Supplementary Information

Content-Aware Image Restoration: Pushing the Limits of Fluorescence Microscopy

Martin Weigert, Uwe Schmidt, Tobias Boothe, Andreas Müller, Alexandr Dibrov, Akanksha Jain, Benjamin Wilhelm, Deborah Schmidt, Coleman Broaddus, Siân Culley, Mauricio Rocha-Martins, Fabián Segovia-Miranda, Caren Norden, Ricardo Henriques, Marino Zerial, Michele Solimena, Jochen Rink, Pavel Tomancak, Loic Royer, Florian Jug & Eugene W. Myers

Notes (December 21, 2017)

1. This document is not finalized and will likely be subject to change. Suggestions for improvements are cordially received via email.
2. Each network should be trained with data for a specific combination of image content (*e.g.*, nuclei, microtubules) and image corruption (camera noise, pixel size, microscope PSF, *etc.*). Hence, applying trained networks to images that are very dissimilar to the training data could lead to unexpected results. The pretrained networks provided via FIJI and KNIME are meant to showcase our method on the accompanying example data.
3. Software for generating training data and training of CARE networks will be released soon. This will enable all users to employ CARE on their own image data.

Supplementary Videos

Supplementary Video 1. Challenges in time-lapse imaging of flatworm *Schmidtea mediterranea*. Image stacks of RedDot1-labeled, anesthetized specimen were acquired every 2 min with a Spinning disk confocal microscope ($NA = 1.05$), at high and low SNR (illumination) conditions (10% laser, 30ms exposure per plane vs 0.5% laser, 10ms per plane). Whereas in the high-SNR case the specimen shows illumination induced twitching, the image quality in the low-SNR case is insufficient for further analysis. Network restoration enables us to recover high-SNR images from images acquired at low-SNR conditions without twitching of the specimen, thus providing a practical framework for live-cell imaging of *Schmidtea mediterranea*.

Supplementary Video 2. Restoration results of low-SNR acquisitions of *Schmidtea mediterranea* and comparison to ground-truth. Shown are a 3D rendering of the results on a multi-tiled acquisition ($8192 \times 3072 \times 100$ pixels) and the comparison with ground truth.

Supplementary Video 3. Restoration of low SNR volumetric time-lapses of developing *Tribolium castaneum* embryos (EFA::nGFP labeled nuclei). Acquisition was done on a Zeiss LSM 710 NLO Multiphoton Laser Scanning Microscope with a time-step of 8 min, and stack-size of a single time-point $760 \times 760 \times 100$ pixels. Shown are maximum-intensity projection and single slices of the raw stacks, the network prediction and the high-SNR ground-truth.

Supplementary Video 4. Joint surface projection and denoising of developing *Drosophila melanogaster* wing epithelia. 3D image stacks of the developing wing of a membrane labeled (Ecad::GFP) fly pupa were acquired with a spinning disk confocal ($63\times$, $NA = 1.3$) microscope. We show the projected epithelial surface obtained by a conventional method (PreMosa [1]), the restoration network, and the projected ground-truth. We applied a random-forest based cell segmentation pipeline and show segmentation/tracking results, demonstrating vastly improved accuracy for the restoration when compared to the conventionally processed raw stacks.

Supplementary Video 5. Isotropic restoration of anisotropic time-lapse acquisitions of hisGFP-tagged developing *Drosophila melanogaster* embryos. We used the original, pre-processed data set of [2], acquired with a light-sheet microscope ($NA = 1.1$) with 5-fold axially undersampled resolution (lateral/axial pixel size: $0.39\mu m/1.96\mu m$). The video shows different axial (xz) regions of a single time-point from both the original (input) stacks and the isotropic restoration.

Supplementary Video 6. Isotropic restoration of anisotropic dual-color acquisitions of developing *Danio rerio* retina. The data was acquired with a Spinning disk confocal microscope (Olympus $60\times, NA = 1.1$) and exhibits a 10-fold axial anisotropy (lateral/axial pixel size: $0.2\mu m/2.0\mu m$); labeled structures are nuclei (DRAQ5, magenta) and nuclear envelope

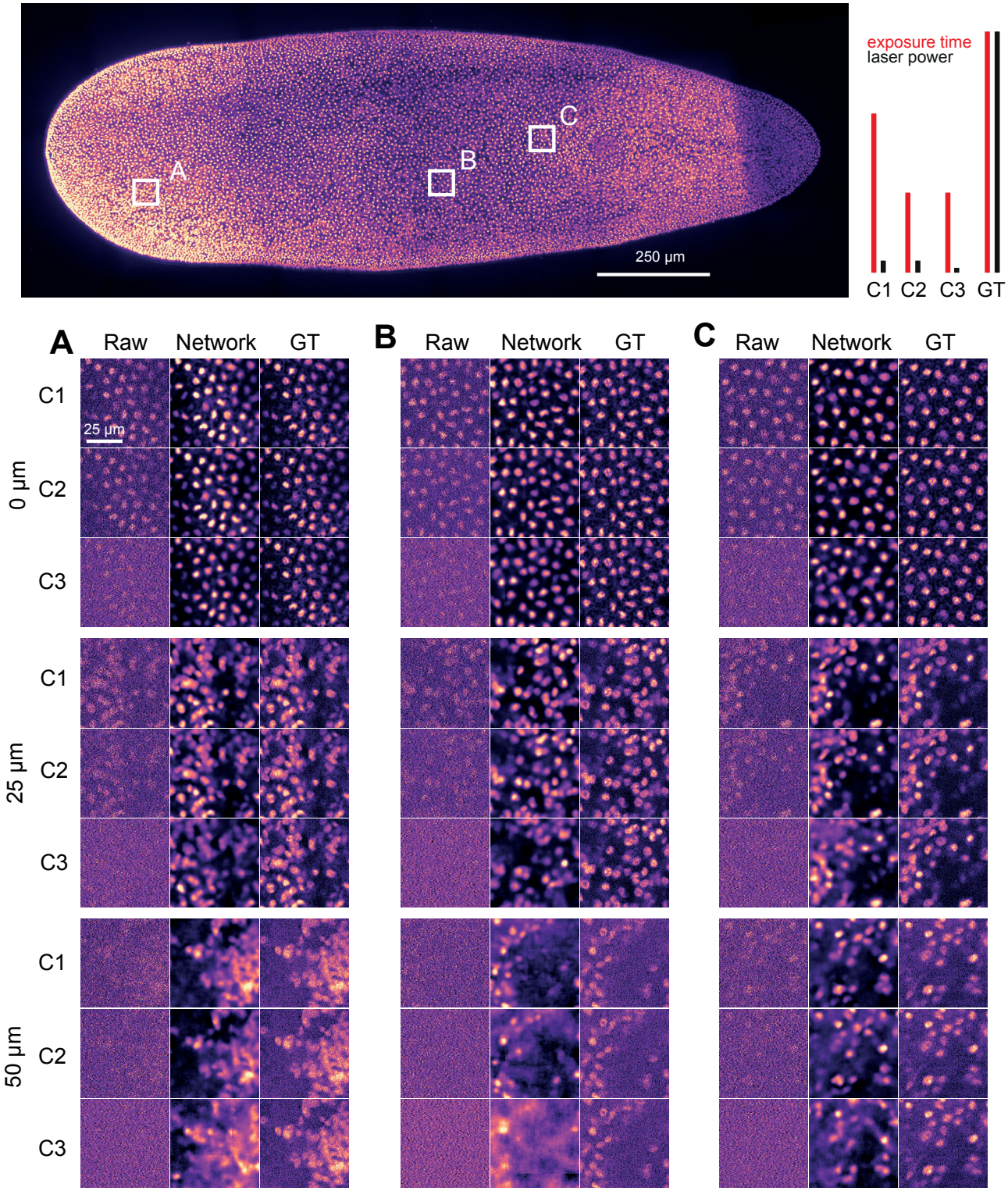
(GFP+LAB2B, green). The video shows a rendering of the dual-color input stack and its isotropic reconstruction.

Supplementary Video 7. Enhancement of diffraction-limited structures in widefield images of rat INS-1 (beta) cells. The video shows time-lapses of several INS-1 cells, acquired with the widefield mode of a DeltaVision OMX microscope (63×, $NA = 1.43$). Labeled are secretory granules (pEG-hIns-SNAP, magenta) and microtubules (SiR-tubulin, green). Next to the time-lapse of the widefield images we show the output of the reconstruction networks.

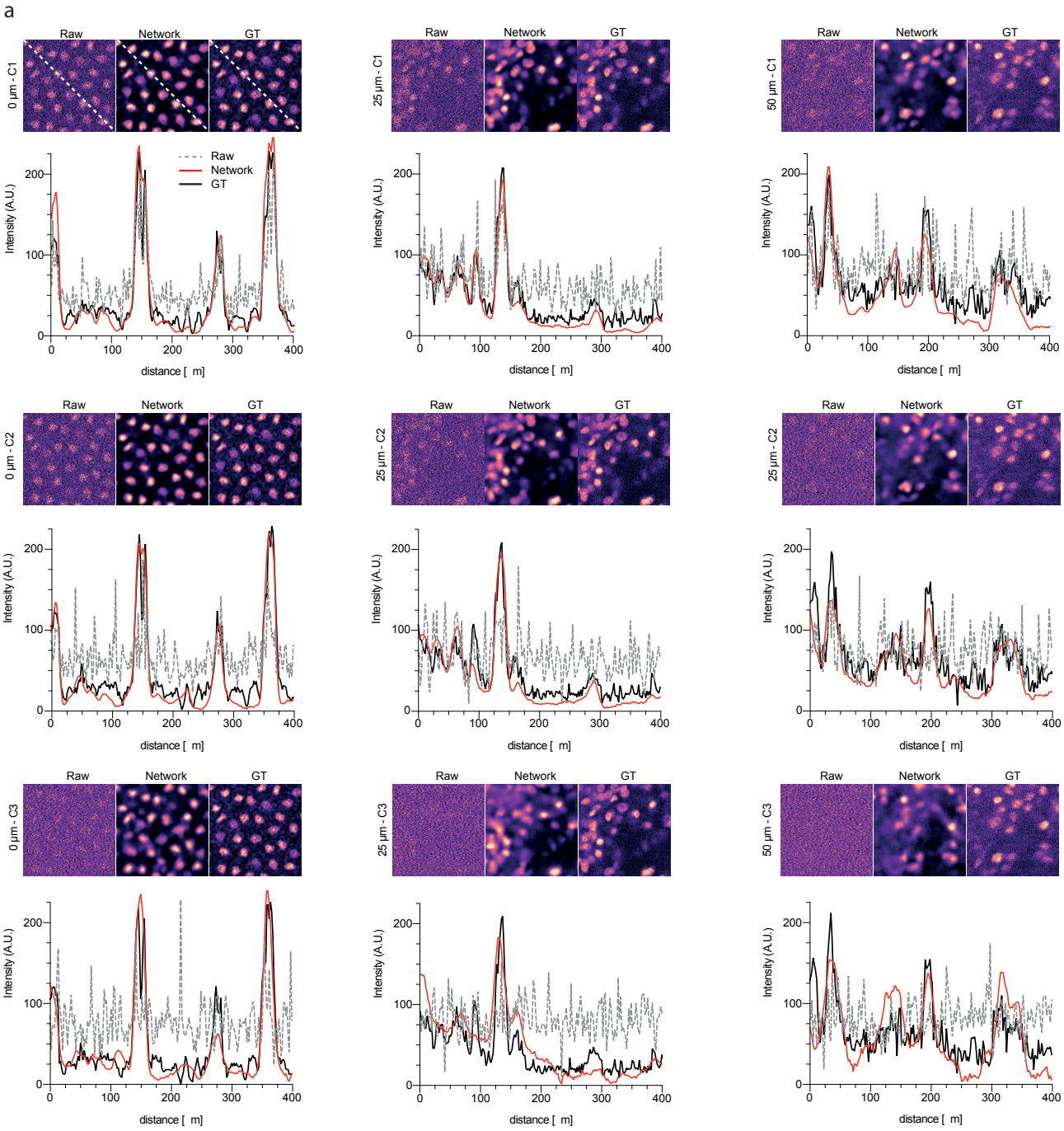
Supplementary Video 8. Enhancement of diffraction-limited widefield images of GFP-labeled microtubules in HeLa cells and comparison with SRRF (super-resolution radial fluctuations [3]). Images were acquired with a Zeiss Elyra PS.1 microscope in TIRF mode (100×, $NA = 1.46$). The video shows the widefield input sequence, the network restoration and the corresponding SRRF images. Note that the time-resolution of the SRRF image sequence is 20 times less then the network restoration, as 20 times more images have to be processed for the same restoration quality.

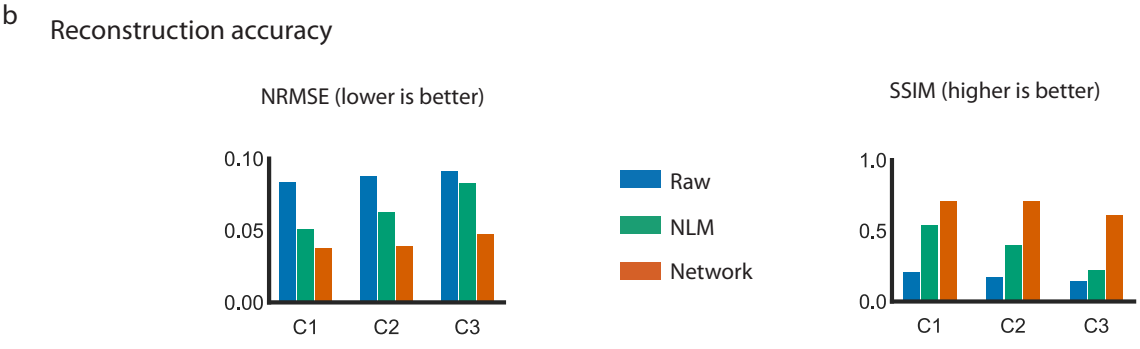
Supplementary Video 9. Visualization of network predictions by sampling from the predicted distribution. The video shows for two examples (Surface projection of fly wing tissue, and microtubule structure restoration in INS-1 cells) that drawing samples from the per-pixel predicted distribution is beneficial for identifying challenging image regions. For each example, we show successively the raw input, the per-pixel mean of the predicted distribution and random samples from the per-pixel distributions.

