Supplement
denoiSplit: a method for joint microscopy image splitting and unsupervised denoising

Ashesh Ashesh and Florian Jug

Fondazione Human Technopole, Viale Rita Levi-Montalcini 1, 20157 Milan, Italy
ashesh276@gmail.com,florian.jug@fht.org

Task	Model	training [h]	Noise level parameters							
			$\lambda = 0$				$\lambda = 1000$			
			$\sigma = 1$	1.5	2	4	$\sigma = 1$	1.5	2	4
T5	μSplit	5.5	25.1	23.7	23.1	22.1	24.8	23.8	23.0	22.1
			0.728	0.633	0.537	0.341	0.697	0.593	0.536	0.307
	HDN⊕μSplit	8	27.3	26.6	**26.1**	**24.6**	27.4	26.4	**26.1**	24.3
			0.793	0.756	**0.731**	**0.645**	**0.794**	0.739	**0.722**	**0.639**
	Altered μSplit ***(ours)***	1.3	**27.9**	26.8	25.9	24.5	27.4	26.5	25.9	**24.4**
			0.799	0.741	0.698	0.601	0.780	0.731	0.697	0.595
	denoiSplit ***(ours)***	1.5	27.7	**27.0**	25.9	24.5	**27.5**	**27.1**	26.0	**24.4**
			0.798	**0.762**	0.707	0.604	0.787	**0.761**	0.710	0.606
T6	μSplit	5	30.5	28.7	27.8	26.7	29.9	28.7	27.9	26.8
			0.869	0.779	0.685	0.422	0.866	0.788	0.708	0.443
	HDN⊕μSplit	7	34.6	33.1	32.1	28.8	33.8	32.6	32.0	28.8
			0.938	0.907	0.885	0.785	0.926	0.899	0.882	0.783
	Altered μSplit ***(ours)***	1.3	35.5	33.3	32.5	**30.4**	34.4	32.8	32.4	**30.1**
			0.950	0.911	0.890	0.825	0.935	0.898	0.887	0.819
	denoiSplit ***(ours)***	1.5	**36.3**	**34.4**	**33.1**	**30.4**	**34.7**	**33.8**	**32.9**	29.9
			0.945	**0.927**	**0.895**	**0.830**	**0.939**	**0.918**	**0.890**	**0.826**

Table 1: Quantitative Results. We show quantitative evaluations for two more tasks which are abbreviated as T5: F-actin vs. MT and T6: F-actin vs CCPs. For all experiments, we show the PSNR (sub-row 1) and MS-SSIM (sub-row 2) metrics across 8 noise levels: Gaussian noise levels of $\sigma \in \{1, 1.5, 2, 4\}$ and Poisson noise levels of $\lambda \in \{0, 1000\}$. The best performance per task and noise level is shown in bold. The third column additionally shows the training time on a single Tesla-V100 GPU (in hours).

1 Performance on more splitting tasks

In this section, we train our models and baselines on four more tasks. We train two tasks from the BioSR dataset. Specifically, we add F-actin *vs*. CCPs and F-actin *vs*. Microtubules tasks. In Tab. 1, we present the quantitative evaluation

on these two tasks. Similar to the results from Table 1 of the main manuscript, here as well, we find our methods, specifically denoi𝕊plit to outperform others in most cases.

We worked on three additional joint denoising-splitting tasks from two other datasets which we describe next.

Hagen et al. Actin-Mitochondria Dataset We picked the high-resolution Actin and Mitochondria channels from Hagen et al. [3] which were also used in [1]. Similar to our tasks from BioSR dataset, we added Gaussian and Poisson noise.

PaviaATN dataset [1] We worked with the Actin and Tubulin channel provided by the dataset. It is worth noting that in terms of PSNR, we picked the hardest of the three tasks worked upon by Ashesh et al. [1]. This is the task on which denoi𝕊plit and all the baselines perform poorly. We discuss more on these results in Sec. 10.

We provide the results on tasks generated from PaviaATN and Hagen et al. datasets in Tab. 4. We show the full-frame predictions for tasks not shown in the main manuscript in Figs. 5 to 8.

2 Details on architecture, hyperparameters, training and evaluation

As stated in the main manuscript, our denoi𝕊plit and *Altered μSplit* are built on top of μSplit architecture. In addition to the major changes that are discussed in the main manuscript, we have enabled *free bits* [5] parameter and have set *free bits* $= 1$. But similar to [1], we also upper bounded the log of variance of the latent space to 20 across all hierarchy levels for stability in training.

denoi𝕊plit, *Altered μSplit* and HDN have been trained with the learning rate of 0.001, batch size of 32, patch size of 128, max epoch of 400 and with 16 bit precision. For every task, 80% of the data was allocated as training data, 10% of the data as validation data, and 10% as the test data. For μSplit baseline and for μSplit used as part of *HDN*$\oplus$*μSplit*, we use the same training configuration as mentioned in [1]. For PSNR, we use the range invariant PSNR formulation which is commonly used in this field [1,14]. When working with Actin vs Mito task (Fig. 5 of the main text), we first scale the predictions in a way described in [14] and then use the Multiscale SSIM metric between the high-SNR groundtruth and the scaled prediction. The scaling is necessary because the high-SNR groundtruth has much higher pixel intensties than the low-SNR data on which the models are trained. Due to this the predictions also have lower pixel intensity values. We do not need to do this for PSNR separately because the version we use already does the scaling.

We have provided code with this supplement where information about how Poisson noise is added is present in the file *vanilla_ dloader.py:L175*. Specifically, we use numpy python package to add Poisson noise as

```
data = np.random.poisson(data / poisson_factor) * poisson_factor
```

We use poisson_factor = 1000. Finally, in Tab. 5, we provide the actual σ values which were used to add Gaussian noise in different tasks.

For the calibration plot shown in Figure 3 of the main text, 50 samples were used to estimate the RMV (Root mean variance).

