μSplit: image decomposition for fluorescence microscopy SUPPLEMENT

Ashesh[1], Alexander Krull[2], Moises Di Sante[3], Francesco Silvio Pasqualini[3], Florian Jug[1,*]
[1]Human Technopole, Italy, [2]University of Birmingham, UK, [3]University of Pavia, Italy
ashesh.ashesh@fht.org, a.f.f.krull@bham.ac.uk, moises.disante@unipv.it
francesco.pasqualini@unipv.it, florian.jug@fht.org

S.1. The Architecture of Lean-LC

In this section, we describe the *Lean-LC* architecture, our most GPU memory efficient LC variation (see Supplementary Figure S.1). In this architecture, lateral contextualization is only used along the bottom-up branch. The top-down branch therefore reduces to a regular HVAE, in our case just as the one used in [2]. To make the laterally contextualized bottom-up branch funnel into the regular (vanilla) HVAE top-down branch, the output of each BottomUp block feeding into the corresponding TopDown block is appropriately centercropped. Hence, the latent tensors in the top-down branch are smaller, leading to the reduced memory footprint.

S.2. Padding used in Tiling

S.2.1. Issue with Outer Padding

Here we introduce two terminologies needed to explain the issue with Outer Padding. Assuming an infinitely large input or intermediate tensor, we define its *theoretical receptive field* to be the subset of tensor entries which can influence a single output pixel (see Figure 3 of the paper). Given a finite tensor size, governed by a fixed input patch size (*e.g.* 64×64) we define the *effective receptive field* analogously as the subset of tensor entries which can influence a single output pixel. Note that the theoretical receptive field is either identical to the effective receptive field or larger (see Figure 3).

As we use a deep network and work with 64×64 sized input patches during training, the theoretical receptive field (with grows up to about 500×500) is much larger than the effective receptive field (which cannot grow beyond 64×64). Given that the network has the capacity to see a large region but a much smaller patch is fed as input, a natural question to ask is: what does the network 'see' beyond the input, *i.e.*, its effective receptive field? The answer is that it sees zeros due to zero padding present in *same-convolution* operations of PyTorch. Hence, the network is accustomed to see lot of zeros during training.

At evaluation time, if we use Outer Padding, we increase the patch size, and therefore the effective receptive field. Now, suddenly, the network sees a lot fewer zeros and it is therefore not surprising that the quality of predictions degrades when so much more input is fed through the network. In such cases, the network will effectively start operating out of distribution (OOD) with respect to the training data that was consistently fed at the same patch size of 64×64.

Importantly, when using Inner Padding, we do not change the patch size during tiled predictions and are therefore avoiding to operate the trained network OOD.

S.2.2. Qualitative Results for Inner Padding

In Supplementary Figure S.3, we compare results obtained with Inner Padding, Outer Padding, and without padding. For this we evaluated on random patches of size 400×400 from the Actin vs Nucleus dataset. One can easily spot square-shaped artefacts when no padding is used. With Inner Padding, they are generally much improved. Outer Padding also leads to a reduction of these artefacts, but generate other artefacts leading to degraded performance in the splitting task at hand.

On the full dataset, PSNR drops from 31.4 dB PSNR without padding to 30.2 dB with Outer Padding. Using Inner Padding, on the other hand, improves the obtained results to a PSNR of 31.8 dB (+0.4 dB).

S.2.3. Deep-LC makes padding less important

We compare results obtained with a vanilla HVAE, our architecture using *Lean-LC*, and our variant using *Deep-LC*. In Table S.1, we report the PSNR we achieve on four datasets with all of the architectures, once using no padding during prediction, the other time using Inner Padding of 24 pixels. On average, the performance when using Inner Padding improved by 0.33 dB PSNR for the vanilla HVAE, 0.18 dB when using *Lean-LC*, and 0.02 dB when using *Deep-LC*. This supports our claim that *Deep-LC* brings

*Corresponding Author.

Dataset	Vanilla		*Lean-LC*		*Deep-LC*	
	P0	P24	P0	P24	P0	P24
Act vs Nuc	31.4	31.8	33.7	33.8	33.9	33.9
Act vs Mito	31.4	31.9	32.4	32.7	34.3	34.4
Act vs Tub	25.0	25.2	27.5	27.7	28.6	28.6
Tub vs Nuc	29.4	29.6	31.8	31.9	32.5	32.5

Table S.1. LC can offset the importance of padding. The table shows results by a vanilla HVAE and our μSplit with *Lean-LC* and *Deep-LC*. For each model, we show the achieved PSNR without padding (P0) and when Inner Padding of 24 pixels (P24) was used during prediction. One can observe that the difference between the two columns for each architecture becomes increasingly smaller, suggesting that LC helps to avoid artefacts typically addressed with padding during tiled predictions.

enough lateral context that padding becomes generally less important.