Supplementary Figures

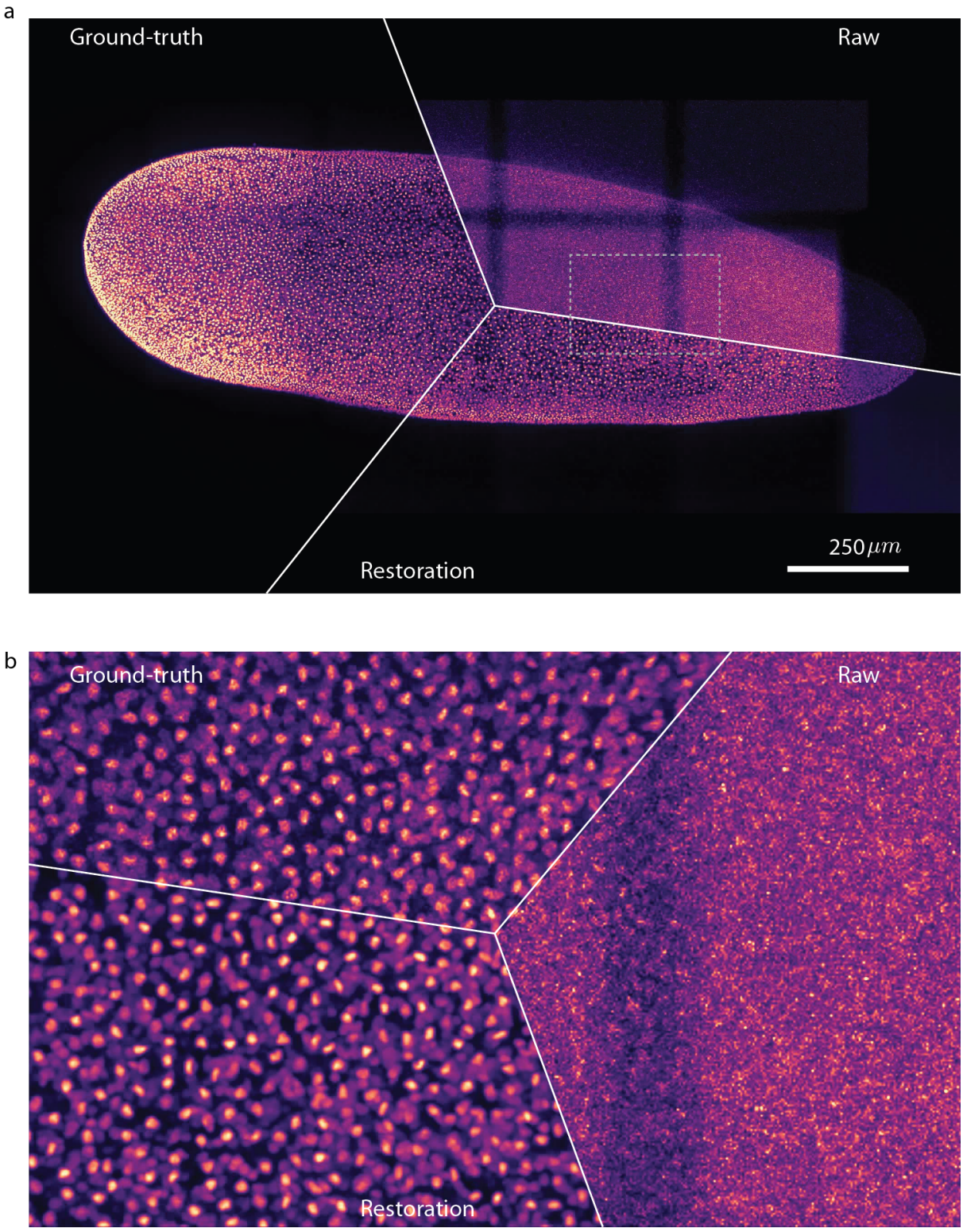


Supplementary Figure 1: Restoration of low-SNR images of *Schmidtea mediterranea* (Planaria). Spinning disk confocal acquisitions of a fixed worm (*RedDot1* stained) at 3 different low-SNR conditions (C1-C3) and a high-SNR condition (ground-truth, GT). Depicted are insets of the input, the network reconstruction and the high-SNR ground-truth at different positions and tissue depths ($0 - 50\mu m$).

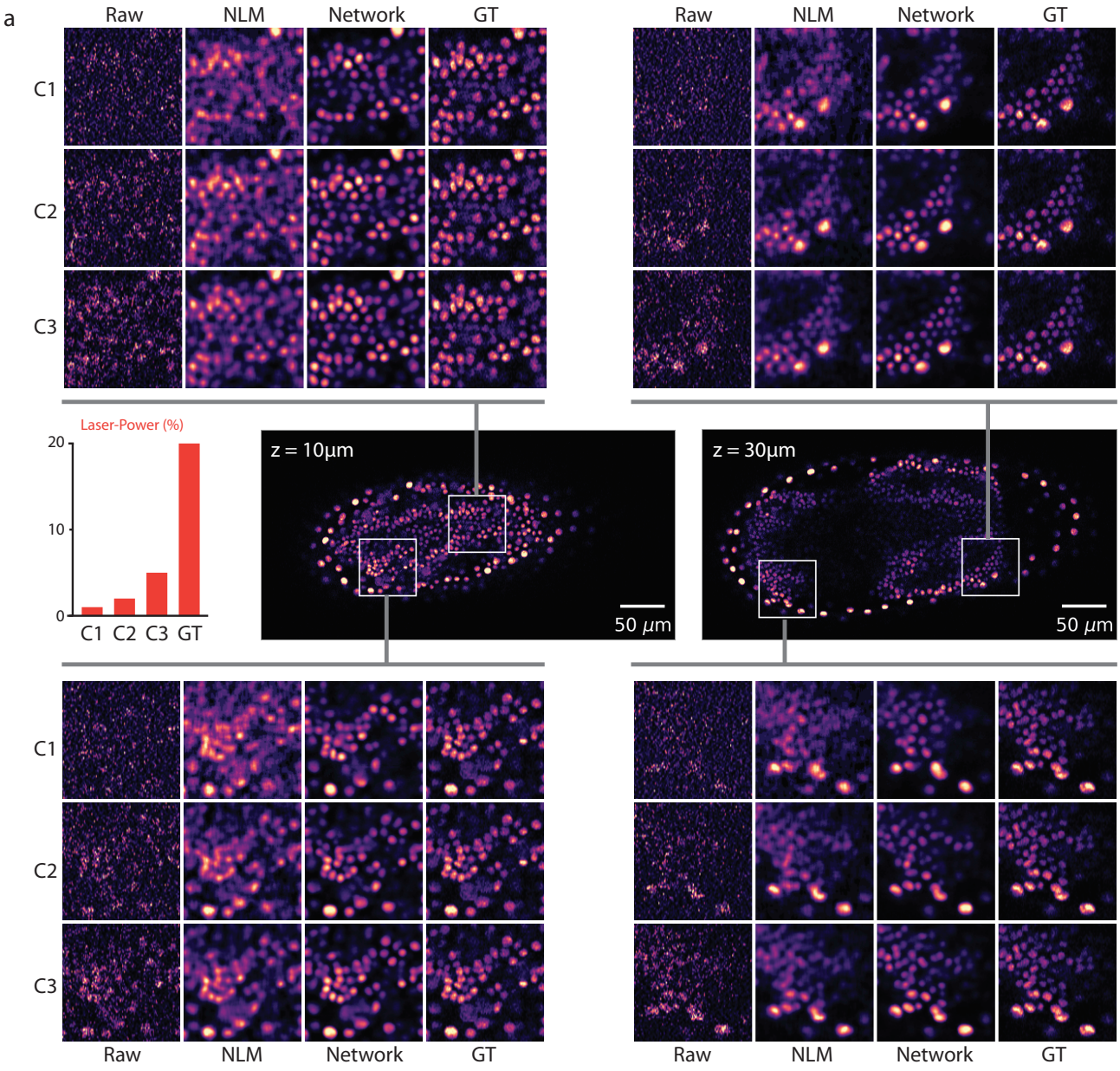


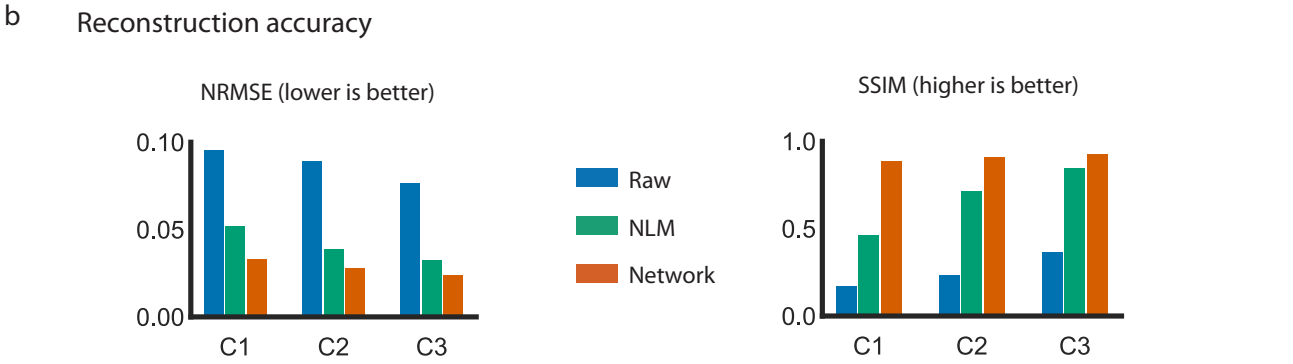


Supplementary Figure 2: **(a)** Reconstructions for different SNR conditions (C1-C3, top to bottom) at different tissue depths ($0 - 50\mu m$, left to right) and corresponding lineplots. As SNR decreases with increasing depth, both the input and the reconstruction quality decreases.**(b)** Reconstruction accuracy measured on held out test set (comprising 22 volumetric stacks for each condition). Shown are normalized root mean squared error (NRMSE, lower is better) and structural similarity (SSIM, higher is better). We compare the error between ground-truth *vs*. raw stacks, the network prediction and a strong denoising baseline (Non-local-means with parameters tuned to the test set).

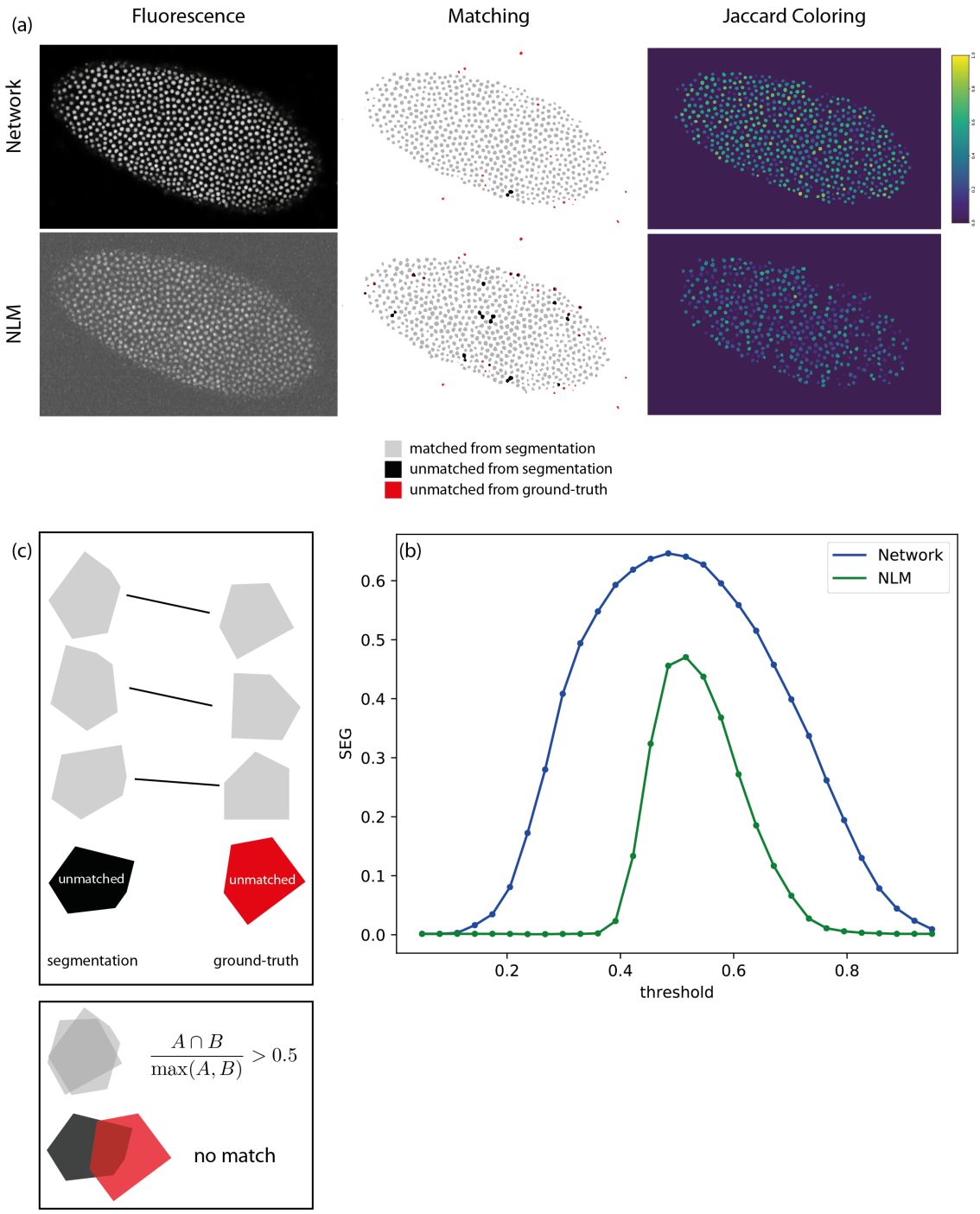


Supplementary Figure 3: Restoration of low-SNR multi-tiled acquisitions of *Schmidtea mediterranea*. **(a)** Comparison of restoration results with the input (low SNR) and ground-truth. **(b)** Zoomed inset from region above.

