 64$ (inference)	5.001G	1.155G	18.978G	15.890G

Table 7. Comparison of the number of parameters and FLOPS between Noise2Same [37], Blind2Unblind [35], SwinIR [19], and SwinIA (ours). The number of FLOPs is calculated separately for training and inference.

SwinIA is well-regularized and aimed at extracting meaningful image features, equally good in global and local contexts. Embedding clusters show our model's potential as a universal self-supervised feature extractor for dense downstream tasks such as semantic segmentation.

C. Training details

We train SwinIA for 50 000 steps with batch size 64 and use Lion [4] with one cycle schedule [32] warming up for 15% of steps and then reducing learning rate from 3×10^{-4} to 10^{-6}. In experiments with FMD, we decrease the number of steps to 20 000 and the peak learning rate to 10^{-4}, because the dataset is small. During training, we randomly cut 64×64 crops from images scaled to $[0, 1]$, rotate them by multiples of $90°$, and flip them horizontally and vertically. Each crop is standardized with μ and σ of the training dataset. During validation, we pad each image for divisibility by 32 with reflection and crop the padding after prediction. For ImageNet in the mixed noise experiment (Sec. 5.2), we denoised overlapping tiles for several largest images and stitched them back before evaluation.

We implemented all models in Python 3.8.3 and PyTorch 1.12.1 [27] and trained them on NVIDIA A100 80GB GPUs (driver version: 470.57.02, CUDA version: 11.4). We used `einops` [29] for tensor permutations.

We compare the training time and average inference time of Noise2Same [37], Blind2Unblind [35], and SwinIA models in Tab. 6 and show the difference in the number of parameters and the number of floating point operations (FLOPs) per grayscale image in Tab. 7.

D. Denoising results visualization

In Figs. 10 to 12, we show additional denoising examples in experiments with synthetic Poisson noise, mixed synthetic noise, and real-world noise on microscopy data, respectively.

E. Model card

We present a model card with the main information and technical details about our SwinIA model in Tab. 8.