2.1 Architecture details of μSplit

For completeness, here we describe all relevant aspects of μSplit [1], the work that we built upon. μSplit was built by modifying a HVAE framework and therefore inherited multiple hierarchy levels of latent spaces and the loss comprised of KL divergence and log-likelihood. In μSplit, Ashesh et al. [1] removed the auto-encoding nature of the HVAE framework by making the network predict two-channel output with input being a single channel. They therefore had two components in their log-likelihood loss, one for each output channel. In the log-likelihood loss of μSplit, a pixel-wise variance was predicted along with the split prediction for each channel. Note that in HVAE implementations, variance in the log-likelihood loss component is typically set to 1. μSplit's prediction, on the other hand, is a 4-channel tensor, two channels being the prediction and the other two being the predicted pixel-wise log-variance. As discussed in the main manuscript, one of our key contributions is that we integrated Noise models, originally developed for unsupervised denoising into the image decomposition task. We therefore did not need to predict the pixelwise log-variance. Next, compared to the classical HVAE formulation [10] of KL divergence loss, Ashesh et. al [1] relax the weights given to KL divergence loss components across different hierarchy levels, arguably to get better high-frequency details in the prediction. In our work, however, we find that this leads to a loss of denoising and sampling properties of the network, properties which HVAEs typically have. Therefore, as discussed in the main manuscript, we changed the KL loss weighting scheme used in μSplit and adopted the formulation used in [10]. Ashesh et al. [1] also showed the theoretical soundness of their approach by deriving the ELBO loss for the image decomposition setup. Please refer to [1] for the proof. Finally, one of their main contributions was the introduction of lateral contextualization (LC) wherein additional low-resolution images centered on the primary input patch, but covering larger spatial regions, were fed to the network through separate input branches. This enabled GPU-memory efficient assimilation of the information about the surrounding spatial context of the input patch. Since we primarily worked with the BioSR dataset which did not have as large structures as the ones used in [1], we disabled the LC module and instead doubled the input patch size.

3 Applications on natural images

While this work focuses on microscopy data, in this section, we briefly explore the utility of μSplit to tasks on natural images. Specifically, we look at deraining and de-hazing tasks. For de-hazing, we used Haze4K dataset [8] and for

de-raining, we used Rain100H dataset [15]. We worked with the original version of Rain100H dataset containing 1800 clean/rainy training image pairs and 100 clean/rainy testing image pairs. Due to the absence of pixel-independent noise in these datasets, we disabled the noise model in denoiSplit.

We present qualitative and quantitative results in Figure 1 and Tab. 2, 3. In the de-raining task, it was encouraging to observe that while training was done on images with synthetic rain, denoiSplit was able to remove rain from real rainy images as well.

Fig. 1: Qualitative results of denoiSplit on De-hazing (Haze4K dataset) and De-raining (Trained on Rain100H dataset) tasks.

	ID [4]	LP [7]	DSC [9]	JORDER-R [15]	denoiSplit
PSNR	14.02	14.26	15.66	23.45	26.2
SSIM	0.5239	0.4225	0.5444	0.7490	0.758

Table 2: Quantitative results on Rain100H dataset [15] for De-raining task. Due to the absence of noise, we disabled the noise model in our denoiSplit. Metric values of all other methods have been taken from [15].

It should be noted that the absence of noise in these datasets did not allow the primary feature of our approach, unsupervised denoising using noise models, to be used in these tasks. However, we believe our approach holds more promise on tasks on natural images having a significant amount of noise.

	DM^2F-Net [2]	FFA-Net [12]	DA [13]	DMT-Net [8]	denoiSplit
PSNR	24.61	26.97	24.03	28.53	27.1
SSIM	0.92	0.95	0.90	0.96	0.90

Table 3: Quantitative results on Haze4K dataset [8] for De-hazing task. Due to the absence of noise, we disabled the noise model in our denoiSplit. Metric values of all other methods have been taken from [8].

4 Practical relevance of denoiSplit

Here, we outline the intended usecase for our work. Similar to [1], our work also aims to enable microscopists to extract multiple structures using a single fluorescence marker. There are two important practical considerations which our work addresses.

Firstly, different microscopy projects, depending upon the nature of underlying specimen and microscope type, have different tolerances for the amount of laser power and the dwell time that can be used during acquisition. This roughly translates to the amount of noise that will be present in the acquired micrographs. In this work, we cater to this necessity by working with different noise levels.

Secondly, in most cases, there will be a need to purchase a single fluorescent marker that can bind to both cellular structure types one is interested in. Additionally, even after the purchase, it might still be challenging to get a decent staining of both structures with the marker. An imperfect staining can lead to under-expression of one of the two structure types in the imaged micrographs. Therefore, there is also an investment of time and expertise in getting a proper staining. That being the case, it makes sense to first inspect the feasibility of the approach before making the investment of buying a new marker followed by getting the staining correct. Our method provides a way to get proof-of-concept splitting without making any of these investments.

We envision that microscopists should image individual channels in the same noise regime as is permitted in their project. They should then train the denoiSplit and inspect the prediction quality. If there is room for adjustment in power and dwell time, they can re-acquire in a different noise regime and train the denoiSplit again. Note that this acquisition can be done with a single-color setup and therefore can be done using their existing microscope configuration.

In case they find the model performance satisfactory in some feasible noise regime, they can then order the relevant fluorescent marker and can subsequently label their structures with it to get superimposed images. Finally, they need to finetune the trained network to this slightly different input data and then they can start using denoiSplit. This problem of finetuning the model is outside the scope of this work and will be taken up in our future work.

denoiSplit also enables sampling which, as stated in the main manuscript, will enable microscopists to inspect the predictions to get a visual feeling about uncertain areas in prediction. To showcase this, we provide with this supplement,

a *sampling.gif* file where one could see 50 samples on a randomly choosen input from MT *vs*. ER task.