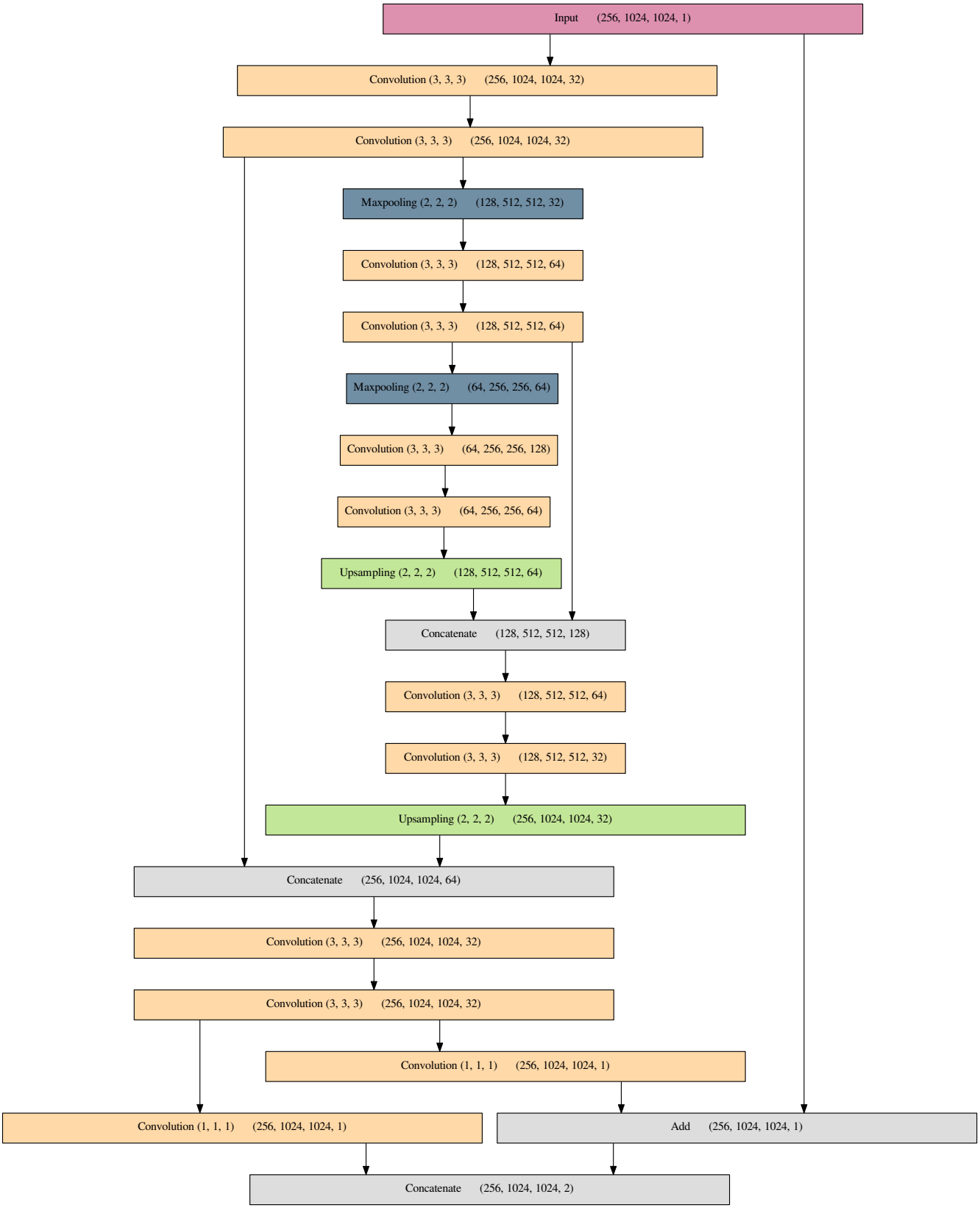




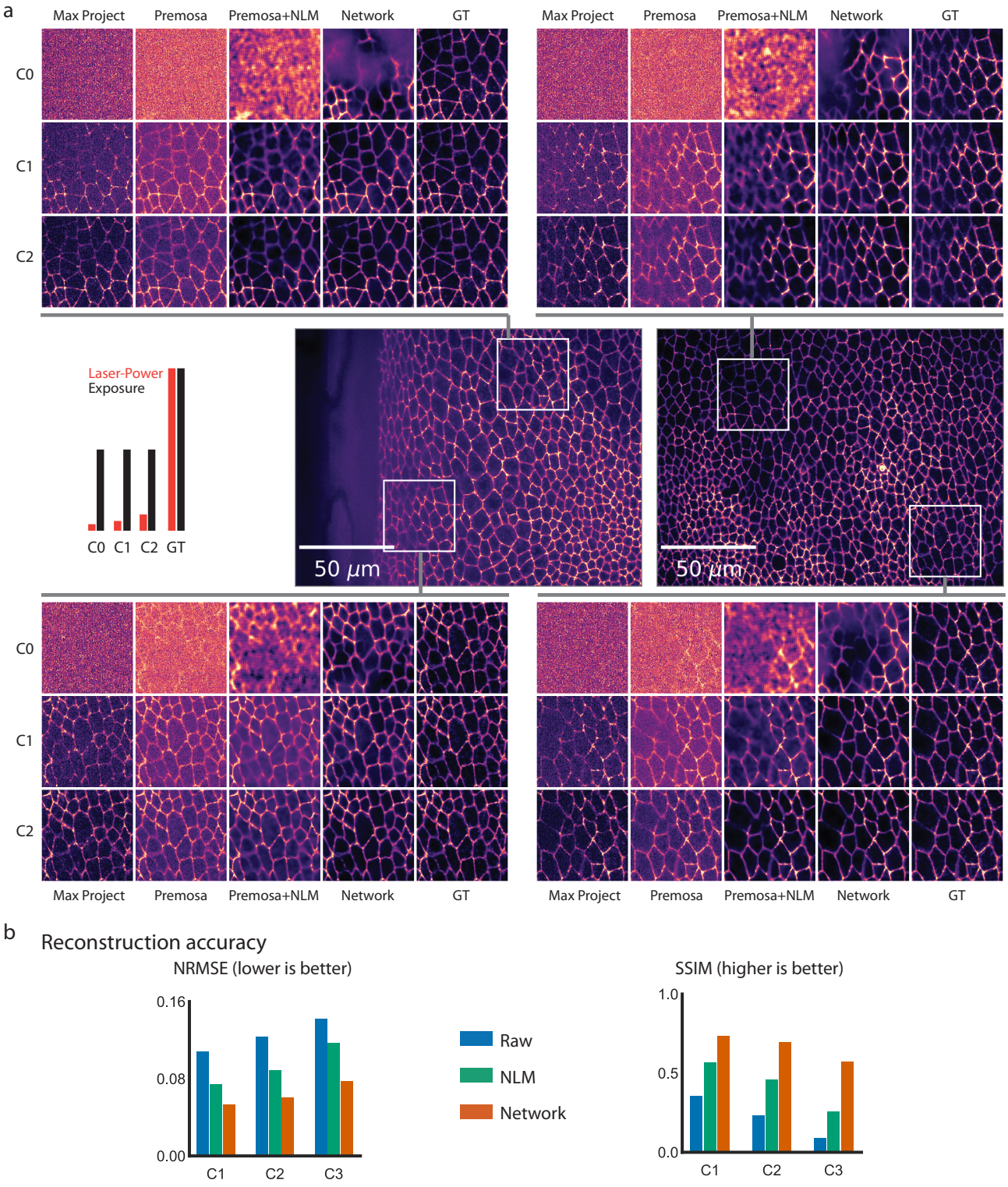
Supplementary Figure 4: Restoration of low-SNR images of *Tribolium castaneum*. Spinning disk confocal acquisitions of a developing embryo (EFA::nGFP labeling) at 3 different low-SNR conditions (C1-C3) and a high-SNR condition (GT). **(a)** Depicted are insets of the input, the best Non-Local-means (NLM) denoising result, the network reconstruction and the high-SNR ground-truth at different positions and tissue depth. **(b)** Reconstruction accuracy measured on held out test set (comprising 6 volumetric stacks for each condition). Shown are normalized root mean squared error (NRMSE, lower is better) and structural similarity (SSIM, higher is better). Note, that the parameters of the NLM baseline *were optimized on the test-set*, which would normally not be possible.



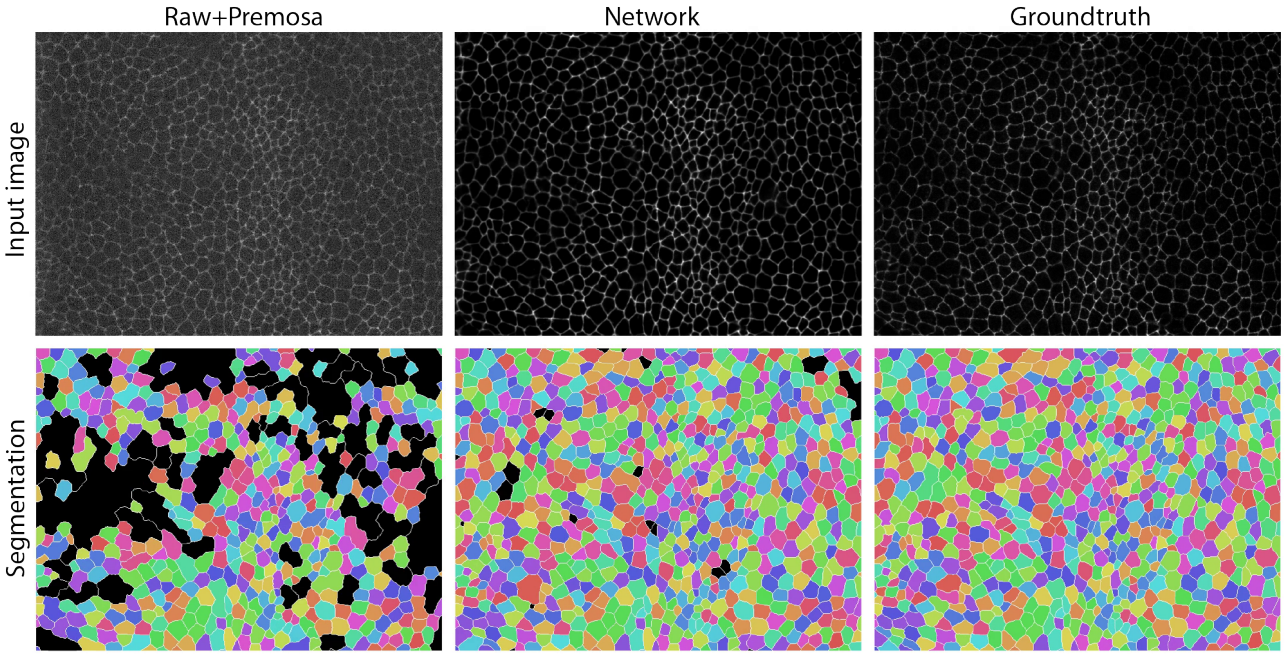
Supplementary Figure 5: Segmentation comparison of CARE network restoration (Network) *vs*. Non-Local-means denoised (NLM) images of *Tribolium castaneum*. **(a)** From left to right: Max projections of fluorescence intensity, segment matching, and Jaccard coloring of individual nuclei. Jaccard coloring assigns maximum Jaccard score (see Eq. (2.4)) to each nucleus. The segment matching allows simultaneous depiction of unmatched nuclei both from the segmentation as well as from ground-truth images. **(b)** SEG score [4] as a function of threshold for network restoration and Non-Local-means denoised images. A score of 1 corresponds to a perfect pixel-wise segmentation. The network and NLM restoration segmentations have optimal SEG scores of 0.65 and 0.47 respectively. Note that the broader peak for the network result is beneficial in the default situation when ground-truth is not available and the user attempts to optimize segmentation quality via visual inspection. **(c)** A schematic depicting the bipartite matching criterion for nuclei.



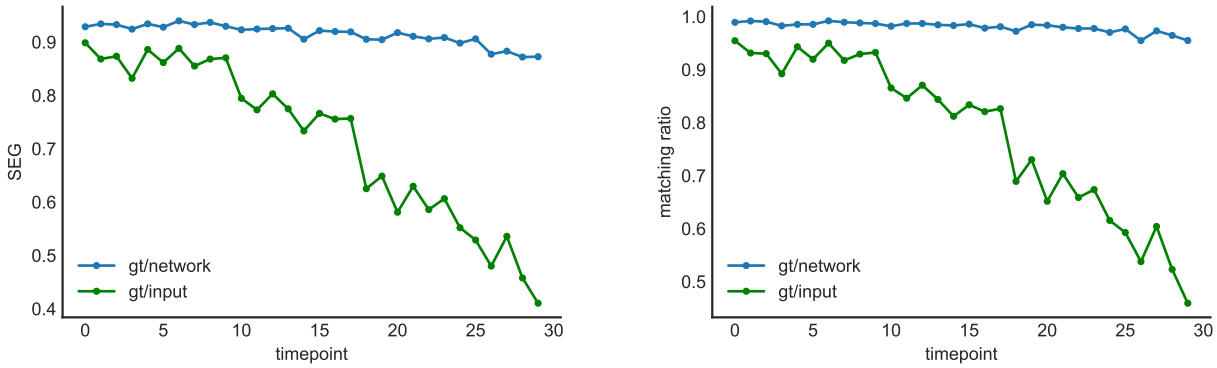
Supplementary Figure 6: Network architecture used for low-SNR restoration (shown for 3D inputs). We use an architecture based on [5], but predict a per-pixel distribution (parameterized by mean and scale) resulting in a 2-channel output. The mean is calculated as the sum of the input and the internal predictions (*residual* addition layer near the end).



Supplementary Figure 7: Joint surface projection and denoising of developing *Drosophila melanogaster* wing epithelia. Spinning disk confocal acquisitions of a developing embryo at 3 different low-SNR conditions (C1-C3) and a high-SNR condition (GT). **(a)** Depicted are insets of the input, the network reconstruction and the high-SNR ground-truth at different locations.**(b)** Reconstruction accuracy measured on held out test set (comprising 26 volumetric stacks for each condition). Shown are normalized root mean squared error (NRMSE, lower is better) and structural similarity (SSIM, higher is better). We compare the error between ground-truth *vs*. projected raw stacks, the network prediction and a strong denoising baseline (Non-local-means with parameters tuned to the test set).

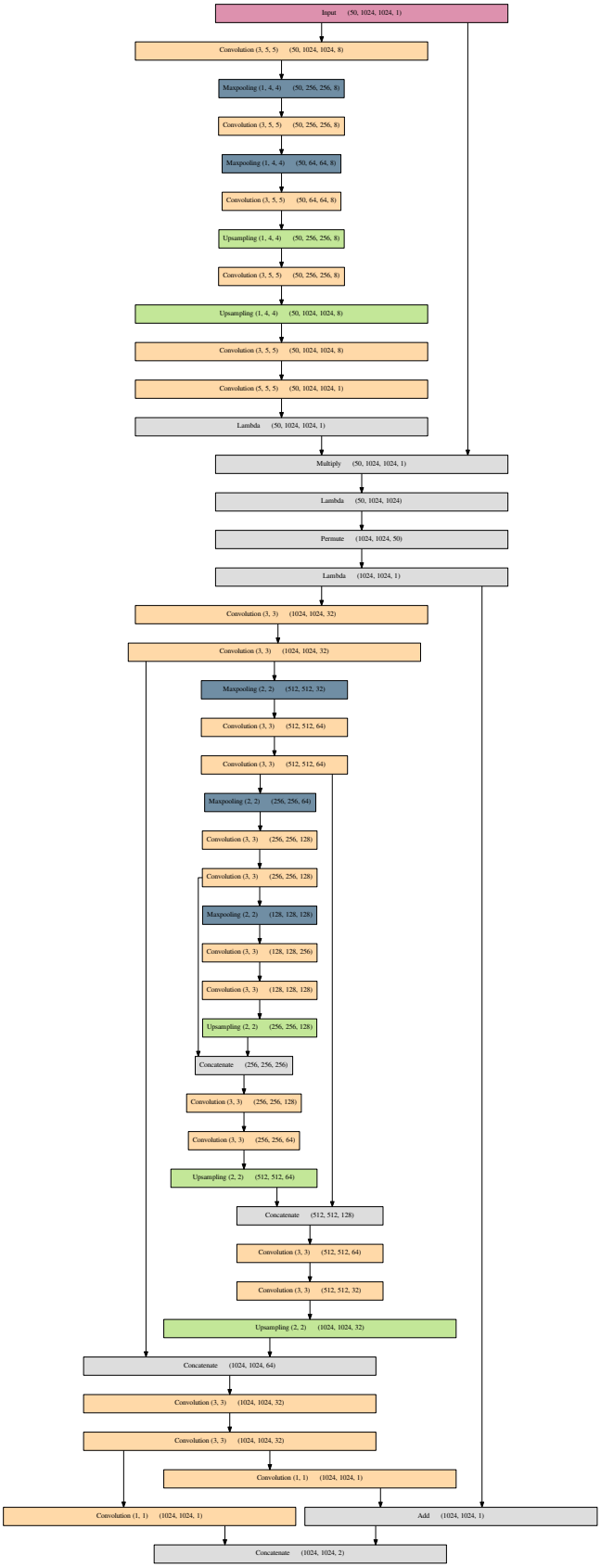


(a) From left to right (upper row): Surface projected input stack (obtained with Premosa [1]), the network projection, and the surface projection ground-truth (attained at 6x higher exposure time, again with Premosa). Bottom row shows the corresponding cell segmentation/tracking results after applying a standard random-forest based pipeline [6, 7]). Cells that have no corresponding match in the ground-truth are labeled black.