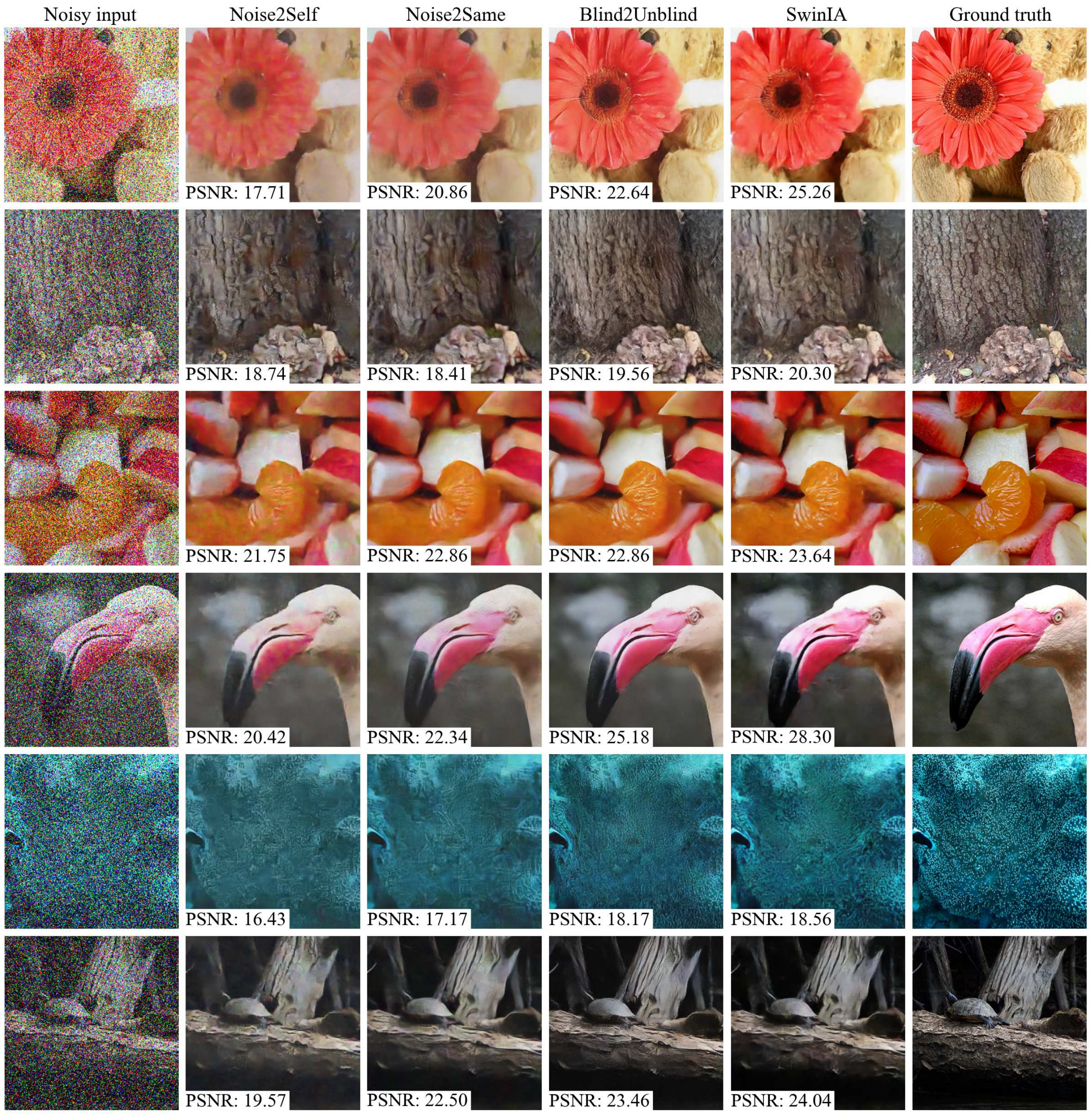

Figure 10. Denoising examples on ImageNet dataset with mixed synthetic noise [37]. Each row contains noisy and ground truth images, along with the predictions of Noise2Self [2], Noise2Same [37], Blind2Unblind [35], and SwinIA (ours) models with corresponding PSNR scores. Each image is center-cropped for visualization.