5 Noise model generation

Depending upon the physical availability of the microscope, [11] described two ways in which the noise model can be generated. In this section, our motivation is to assess the performance difference that one should expect between these two ways. We find that different choices of noise model generation do not lead to very different performances. This result enables denoi$\mathbb{S}$plit to be used across a wide variety of scenarios.

5.1 Physical availability of Microscope

The simpler and the better case is when we have access to the microscope which generated the noisy data. In this case, one needs to image the same content N times which would result in acquisition of N noisy versions of the same underlying specimen. The high SNR version is then computed by simply taking the pixel-wise average of these noisy samples. In this situation, for every clean signal value, one has N noisy intensities. This data is then used to train the Gaussian-mixture based noise model.

To simulate this condition, for every intensity value in the range $[0, 65535]$, we obtain multiple noisy intensity values by adding the noise (Gaussian and/or Poisson) multiple times independently.

5.2 Physical unavailability of Microscope

In case the microscope is not available, which is true for all publicly available microscopy datasets, [11] proposed a bootstrap noise model approach. In this method, the idea was to denoise the noisy data using some unsupervised/self-supervised denoising technique and use the noisy data and the predicted denoised data to generate the noise model.

To simulate this condition, we added noise to the noise free training data. This gave us the noisy data and clean data pair which we used to train the noise model. To showcase the applicability of our method to all publicly available datasets, we have generated all noise models using this approach.

It is worth noting that in the way described above, there is a source of error which has not been captured. This is the error introduced by the denoising process for getting the clean data from its noisy counterpart. To assess the effect of denoising, we generate the noise model in yet another way. In this way, we added the noise to the noise free training data. We then use N2V [6], an off-the-shelf denoiser to denoise the images. We use the noisy data and the denoised images to generate the noise model.

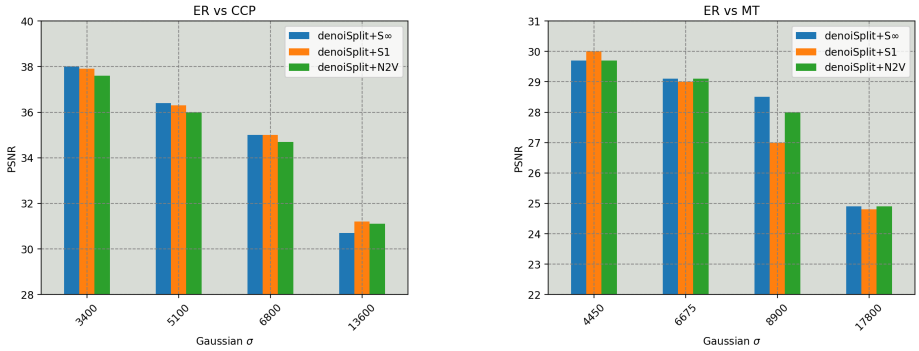

Fig. 2: Quantitative comparison of different noise model generation methodologies: Here, we compare three ways in which noise models can be generated. (a) denoiSplit+S_{∞}: When for every clean pixel intensity, one has access to multiple noisy intensities. This corresponds to the case when one has access to the microscope which has generated the data. (b) denoiSplit+$S1$: This corresponds the case when one has access to clean data and its corresponding noisy data. (c) denoiSplit+$N2V$: This corresponds to the case when one has access to just noisy data. We use N2V to denoise them which we use as clean data. We compare denoiSplit trained using each of the three noise model variants. We show PSNR performance on two tasks.

5.3 Performance comparison among different noise model generation procedures

To assess how performance varies depending upon the methodology used for noise model creation which have been described above subsections, we train denoiSplit three times, each time with its noise model computed using a different way. In Fig. 2, we show the performance comparison. denoiSplit+S_{∞} simulates the case when one has access to a microscope. For every clean pixel, one has multiple corresponding noisy pixels. denoiSplit+$S1$ denotes the case when one works with a noisy and the corresponding clean data pair. Note that, unlike the previous case, for every clean pixel, there is exactly one noisy pixel.

denoiSplit+$N2V$ denotes the case when one obtains the clean data after denoising with N2V. We train them on two tasks namely ER *vs*. CCPs and ER *vs*. MT over four Gaussian noise levels with Poisson noise ($\lambda = 1000$) enabled. As can be observed from the plots, we don't see a large difference between the three approaches.

This performance evaluation encourages denoiSplit to be used on publicly available datasets and for proof-of-concept evaluations since it shows that having access to the microscope does not give large performance improvements and the **bottleneck in denoising-splitting is essentially the splitting task**.

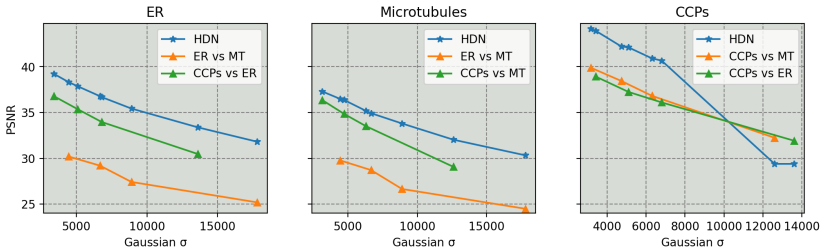

Fig. 3: Problem comparison: Joint Denoising-Splitting *vs*. Denoising In this figure, we compare the PSNR with respect to high-SNR micrographs for two image restoration tasks: (a) Self-supervised denoising for which we use HDN and (b) joint denoising-splitting where denoi𝕊plit is used. We evaluate the two models over multiple noise levels of Gaussian noise (x-axis) with Poisson($\lambda = 1000$) noise present in all cases. For denoi𝕊plit, we use three tasks namely: ER *vs*. MT, CCPs *vs*. ER and CCPs vs MT. We show one plot for each structure type (ER, Microtubules and CCPs). Two things are evident: (a) Judging just from PSNR numbers, we can say that Denoising is a simpler task than joint Denoising-Splitting. (b) Performance of denoi𝕊plit for one channel depends on the other channel as well. For example, for denoi𝕊plit, PSNR on ER channel (first plot) is higher when the task is CCPs *vs*. ER (green) as opposed to ER *vs*. MT (orange) task.

