μSplit: image decomposition for fluorescence microscopy

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

Abstract

We present μSplit, a dedicated approach for trained image decomposition in the context of fluorescence microscopy images. We find that best results using regular deep architectures are achieved when large image patches are used during training, making memory consumption the limiting factor to further improving performance. We therefore introduce lateral contextualization (LC), a novel meta-architecture that enables the memory efficient incorporation of large image-context, which we observe is a key ingredient to solving the image decomposition task at hand. We integrate LC with U-Nets, Hierarchical AEs, and Hierarchical VAEs, for which we formulate a modified ELBO loss. Additionally, LC enables training deeper hierarchical models than otherwise possible and, interestingly, helps to reduce tiling artefacts that are inherently impossible to avoid when using tiled VAE predictions. We apply μSplit to five decomposition tasks, one on a synthetic dataset, four others derived from real microscopy data. Our method consistently achieves best results (average improvements to the best baseline of 2.25 dB PSNR), while simultaneously requiring considerably less GPU memory. Our code and datasets can be found at https://github.com/juglab/uSplit.

1. Introduction

Fluorescence microscopy [10] is routinely used to look at living cells and biological tissues at cellular and sub-cellular resolution [18]. Components of the imaged cells can be highlighted using fluorescent labels, allowing biologists to investigate individual structures of interest. Given the complexity of biological processes, it is typically necessary to look at multiple structures simultaneously, typically via a temporal multiplexing scheme [10] that separates them into different image channels.

Imaging more than 3 or 4 structures in this way is difficult for technical reasons, limiting the rate of scientific progress in the life sciences. One way to circumvent this limitation would be to label two cellular components with the same fluorophore, *i.e.* image them in the same image channel. Hence, a computational method to split apart (decompose) superimposed biological structures acquired in a single image channel, *i.e.* without temporal multiplexing, would have tremendous impact (see Figure 1).

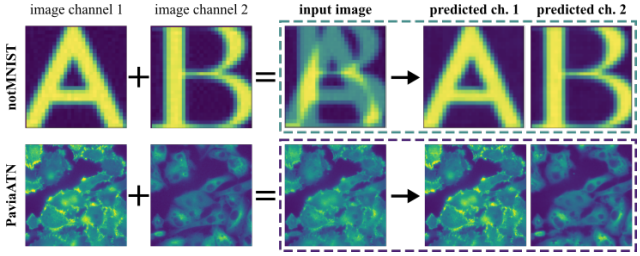

Figure 1. Splitting of superimposed image channels. The input image is the sum of two image channels, each channel containing structures from one given object class. The task of μSplit is to identify and split the structures superimposed in the given input image (dashed rectangles).

Historically, image decomposition has found applications on natural images [9, 8, 1, 3]. Our approach for image decomposition, called μSplit, rests on the idea of learning structural priors for the two unmixed target image channels, and then using these to guide the decomposition of the superimposed (added) pixel intensities. Such content-aware priors have previously been used for tasks such as image restoration [29, 4, 28], denoising [14, 2, 15, 11, 20, 19], and segmentation [5, 25, 30].

In many of these cases, the achievable performance heavily depends on the portion of the image a network can see before having to make a prediction. As we show in this work, the need for large spatial context, *i.e.* receptive field and patch size, is particularly pronounced for image decomposition. Biological structures in microscopy images can easily extend over distances of several hundred pixels. Accordingly, we observe that results improve with larger training patch sizes and deeper architectures (see Figure 6(a)). Naturally, this leads to models having a huge GPU memory footprint, which limits their applicability to selected compute-savvy life-science labs.

The importance of context has previously been utilized in the field of image segmentation [16, 13]. Leng *et al.* [16] devised a method to efficiently use the available context of

*Corresponding Author, (florian.jug@fht.org).

the input image for a segmentation task. However, they did not use additional inputs for having access to a larger context than what is already present in the given input patch. Hilbert *et al*. [13] worked with 3D images and used an additional lower resolution image to improve overall segmentation performance.

Also for μSplit we observe that additional image context is important. In contrast to the previously mentioned architectures, we introduce Lateral Contextualization (LC), a novel meta-architecture that feeds additional image context at multiple processing steps. We introduce three variants, *Lean-LC*, *Regular-LC*, and *Deep-LC*, differing from each other in terms of GPU memory requirements and achievable prediction quality. As we elaborate below, *Deep-LC* additionally offers the possibility to instantiate a more powerful HIERARCHICAL VAE with more hierarchy layers than otherwise possible, and show that this leads to improved performance on the image splitting task at hand.

Since μSplit needs to be applicable to large microscopy images, tiled predictions are required. In tiled predictions, input image is divided into overlapping patches on which predictions are performed individually. Those predictions are then appropriately center-cropped into non-overlapping tiles which can then be appended to form the final prediction. Overlapping patches have to be used to ensure that sufficient image context is available to address border artifacts to occur in the non-overlapping central region.

In Section 3, we argue that for deep networks operating on relatively small patches, overlapping regions should not be created by making tiles larger (Outer Padding) which is arguably the most common way, but that it is better to instead center-crop regions smaller than the original patch size (Inner Padding).

Since HIERARCHICAL VAEs (HVAEs) [26] have recently gained popularity, *e.g*. for microscopy image denoising and restoration [20, 19], we made these powerful architectures also available to the image decomposition task by modifying the default VAE ELBO loss, incorporating the fact that the fed input is different from the decoded output.

2. Problem Statement

A dataset $D_{mix} = (x^1, x^2, .., x^N)$ of N images is created by superimposing sampled pairs of image channels (D_1, D_2), such that

$$x^i = (d_1^i + d_2^i)/2, \forall i \in [1, N], \tag{1}$$

with $D_1 = (d_1^1, d_1^2, ...d_1^N)$ and $D_2 = (d_2^1, d_2^2..., d_2^N)$.

Given a newly sampled $x = (d_1 + d_2)/2$, the task is to decompose x into estimates of d_1 and d_2.