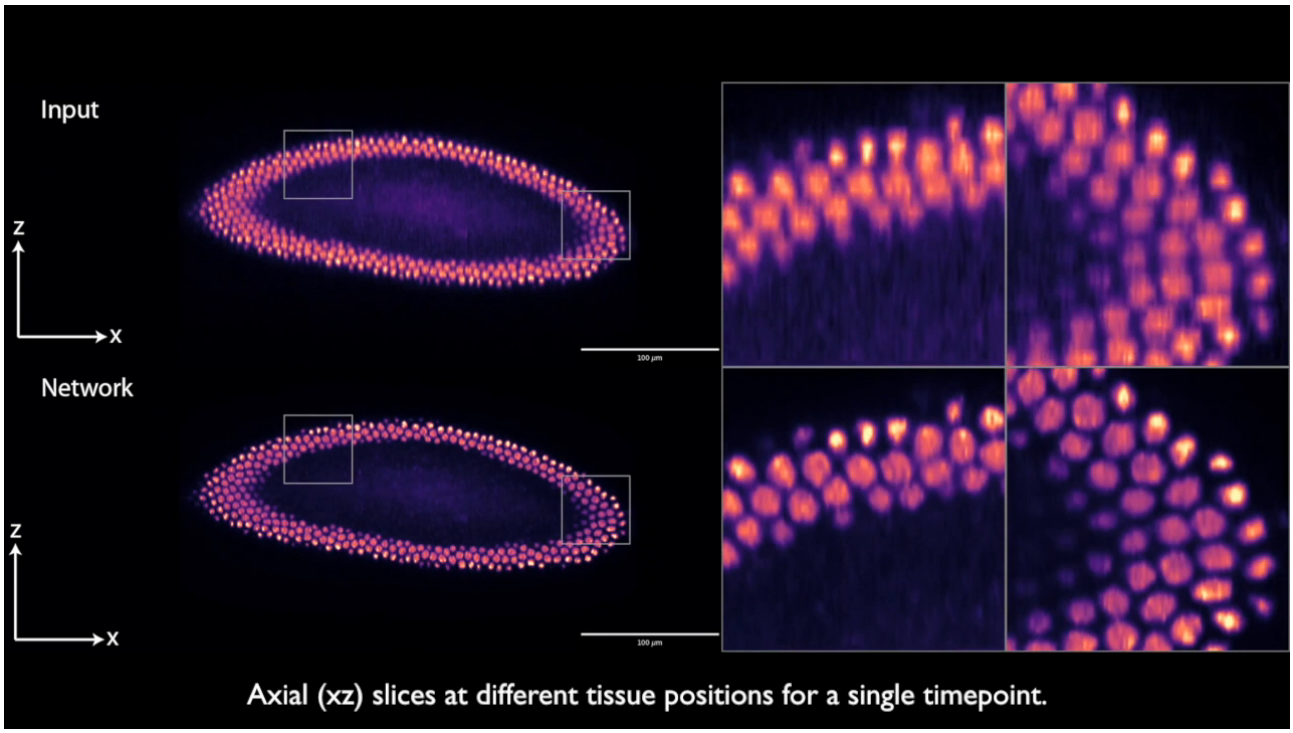


(b) Segmentation error *vs*. ground-truth over time. Both SEG-Score [4] and fraction of correctly matched cells is improved for the network reconstruction. Note that the simultaneous acquisition of high exposure ground-truth and the subsequent photo-bleaching leads to impoverished segmentation results for later time-points on input stacks, but not on the network restorations.

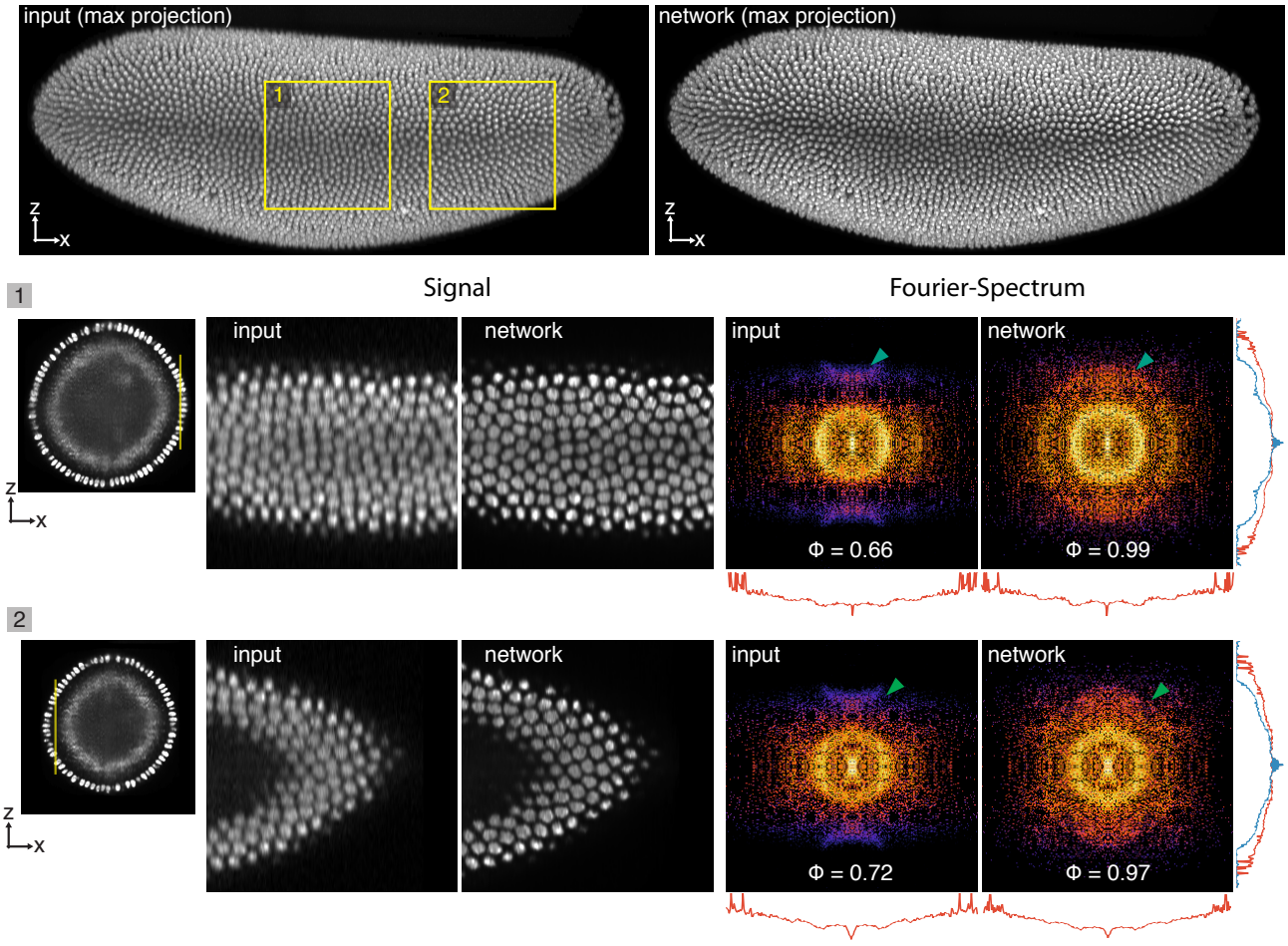
Supplementary Figure 8: Cell segmentation and tracking of a projected time-lapse of developing *Drosophila* wing epithelia.



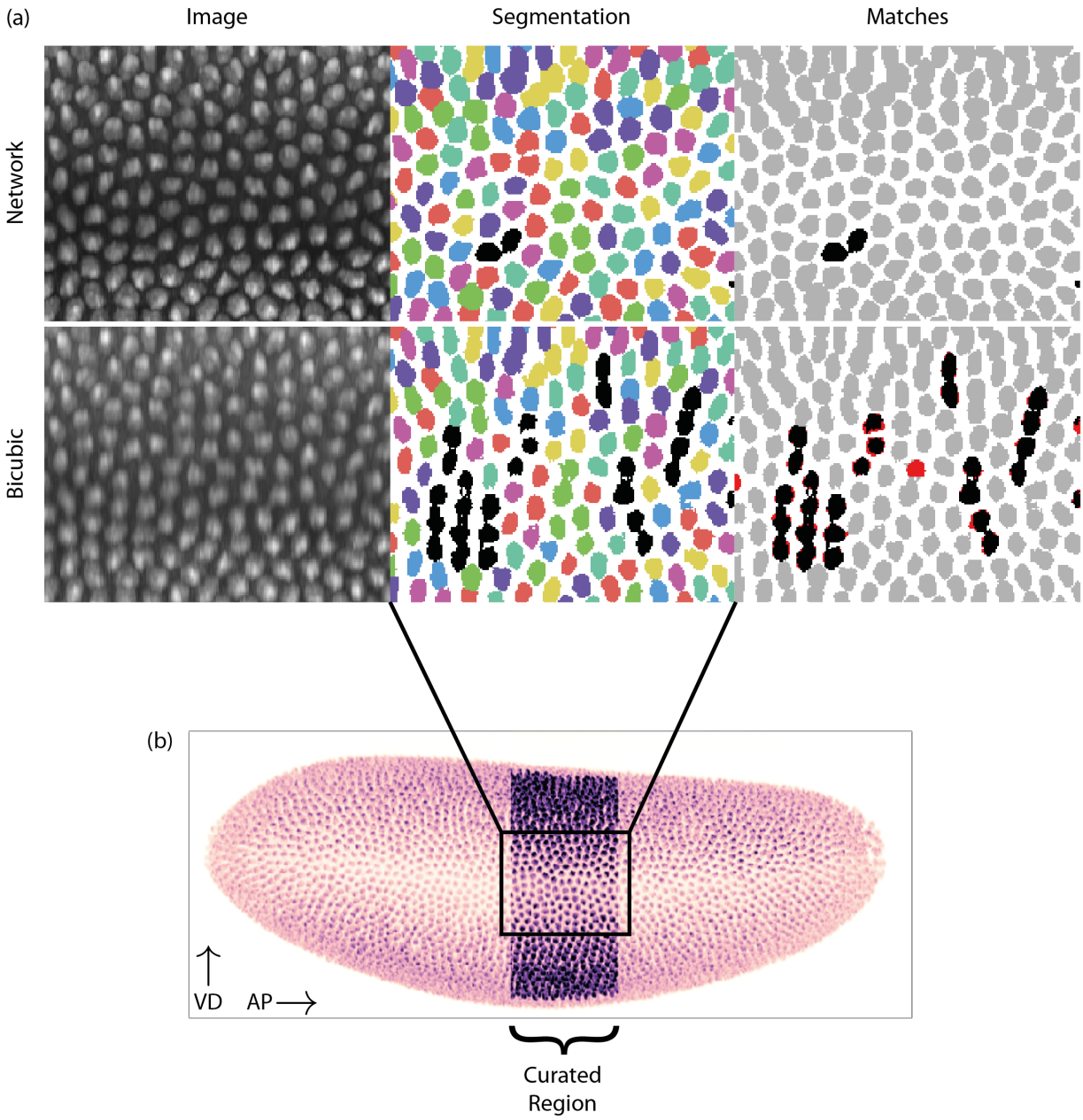
Supplementary Figure 9: Network architecture used for joint surface projection and denoising. The first encoder-decoder part (until "Multiply"-layer) estimates the probability of a certain pixel to belong to the surface, whereas the later parts restore and denoise the projected signal.



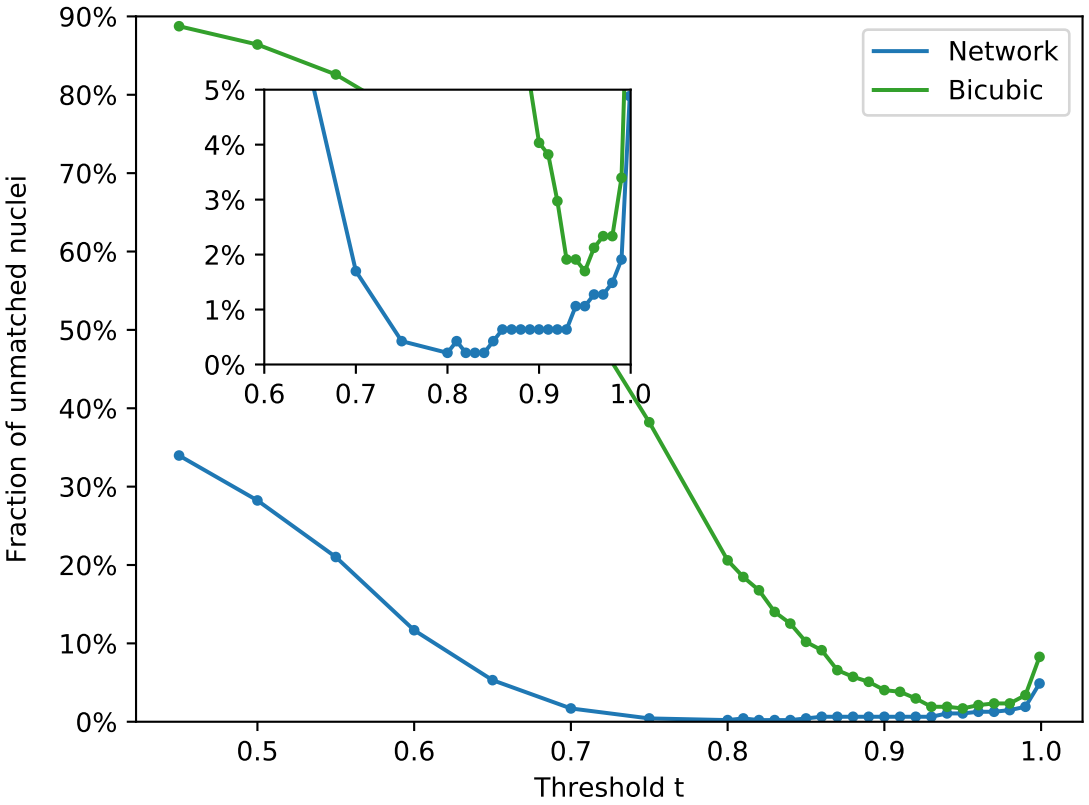
Supplementary Figure 10: Isotropic reconstruction of time-lapsed acquisitions of *Drosophila melanogaster* taken from [2]. Selected (xz) views from the raw (yet already pre-processed) acquisitions (5-fold under-sampled in z, top row) and the isotropic restoration (bottom row). Note that the restoration clearly improves resolution in areas where nuclei are densely packed.