S.3. The PaviaATN Data

The PaviaATN dataset comprises static lambda-stacks from a human keratinocytes cell line (HaCaT) expressing GFP-tubulin, RFP-LifeAct, and a customized version of the cell cycle indicator FastFUCCI that uses various combinations of a yellow fluorescent protein (YPF, mTurquoise2) and a far-red fluorescent protein (iRFP, miRFP670) to indicate multiple phases of the cell cycle. While the details of how this cell line was genetically engineered will be published separately, here we used it to create a challenging dataset for μSplit which we planning publish with this manuscript. In fact, when a cell is in the G1 phase, increasing intensities of YFP fluorescence are detected in the nucleus. As a cell moves from G1 to S phase (G1/S), both YFP and iRFP fluorescence are detected in the nucleus of the cell. Finally, the sole iRFP fluorescence is detected in the nucleus during the S-G2-M phase. At the onset of the G1 phase, the nucleus shows no visible fluorescence intensity. All Images are acquired through the 100x silicon oil objective of a Nikon Ti2 microscope (100x silicon oil objective) equipped with an Okolab environmental control chamber and a Crest V3 spinning disk confocal in widefield mode. Excitation light was provided by a Lumencor Celesta laser engine set up to provide 5% laser power to the 446, 477, 546, and 637 nm lines. Emission light was collected through the following filters Semrock FF01-511/20, 595/31, 685/40. This configuration can spectrally separate the signals from actin (RFP) and S-G2-M cell cycle phases (iRFP). Instead, GFP and YFP exhibit a degree of overlap in excitation (446 and 477) and emissions (through the Semrock 595/31 filter), which we seek to resolve with μSplit. Since all combinations of YFP and GFP variants have spectral overlap, we expect the μSplit results to be very relevant for the field.

S.4. Metrics

We use PSNR and SSIM (structural similarity) to quantitatively measure the quality of predictions. When reporting PSNR, we use the commonly used shift invariant variation introduced in [3]. We compute both SSIM and PSNR metrics on normalized data.

S.5. More Quantitative Results

Figure S.2 shows the achievable results of a U-Net, a vanilla HVAE, and our μSplit variations for the PaviaATN Actin vs Nucleus data. Plots are as the ones in the main figure. One can observe the outperformance of LC variants with respect to the Vanilla baseline. On this task we observe that the *Deep-LC* architecture does not lead to additional improvements over the other LC variations. This can be explained by the nucleus channel being relatively easy to separate from the actin channel, without requiring much lateral image context to perform the task well.

S.6. More Qualitative Results

All qualitative results in Supplementary Figures S.4, S.5, S.6 and S.7 are showing predictions on randomly chosen patches of size 300×300.

All qualitative results figures show randomly chosen patches in two rows, each one showing one of the superimposed image channels. The superimposed input region is shown in the first column and the last column shows ground truth. All other columns show predictions from various model configurations.

Supplementary Figures S.4, S.5, S.6 and S.7 all show results for HVAE variations, *i.e.* comparing the vanilla architecture, with the ones utilizing *Lean-LC*, and *Deep-LC*.

In Supplementary Figure S.8, we show performance on random inputs of our SinosoidalCritters data.

S.6.1. Comparison with Double-DIP

In Supplementary Figure S.9, we qualitatively compare μSplit's performance with Double-DIP's performance. We note that Double-DIP, being a completely unsupervised approach, naturally finds it difficult to know the 'correct' split, the split which exists in nature. It simply returns one of the many plausible splitting options. Its inferior performance argues for some form of supervision for our problem.

S.7. Different Neural Network Submodules

Residual Block We've taken the residual block formulation from [2]. The schema for the residual block is shown in Supplementary Figure S.1 **(b)**. The last layer in the residual block is the GatedLayer2D which doubles the number of channels through a convolutional layer, then use half the channels as gate for the other half.

BU Blocks	PSNR
1	29.8
2	31.3
4	33.2
5	33.2
6	33.0

Table S.2. The achievable performance using a U-Net using various numbers of bottom-up (BU) blocks. For the results reported in the main text, 5 BU blocks have been used.

Stochastic Block The channels of the input of this block are divided into two equal groups. The first half is used as the mean for the Gaussian distribution of the latent space. The second half is used to get the variance of this distribution, implemented via the σ_{ExpLin} reformulation introduced in [1].

S.8. U-Net Tuning

We varied the depth of the used U-Net. For consistency with the other used architectures, we decided to still call it BottomUp (BU) blocks (HAEs and HVAEs grow upwards, not downwards.) Table S.2 shows the achievable performance with U-Nets of different depth (number of BU blocks).

Other relevant hyperparameter values used for U-Nets are patience $=$ 200 for early stopping , patience $=$ 30 for the learning rate scheduler (ReduceLROnPlateau).

References

[1] David Dehaene and Rémy Brossard. Re-parameterizing VAEs for stability. June 2021. S.3

[2] Mangal Prakash, Alexander Krull, and Florian Jug. DivNoising: Diversity denoising with fully convolutional variational autoencoders. *ICLR 2020*, June 2020. S.1, S.2, S.4

[3] Martin Weigert, Uwe Schmidt, Tobias Boothe, Andreas M uuml ller, Alexander Dibrov, Akanksha Jain, Benjamin Wilhelm, Deborah Schmidt, Coleman Broaddus, Siân Culley, Maurício Rocha-Martins, Fabián Segovia-Miranda, Caren Norden, Ricardo Henriques, Marino Zerial, Michele Solimena, Jochen Rink, Pavel Tomancak, Loïc Royer, Florian Jug, and Eugene W Myers. Content-aware image restoration: pushing the limits of fluorescence microscopy. *Nature Publishing Group*, 15(12):1090–1097, Dec. 2018. S.2