6 Quantifying model uncertainty

In this section, we quantify how much the performance varies between multiple models trained independently on the same task under identical configuration. For this, we picked ER vs MT task from BioSR dataset with $\sigma = 1, \lambda = 1000$. We trained the model 10 times and computed the PSNR and SSIM metrics on the test data. The mean and standard deviation of the PSNR values across different runs came out to be 29.84 ± 0.18 dB with individual PSNR values lying in $[29.7, 30.3]$ dB. For SSIM, it was 0.905 ± 0.004. Due to computing limitations, all the main text and the supplement tables use a single trained model per configuration to generate the metric values.

7 Comparison between denoising task and splitting task

In this section, we compare between two computer vision tasks which are of relevance to us: (a) unsupervised denoising and (b) joint denoising-splitting. From Fig. 3, we observe that unsupervised denoising is arguably a simpler task when compared to joint denoising-splitting. The PSNR between the prediction and high SNR micrographs is much better for HDN as compared to denoi𝕊plit for most cases. We note that it is expected because in joint denoising-splitting, besides denoising, which is the sole task in unsupervised denoising, one needs to additionally do the job of image decomposition. But more interestingly, we observe that the prediction quality of one channel depends upon the other channel.

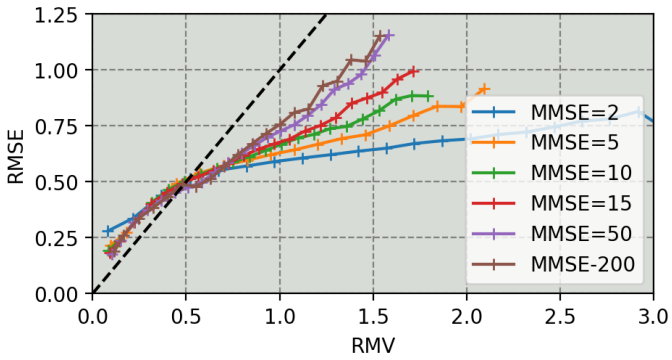

Fig. 4: In this figure, we show that as we increase the sample count to get the uncalibrated estimate of pixelwise uncertainty, the calibration diagram as shown in this plot becomes better and better, *i.e.*, gets more and more closer to $y = x$.

We note that this observation is of considerable importance because it opens up the question of best pairing strategy: which two structures should be imaged by a single color fluorescent marker? This will be part of our future work.

8 On usefulness of using sampling for calibration

In this section, we investigate our choice for estimating un-calibrated uncertainty using sampled predictions. For this, we estimate the uncalibrated pixelwise uncertainty using varying number of samples. We then follow our calibration procedure and learn the channelwise scalar to get the calibration plot. As can be observed in Fig. 4, as we increase the number of samples, the calibrated plot also improves thereby validating our choice.

9 Qualitative evaluation of HDN denoising

Here, we qualitatively evaluate the denoising behaviour of HDN. We show three random input patches and the corresponding channel first and channel second crops from 6 different splitting tasks in Figs. 9 to 14. We show the results on noisy dataset having Poisson($\lambda = 1000$) noise and Gaussian noise of relative scale 1.5. Since main manuscript also shows the qualitative figures on this noise level, we believe this can be used together with the figures present in main manuscript to better understand the behaviour of *HDN⊕μSplit*.

10 Failure cases: Avenues for future work

In this section, we inspect the worse performing cases in our work. One clear example is Actin *vs*. Tubulin task from PaviaATN [1] dataset whose quantitative

evaluation is present in Tab. 4. However, as can be seen from Fig. 15, we find performance of both *HDN⊕μSplit* and denoiSplit to be unsatisfactory for it to be used by microscopists. We note a striking difference of this task with the tasks from BioSR dataset. Looking at 128×128 input patches for tasks from BioSR data, one could visually form an opinion about which structures in the input patch should belong to first channel and which to the second channel. In case of Actin *vs*. Tubulin task, we observe that making this opinion is much more difficult since local structures are much less discriminative. It is only when looking at a larger context, one can form some opinion about which structures should be present in which channel. There seems to be another factor related to the nature of the structures which the channels are composed of. Informally speaking, individual channels have a "surface" like structure in Actin and Tubulin images of PaviaATN dataset as opposed to "curved lines", "mesh" and "dots" like structures in BioSR dataset.

Between denoiSplit and *HDN⊕μSplit*, we observe that for this task, denoiSplit has more tiling artefacts. It is not surprising for that to be the case because in [1], the Lateral contextualization (LC) approach which incorporates context in a memory efficient way, worked well on Actin *vs*. Tubulin. Compared to *HDN⊕μSplit*, our model is naturally at disadvantage because we have disabled the LC module but *HDN⊕μSplit* uses it. We believe that increasing the patch size can help our denoiSplit reduce the tiling artefacts.

In general, we also find cases where μSplit has retained some fine structures, albeit with noise, which the denoising based approaches have omitted from the prediction. We argue this to be a natural consequence of restricting the expressivity of latent spaces with KL divergence loss, which is pivotal for denoising.

As joint denoising-Splitting is a new task, there is much that needs to be done. We humbly acknowledge the challenges mentioned above which we hope to tackle in our future works.