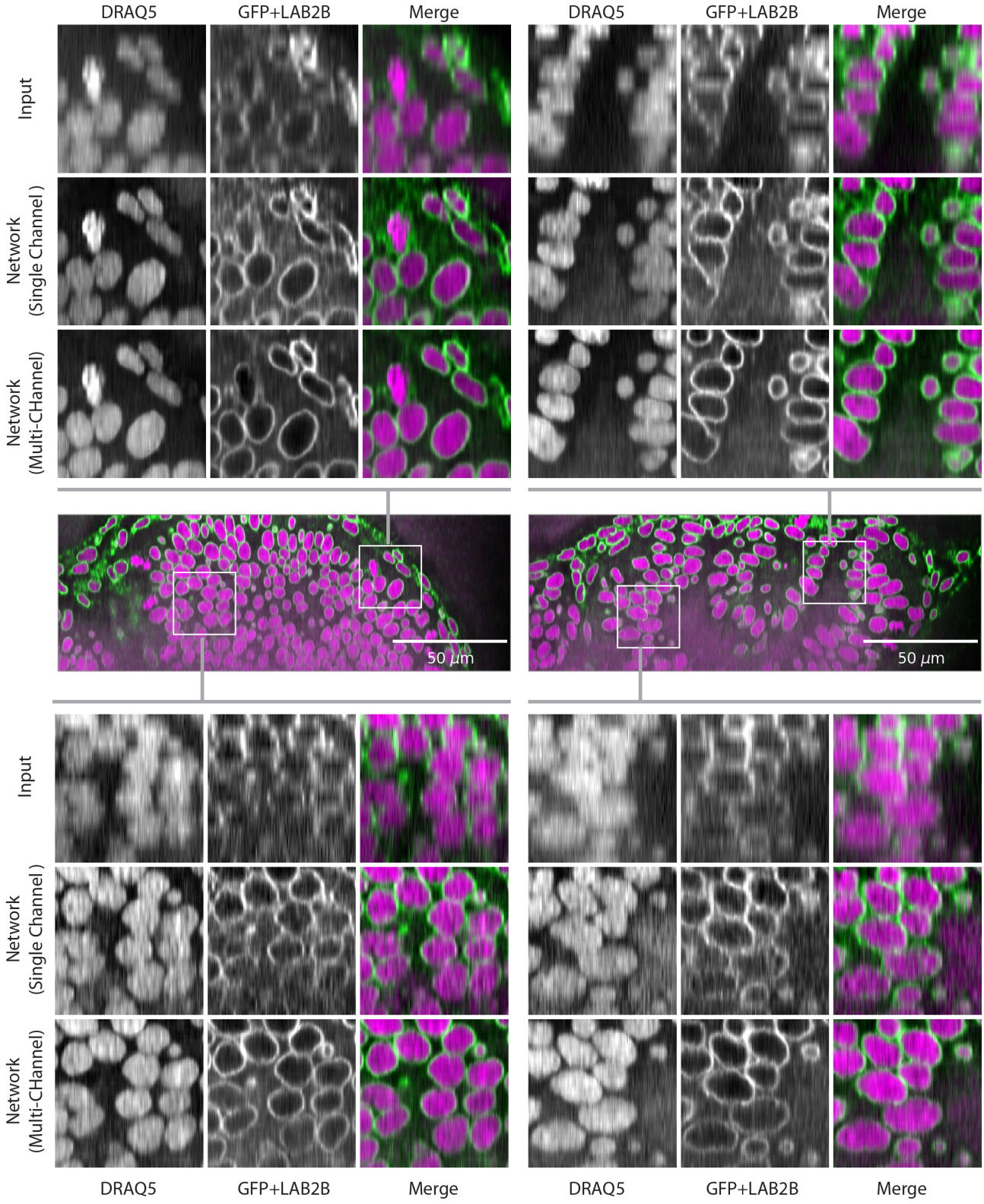
Supplementary Figure 11: Isotropic reconstruction of time-lapsed acquisitions of *Drosophila melanogaster* taken from [2] (5-fold under-sampled in z). Axial resolution restoration as quantified by analyzing the ratio Φ of axial *vs*. lateral spectral energy in Fourier space: $\Phi = 0.66/0.72$ (Input), $\Phi = 0.99/0.97$ (Network). Note that the missing axial information along the z-axis in Fourier-space is filled up by the restoration.



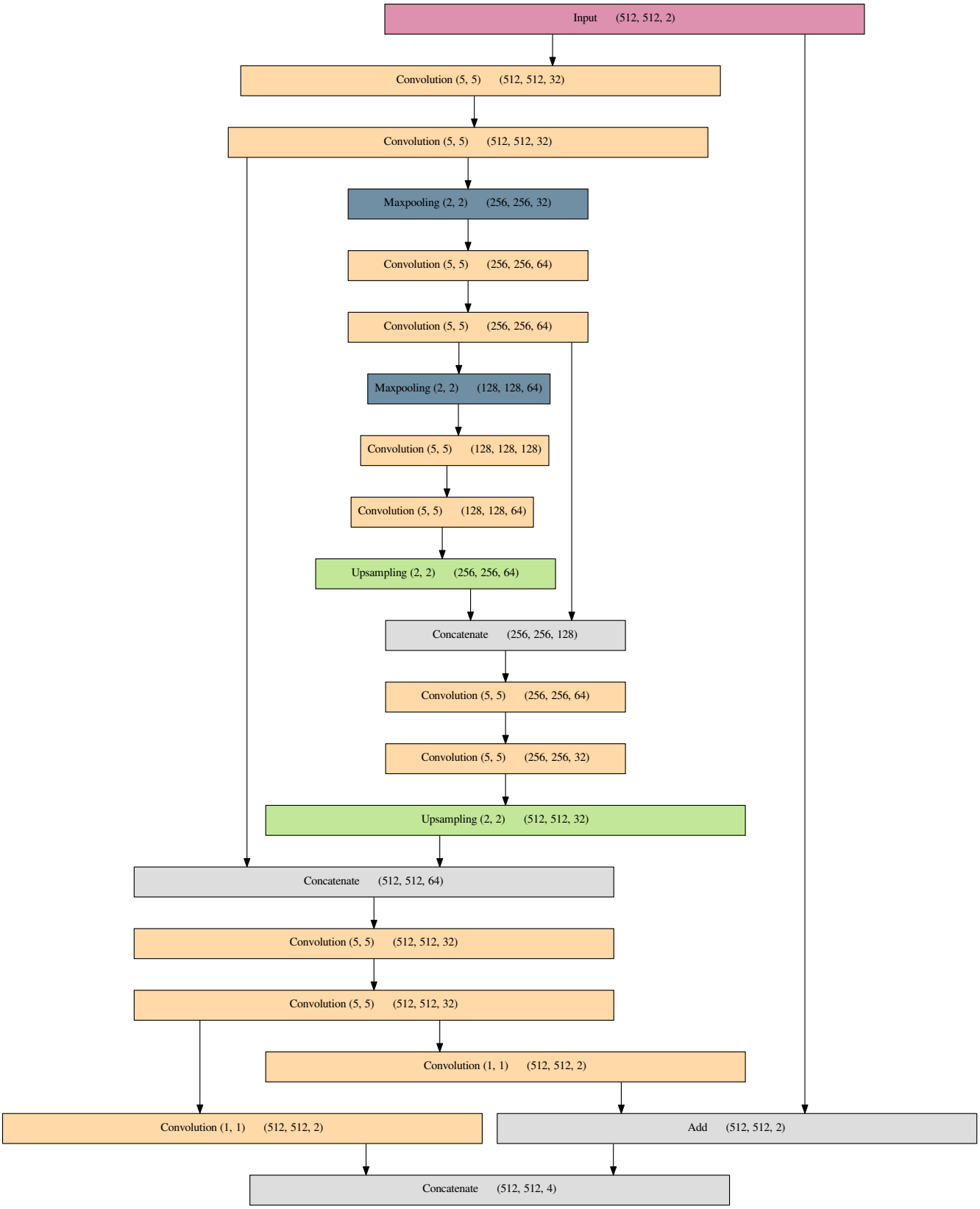
Supplementary Figure 12: Comparison of 3D nuclei segmentation of bicubically upsampled raw (Bicubic, bottom) and isotropically reconstructed (Network, top) images of *Drosophila* embryos. **(a)** From left to right: Fluorescence intensity, segment coloring of individual nuclei, and matched (gray) and unmatched (black/red) segments when compared to manual ground-truth segmentation. The unmatched cells consist of poorly segmented regions that have no ground-truth counterpart (black) or ground-truth segments that failed to be segmented (red). Note how the black, undersegmented nuclei fuse along the vertical axis in the raw segmentation, orthogonal to the detection plane. Isotropic restoration allows accurate segmentations even in regions of the embryo where nuclei are densely packed along the vertical axis. **(b)** Drosophila embryo with RFP-tagged histone imaged at 3 hrs. Dark vertical band shows region of the embryo that was curated for use in the scoring of segmentations.



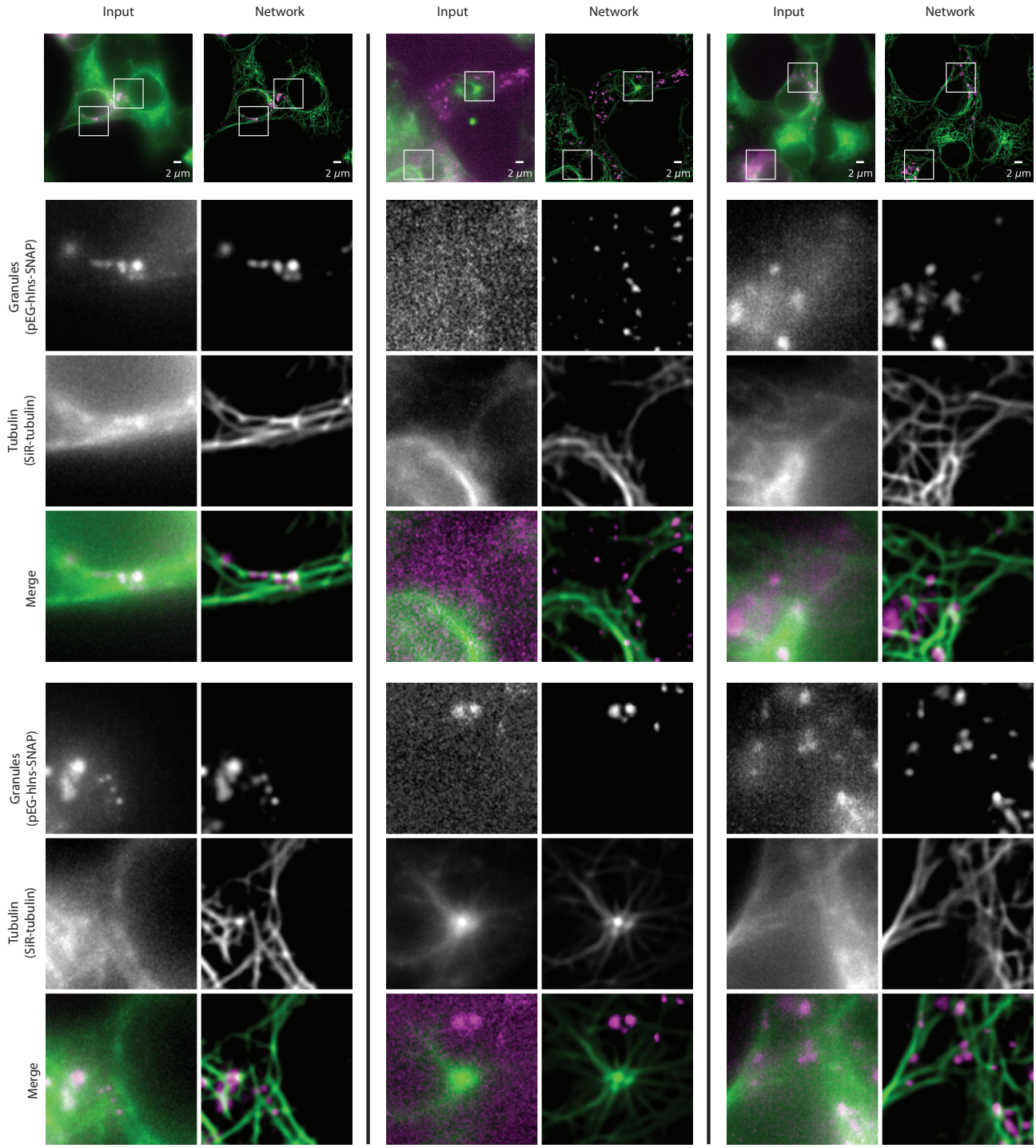
Supplementary Figure 13: Segmentation error for isotropically restored (Network) and bicubically upsampled (Bicubic) images of *Drosophila* embryos. Shown is the fraction of unmatched nuclei (out of all 471 ground-truth nuclei) for varying levels of the threshold t. The lowest error is 0.2% (Network, $t = 0.8$), and 1.7% (Bicubic, $t = 0.95$). Note the wide region over which isotropically restored images produce a lower fraction of unmatched nuclei relative to bicubically upsampled images. For a visual comparison of the optimal segmentations see Supp. Fig. 12.



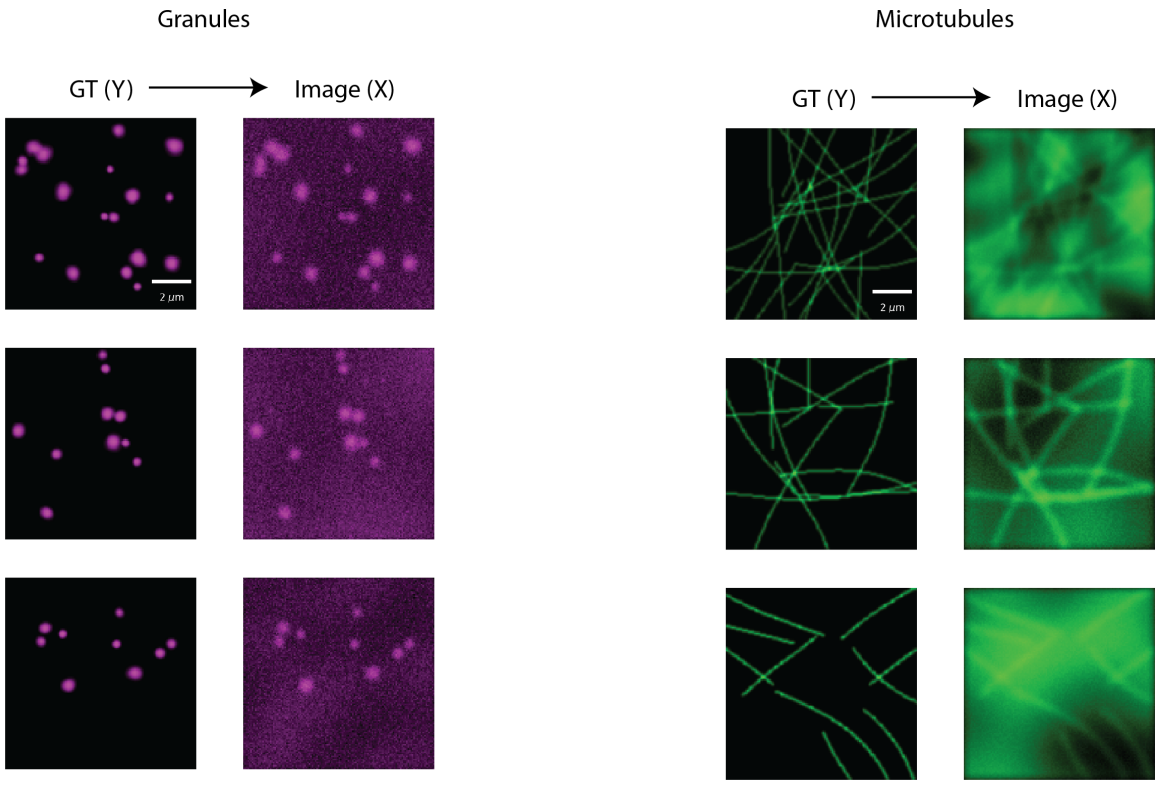
Supplementary Figure 14: Nucleus (DRAQ5, magenta) and nuclear envelope (GFP+LAB2B, green) labeled volumes of a developing *Danio rerio* retina. The volumetric stacks exhibit $\sigma = 10.2$ anisotropic z sampling. Different (xz) slices for different stacks/positions are shown, for the input (top row in each sub-panel), the network reconstruction when training a separate network for each channel (middle) and when training a network that restores both channels simultaneously (bottom row). The reconstruction quality of the multi-channel network is visibly superior compared to the single-channel case, showing that cross-channel information is beneficial for restoration.