3. Our Approach

A Sound ELBO for μSplit. We train our VAE to describe the joint distribution for both channel images d_1 and d_2. We modify the VAE's ELBO objective to incorporate the fact that input and output are not the same (as they are for autoencoders). When training the VAE, our objective is to find

$$\arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log P(d_1^i, d_2^i; \boldsymbol{\theta}),$$

based on our training examples (d_1^i, d_2^i). Here, $\boldsymbol{\theta}$ are the decoder parameters of our VAE, which define the distribution. Next, we expand $\log P(d_1, d_2; \boldsymbol{\theta})$ as

$$\begin{aligned} &\log \int P(d_1, d_2, z; \boldsymbol{\theta}) dz \\ &= \log \int q(z|x; \boldsymbol{\phi}) * \frac{P(d_1, d_2, z; \boldsymbol{\theta})}{q(z|x; \boldsymbol{\phi})} dz \\ &>= \int q(z|x; \boldsymbol{\phi}) * \log \frac{P(d_1, d_2, z; \boldsymbol{\theta})}{q(z|x; \boldsymbol{\phi})} dz, \end{aligned} \tag{2}$$

where $q(z|x; \boldsymbol{\phi})$ is our encoder network with parameters $\boldsymbol{\phi}$. It can be shown that the evidence lower bound in Eq. 2 is equal to

$$E_{q(z|x;\boldsymbol{\phi})}[\log P(d_1, d_2|z; \boldsymbol{\theta})] - KL(q(z|x; \boldsymbol{\phi}), P(z)).$$

By making the assumption of conditional independence of d_1 and d_2 given z, we can simplify the expression to

$$\begin{aligned} E_{q(z|x;\boldsymbol{\phi})}[\log P(d_1|z; \boldsymbol{\theta}) + \log P(d_2|z; \boldsymbol{\theta})] \\ -KL(q(z|x; \boldsymbol{\phi}), P(z)). \end{aligned} \tag{3}$$

Expression 3 is what we end up maximizing during training. Note that this analysis can be seamlessly extended to the case where one has a hierarchy of latent vectors [26] instead of just one.

For modelling $q(z|x; \boldsymbol{\phi})$, we use the identical setup of the bottom-up branch used in [20] with the input being x, the superimposed input. For modeling $P(d_1|z; \boldsymbol{\theta})$ and $P(d_2|z; \boldsymbol{\theta})$, we again use the top-down branch design used in [20] but make the top-down branch output two channels for mean and two more for the pixelwise $\log(var)$, one each for d_1 and d_2. So, the output of our model is a 4 channel tensor with identical spatial dimensions as the input. Note that to encorporate LC, we modify both $q(z|x; \boldsymbol{\phi})$ and $P(d_2|z; \boldsymbol{\theta})$ which we describe next.

Lateral Contextualisation (LC). We introduce LC, allowing μSplit to see large portions of the input image at increasingly downscaled pixel resolutions. LC only requires small full resolution patches, rendering the network considerably more memory efficient.

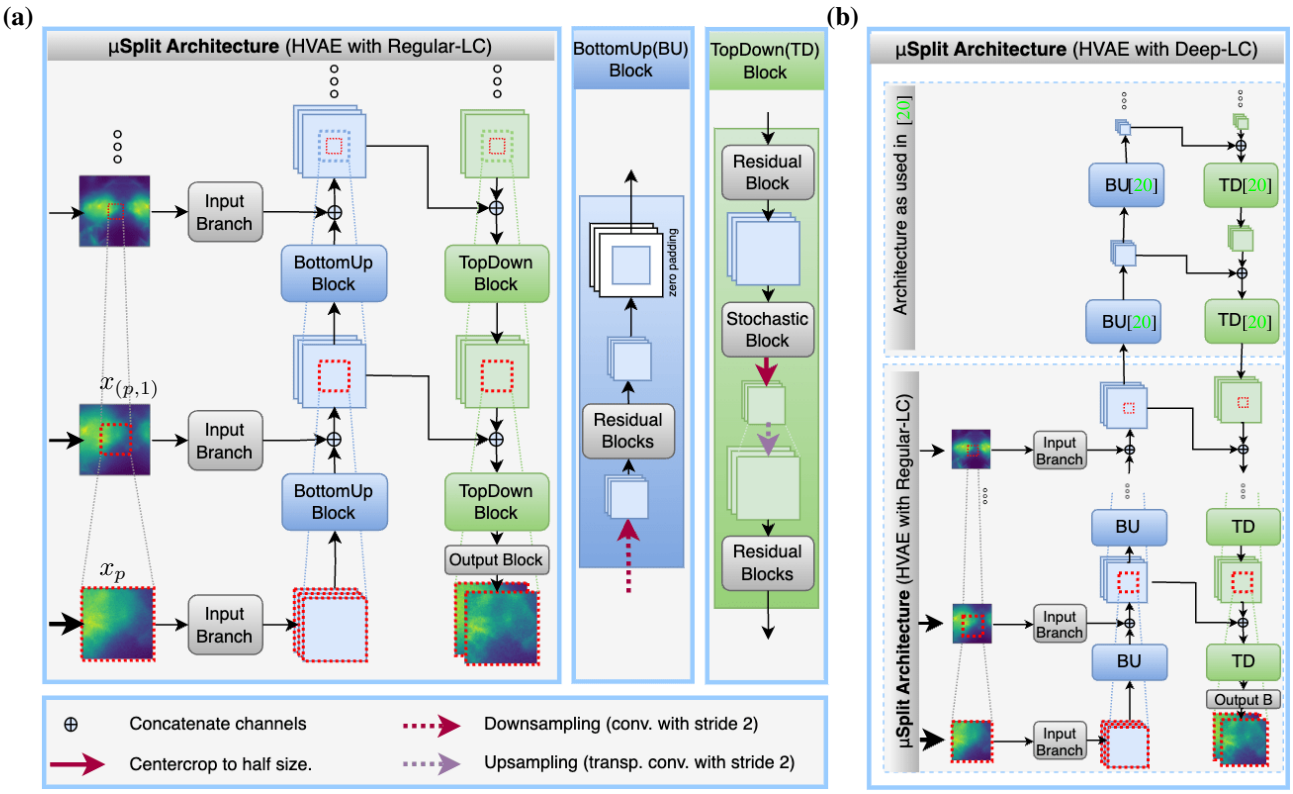

Figure 2. Network architecture of μSplit. In **(a)**, we show the network architecture employed by *Regular-LC*. The input (left side) consists of a core image patch x_p, together with downscaled version of the patch surroundings – the lateral context (LC). We show the area corresponding to the original patch as red dotted box throughout the figure. **(b)** The network architecture of *Deep-LC*. The architecture used in [20] is stacked on top of the *Regular-LC* architecture shown in (a). Note that this is only possible because the latent space in *Regular-LC* retained the spatial dimensions of all layers by means of using the proposed LC. *Note:* a sketch of the *Lean-LC* architecture, our third LC variant, can be found in the Supp. Figure *S.1*.

Many popular architectures, such as U-NETS [23] or HVAEs [20, 6, 27] are composed of a hierarchy of levels that operate on increasingly downsampled and therefore also increasingly smaller layers. The basic idea of LC is to pad each downsampled layer by additional image context, *i.e*. additional input from an available larger input image, such that each layer at each hierarchy level maintains the same spatial dimensions. (In Figure 2 (a), the red dashed squares in the stack of inputs (leftmost column) indicate the location of the original patch (x_p) within the downscaled and laterally contextualized inputs at higher hierarchy levels ($x_{(p,i)}$).)