Model	Tasks							
	T7				T8			
	1	1.5	2	4	1	1.5	2	4
μSplit	22.6	21.1	20.2	19.1	28.7	27.0	26.2	25.2
	0.555	0.442	0.361	0.189	0.905	0.825	0.747	0.489
HDN⊕μSplit	27.8	27.3	27.0	26.4	-	-	-	-
	0.880	0.871	0.865	0.843	-	-	-	-
(Ours) *Altered μSplit*	26.3	26.2	25.9	25.2	34.7	34.2	33.6	32.3
	0.838	0.826	0.820	0.807	0.975	0.971	0.966	0.951
(Ours) denoisplit	26.4	26.1	26.0	25.2	35.5	33.7	33.5	32.2
	0.835	0.827	0.825	0.807	0.979	0.974	0.968	0.951

Table 4: T7: Actin vs Tubulin from PaviaATN dataset, T8: Actin vs Mito High-SNR. For T8, HDN training was quite unstable and crashed multiple times due to NaNs. Due to this reason, there are no entries for *HDN⊕μSplit* for the task T8. Note that Actin vs Mito los-SNR task, which is present in the main text, also had trouble training HDN.

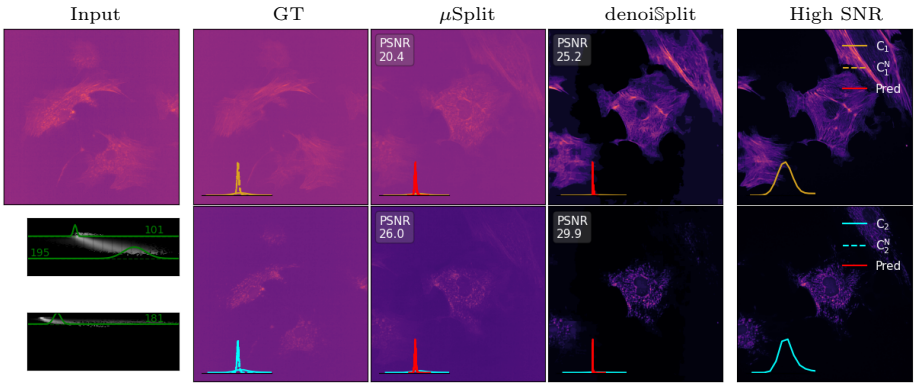

Fig. 5: Qualitative Results Actin *vs.* Mito: In this figure, we show full frame prediction on Actin *vs.* Mitochondria task. Here, the noise in the target channels is not synthetic but is real microscopy noise. We show noisy input (column one), individual noisy channel training data (column two), and predictions by one of the baselines μSplit (column three) and our own results obtained with denoiSplit (column four). Additionally we show high SNR channel images (not used during training) as the last column and show PSNR values w.r.t. these images. Additionally, we plot histograms of pixel intensities various panels for comparison (see legend on the right). The second row, first column shows the used noise models. The superimposed plots (green) show the distribution of noisy observations (c_i^N) for two clean signal intensities.

Task	$\sigma = 1$	$\sigma = 1.5$	$\sigma = 2$	$\sigma = 4$
ER*vs.* CCPs	3400	5100	6800	13600
ER *vs.* MT	4450	6675	8900	17800
CCPs *vs.* MT	3150	4725	6300	12600
F-actin *vs.* ER	4450	6675	8900	17800
F-actin *vs.* CCPs	3050	4575	6100	12200
F-actin *vs.* MT	4300	6450	8600	17200

Table 5: Gaussian σ values for the different tasks. Note that they have been estimated by computing the standard deviation on the input images of these tasks.

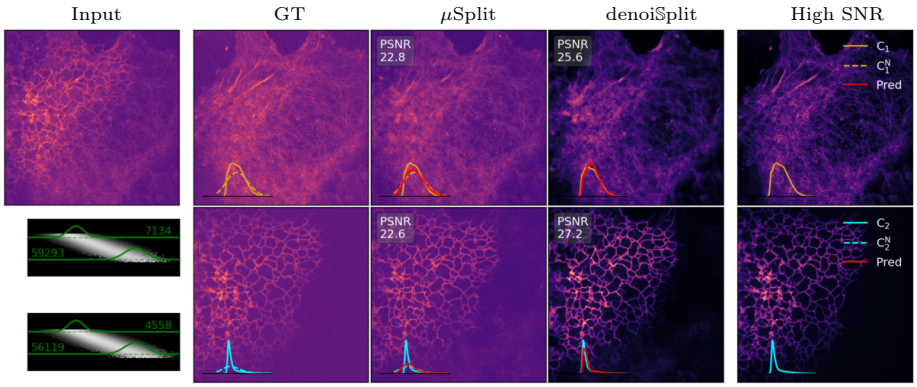

Fig. 6: Qualitative Results F-actin *vs.* ER: In this figure, we show full frame prediction on F-actin *vs.* ER task. We show noisy input (column one), individual noisy channel training data (column two), and predictions by one of the baselines μSplit (column three) and our own results obtained with denoiSplit (column four). Additionally we show high SNR channel images (not used during training) as the last column and show PSNR values w.r.t. these images. Additionally, we plot histograms of pixel intensities various panels for comparison (see legend on the right). The second row, first column shows the used noise models. The superimposed plots (green) show the distribution of noisy observations (c_i^N) for two clean signal intensities.

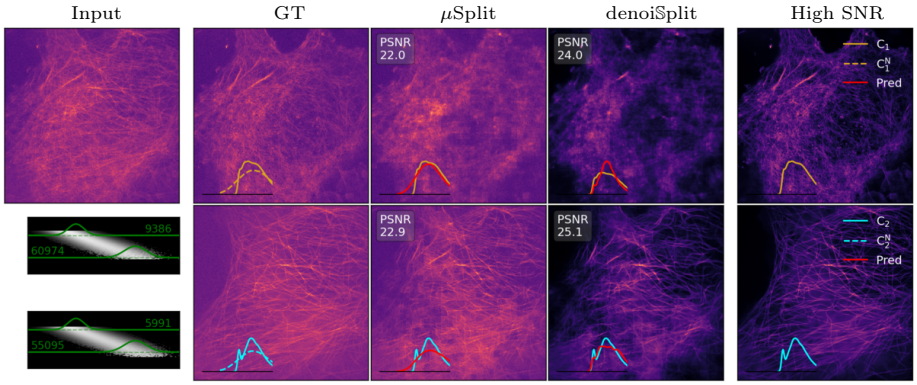

Fig. 7: Qualitative Results F-actin *vs.* MT: In this figure, we show full frame prediction on F-actin *vs.* MT task. We show noisy input (column one), individual noisy channel training data (column two), and predictions by one of the baselines μSplit (column three) and our own results obtained with denoiSplit (column four). Additionally we show high SNR channel images (not used during training) as the last column and show PSNR values w.r.t. these images. Additionally, we plot histograms of pixel intensities various panels for comparison (see legend on the right). The second row, first column shows the used noise models. The superimposed plots (green) show the distribution of noisy observations (c_i^N) for two clean signal intensities.