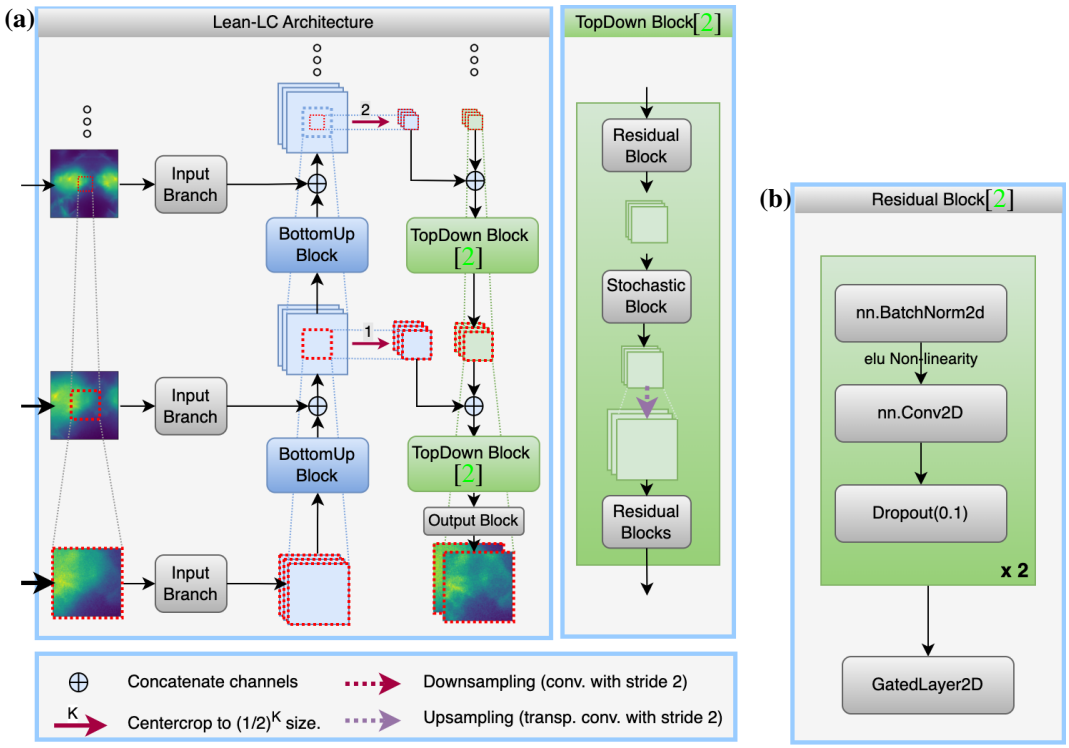

Figure S.1. Lean-LC and Residual block architecture. **(a)** The network architecture of Lean-LC variant is shown here. The Bottom-Up block remains unchanged from the architecture of LC and it is Top-Down block which has changed. If we look at k^{th} Bottom-Up block (from bottom), then as before, the output from the k^{th} Bottom-Up block is passed to the next Bottom-Up block and also to the Top-Down block to the same level. However, before feeding to the Top-Down block, the output is center-cropped to $(1/2^k)^{\text{th}}$ size. The Top-Down block of LC-Lean is identical to the Top-Down block of /citePrakash2020-wr. Input to the block passes through Residual blocks and then through the stochastic block. The output of the Stochastic block is upsampled to twice its size through Transposed convolution with stride of 2. **(b)** The schema for the Residual Block. This is directly taken from [2].

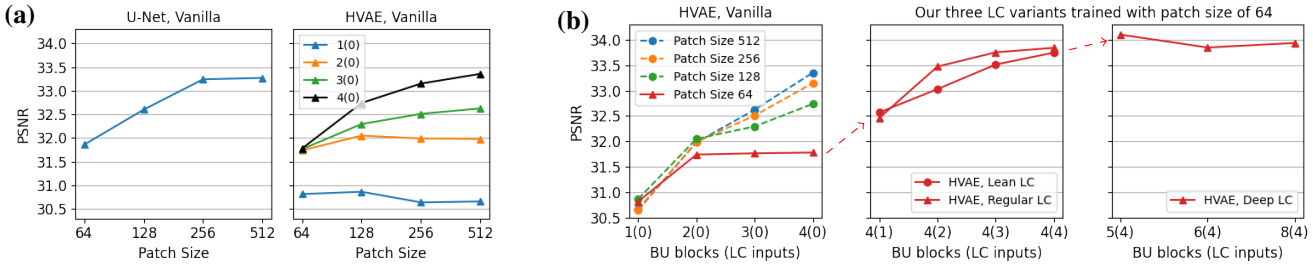

Figure S.2. **Benefits of μSplit in one glance:** Quantitative results of baselines *vs.* μSplit variants on our *PaviaATN* Act *vs.* Nuc task. In Figure 6 of the paper, we do the identical analysis on our *PaviaATN* Act *vs.* Tub task. **(a)** We plot the performance of the vanilla U-NET and the vanilla HVAE baseline trained on increasingly larger patch sizes. The U-NET performance plateaus roughly at a patch size of about 256. The performance of the vanilla HVAE (not using LC) depends on how many hierarchy layers we use (1 to 4, different colored plots), but then plateaus as well, or requiring a tremendous amount of GPU memory (black plot, also see Table 1. **(b)** The left plot displays the data as shown in the HVAE plot in (a), but now as a function of hierarchy levels in the used architecture. Each curve is now representing a given patch size. X-axis ticks express how many hierarchy levels the HVAE has, and how many of those make use of LC (number in brackets). The rightmost two plots show results obtained with μSplit using an HVAE with a patch size of only 64. Each plot shows results results obtained with one of our LC variations being used. Not only do networks using LC outperform all baselines, they do so already when using the smallest patch size (64), thereby requiring only a moderate amount of GPU memory (see Table 1). Note that here, *Deep-LC* has lesser advantage as was the case with Act vs Tub data.