Creating downsampled LC inputs. Let $x_p = x_{[c,h]}$ denote a patch of size $h \times h$ from $x \in D_{mix}$ centered around pixel location c. To decompose the patch x_p, we additionally use a sequence of successively downscaled and cropped versions of x, $X_p^{\text{lowres}} = (x_{(p,1)}, x_{(p,2)}, \dots, x_{(p,n_{\text{LC}})})$, where $x_{(p,k)}$ is $x_{[c,2^k \cdot h]}$, downsampled to the same pixel resolution of $h \times h$, and n_{LC} denotes the total number of used LC inputs.

Implementation of *Regular-LC*. Overall architecture is shown in Figure 2(a). Primary input patch x_p is fed to the first input branch (IB). The output of this IB is fed to the first bottom up (BU) block, which downsamples the input via strided convolutions, whose output is then passed to some residual blocks (see Supp. figure *S.1*), and finally zero padded to regain the same spatial dimension as the input it received. The output of the first BU block is concatenated with the output of the second IB, which has received the first lower resolution input containing additional lateral context, $x_{(p,1)}$. Zero-padding followed by concatenation ensures pixelwise alignment between IB's output and BU's output. We use 1×1-convolutions to merge these concatenated channels and feed the resulting layer into the next BU block. This procedure gets repeated for every hierarchy level in the given HVAE.

Once the topmost hierarchy level is reached, the last layer is fed into the topmost top down (TD) block. A TD block consist of some residual layers, followed by a stochastic block as they are used in HVAEs. The output of the stochastic block is center-cropped to half size and upsampled via transpose convolutions before again being fed through some residual layers ((see Supp. figure *S.1*)). Cropping and upsampling ensures that the output of the TD block matches the next lower hierarchy level. The output of the

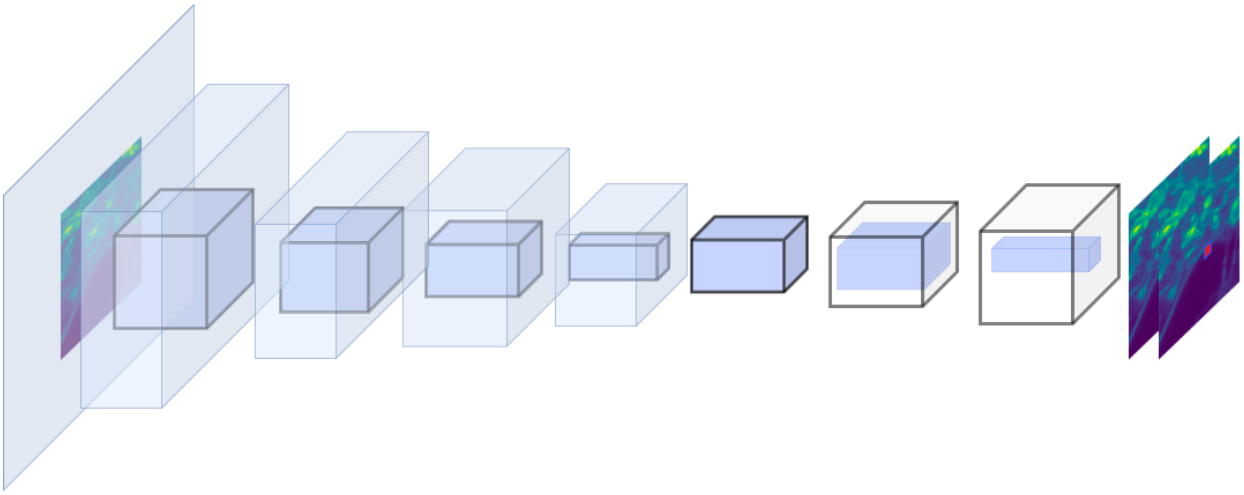

Figure 3. Cartoon of a generic hierarchical network with an encoder-decoder architecture illustrating the relationship between the input patch size, the effective receptive field, and the theoretical receptive field. The *input patch*, shown at the very left in the center of the light blue area, is processed and downsampled multiple times (encoder) before being upsampled multiple times (decoder) to allow the output, shown on the very right, to have the same pixel dimensions as the input patch. Cuboids shown by solid black lines represent the tensors the network computes during its execution. Solid blue cuboids show the *effective receptive field*, *i.e.* the areas within each tensor that can influence the center-most pixels in the two output layers (depicted by red rectangles). All but the last two tensors are fully 'visible' to those pixels, since the *theoretical receptive field*, *i.e.* the maximum area that would influence those pixels if the respective tensor would be sufficiently large, grows beyond their bounds (shown as light-blue solid cuboids). Note that working with larger input patches will fill a larger portion of the theoretical receptive field. If theoretical and effective receptive fields diverge, as shown in this cartoon, padded predictions on input patches larger then the training patch size will cause the network to operate out-of-distribution (OOD) and therefore lead to degraded prediction quality (see main text and Supp. Section S.2.1).

TD block is, similar to before, first concatenated with the output of the bottom up computations and then fed through 1×1-convolutions. Once we reach the bottom hierarchy level, the output of the last TD block is fed through an output block (OB) composed of some additional convolutional layers, giving us the final predictions of d_1 and d_2.

We've integrated LC into HVAE, HAE and the classic U-NET architecture. Note that the difference between HVAEs and HIERARCHICAL AUTOENCODERS (HAEs) is that the stochastic block is replaced by the identity. We use the term *Vanilla* to denote the underlying architecture on which we then enable LC.

***Deep-LC*: deeper performs better.** We observe empirically that having deeper hierarchies is beneficial (see Figure 6(a)). Since in U-NETS, HAEs, and HVAEs, each consecutive hierarchical level halves the input tensor in all spatial dimensions, a natural limit to the maximum hierarchy level is given by the fed patch size[1]. By making use of additional lower resolution image context at each hierarchy level, we've designed μSplit such that spatial dimensions of latent tensors stay constant across all hierarchy levels. This enables *Deep-LC* (see Figure 2(b)), our most potent architecture variant, to have additional hierarchy levels over what a vanilla HVAE can have, typically showing best results in our experiments (see Figure 6(b) and Figure 7).

More concretely, in our *Deep-LC* network, we stack a default HVAE (like the one used in [20]) on top of our *Regular-LC* variant (Figure 2(a)). This means that starting from the highest hierarchy level using LC, any further hierarchy level is built like a regular HVAE hierarchy stack.