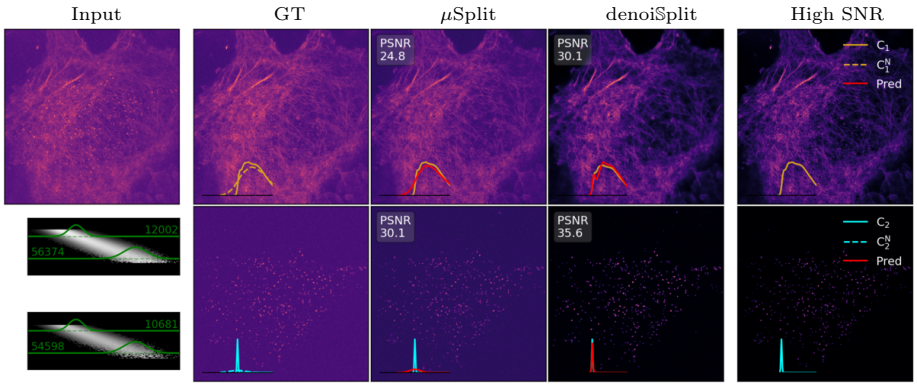

Fig. 8: Qualitative Results F-actin *vs.* CCPs: In this figure, we show full frame prediction on F-actin *vs.* CCPs task. We show noisy input (column one), individual noisy channel training data (column two), and predictions by one of the baselines μSplit (column three) and our own results obtained with denoiSplit (column four). Additionally we show high SNR channel images (not used during training) as the last column and show PSNR values w.r.t. these images. Additionally, we plot histograms of pixel intensities various panels for comparison (see legend on the right). The second row, first column shows the used noise models. The superimposed plots (green) show the distribution of noisy observations (c_i^N) for two clean signal intensities.

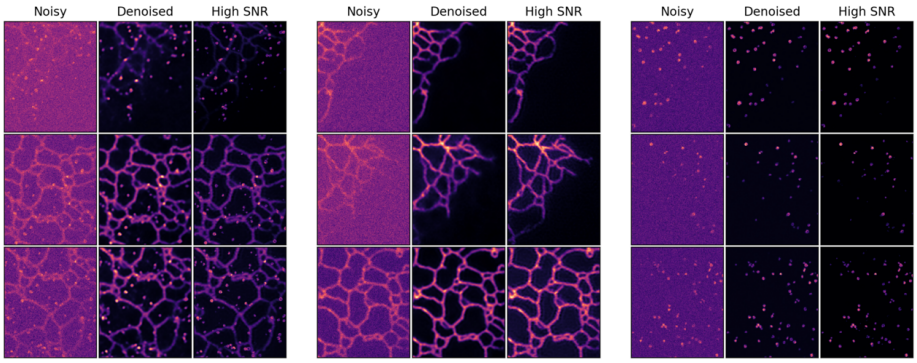

Fig. 9: Qualitative performance of HDN on ER *vs.* CCPs task input and its two constituent channels. Left panel shows the denoising performance on input for our splitting task and central and right panel shows its denoising performance on its two constituent channels. Within each panel, we show three random patches (rows) of size 256×213. Specifically, we show the input (first column), denoised predictions (second column) and high SNR patch (last column).

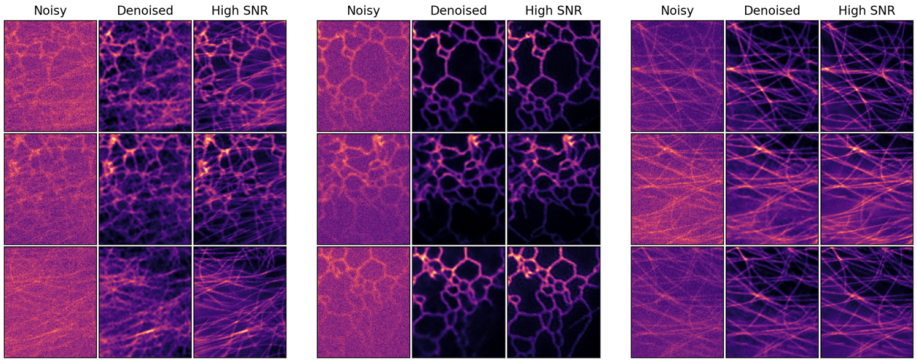

Fig. 10: Qualitative performance of HDN on ER *vs*. MT task input and its two constituent channels. Left panel shows the denoising performance on input for our splitting task and central and right panel shows its denoising performance on its two constituent channels. Within each panel, we show three random patches (rows) of size 256×213. Specifically, we show the input (first column), denoised predictions (second column) and high SNR patch (last column).