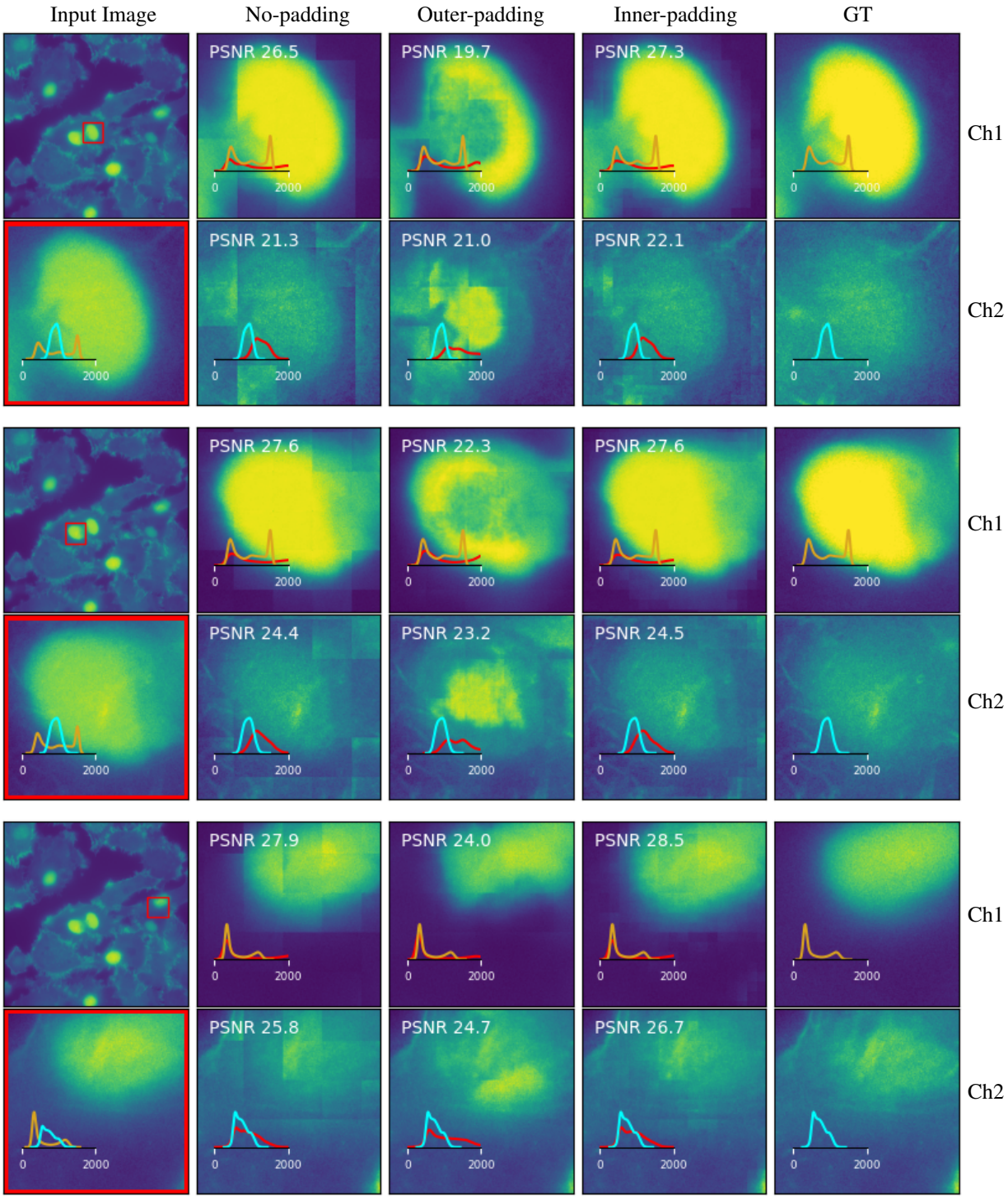

Figure S.3. Comparing *No Padding*, *Inner Padding* and *Outer Padding* on the Actin vs Nucleus task. Here, we disentangle the region present inside the red square (column one) using Vanilla HVAE model. One can see square shaped artefacts arising during tiling of predictions in 'No Padding' case, a case where no padding is used during the tiling operation. But more importantly, use of 'Outer Padding' leads to quite inferior splitting results where pixel intensity which was supposed to be present in the nuclues region 'leaks' into the actin channel (unexpected bright region in actin channel's prediction). This naturally degrades the splitting results of both channels. Inner tiling, on the other hand yields most consistent splitting results with respect to the ground truth (last column).