***Lean-LC*: minimal memory footprint.** *Lean-LC*, our most memory efficient LC variation, does not use the lateral context introduced in the bottom-up branch within the top-down branch (see Supp. Figure S.1 for its architecture). More specifically, the bottom-up branch is identical to *Regular-LC*, but the top-down branch reduces to the default HVAE implementation, very similar to how it was also used in [20]. This is enabled by centercropping the output of each BU block going into the TD block.

Tiled Predictions. For virtually all tasks using fully convolutional architectures, trained networks are often used to predict results on inputs much larger then the patches they were trained on. Whenever an input image is so large that the network in question cannot scale without running out-of-memory, predictions are typically performed on overlapping patches and later suitably cropped and appended. When applied to relatively shallow [24] and non-variational networks, results can be pixel-perfect, *i.e.* not containing any tiling artifacts. But we observe that there are two cases wherein tiling artefacts are not easily avoidable.

[1] Using a patch size of 64, for example, can at most give rise to 5 hierarchy levels ($2^{5+1} = 64$).

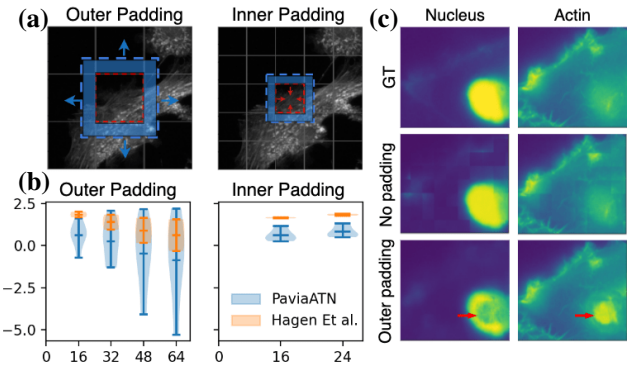

Figure 4. **Strategies for tiled predictions.** **(a)** The difference between Inner and Outer Padding. The blue dashed rectangle represents one patch used for tiled predictions. For each cell in the faint gray grid superimposed on the input image one such patch exists. The red dashed rectangle represents the center-crop region used to tile the final prediction of the entire input image. The blue shaded area is therefore the part of the patch that overlaps with neighboring patches, *i.e.* it is the padding area for the red rectangle. *Outer Padding* uses a tile size equivalent to the training patch size and introduces overlap by enlarging the patch being fed to the network. *Inner Padding*, in contrast, maintains the original patch size, and uses only an inner crop to tile the given input image. **(b)** Percentange variation (of PSNR measurements) when using different amounts of Outer or Inner padding (for HAE and HVAE vanilla setups using a patch size of 64). For varying amounts of padding (x-axis), we plot how 6 data points for the *PaviaATN* data (3 tasks$*2 = 6$) and 2 data points for *Hagen* et al. data (1 task) are distributed. Note how distributions for Inner Padding are consistently better. **(c)** Using Outer Padding, predictions are performed on patches larger than the ones used during training, leading to out-of-distribution (OOD) inputs and therefore to inferior predictions (red arrows). First and second row are the ground truth and prediction made without any padding respectively. See Supp. Figure *S.3* for more examples.

The first is caused by networks that have huge receptive fields (see Figure 3). When trained with a patch size much smaller than the theoretical receptive field size, large parts of the theoretical receptive field will be empty (*i.e.* zero). See also Supp. Section S.2.1 for a more detailed description.

When such trained networks are later used for tiled predictions, a problem arises whenever the input patches, on which predictions are made, are larger than the patch size used during training (which typically is the case because patch sizes is chosen such that GPU memory is best utilized, and input patches need to overlap sufficiently to avoid border artifacts). These patches will fill a larger portion of the theoretical receptive field than training patches did, resulting in out-of-distribution (OOD) predictions and worsened performance (see Figure 4 (b) for quantitative assessment).

The second case for tiling artifacts arises when variational networks like HVAEs are used. These architectures sample from the variational latent space of encoded tiles, with samples for neighboring tiles not necessarily decoding into consistent image contents along the borders of predicted tiles.

The solution we propose is twofold: (i) Instead of tiled prediction on large patches (Outer Padding), which is arguably the most often used tiling scheme, we propose to use Inner Padding instead, an approach that uses patches of the same size as the ones used during training, thereby solving the OOD issue introduced above. More specifically, in both tiling schemes, the input image is divided into overlapping patches. The predictions on these patches are then centercropped and these crops are put right next to each other in order to create a prediction for the entire input image. To enlarge the overlap between neighboring patches, Outer Padding enlarges the patch size. Inner Padding does not alter the size of patches, but instead only uses a smaller central area of their respective predictions. See Figure 4 (a) for a visual depiction of Inner and Outer Padding. In our experiments (see Section 5), we have used Inner Padding of 24 pixels, determined via grid-search. (ii) Overlap amount with Inner Padding are constrained to be small. Small overlap would usually cause artifacts due to insufficient image context at tile boundaries. However, due to our LC approach, μSplit is fed a very large and consistent image context at both sides of all patch boundaries, allowing us to operate with minimal artifacts even with small overlaps[2]. In supplement, we empirically show the lower need of overlap for our LC variants.

Training Details. For every dataset, we use 80%, 10% and 10% of the data as train-validation-test split. All models are trained using 16-bit precision on a Tesla V100 GPU. Unless otherwise mentioned, all models are trained with batch size of 32 and input patch size of 64. For all HVAEs, we lower-bound σs of $P(d_1, d_2, \theta)$ to $\exp(-5)$. This avoids numerical problems arising from these σs going to zero, as reported in [22]. Next, we re-parameterize the normal distributions for the BU branch using σ_{ExpLin} reformulation introduced in [7]. We additionally upper-bound the input to σ_{ExpLin} to 20. For training μSplit with *Deep-LC*, we follow the suggestions in [6, 21], and divide the output of each BU block by $\sqrt{2i}$, with i being the index of the hierarchy level the BU block is part of.

4. Datasets

SinosoidalCritters.

We created this synthetic dataset explicitly to demonstrate the importance of context for the splitting task and the usefulness of using LC within μSplit. Images in this dataset can only correctly be decomposed when sufficient lateral image context is available during prediction time.

[2]Note that artifacts arising from independently sampling the latent space in HVAEs remains an unsolved problem.