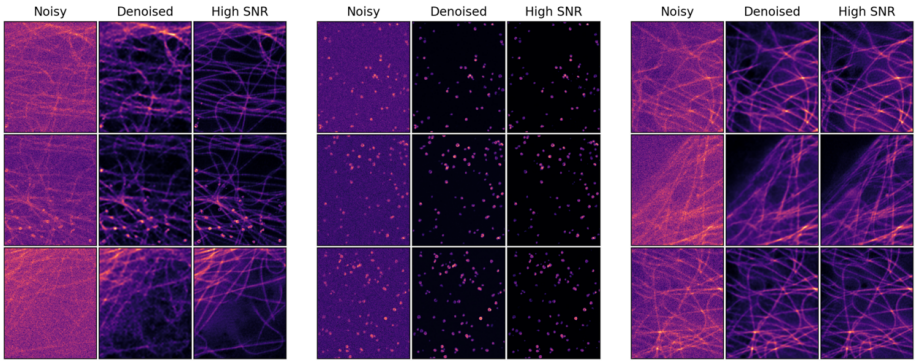

Fig. 11: Qualitative performance of HDN on CCPs *vs*. MT task input and its two constituent channels. Left panel shows the denoising performance on input for our splitting task and central and right panel shows its denoising performance on its two constituent channels. Within each panel, we show three random patches (rows) of size 256×213. Specifically, we show the input (first column), denoised predictions (second column) and high SNR patch (last column).

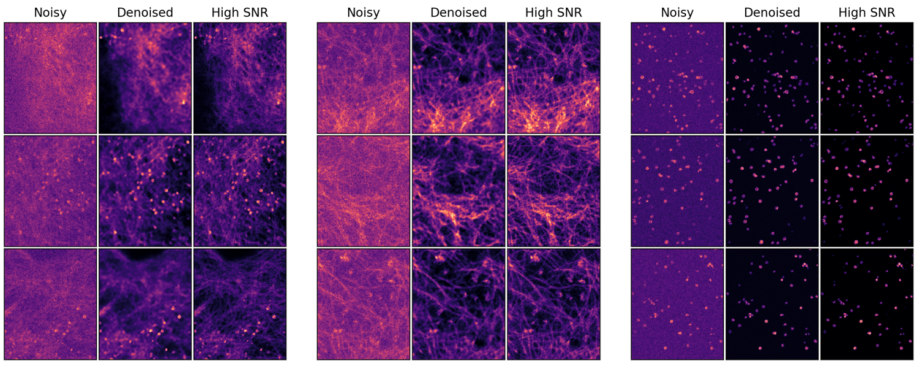

Fig. 12: Qualitative performance of HDN on F-actin *vs*. CCPs task input and its two constituent channels. Left panel shows the denoising performance on input for our splitting task and central and right panel shows its denoising performance on its two constituent channels. Within each panel, we show three random patches (rows) of size 256×213. Specifically, we show the input (first column), denoised predictions (second column) and high SNR patch (last column).

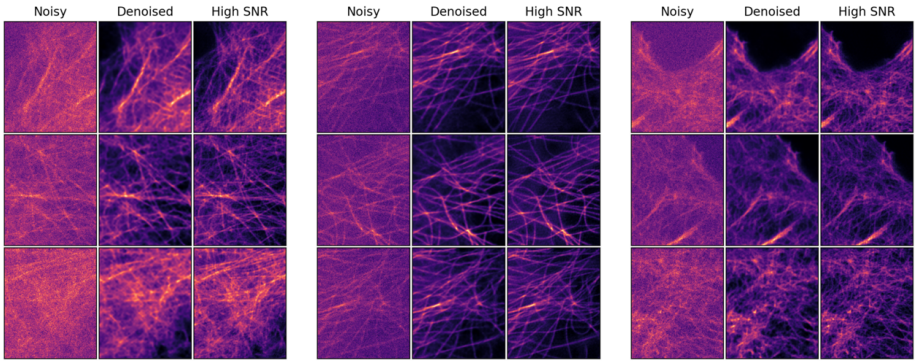

Fig. 13: Qualitative performance of HDN on F-actin *vs*. MT task input and its two constituent channels. Left panel shows the denoising performance on input for our splitting task and central and right panel shows its denoising performance on its two constituent channels. Within each panel, we show three random patches (rows) of size 256×213. Specifically, we show the input (first column), denoised predictions (second column) and high SNR patch (last column).

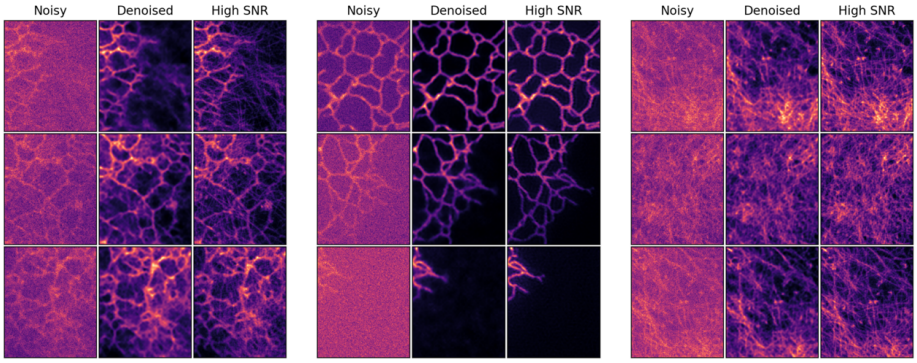

Fig. 14: Qualitative performance of HDN on F-actin *vs*. ER task input and its two constituent channels. Left panel shows the denoising performance on input for our splitting task and central and right panel shows its denoising performance on its two constituent channels. Within each panel, we show three random patches (rows) of size 256×213. Specifically, we show the input (first column), denoised predictions (second column) and high SNR patch (last column).