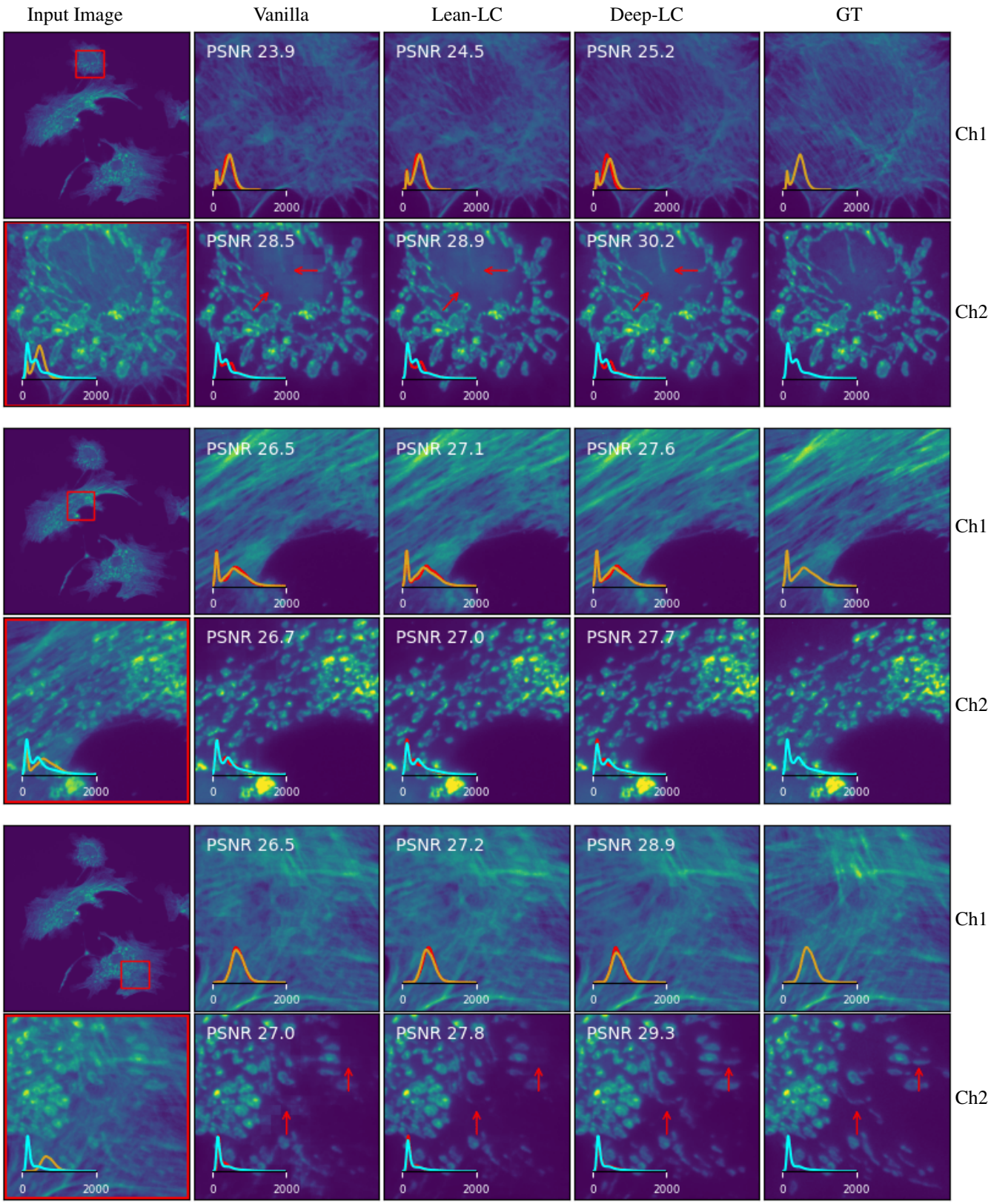

Figure S.4. Qualitative evaluation of Vanilla HVAE and our LC variants (also integrated to HVAE architecture) on Actin vs Mitochondria task. Here, we show results on three random crops of size 300×300. Input to all models is the region inside red square, as seen in column one. Last column has the ground truth for both channels. Red arrows highlight few interesting areas where we observe our *Deep-LC* performs better than others.