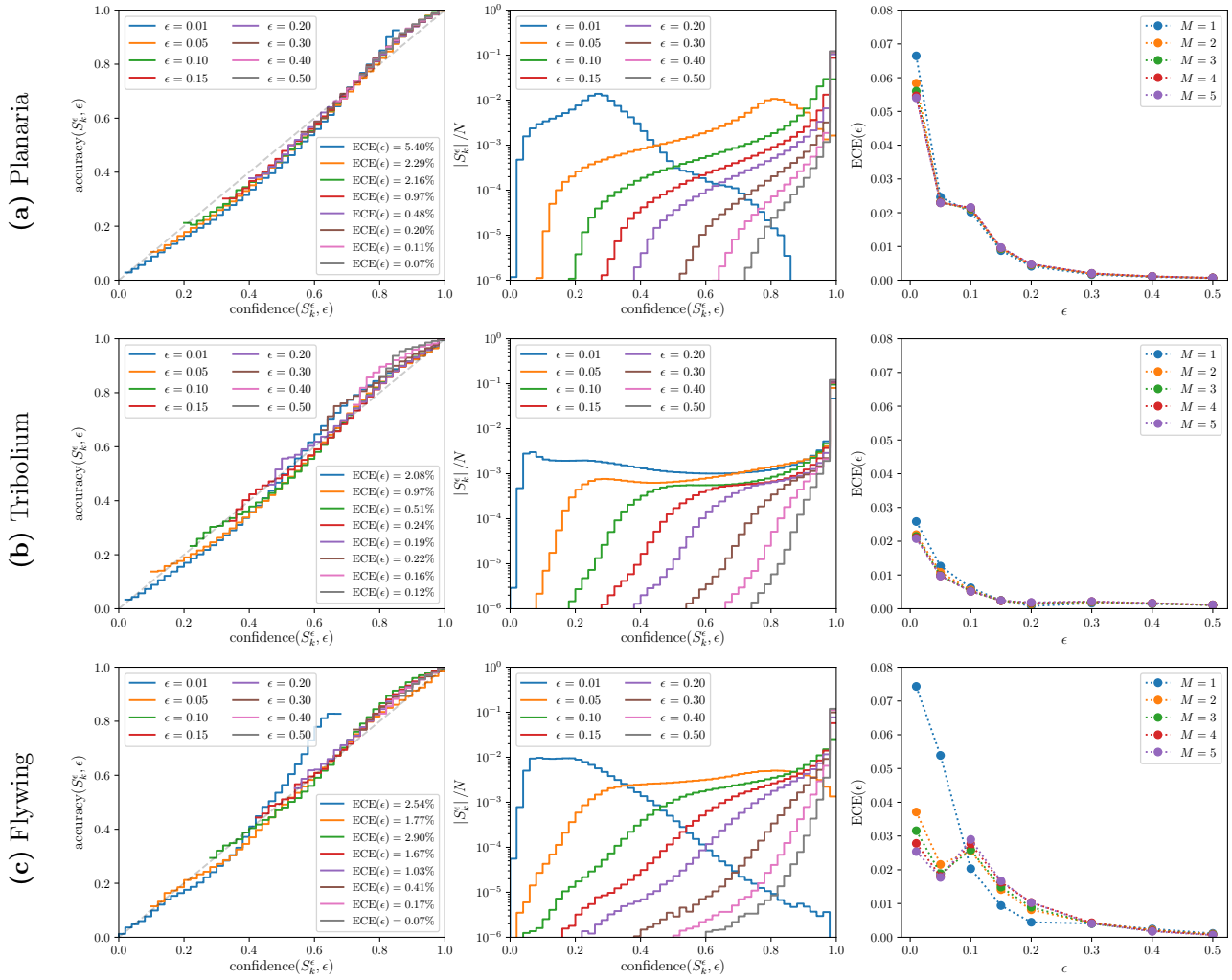
Supplementary Figure 15: Network architecture used for isotropic reconstruction.



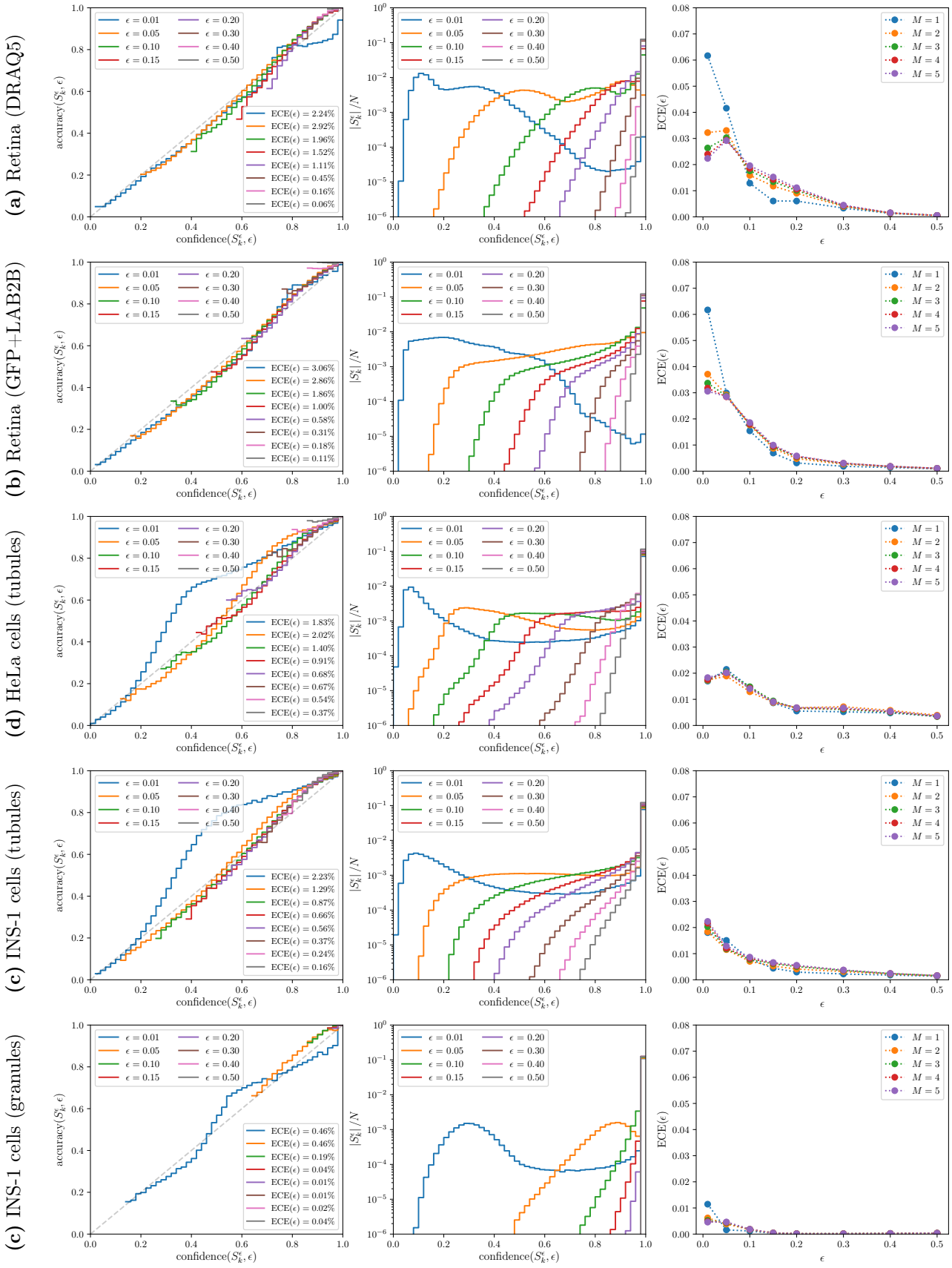
Supplementary Figure 16: Enhancement of diffraction-limited structures in widefield images. Secretory granules (pEG-hIns-SNAP) and microtubules (SiR-tubulin) in insulin-secreting rat INS-1 cells. Shown are reconstruction results for 3 different cells, acquired with DeltaVision OMX microscope in widefield mode. Input images (left columns), network reconstructions (right columns).



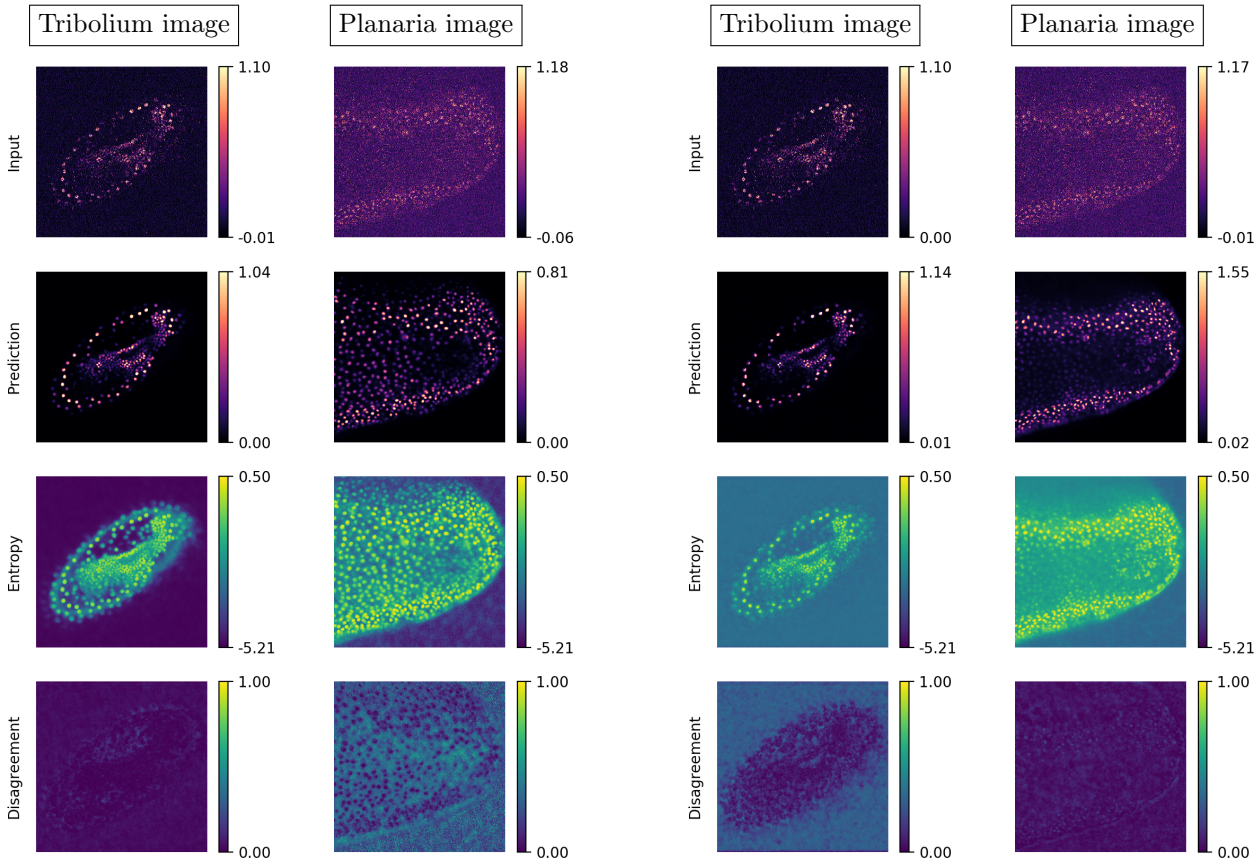
Supplementary Figure 17: Synthetically generated training data for secretory granules and microtubules structures. We first generate the ground-truth images (Y) by simulating granular regions (left) or tracing tubular bundles of different sizes (right) on pixel grids of size 128×128 and pixel-size $0.08\mu m$. We then simulate the microscopy image (X) by adding low frequency perlin-noise resembling background fluorescence, convolving with the microscope PSF, and adding Gaussian and Poisson camera noise.



Supplementary Figure 18: Evaluating uncertainty of our network ensembles. Given an interval around each predicted restored pixel value (with size based on ϵ), we count a *success* if the true value is indeed contained in the interval. Furthermore, we compute the *confidence* of the ensemble as the probability that the true value will fall into this interval. The predicted per-pixel distributions are well-calibrated if the accuracy, *i.e.* the fraction of successfully predicted pixels, is close to the average confidence. Furthermore, we compare confidence and accuracy more granularly for subsets of pixels within a certain confidence range. For each of these subsets, we plot average confidence *vs.* accuracy, which gives rise to a *reliability diagram* (left column); note that we do this for intervals of various sizes as defined by ϵ. The network ensemble is well-calibrated if the plotted curves are close to the identity function (dashed, gray). Additionally, we measure the *expected calibration error (ECE)*, which is essentially the average (weighted) difference of a curve to the identity function. The middle column shows the fraction of pixels that fall into each of the confidence ranges. The right column shows the expected calibration error as a function of interval size ϵ for ensembles of sizes $M = 1, \ldots, 5$.



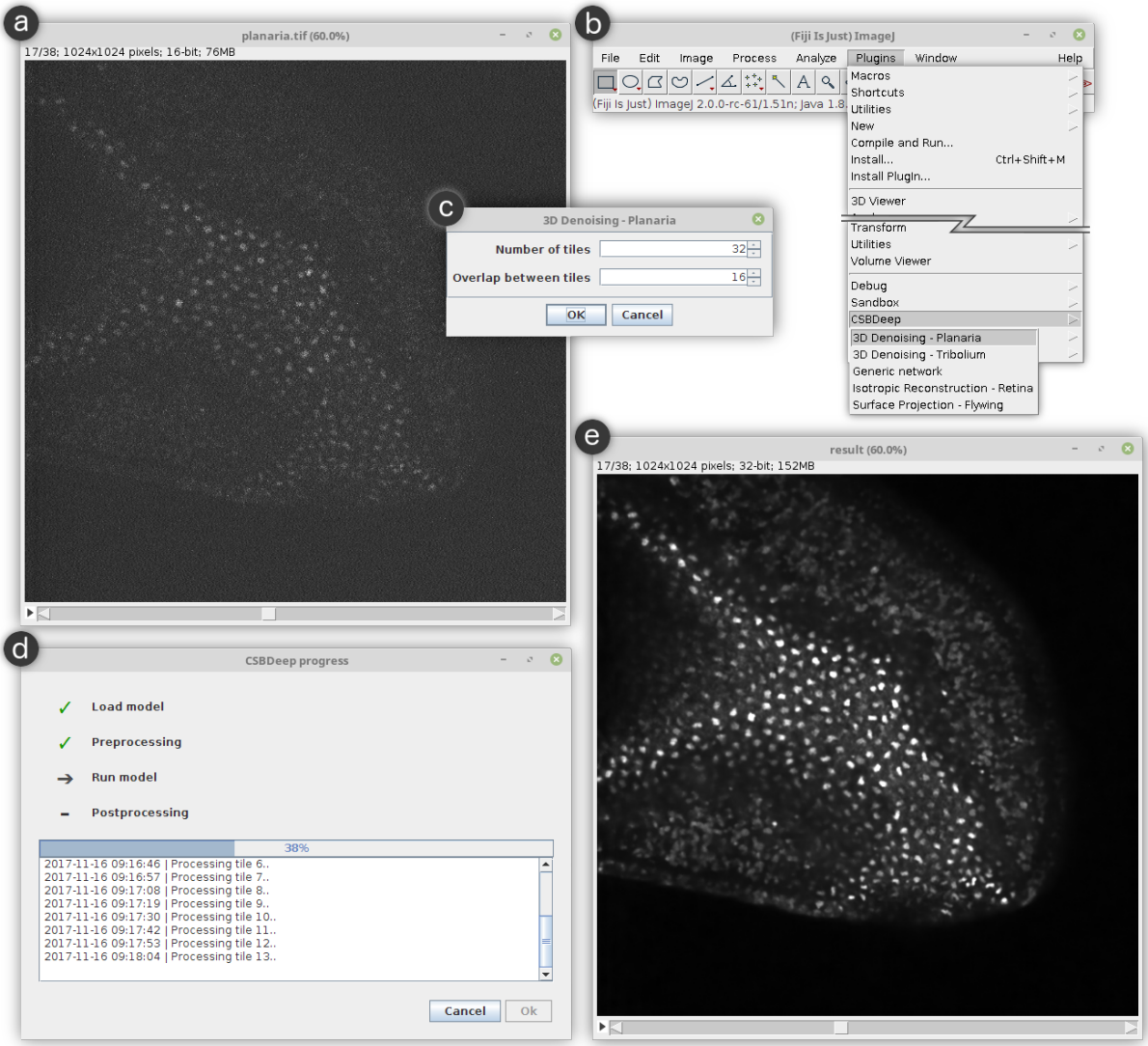
Supplementary Figure 19: Evaluating uncertainty of our network ensembles (continued from Supp. Fig. 18).