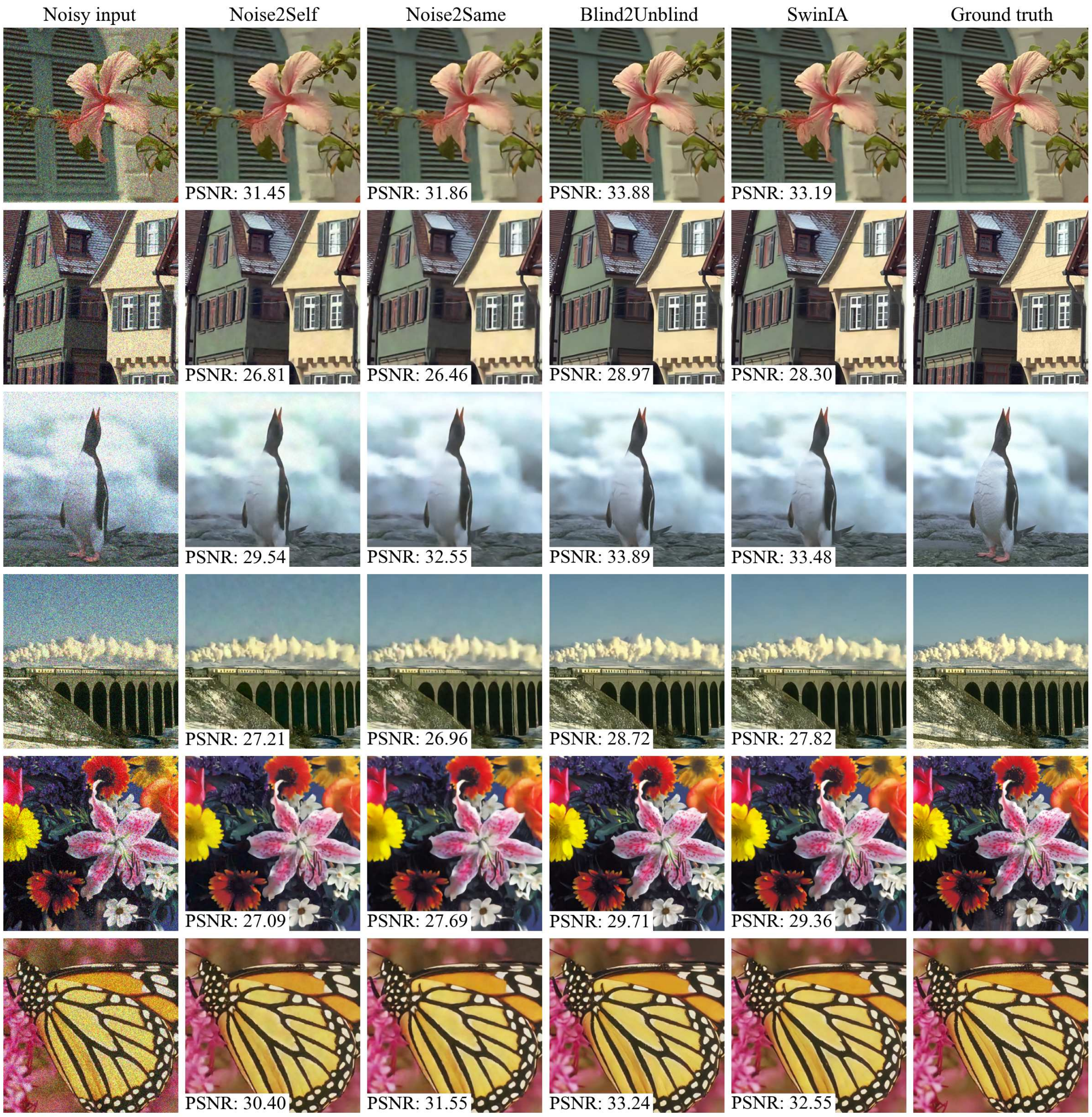

Figure 11. Denoising examples on sRGB data with synthetic Poisson noise ($\lambda = 30$). Each row contains noisy and ground truth images, along with the predictions of Noise2Self [2], Noise2Same [37], Blind2Unblind [35], and SwinIA (ours) models with corresponding PSNR scores. Each image is center-cropped for visualization.