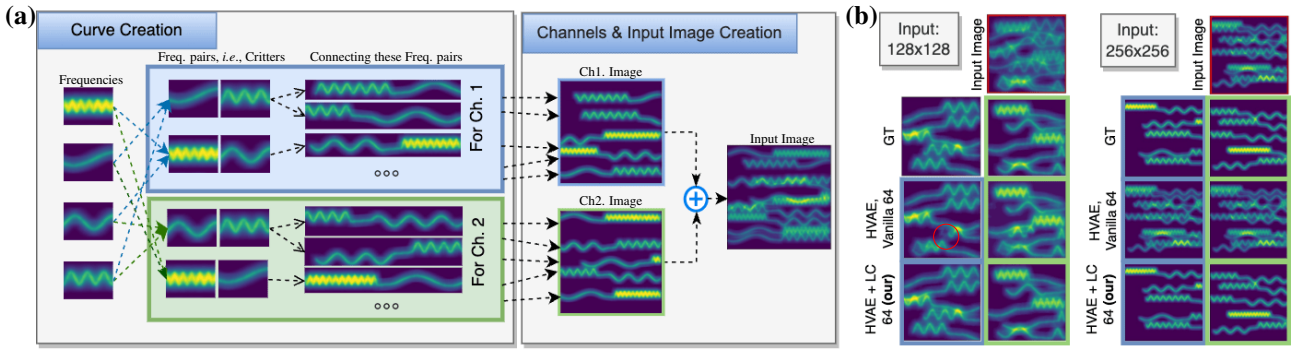

Figure 5. The synthetic *SinosoidalCritters* dataset is designed in such a way that large lateral image context is needed in order to perform correct channel splitting. **(a)** A schema illustrating how we created the *SinosoidalCritters* dataset. A detailed description is provided in Section 4. **(b)** We show two sample *SinosoidalCritters* input images (row 1) of size 128×128 and 256×256 pixels and the two channels that created them (row 2), respectively. Below, we show the decomposition results obtained with a trained vanilla HVAE with input patch size 64 (row 3), and results obtained with the same architecture but using *Regular-LC* (row 4). To recognise which critter is depicted and assign it to a channel, the network has to see both wave forms, hence requiring long range lateral image context.

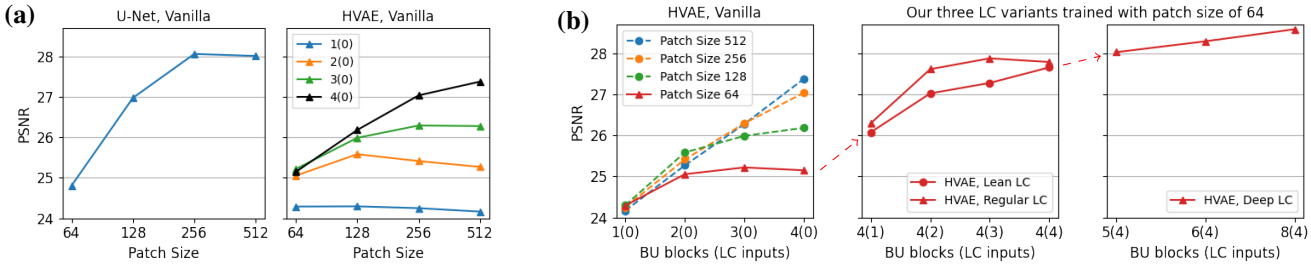

Figure 6. **Benefits of μSplit in one glance:** Quantitative results of baselines *vs*. μSplit variants. **(a)** We plot the performance of the vanilla U-NET and the vanilla HVAE baseline trained on increasingly larger patch sizes on our *PaviaATN* Act *vs*. Tub data. The U-NET performance plateaus roughly 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 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).

We first choose 4 different frequencies and combine them into 4 unique pairs. Two pairs are dedicated for image channel 1 (blue box), the other two for image channel 2 (green box). We call these pairs *critters*. The assignment of these critters to channels is done such that each frequency is assigned exactly once to each channel. We connect the two sinosoids of each critter with a low frequency curve of controllable length (later denoted by N_{join} in Table 2). Note that it is the specific combination of sinosoid frequencies present in the curve which decides whether it belongs to Channel 1 or 2 since the individual sinosoids themselves occur in both channels in equal amount. Next, we assemble channel images by placing a predefined number of randomly chosen curves at random positions in the respective image channel. The final input image is created as the sum of the two channels. See Figure 5(a) for dataset construction.

***PaviaATN* Microscopy Dataset.** We've created *PaviaATN* dataset. It has been imaged in the Synthetic Physiology Laboratory at University of Pavia, and is composed of 62 4-channel fluorescence microscopy images of size 2720×2720. Notably, this dataset has higher pixel resolution than most publicly available fluoroscence microscopy datasets [12, 17, 31]. The three channels we use label Actin, Tubulin and Nuclei, respectively, yielding three decomposition tasks we refer to as Actin *vs*. Tubulin, Actin *vs*. Nuclues, and Tubulin *vs*. Nucleus. Note that the dataset has two channels labelling Nuclei from which we picked one. See supplement for more details.

***Hagen* et al. Actin-Mitochondria Dataset.** From many sub-datasets provided by Hagen and colleagues [12], we picked the one with Mitochondria and Actin channels, the

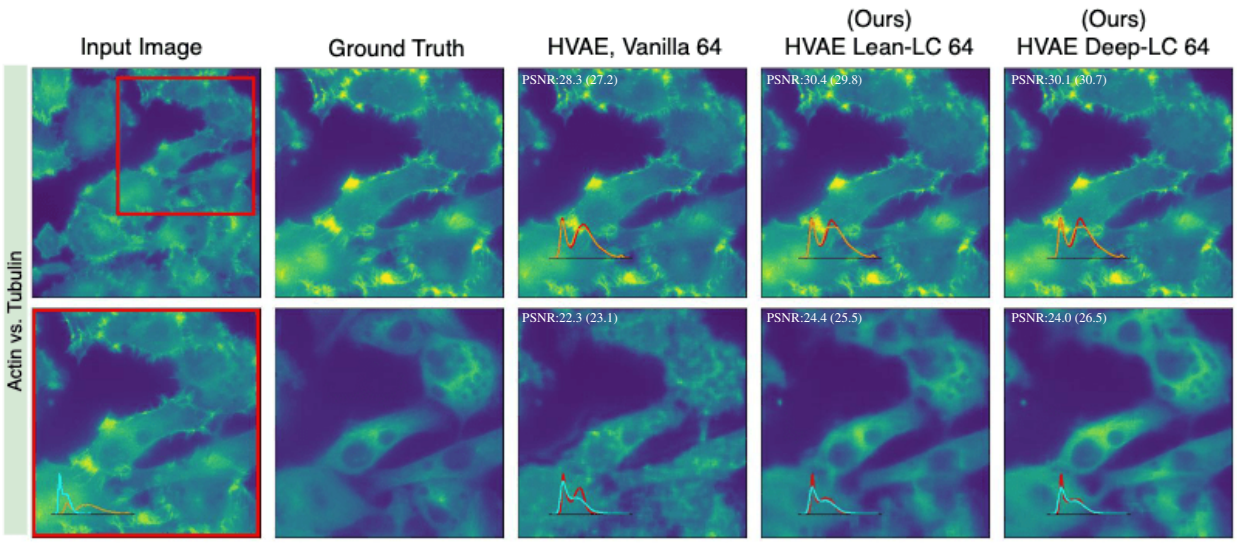

Figure 7. Qualitative results on the Act vs. Tub task from our *PaviaATN* dataset. We compare ground truth to results obtained with the vanilla HVAE baseline trained with a patch size of 64 to results obtained with two variations of μSplit (HVAEs using lean and deep LC, both also using a patch size of 64). 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). The given PSNR are for the individual prediction (full input image) and for the entire dataset (in brackets).