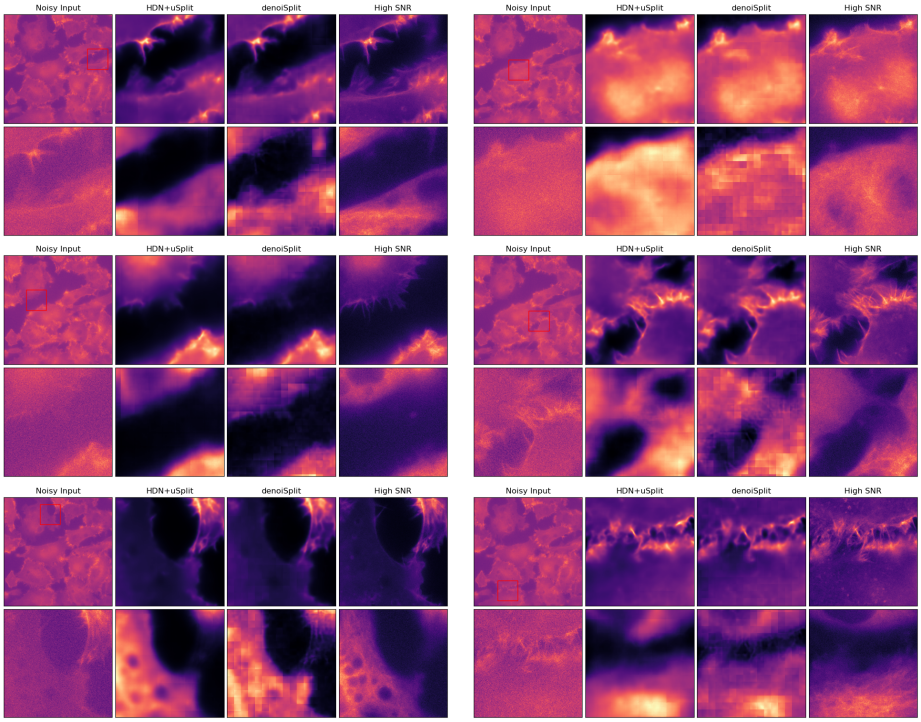

Fig. 15: PaviaATN Actin *vs.* Tubulin task Here, we show performance of denoiSplit and *HDN⊕µSplit* for six random input patches of size 500×500 in six panels. Within each panel, we show the full input frame and its crop for which we do the predictions (column one). Next two columns have the predictions of *HDN⊕µSplit* and denoiSplit respectively. The last column is the high SNR ground truth. We observe that the splitting performance of both *HDN⊕µSplit* and denoiSplit does not reach the quality at which microscopists would find it useful. Between *HDN⊕µSplit* and denoiSplit, we see more tiling artefacts for denoiSplit.

References

1. Ashesh, Krull, A., Sante, M.D., Pasqualini, F.S., Jug, F.: μSplit: image decomposition for fluorescence microscopy (2023)
2. Deng, Z., Zhu, L., Hu, X., Fu, C.W., Xu, X., Zhang, Q., Qin, J., Heng, P.A.: Deep multi-model fusion for single-image dehazing. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 2453–2462 (2019). https://doi.org/10.1109/ICCV.2019.00254
3. Hagen, G.M., Bendesky, J., Machado, R., Nguyen, T.A., Kumar, T., Ventura, J.: Fluorescence microscopy datasets for training deep neural networks. Gigascience **10**(5) (May 2021)
4. Kang, L.W., Lin, C.W., Fu, Y.H.: Automatic single-image-based rain streaks removal via image decomposition. IEEE Transactions on Image Processing **21**(4), 1742–1755 (2012). https://doi.org/10.1109/TIP.2011.2179057
5. Kingma, D.P., Salimans, T., Welling, M.: Improving variational inference with inverse autoregressive flow. CoRR **abs/1606.04934** (2016), http://arxiv.org/abs/1606.04934
6. Krull, A., Buchholz, T.O., Jug, F.: Noise2Void - learning denoising from single noisy images. arXiv **cs.CV**, 2129–2137 (Nov 2018)
7. Li, Y., Tan, R.T., Guo, X., Lu, J., Brown, M.S.: Rain streak removal using layer priors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)
8. Liu, Y., Zhu, L., Pei, S., Fu, H., Qin, J., Zhang, Q., Wan, L., Feng, W.: From synthetic to real: Image dehazing collaborating with unlabeled real data. In: Proceedings of the 29th ACM International Conference on Multimedia. p. 50–58. MM '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3474085.3475331, https://doi.org/10.1145/3474085.3475331
9. Luo, Y., Xu, Y., Ji, H.: Removing rain from a single image via discriminative sparse coding. In: 2015 IEEE International Conference on Computer Vision (ICCV). pp. 3397–3405 (2015). https://doi.org/10.1109/ICCV.2015.388
10. Prakash, M., Delbracio, M., Milanfar, P., Jug, F.: Interpretable unsupervised diversity denoising and artefact removal (Apr 2021)
11. Prakash, M., Krull, A., Jug, F.: DivNoising: Diversity denoising with fully convolutional variational autoencoders. ICLR 2020 (Jun 2020)
12. Qin, X., Wang, Z., Bai, Y., Xie, X., Jia, H.: Ffa-net: Feature fusion attention network for single image dehazing. ArXiv **abs/1911.07559** (2019), https://api.semanticscholar.org/CorpusID:208138077
13. Shao, Y., Li, L., Ren, W., Gao, C., Sang, N.: Domain adaptation for image dehazing. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2805–2814 (2020). https://doi.org/10.1109/CVPR42600.2020.00288
14. Weigert, M., Schmidt, U., Boothe, T., M uuml ller, A., Dibrov, A., Jain, A., Wilhelm, B., Schmidt, D., Broaddus, C., Culley, S., Rocha-Martins, M., Segovia-Miranda, F., Norden, C., Henriques, R., Zerial, M., Solimena, M., Rink, J., Tomancak, P., Royer, L., Jug, F., Myers, E.W.: Content-aware image restoration: pushing the limits of fluorescence microscopy. Nature Publishing Group **15**(12), 1090–1097 (Dec 2018)
15. Yang, W., Tan, R.T., Feng, J., Liu, J., Guo, Z., Yan, S.: Deep joint rain detection and removal from a single image. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1685–1694 (2017). https://doi.org/10.1109/CVPR.2017.183