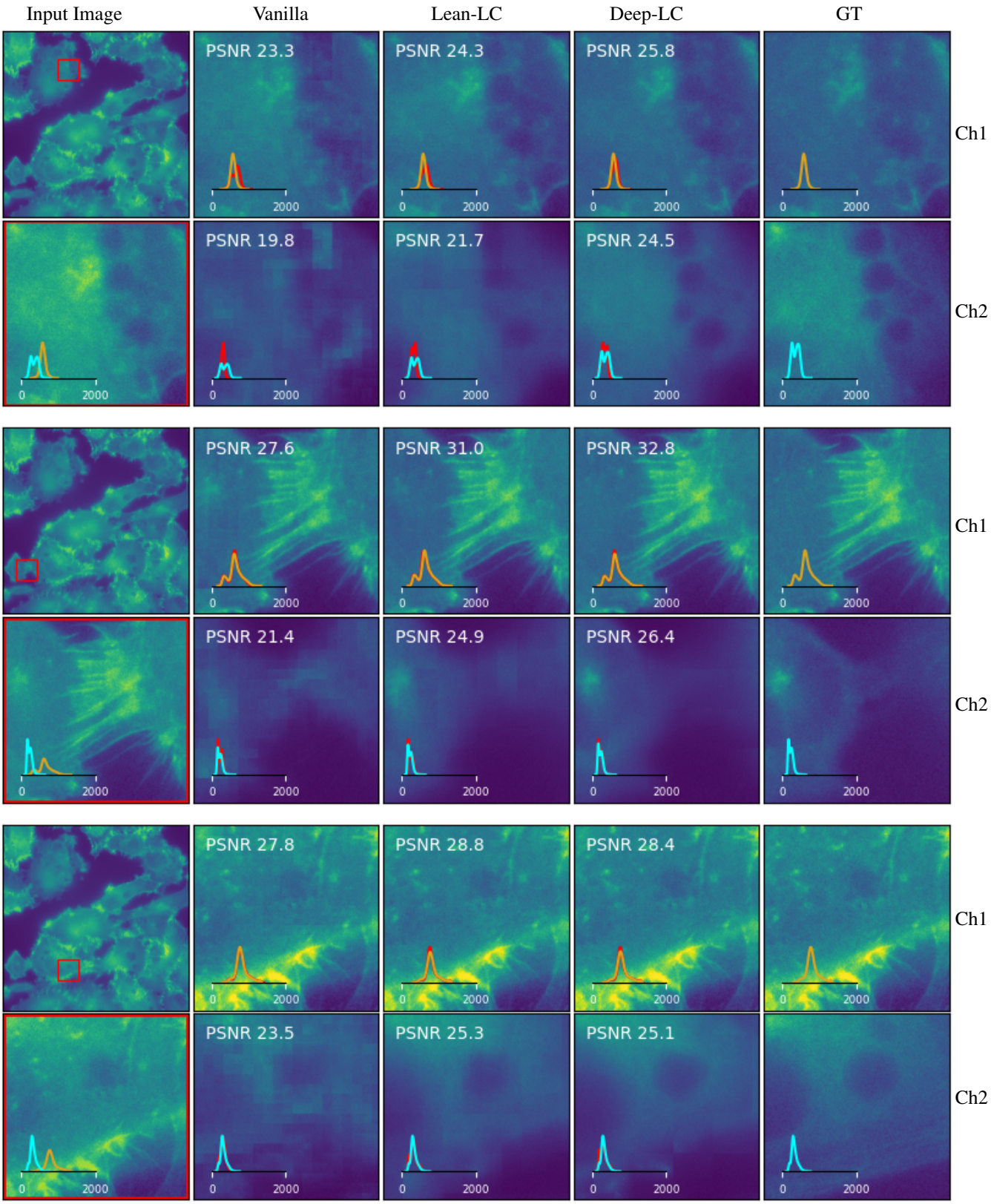

Figure S.5. Qualitative evaluation of Vanilla HVAE and our LC variants (also integrated to HVAE architecture) on Actin vs Tubulin task. Here, we show results on three random crops of size 300×300. We disentangle the region inside red square, which is shown in column one. Last column has the ground truth for both channels.