one with the highest pixel resolution (2048×2048).

5. Experiments and Results

Incrementally Introducing LC. In left panel of Figure 6(b), we show that for Vanilla HVAE, as hierarchy levels increase (BU blocks), so does the performance, provided we've large enough patch size. For patch size of 64, increasing hierarchy levels does not bring any benefit after a point.

In central panel of Figure 6(b), keeping the patch size and hierarchy levels fixed to 64 and 4 respectively, we introduce LC to an increasing number of hierarchy levels (denoted by the number in the brackets along the x-axis). This gives us a cumulative gain of around 2dB PSNR. Furthermore, with *Deep-LC* (right panel), we increase the hierarchy level even further which gives us further improvements. Two things are worth noting here for the patch size of 64: (i) There is not much benefit in increasing hierarchy levels for Vanilla HVAE. Using LC, on the other hand, leads to additional improvements, and (ii) Vanilla HVAE, cannot employ as many hierarchy levels as we can do using *Deep-LC*, and the results gain substantially from those extra levels. The Vanilla-XL model denotes Vanilla model trained with a patch size of 512. The *Deep-LC* results outperform the Vanilla-XL HVAE, see Figure 6(a), while also having a much smaller GPU memory footprint (see Table 1).

Experiments on Microscopy Data. We present results on 3 decomposition tasks on the *PaviaATN* dataset and 1 decomposition task on the *Hagen* et al. dataset. Table 1 summarizes our findings. As baselines, we've adapted the works of [16, 13] and find that μSplit outperforms them. It is worth noting that architecture used in [13], unlike ours, did not generalize to using a hierarchy of lower resolution inputs and worked with just one additional low resolution input. It also, unlike us, did not respect pixel alignments while concatenating the latent space tensors of the two resolution levels. We have also applied the unsupervised Double-DIP [9] baseline to random sampled 6 crops of size 256×256 for each test-set image of the *PaviaATN* and *Hagen* et al. datasets (see Table 1 and supplementary figure).

Over all four tasks, the best performing LC variant with HVAE architecture outperforms the best LC variant with HAE architecture by 0.5 PSNR on average. Using the HVAE architecture, *Deep-LC* outperforms *Lean-LC* on average by 0.8 PSNR. For the HAE architecture this difference is 0.1 PSNR. Qualitative results are shown in Figure 7 and in the supplement.

Outer *vs*. Inner Padding and Runtime Performance. In Figure 4(c), we show the percentage change in PSNR with different amounts of padding and see that the vanilla HAE and HVAE setup performances degrade (left plot) when Outer Padding is used with large padding amounts. But with Inner Padding (right plot), we see improvement saturation with increase in padding amount. In Figure 4(b), one can observe an artefact appearing solely due to Outer Padding (artifact does not exist in 'No padding'). These results support our claim about OOD issue as described in Section 3.

Note that Inner Padding requires a larger number of individual predictions, indicated by the smaller grid size seen in Figure 4(a) (denoted by red dashed rectangle). Specifically,

Model + Patch Size			GPU	*PaviaATN*						*Hagen* et al.	
				Act vs Nuc		Tub vs Nuc		Act vs Tub		Act vs Mit	
			(GiB)	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Double-DIP [9]			-	22.8	0.30	21.2	0.20	20.9	0.30	25.3	0.56
BraveNet [13]		64	2.8	31.7	0.73	30.3	0.61	25.9	0.62	33.0	0.92
Context-Aware U-Net [16]		64	4.7	31.5	0.74	29.0	0.61	25.1	0.61	31.1	0.91
U-Net		256	9.4	33.2	0.79	31.4	0.71	28.1	0.69	34.2	**0.95**
U-Net		512	28.7	33.3	0.79	31.1	0.72	27.9	0.69	34.1	0.94
U-Net	Regular-LC	64	12.5	33.5	0.79	32.0	0.71	27.6	0.68	32.7	0.93
HAE	Vanilla	64	2.3	31.7	0.74	29.5	0.64	25.4	0.63	31.9	0.92
	Lean-LC	64	3.9	33.6	0.78	31.9	0.70	27.7	0.67	32.9	0.94
	Regular-LC	64	6.0	33.5	0.79	31.6	0.71	27.9	0.68	33.4	0.94
	Deep-LC	64	6.9	33.7	0.80	31.8	0.72	28.3	0.69	32.8	0.94
	Vanilla-XL	512	31.2	33.2	0.79	30.2	0.68	27.6	0.67	34.2	**0.95**
HVAE	Vanilla	64	2.8	31.8	0.75	29.6	0.64	25.2	0.61	31.9	0.93
	Lean-LC	64	4.4	33.8	0.79	31.9	0.71	27.7	0.68	32.7	0.94
	Regular-LC	64	11.1	**33.9**	0.80	32.1	0.72	27.8	0.68	34.1	**0.95**
	Deep-LC	64	12.8	**33.9**	**0.81**	32.5	**0.73**	**28.6**	**0.70**	**34.3**	**0.95**
	Vanilla-XL	512	(*)	33.4	0.78	**32.9**	0.69	27.6	0.67	**34.3**	**0.95**

Table 1. Quantitative results on fluorescent image decomposition tasks derived from the *PaviaATN* and *Hagen* et al. datasets. All results are reported in terms of peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM). For each model we also report the used training patch size and GPU memory usage during training. The baselines we use are Double-DIP [9], BraveNet [13], Context-Aware U-Net [16], as well as vanilla HAEs and HVAEs using four hierarchy levels. Additionally, we show results for U-NETS [23], HAEs, and HVAEs trained on much larger patch sizes (256 and 512). The results of μSplit are also obtained with the same HAE and HVAE architectures trained on patches of size 64×64, but with all hierarchy levels also employing either *Lean-LC*, *Regular-LC*, or *Deep-LC* (see main text for details). Bold numbers denote the best result for any given task (column). In all but one case (*PaviaATN*, Tubulin *vs.* Nuclei), our results outperform all baselines despite having a comparatively lean memory footprint. Note that the Vanilla-XL HVAE with patch size of 512 and batch size of 32 did not fit in 32 GiB of GPU memory and so we lowered the batch size such that the model did fit in memory.