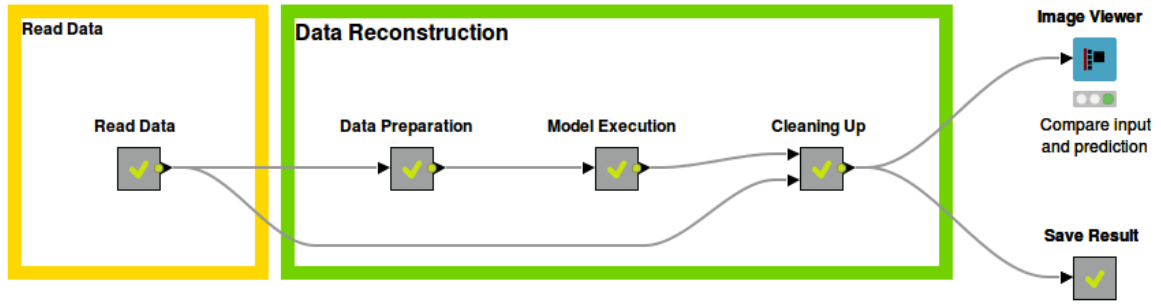
(a) Network ensemble for Tribolium

(b) Network ensemble for Planaria

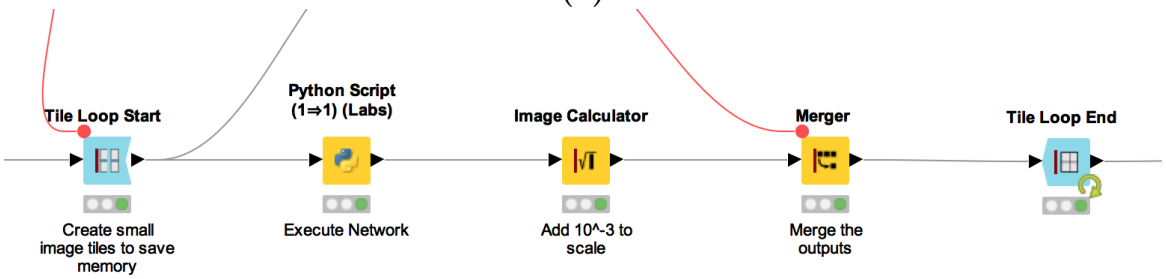
Supplementary Figure 20: Two network ensembles applied to two input image stacks, yielding four different restorations. From top to bottom, we show slices of the input, the predicted restored image, the entropy of the ensemble, and the ensemble disagreement. We applied each network ensemble to an in-distribution (ID) image (*i.e.*, an image similar to those the ensemble was trained on; first and last column) and an out-of-distribution (OD) image (*i.e.*, an image dissimilar to the training data; two middle columns). Disagreement (last row) is often substantially higher when an ensemble is applied to an OD input image.



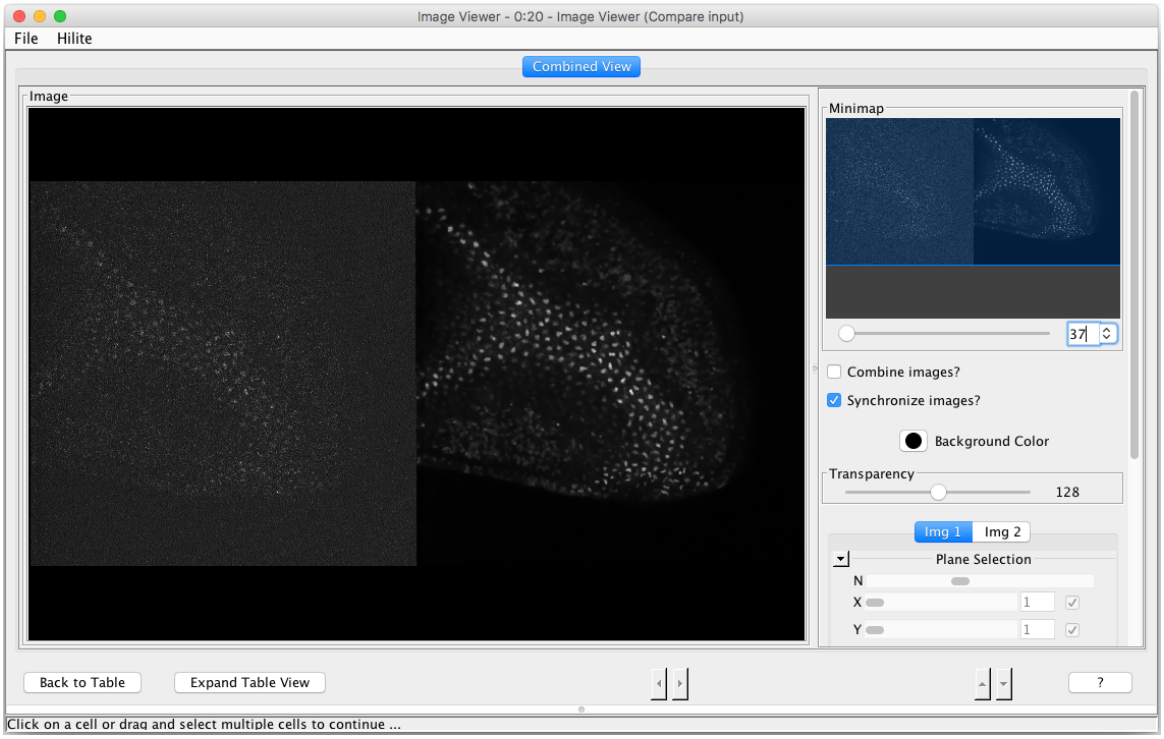
Supplementary Figure 21: Fiji integration of an example denoising workflow. **(a)** is a raw input image, loaded in Fiji. **(b)** the adequate plugin can be started from Fiji's menu *Plugins* ▷ *CSBDeep*. **(c)** Fiji asks the user in how many tiles the input image should be subdivided for processing (see Section 5 for details). **(d)** the started CSBDeep plugin shows progress and status messages during execution of the network. **(e)** once terminated, the final result is shown in a separate Fiji window.



(a)



(b)



(c)

Supplementary Figure 22: KNIME integration of an example denoising workflow. **(a)** shows the overall structure of a CSBDeep KNIME workflow. **(b)** shows the content of the 'Model Execution' node, containing the main computation. **(c)** finally compares the input and result of our Tribolium denoising network in KNIME's image viewer.

Supplementary Tables

Experiment	Restoration Task	Organism	Biological structure	Microscopy
Planaria	Denoising	*Schmidtea mediterranea*	Whole worm	Spinning-disk confocal
Tribolium	Denoising	*Tribolium castaneum*	Whole embryo	Confocal
Flywing	Surface Projection	*Drosophila melanogaster*	Epithelium	Spinning-disk confocal
Drosophila	Isotropic Reconstruction	*Drosophila melanogaster*	Whole embryo	Light-sheet microscopy
Retina	Isotropic Reconstruction	*Danio rerio*	Nuclei, Nuclear envelope	Spinning-disk confocal
Liver	Isotropic Reconstruction	*Mus musculus*	Nuclei, Sinusoidal network	Confocal
INS-1 cells	Structural Enhancement	*Rattus norvegicus*	Secretory granules, Microtubules	DeltaVision widefield mode
HeLa cells	Structural Enhancement	*Homo sapiens*	Microtubules	TIRF

Supplementary Table 1: Overview of experimental data sets.

Supplementary Notes

1 Image Restoration with Convolutional Neural Networks **29**

1.1 Convolutional Neural Networks 30

1.1.1 CNN architecture 30

1.1.2 CNN training 31

2 Restoration Experiments / Online Methods **32**

2.1 Overview 32

2.2 Preliminaries 32

2.3 Image Restoration with Physically Acquired Training Data 33

2.3.1 Planaria (Denoising) 33

2.3.2 Tribolium (Denoising) 34

2.3.3 Flywing (Joint Surface Projection and Denoising) 36

2.4 Image Restoration with Semi-synthetic Training Data 38

2.4.1 Drosophila (Restoration of Anisotropic Volumes) 39

2.4.2 Retina (Restoration of Anisotropic Volumes) 41

2.4.3 Liver (Restoration of Anisotropic Volumes) 41

2.5 Image Restoration with Synthetic Training Data 42

2.5.1 INS-1 cells (Structural enhancement beyond the diffraction limit) 42

2.5.2 HeLa cells (Structural enhancement beyond the diffraction limit) 43

3 Reliability of Image Restoration **44**

3.1 Modeling Uncertainty 45

3.1.1 Aleatoric uncertainty 45

3.1.2 Epistemic uncertainty of model parameters 48

3.1.3 Validation and visualization 49

3.2 Identifying Problematic Image Regions 52

3.2.1 Ensemble disagreement 52

3.3 Discussion and Empirical Validation 55

4 Sample Preparation and Imaging Procedures **57**

4.1 Planaria 57

4.2 Tribolium 57

4.3 Flywing 57

4.4 Liver 58

4.5 Retina 58

4.6 Drosophila 59
4.7 INS-1 cells 59
4.8 HeLa cells 59

5 Deployment of Image Restoration Models **60**
5.1 Fiji Integration 60
5.1.1 Extending the available functionality 61
5.2 Knime Integration 62

1 Image Restoration with Convolutional Neural Networks

Image restoration, in general, is the problem of reconstructing an (latent) image given a distorted (corrupted) version of it. In the context of fluorescent microscopy, typical distortions include the effects of camera and photon noise, the blur of the optical point-spread-function (PSF), the resolution loss due to under-sampling, or the optical aberrations induced by the heterogeneity of the tissue. The distorted, observed image x can be thought of being the result of a function $x = f(y)$ (the *forward model*) applied to the true (latent) image y. The problem then is to recover y, given x. Typical image restoration tasks with their corresponding forward models are:

- Denoising: $f(y_i) = \mathcal{P}(x_i) + \eta$ (with Poisson noise $\mathcal{P}$ and camera noise η)
- Deconvolution: $f(y) = h \otimes x$ (with the PSF h)
- Upsampling: $f(y) = \mathcal{S}_\alpha x$ (with subsampling factor α)

Although f is often easy to evaluate, the inverse $y = f^{-1}(x)$ is typically hard to compute, and often it is not even uniquely defined (*i.e.* there are a multitude of potential latent images). A classical approach to solve this problem to find the minimizer of a cost function $\mathcal{E}$

$$\hat{y} = \arg\min_y \mathcal{E}(y|x), \quad \text{where} \quad \mathcal{E}(y|x) = \textsc{DataTerm}_f(x, y) + \textsc{RegularizationTerm}(y)$$

consisting of a *data term* that encourages the restored image to adhere to the forward model f, and a *regularization term* that favors solutions based on prior assumptions about the latent image. Although this approach can work well in many cases, it has major problems. First, the forward model (and thus the data term) is sometimes not fully accurate by neglecting to model some sources of image corruption which are either unknown or hard to describe mathematically. Second, and more important, the regularization term typically encodes rather *weak* assumptions about the latent image, often simply favoring smooth images by penalizing strong differences of neighboring pixels. Third, minimizing the cost function $\mathcal{E}$ is often computationally demanding, resulting in long runtimes or the need for approximations to find solutions more quickly. In fact, this issue exacerbates the first two problems since data and regularization terms are sometimes modified to make minimizing $\mathcal{E}$ easier.

In this work, we take an alternative approach and directly *learn* a function g that computes the restored image $\hat{y} = g(x) \approx \bar{f}^{-1}(x)$ for input image x, thereby effectively approximating the inverse of the true forward model $\bar{f}$ (which may be unknown). We do on the basis of a sufficiently large set of pairs (x, y) of corrupted images x and their corresponding true images y, and by selecting g as a *convolutional neural network*. Note that this general approach has already been pursued by [8–10] in different and more specialized contexts.

1.1 Convolutional Neural Networks

Deep learning [*cf.* 11, 12] is currently experiencing a renaissance in computer vision and machine learning. For problems involving images, convolutional neural networks (CNNs) have demonstrated impressive results for many problems, such as image classification [*e.g.*, 13]. Furthermore, CNNs have also been shown to work well for image restoration problems (see [9] for a recent review).

Their main building blocks (called *layers*) are *convolutions* (with small *filters*, *e.g.* $3 \times 3 \times c$ pixels for a 2D image with c channels) and non-linear *activation functions* (which are applied per-pixel). While activation functions are often fixed, the convolutional filters are typically learned. A CNN can compute complicated functions (of the local image content) by simply interleaving convolutional layers and activation functions. Note that internal convolutional layers of the CNN are also called *feature layers*, because they often represent certain structural image features (*e.g.*, edges or corners).

On top of this, an *encoder-decoder* architecture is often employed, which has been found to increase the modeling power of CNNs. To that end, internal feature layers are first compressed in the encoder part by reducing their spatial resolution with *pooling* layers (typically by taking the maximum in a local neighborhood), hoping that this will lead to better (more high-level) features of the image. The decoder part of the network uses *upsampling* to successively enlarge feature layers to eventually yield an output that is typically of the same resolution as the input. However, using upsampling layers can lead to blurred CNN outputs that cannot delineate object boundaries well anymore. Using high-resolution feature layers in the decoder part can remedy this issue. One approach to do this, that we also adopt in this work, is to use so-called *skip connections* that concatenate the feature layers from the encoder part with the upsampled layers in the decoder part [5, 8].

Finally, it has often been shown advantageous (originally by [14]) to not directly learn the output of interest (such as the restored image), but instead the *residual* $\tilde{g}$ to the input of the network, *i.e.* $g(x) = x + \tilde{g}(x)$.

1.1.1 CNN architecture

Our CNN is based on *U-Net* [5], which is an encoder-decoder architecture with skip-connections. In contrast to most CNNs for image restoration, we do not predict a single value per output pixel, but instead a *distribution* (please see Section 3 for a detailed explanation). In particular, we predict the location μ and scale σ of a Laplace probability density function $p(z; \mu, \sigma) = \exp(-|z - \mu|/\sigma)/(2\sigma)$. Hence, our network output for every pixel i is

$$g(x)_i = p\big(\mu(x)_i, \sigma(x)_i\big),$$

where $\mu(x) = x + \tilde{\mu}(x)$ is defined via a residual term. Although our network produces two distinct outputs $\mu(x)$ and $\sigma(x)$, most layers are shared (*e.g.*, Supp. Fig. 6).

	Planaria	Tribolium	Flywing	Retina	INS-1 (tubules)	INS-1 (granules)	HeLa cells
Pooling layers	2	2	3	2	2	2	2
Filter size	3	3	3	5	5	5	5
Init. conv. feat.	32	32	32	32	32	32	32
Topology	Supp. Fig. 6	Supp. Fig. 6	Supp. Fig. 9	Supp. Fig. 15	Supp. Fig. 6	Supp. Fig. 6	Supp. Fig. 6
No. of param.	996802	996802	1386347	923876	923010	923010	923010
Input size	$(16, 64, 64, 1)$	$(16, 64, 64, 1)$	$(50, 64, 64, 1)$	$(128, 128, 2)$	$(128, 128, 1)$	$(128, 128, 1)$	$(128, 128, 1)$
Output size	$(16, 64, 64, 2)$	$(16, 64, 64, 2)$	$(64, 64, 2)$	$(128, 128, 4)$	$(128, 128, 2)$	$(128, 128, 2)$	$(128, 128, 2)$
No. of images	17005	14725	16891	24121	5578	2728	5578
Batch size	16	16	16	64	32	32	32
No. of epochs	200	200	100	100	100	100	100
Learning rate	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004
Training time	30h 06m	26h 12m	10h 17m	06h 15m	01h 23m	00h 39m	01h 23m

Table 1: Details for network architecture and training. The top part of the table provides details w.r.t. the encoder-decoder architecture. Specifically, the number of pooling/upsampling layers (indicating the number of resolution levels), the size of the convolution filters in all image dimensions, and the number of initial convolution filters, which is doubled after each pooling layer, and halved after each upsampling layer. It also refers to the figure where the network topology is shown and lists the total number of network parameters. The bottom part pertains to learning details, first showing the input size of the training images and the corresponding output size of the network; note that a network can be applied to different image sizes after training. Further shown are the number of training images, batch size, number of epochs, learning rate, and total training time on a single Nvidia 1080 GPU.