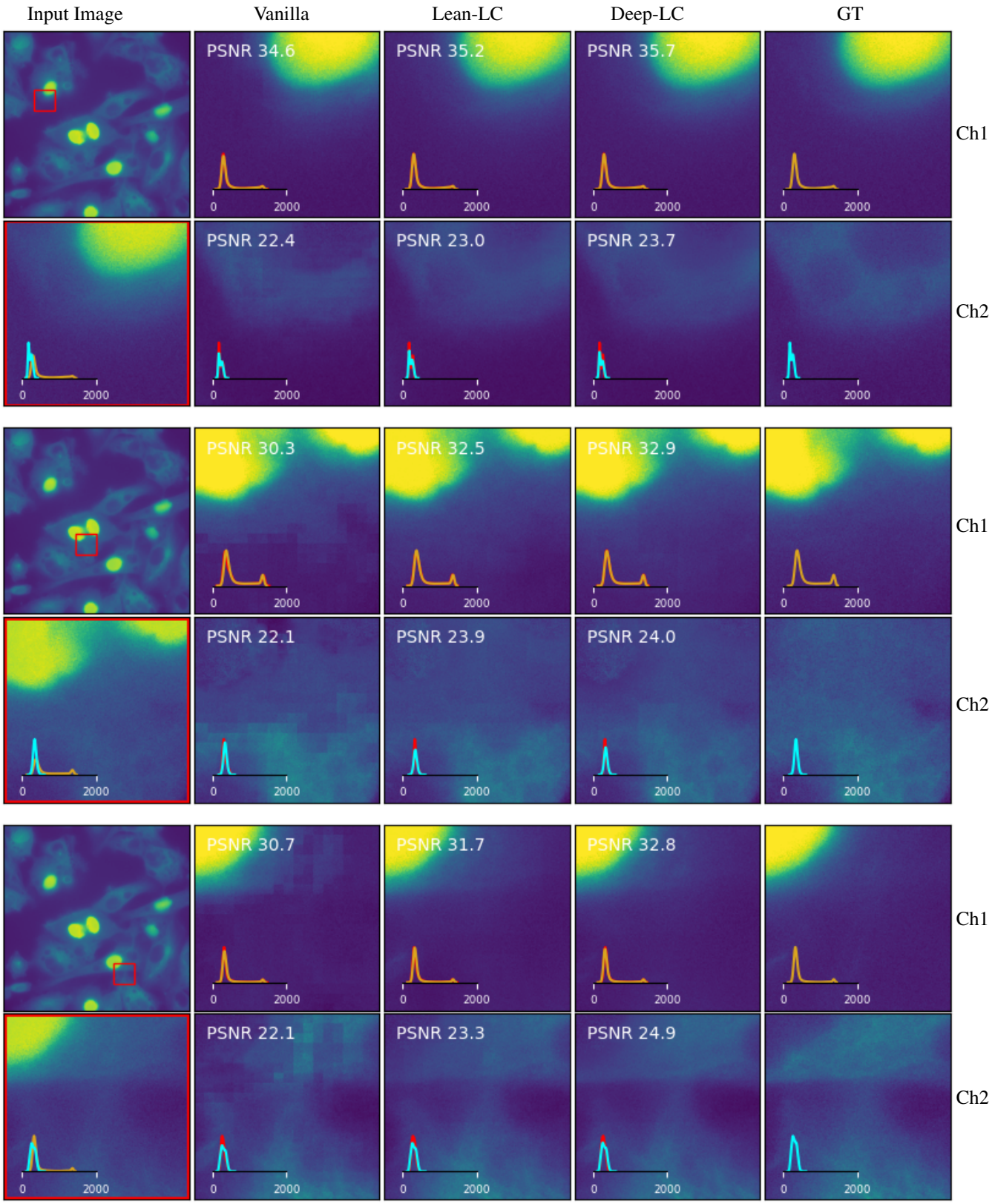

Figure S.6. Qualitative evaluation of Vanilla HVAE and our LC variants (also integrated to HVAE architecture) on Tubulin vs Nucleus task. Here, we show results on three random crops of size 300×300. We disentangle the region inside red square, which is shown in column one. Last column has the ground truth for both channels.

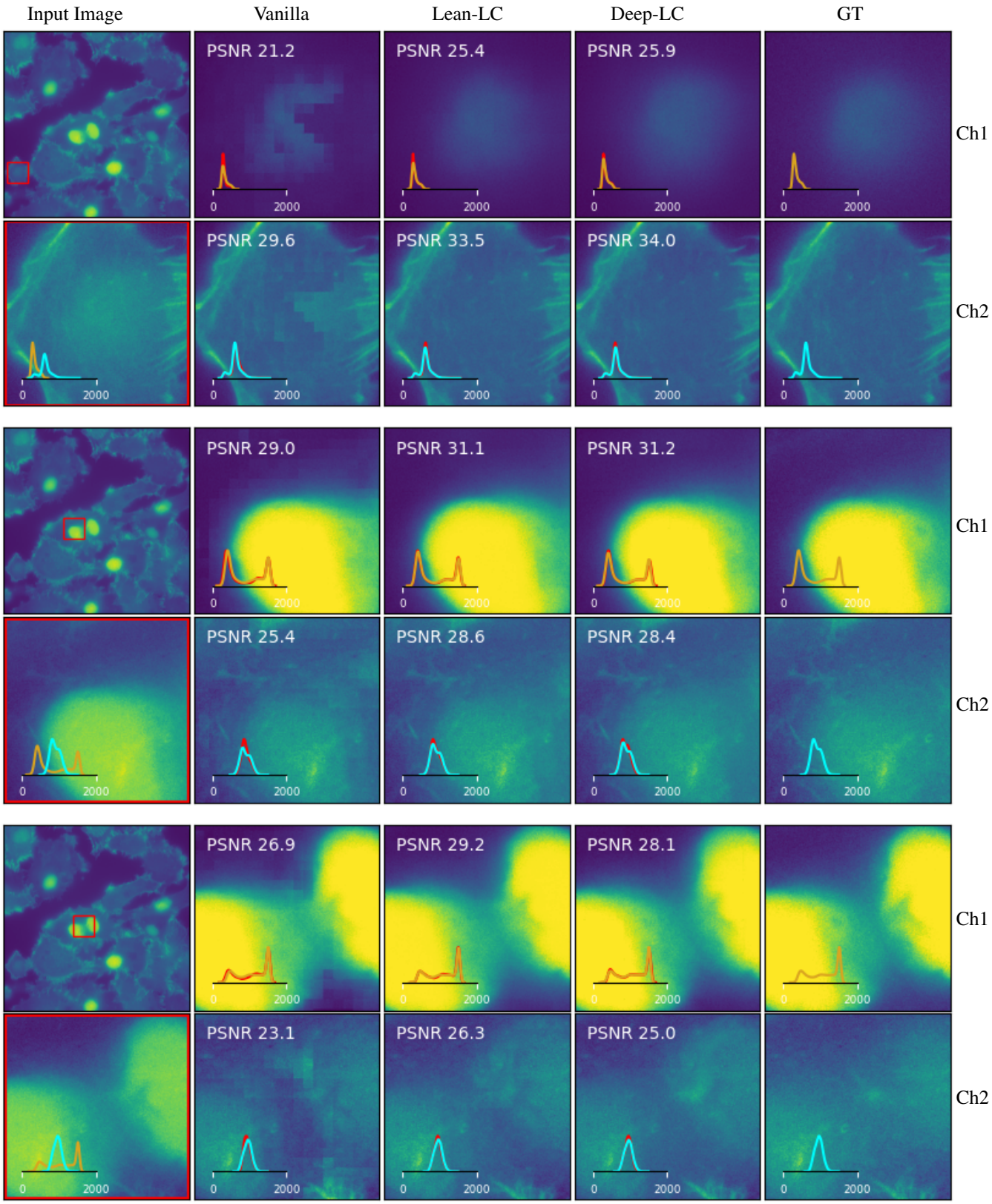

Figure S.7. Qualitative evaluation of Vanilla HVAE and our LC variants (also integrated to HVAE architecture) on Actin vs Nucleus task. Here, we show results on three random crops of size 300×300. We disentangle the region inside red square, which is shown in column one. Last column has the ground truth for both channels.