Image Size	Model	$N_{join} = 0$		$N_{join} = 25$	
		PSNR	SSIM	PSNR	SSIM
128	Vanilla	28.3	0.90	25.5	0.85
	Lean-LC	**37.3**	0.97	35.1	0.96
	Regular-LC	37.0	**0.98**	**39.2**	**0.98**
256	Vanilla	19.4	0.75	15.8	0.43
	Lean-LC	34.1	0.97	32.2	0.97
	Regular-LC	**41.5**	**0.99**	**41.6**	**0.98**

Table 2. Quantitative results on the *SinosoidalCritters* dataset. We compare results obtained with vanilla HVAEs that do not use LC, and HVAEs employing either *Lean-LC* or *Regular-LC* (*i.e.* μSplit results, see main text for details). All experiments are performed using a patch size of 64. Bold numbers denote the best result for any given task (columns), showing that our results consistently outperform the vanilla baselines.

using an Inner Padding of 24 pixels with a patch size of 64 will use 16×16 center-crop per patch. Hence, we will need to predict 16 ($(64/16 = 4)^2$) times more patches to cover the entire input image.

Interestingly, we found padding giving minor benefits for *Deep-LC* quantitatively and so *Deep-LC* results in Table 1 were computed without padding thereby leading to a better runtime for *Deep-LC*. However, we still find few tiling artefacts with *Deep-LC* and in those cases Inner Padding helps. Other two LC variants benefit both quantitatively and qualitatively from Inner Padding.

BU Blocks	vanilla 64	rLC 64	rLC 128
1	24.3	24.7	**24.8**
2	25.1	**25.9**	**25.9**
3	25.2	**27.0**	**27.0**
4	25.4	27.8	**27.9**

Table 3. Performance of HVAE + *Regular-LC* trained with patch size of 64 (col 3) and 128 (col 4) on Act vs Tub data. The larger patch size shows diminishing returns, indicating that LC is providing enough image context, showcasing the value of our approach.

Effects of Larger Training Patch Sizes. In Figure 6(a) we show that increasing the training patch size improves the performance of a U-NET and vanilla HVAEs across different hierarchy levels. While the U-NET baseline performance saturates, HVAEs' improvement with increasing hierarchy levels does not, but quickly reach a hard limit in terms of GPU memory requirement (see Table 1).

Performance of LC with larger patch sizes. Using μSplit, microscopy labs having limited GPU compute will still get similar performance to labs with ample resources, labs capable of using networks employing larger patch sizes. So far, all our LC variants have been trained with a patch size of 64. A natural question to ask is whether there is still some benefit in using larger patch sizes when also using LC. While the answer to this question depends upon multiple factors like how much long range interactions are present in the data, the receptive field size of the network etc, we did an ablation to empirically investigate this in Table 3. One can observe that for HVAE + *Lean-LC*, across different hierarchy levels (BU Block count), using a patch size of 128 only provides a minor performance improvement over a patch size of 64. This implies that for a pixel's prediction, only a small amount of neighbourhood context needs to be given at native pixel resolution and most of the context can be given via lower-resolution lateral image context.

Experiments on Synthetic Data. In Table 2 we show the results obtained on the *SinosoidalCritters* dataset. We used two input image sizes, 128×128 and 256×256, and two values for N_{join}, namely 0 and 25 pixels. On average, μSplit outperforms the vanilla HVAE by 18 PSNR. Also note that the larger input size, constituting a harder problem to solve, is resulting in a drop of performance for the vanilla HVAE. Using μSplit, instead, the performance increases. To recognise which critter is depicted and assign it to a channel, the network has to see both wave forms. The vanilla HVAE is able to do splitting on 128×128, but it has artefacts (red circle in Figure 5(b)). For the 256×256 pixel images, it completely fails because it is unable distinguish between the critters since it cannot simultaneously process a sufficiently large part of the image. In contrast, by using LC we are able to successfully split both images.

U-NET Hyperparameter Tuning. We tuned depth and patch size of a classic U-NET to achieve optimal performance for the tasks at hand (see supplement for details).

6. Discussion

In this work, on our dataset we show that μSplit performs better when deeper architectures, *i.e*. HAEs and HVAEs, are employed and enabled to process additional image context via the memory efficient lateral contextualization (LC) schemes we propose.

The deeper such networks become, the larger will the receptive field (RF) sizes grow, in our case routinely exceeding sizes of 512×512 pixels. An immediate consequence of this is that we cannot easily employ common tiling schemes (*i.e*. Outer Padding) without running into out-of-distribution (OOD) issues (see Section 3). Hence, we propose to use Inner Padding to circumvent this problem. Additionally, we observe that *Deep-LC* does even perform quite well without padded tiled predictions (no additional overlap between patches). The reason for this is that the patch context typically given by overlapping regions is now substituted by context being fed via *Deep-LC*. Still, best performance is typically obtained using *Deep-LC* and Inner Padding during tiled predictions.

It is important to point out that for any variational models, such as HVAEs, tiled predictions suffer from the additional problem that neighboring tiles will likely not be consistent due to the sampling step performed independently per tile. While Inner Padding still is the better strategy to employ (for the same argument as for any other model with huge receptive fields), sampling inconsistencies cannot be fully avoided. The strength of these artifacts will depend on the data uncertainty (*i.e*. the ambiguity in the fed inputs w.r.t. the trained model).

In summary, we have proposed a powerful new method to efficiently use image context. We have then applied this method to an impactful new image decomposition task on fluorescence microscopy data. We believe that the presented ideas will prove to also be useful in the context of other computer vision problems. We will explore the applicability of LC to other problem domains in future work. Additionally, we will make μSplit more amenable to noisy fluorescence data and to disentanglement tasks where more than two image channels are superimposed.

Acknowledgements

This work was supported by the European Commission through the Horizon Europe program (IMAGINE project, grant agreement 101094250-IMAGINE and AI4LIFE project, grant agreement 101057970-AI4LIFE) as well as the compute infrastructure of the BMBF-funded de.NBI Cloud within the German Network for Bioinformatics Infrastructure (de.NBI) (031A532B, 031A533A, 031A533B, 031A534A, 031A535A, 031A537A, 031A537B, 031A537C, 031A537D, 031A538A). Additionally, the authors also want to thank Damian Dalle Nogare of the Image Analysis Facility at Human Technopole for useful guidance and discussions and the IT and HPC teams at HT for the compute infrastructure they make available to us.