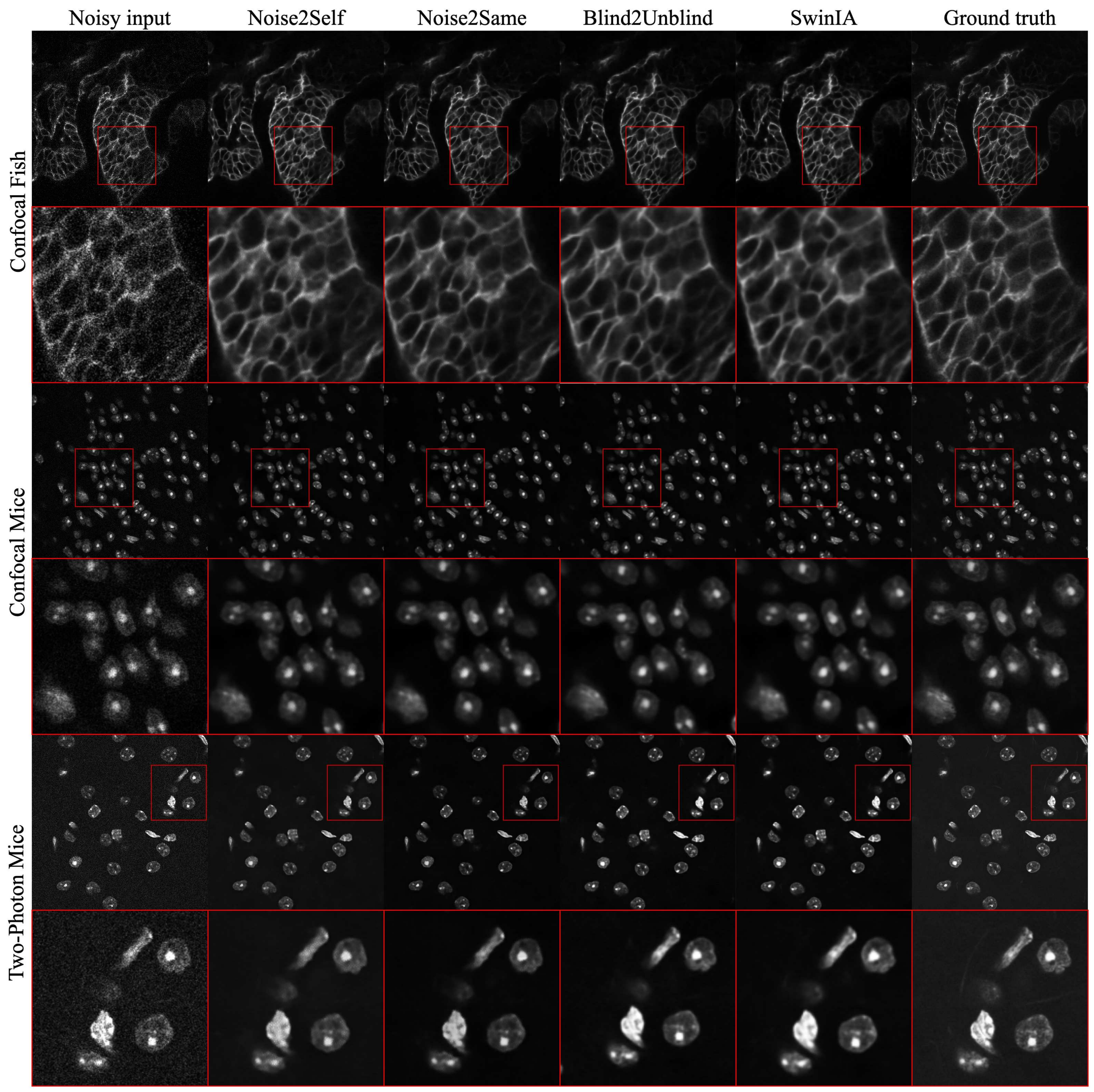

Figure 12. Denoising examples on fluorescent microscopy images with natural noise [42]. Each pair of rows contains noisy and ground truth images from the three used datasets (top — full size, bottom — zoomed in for a better view), along with the predictions of Noise2Self [2], Noise2Same [37], Blind2Unblind [35], and SwinIA (ours).

Table 8. Model Card of SwinIA.

Model Summary	
Model Architecture	Fully transformer-based image autoencoder model for end-to-end self-supervised image denoising with no convolutions. For details, see Sec. 4.
Input(s)	The model takes noisy images as input, batch and channel dimensions go first.
Output(s)	The model outputs a batch of denoised images of the same shape as input.
Usage	
Application	The model can be used in self-supervised image denoising for any type of spatially-uncorrelated synthetic and natural noise on both grayscale and colored images. Also, it is theoretically possible to use the model in self-supervised pre-training for extracting features from images for downstream vision tasks.
Known Limitations	The model is computationally expensive and requires both powerful GPU hardware and considerable training time. Also, small datasets will most probably lead to overfitting. Finally, the model will most likely learn an identity function on the data with spatially correlated noise without proper tuning of patch sizes (as mentioned in Sec. 6).
System Type	
System Description	This is a standalone model.
Dependencies	None.
Implementation Frameworks	
Hardware & Software	Hardware: NVIDIA A100 80GB GPUs (driver version: 470.57.02, CUDA version: 11.4). Software: Python 3.8.3, PyTorch 1.12.1 [27], einops [29].
Compute Requirements	In every experiment, SwinIA was trained on one NVIDIA A100 80GB GPU for different numbers of steps (see Sec. 5 for details).
Model Characteristics	
Model Initialization	The model is trained from a random initialization.
Model Status	This is a static model trained on offline datasets.
Model Stats	SwinIA model has 3.966 million trainable parameters and performs 15.89 GFLOPS (floating point operations per second) per 64×64 grayscale image.
Data Overview	
Training Datasets	Synthetic noise (sRGB): ILSVRC2012 [6] validation set. Synthetic noise (grayscale): BSD400 [41]. Mixture synthetic noise: ImageNet [37], HànZì [2, 37]. Natural noise (grayscale): Confocal Fish, Confocal Mice, and Two-Photon Mice datasets from the Fluorescent Microscopy Denoising Dataset [42].

Evaluation Datasets	Synthetic noise (sRGB): Kodak [9], BSD300 [23], Set14 [39]. Synthetic noise (grayscale): Set12 and BSD68 [30]. Mixture synthetic noise: ImageNet [37], HànZì [2, 37]. Natural noise (grayscale): Confocal Fish, Confocal Mice, and Two-Photon Mice datasets from the Fluorescent Microscopy Denoising Dataset [42].