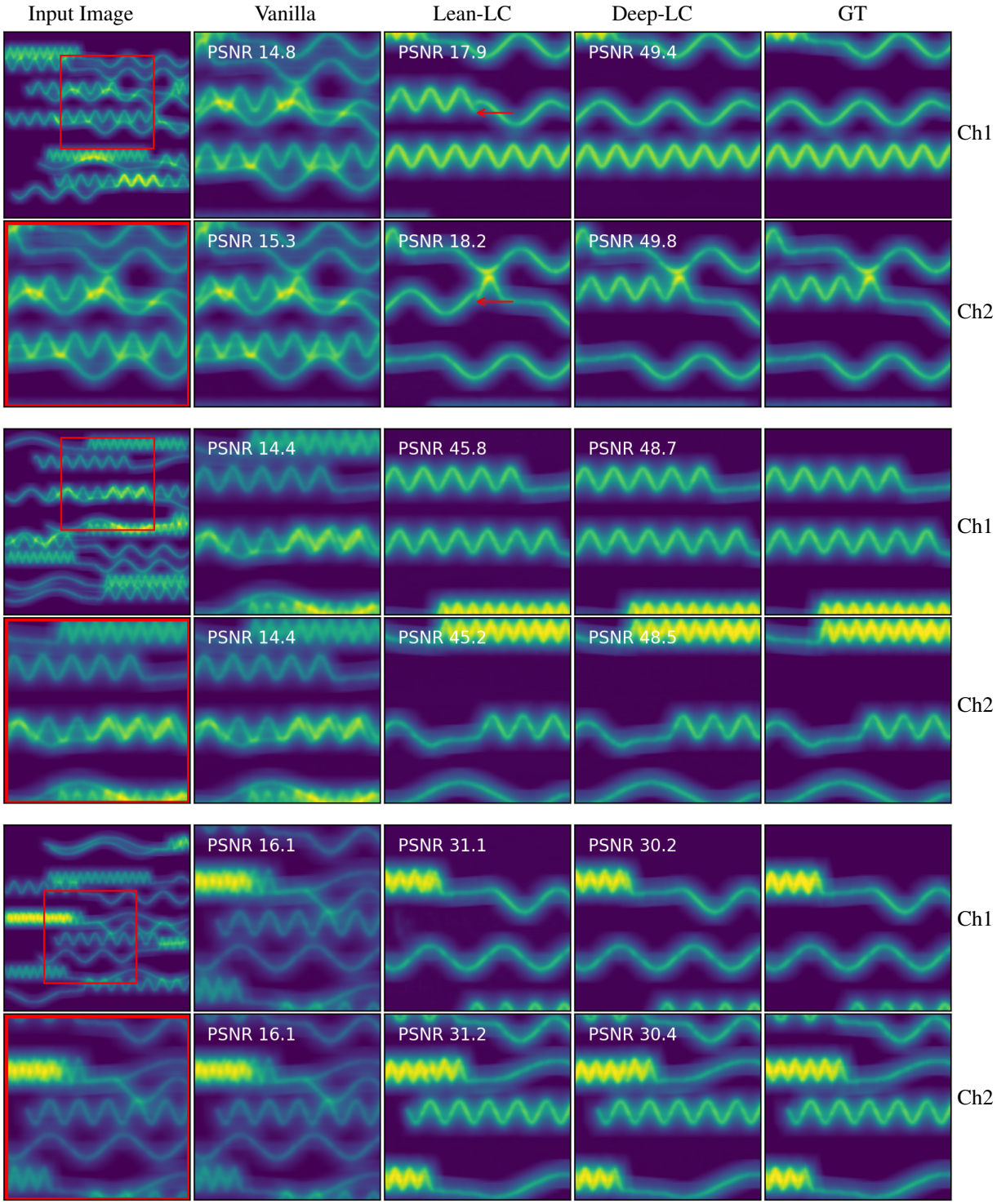

Figure S.8. Qualitative evaluation of Vanilla HVAE and our LC variants (also integrated to HVAE architecture) on SinosoidalCritters dataset. Here, we show results on three random crops of size 200×200. We disentangle the region inside red square, which is shown in column one. Last column has the ground truth for both channels. Red arrows highlight few interesting areas where we observe our *Deep-LC* performs better than others.

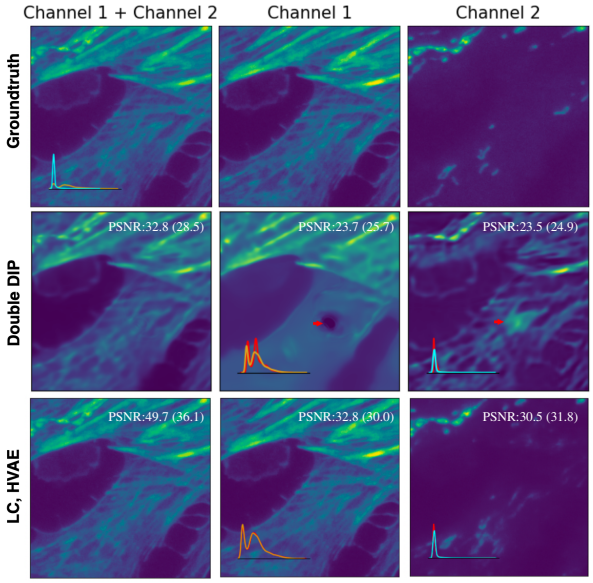

Figure S.9. Qualitative image decomposition results using the Double-DIP baseline (row 2) on an 256×256 image crop from *Hagen* et al. dataset. The overlaid histograms shows either the intensity distribution of the two channels (column 1) or the intensity distribution of the ground truth and the prediction (red). *Regular-LC*, on the other hand, performs well. Note that Double-DIP is solving a much harder task since it is an unsupervised method trained on a single input images.