References

[1] Yuval Bahat and Michal Irani. Blind dehazing using internal patch recurrence. In *2016 IEEE International Conference on Computational Photography (ICCP)*, pages 1–9, May 2016. 1

[2] Joshua Batson and Loïc Royer. Noise2Self: Blind denoising by Self-Supervision. pages 1–16, Jan. 2019. 1

[3] Dana Berman, Tali Treibitz, and Shai Avidan. Non-local image dehazing. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1674–1682. IEEE, June 2016. 1

[4] Tim-Oliver Buchholz, Alexander Krull, Réza Shahidi, Gaia Pigino, Gáspár Jékely, and Florian Jug. Content-aware image restoration for electron microscopy. *Methods Cell Biol.*, 152:277–289, July 2019. 1

[5] Tim-Oliver Buchholz, Mangal Prakash, Deborah Schmidt, Alexander Krull, and Florian Jug. DenoiSeg: Joint denoising and segmentation. In *Computer Vision – ECCV 2020 Workshops*, pages 324–337. Springer International Publishing, 2020. 1

[6] Rewon Child. Very deep VAEs generalize autoregressive models and can outperform them on images. Nov. 2020. 3, 5

[7] David Dehaene and Rémy Brossard. Re-parameterizing VAEs for stability. June 2021. 5

[8] Tali Dekel, Michael Rubinstein, Ce Liu, and William T Freeman. On the effectiveness of visible watermarks, 2017. 1

[9] Yossi Gandelsman, Assaf Shocher, and Michal Irani. "Double-DIP" : Unsupervised image decomposition via coupled deep-image-priors, 2019. Accessed: 2022-2-14. 1, 7, 8

[10] Ionita C Ghiran. Introduction to fluorescence microscopy. *Methods Mol. Biol.*, 689:93–136, 2011. 1

[11] Anna S Goncharova, Alf Honigmann, Florian Jug, and Alexander Krull. Improving blind spot denoising for microscopy. In *Computer Vision – ECCV 2020 Workshops*, pages 380–393. Springer International Publishing, 2020. 1

[12] Guy M Hagen, Justin Bendesky, Rosa Machado, Tram-Anh Nguyen, Tanmay Kumar, and Jonathan Ventura. Fluorescence microscopy datasets for training deep neural networks. *Gigascience*, 10(5), May 2021. 6

[13] Adam Hilbert, Vince I Madai, Ela M Akay, Orhun U Aydin, Jonas Behland, Jan Sobesky, Ivana Galinovic, Ahmed A Khalil, Abdel A Taha, Jens Wuerfel, Petr Dusek, Thoralf Niendorf, Jochen B Fiebach, Dietmar Frey, and Michelle Livne. BRAVE-NET: Fully automated arterial brain vessel segmentation in patients with cerebrovascular disease. *Front Artif Intell*, 3:552258, Sept. 2020. 1, 2, 7, 8

[14] Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. Noise2Void - learning denoising from single noisy images. *arXiv*, cs.CV:2129–2137, Nov. 2018. 1

[15] Alexander Krull, Tomas Vicar, Mangal Prakash, Manan Lalit, and Florian Jug. Probabilistic Noise2Void: Unsupervised Content-Aware denoising. *Frontiers in Computer Science*, 2:60, Feb. 2020. 1

[16] Jiaxu Leng, Ying Liu, Tianlin Zhang, Pei Quan, and Zhenyu Cui. Context-Aware U-Net for biomedical image segmentation. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, Dec. 2018. 1, 7, 8

[17] Chawin Ounkomol, Sharmishtaa Seshamani, Mary M. Maleckar, Forrest Collman, and Gregory R. Johnson. Label-free prediction of three-dimensional fluorescence images from transmitted-light microscopy. *Nature Methods*, 15(11):917–920, Nov. 2018. 6

[18] Wei Ouyang, Fynn Beuttenmueller, Estibaliz Gómez-de Mariscal, Constantin Pape, Tom Burke, Carlos Garcia-López-de Haro, Craig Russell, Lucía Moya-Sans, Cristina de-la Torre-Gutiérrez, Deborah Schmidt, Dominik Kutra, Maksim Novikov, Martin Weigert, Uwe Schmidt, Peter Bankhead, Guillaume Jacquemet, Daniel Sage, Ricardo Henriques, Arrate Muñoz-Barrutia, Emma Lundberg, Florian Jug, and Anna Kreshuk. BioImage model zoo: A Community-Driven resource for accessible deep learning in BioImage analysis. June 2022. 1

[19] Mangal Prakash, Mauricio Delbracio, Peyman Milanfar, and Florian Jug. Interpretable unsupervised diversity denoising and artefact removal. Apr. 2021. 1, 2

[20] Mangal Prakash, Alexander Krull, and Florian Jug. DivNoising: Diversity denoising with fully convolutional variational autoencoders. *ICLR 2020*, June 2020. 1, 2, 3, 4

[21] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and Others. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 5

[22] Danilo Jimenez Rezende and Fabio Viola. Taming VAEs. Oct. 2018. 5

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, volume 9351, pages 234–241. Springer International Publishing, Cham, Oct. 2015. 3, 8

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, May 2015. arXiv:1505.04597 [cs]. 4

[25] Uwe Schmidt, Martin Weigert, Coleman Broaddus, and Gene Myers. Cell detection with Star-Convex polygons. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, pages 265–273. Springer International Publishing, 2018. 1

[26] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. *Adv. Neural Inf. Process. Syst.*, 29:3738–3746, Jan. 2016. 2

[27] Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. July 2020. 3

[28] Martin Weigert, Loic Royer, Florian Jug, and Gene Myers. Isotropic reconstruction of 3D fluorescence microscopy images using convolutional neural networks. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2017*, pages 126–134. Springer International Publishing, 2017. 1

[29] 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. 1

[30] Martin Weigert, Uwe Schmidt, Robert Haase, Ko Sugawara, and Gene Myers. Star-convex polyhedra for 3D object detection and segmentation in microscopy. *arXiv*, cs.CV, Aug. 2019. 1

[31] Yide Zhang, Yinhao Zhu, Evan Nichols, Qingfei Wang, Siyuan Zhang, Cody Smith, and Scott Howard. A Poisson-Gaussian Denoising Dataset with Real Fluorescence Microscopy Images, Apr. 2019. arXiv:1812.10366 [cs, eess, stat]. 6