1.1.2 CNN training

A CNN is defined via the parameters θ of all its layers (here, mostly the convolution filters) and we can use $g_\theta \equiv g$ to make this dependence explicit. Given training data comprised of T input-output image pairs $\{(x^t, y^t)\}_{t=1}^T$, the parameters are typically chosen by minimizing a loss function $L(\theta)$ that assigns a cost to each prediction $g_\theta(x^t)$ based on the knowledge that y^t is the correct output. The loss function $L(\theta)$ is typically minimized via stochastic gradient descent. Stochastic means that the parameters are repeatedly updated based only on random subsets of the training data, called *batches*. Furthermore, an *epoch* denotes the partitioning of the training into batches and processing all of them. The gradient-based parameter updates are guided by a step size, called the *learning rate*.

In contrast to common CNNs for image restoration, we learn the parameters of our network by minimizing a loss function that averages the per-pixel negative log-likelihood of the Laplace distributions (again, see Section 3 for details). We choose Adam [15], a common stochastic gradient-based descent method, to minimize the loss.

2 Restoration Experiments / Online Methods

2.1 Overview

For each of the described experiments, our method consists of the following steps:

1. Generation of training data that reflects both the biological structures and the restoration task under consideration.
2. Training a neural network (or ensemble of networks) on this training data.
3. Applying the learned model on unseen data of the same kind.

In particular, each CARE network is trained for a specific combination of image content (e.g., nuclei, microtubules) and image corruption (camera noise, pixel size, microscope PSF, etc.).

2.2 Preliminaries

Before we describe our restoration experiments in details, we need to first address a few preliminaries. First, please find the details of the used convolutional neural networks as well as training specifics in Table 1.

Image normalization. For both training and prediction it is important to normalize the input images to a common range, which is typically done by scaling the input to a given minimum and maximum value. Microscopy images, however, often exhibit isolated pixels with extremely dim and bright values that do not represent the actual image content (*dead* and *hot* pixels of the camera). Hence, we use a simple *percentile*-based normalization instead, which we define for an image u as

$$\mathrm{N}(u; p_{\mathrm{low}}, p_{\mathrm{high}}) = \frac{u - \mathrm{perc}(u, p_{\mathrm{low}})}{\mathrm{perc}(u, p_{\mathrm{high}}) - \mathrm{perc}(u, p_{\mathrm{low}})}, \tag{2.1}$$

where $\mathrm{perc}(u, p)$ is the p-th percentile of all pixel values of u. We typically use values of $p_{\mathrm{low}} \in (1, 3)$ and $p_{\mathrm{high}} \in (99.5, 99.9)$.

Image quality assessment. Since the restored image $\hat{y}$ predicted by our network and the corresponding ground-truth image y typically differ considerably in the dynamic range of their pixel values, we cannot simply use standard image quality metrics without normalizing the images first. To that end, we first normalize the ground truth image y by using Eq. (2.1) with $p_{\mathrm{low}} = 0.1$ and $p_{\mathrm{high}} = 99.9$. Secondly, we use a transformation $\varphi(\hat{y}) = \alpha\hat{y} + \beta$ that scales and translates every pixel of the restored image based on parameters

$$\alpha, \beta = \arg\min_{\alpha', \beta'} \mathrm{MSE}\big(\mathrm{N}(y, 0.1, 99.9), \alpha'\hat{y} + \beta'\big) \quad \text{with} \quad \mathrm{MSE}(u, v) = \frac{1}{N}\sum_{i=1}^{N}(u_i - v_i)^2. \tag{2.2}$$

In other words, α, β are chosen such that the *mean squared error (MSE)* between $\varphi(\hat{y})$ and $\mathrm{N}(y, 0.1, 99.9)$ is minimal. Note that α, β can be easily computed in closed-form.

Based on above definitions, we define the *normalized root-mean-square error (NRMSE)* between the ground-truth y and the restored image $\hat{y}$ as follows:

$$\mathrm{NRMSE}(y, \hat{y}) = \sqrt{\mathrm{MSE}\big(\varphi(\hat{y}), \mathrm{N}(y, 0.1, 99.9)\big)}. \tag{2.3}$$

Furthermore, we apply the same normalizations to y and $\hat{y}$ before we evaluate the *structural similarity index (SSIM)* [16], which measures the perceived similarity between two images.

2.3 Image Restoration with Physically Acquired Training Data

2.3.1 Planaria (Denoising)

We first considered the problem of restoring volumetric images from low to very low signal-to-noise ratio (SNR) acquisitions in the case of the flatworm *Schmidtea mediterranea*, that is exceptionally sensitive to even moderate amounts of laser light (Supp. Vid. 1).

To generate the training data, we imaged *fixed* samples of nucleus-labeled (*RedDot1*) *Schmidtea mediterranea* (at different developmental stages) under high SNR conditions using high laser power and camera exposure time. As it is common nomenclature in many fields, we refer to the high quality, high SNR acquisitions as *ground-truth* (GT) data. This is justified since they do indeed represent the desired quality of our results. Next we imaged the same samples at 3 different levels of reduced laser power and/or reduced exposure times (C1–C3, *cf.* Table 2):

Condition	Exposure (ms)	Laser-power (mW)
GT	30	2.31
C1	20	0.12
C2	10	0.12
C3	10	0.05

Table 2: Experimental conditions for *Schmidtea mediterranea* acquisitions.

We found it crucial that corresponding image stacks are well aligned for all imaging conditions. We thus interleaved the different imaging (laser power) conditions during plane-by-plane acquisition, thus ensuring properly registered input and ground-truth stacks in all cases. In total, we acquired 96 stacks of average size $1024 \times 1024 \times 400$, resulting in ~ 80 GB of imaged training data. From the so produced well-aligned pairs of high/low SNR volumes, we extracted on the order of ~ 17000 randomly positioned sub-volume pairs (x_i, y_i) of size $64 \times 64 \times 16$ voxels, which were subsequently used for training and validation of a CARE network. As network architecture we used a residual version of a 3D-U-Net type topology (Supp. Fig. 6, [5]), modified to predict the parameters of a Laplace distribution (whose mean we use as final prediction, see

Section 3 for details). The parameters of the training (*e.g.* batch-size, learning rate *etc.*) can be found in Table 1.

After training we applied the network to 22 previously unseen volumes of *Schmidtea mediterranea* images for each of the 3 different conditions. We consistently observed that the reconstructed image data was of very high quality, even if the SNR of the input data to be reconstructed was very low, *e.g.* being acquired with a 60-fold reduced light-dosage (*cf.* Main Figure 1 c, Supp. Figs. 1 to 3 and Supp. Vid. 2). For the reconstruction error between prediction and previously unseen ground-truth we used the NRMSE and SSIM defined before and compared it to results obtained by a strong baseline denoising method (Non-local-means, see Main Figure 1 d and Supp. Fig. 2). To ensure the strongest baseline results possible, we tuned the parameters of Non-local-means (σ, and search-size and -windows) to the test set, thus giving us an upper bound of its performance. In reality, this would of course not be possible (one would have to guess the right parameters), thus the improvement of CARE over that baseline are likely to be higher in practice.

	NRMSE			SSIM		
Condition	Input	NLM	Network	Input	NLM	Network
C0	0.083837	0.051305	0.03768	0.211759	0.542508	0.707738
C1	0.088184	0.062856	0.039419	0.170711	0.402244	0.708026
C2	0.091028	0.08295	0.047804	0.144146	0.221972	0.609718

Table 3: Restoration accuracy for *Schmidtea mediterranea* for the 3 different conditions C1–C3. Shown are NRMSE (lower is better) and SSIM (higher is better) for the input, the network restoration and the Non-local-means (NLM) denoising baseline.

Prediction time for a volume of size $1024 \times 1024 \times 100$ was less than 20 seconds on a single Nvidia Titan X GPU, which allowed for processing of a volumetric 2 hour time-lapse recording in approximately 10 minutes.

2.3.2 Tribolium (Denoising)

As a second example, we considered live-cell imaging of developing *Tribolium castaneum* (red flour beetle) embryos. Again, we used 3 different low-SNR illumination/exposure condition and a corresponding high-SNR ground-truth condition (Table 4). We interleaved all conditions during acquisition, thus ensuring that all images were well aligned.

Condition	Laser-power (%)
GT	20
C1	0.1
C2	0.2
C3	0.5

Table 4: Experimental conditions for *T. castaneum* acquisitions.

In total, we collected 26 training stacks for each condition, acquired from different samples at different developmental stages. The stack size was $\sim 700 \times 700 \times 50$ for each condition, totaling in 14 GB of training data. For testing, we used 6 different volumes per condition, again acquired at different developmental stages. Image reconstruction performs well even on extremely noisy input data with up to 70-fold reduced light-dosage when compared to typical imaging protocols (*cf.* Table 4, Supp. Fig. 4, and Supp. Vid. 3). As a baseline denoising method, we again used Non-local-means denoising tuned on the test set.

	NRMSE			SSIM		
Condition	Input	NLM	Network	Input	NLM	Network
C0	0.095561	0.051896	0.033009	0.170476	0.463562	0.882782
C1	0.089105	0.038485	0.027509	0.22931	0.715711	0.906698
C2	0.076288	0.03259	0.023556	0.366174	0.843744	0.922166

Table 5: Restoration accuracy for *Tribolium castaneum* for the 3 different conditions C1–C3. Shown are NRMSE (lower is better) and SSIM (higher is better) for the input, the network restoration and the Non-local-means denoising baseline.

Segmentation We next exemplify how reconstructed images can benefit subsequent image analysis and downstream data interpretation tasks. To that end we investigated a simple thresholding-based workflow to segment nuclei in image stacks of Tribolium embryos. We perform segmentation based on both network-restored images and images denoised with a strong baseline and compare the accuracy of the resulting segmentation against ground-truth manually curated by domain experts.

Tribolium images contain a roughly planar sheet of spherical nuclei from the embryo's outermost cell layer. The details of sample preparation and image acquisition are described in Section 4.2. The Tribolium embryos were imaged at four different laser powers as described in Table 4. We use the highest quality images (GT) for generating the segmentation ground-truth, and attempt to segment nuclei in denoised versions of the lowest quality image (C1).

Tribolium stacks are pre-processed uniformly for both high intensity acquisitions used in the ground-truth as well as noisy, low-SNR acquisitions used in the trial segmentations. To that end, we first upscale the physically undersampled dimension of each image by a factor of 1.5 to restore isotropic voxel dimensions. We then use the percentile-based normalization $\mathrm{N}(u; 3, 99)$

as defined in Eq. (2.1) for each image u. Third, we apply *non-local means* (NLM) denoising [17] (with $\sigma = 1.0$) to the high-SNR image used for ground-truth generation, as well as to the low-SNR image used as a control in the trial segmentation.

To create the segmentation ground-truth, we first use *ilastik* [18] to train a pixel-wise *random forest* (RF) classifier to distinguish nuclei and background pixels in the high-SNR image, and perform an initial segmentation after applying Gaussian blur with $\sigma = 1$. We curate this initial segmentation using a combination of segmentation tools from *SciPy* [19], the 3D volume rendering software *spimagine*[1], and manual, pixel-wise corrections.

We compare the segmentability of nuclei in network-restored images (denoted as *Network*) with non-local means denoised images (denoted as *NLM*) using the following simple segmentation pipeline:

1. Threshold the intensities across a range of values $\in [0, 1]$
2. Label connected components of pixels above the threshold as individual nuclei
3. Compare and score the proposed segmentations against the curated ground-truth

To compare the quality of the proposed segmentations, we match nuclei in the threshold-based segmentation and ground-truth images and compute the standard segmentation score SEG [4] between all valid nuclei[2]. We then try a range of threshold values (see Supp. Fig. 5 b) to maximize SEG for both Network and NLM images. Network and NLM images had optimal SEG scores of 0.65 and 0.47 respectively. The optimal segmentations are compared visually in Supp. Fig. 5 a. Here we introduce an additional visual representation of segmentation quality by coloring nuclei with a brightness proportional to their *maximal Jaccard score*,

$$J(i) = \max_{j \in GT} \frac{|A_i \cap B_j|}{|A_i \cup B_j|} \,, \tag{2.4}$$

where i, j are the indices of nuclei in the proposed segmentation and ground-truth respectively, and A_i, B_j are their associated pixel sets. Note the overall higher quality of individual nuclei segmentations of Network as represented by the Jaccard-colored images in the rightmost panel of Supp. Fig. 5 a as well as their smoother shapes with fewer holes and less rough edges.

2.3.3 Flywing (Joint Surface Projection and Denoising)

Surface projections, *e.g.* max-projections, are often used to look at biological image data. Here we wish to jointly project and denoise acquired stacks of a developing epithelium of the fruit fly *Drosophila melanogaster* [20–22].

To train a content-aware image reconstruction and projection model we acquired pairs of high and low SNR image stacks for several conditions (see Table 6). For each condition, we acquired 180 different 3D stacks (of size $\sim 700 \times 700 \times 50$), totaling 35 GB of training data.

[1] `https://github.com/maweigert/spimagine`

[2] Visible nuclei with more than half of the volume outside of the image boundary were not included.

Condition	Exposure (ms)	Laser-power (%)
GT	240	20
C1	120	2
C2	120	3
C3	120	5

Table 6: Experimental conditions for flywing acquisitions.

As prediction target we used the surface projected 2D ground-truth signal obtained via *PreMosa* [1].

Again we extracted $\sim$ 17000 random 3D patches of size $64 \times 64 \times 50$ from input stacks and the corresponding patches of size 64×64 from the ground-truth 2D projection. For the composite task of joint projection and denoising, we designed a respective network architecture that consists of two parts: *(i)* an image restoration part, similar to the architecture used before, and *(ii)* an additional projection subnetwork that specializes on content-aware data projection (Supp. Figure 9).

To quantify the error we used the same error metrics as before and computed them on a held out test set (comprising 26 volumes). As baseline we chose again fine-tuned Non-local-means denoising on the PreMosa projections of the input.

	NRMSE			SSIM		
Condition	Input	PreMosa+NLM	Network	Input	PreMosa+NLM	Network
C0	0.141633	0.116893	0.079895	0.092568	0.256161	0.569576
C1	0.107947	0.074497	0.069053	0.356259	0.566829	0.623217
C2	0.123525	0.088274	0.073256	0.235876	0.462033	0.613001

Table 7: Restoration accuracy for surface projection of developing fly wing of *Drosophila melanogaster* for the 3 different conditions C1–C3. Shown are NRMSE (lower is better) and SSIM (higher is better) for the input, the network restoration and the Non-local-means denoising baseline.

Segmentation and Tracking To further investigate the restoration quality, we compare the cell segmentation and tracking accuracy based on the raw low-SNR input images against that of the restored images from our CARE network. The high-SNR PreMosa-projected images serve as the ground truth in this analysis.

First, the *Trainable Weka Segmentation* FIJI plugin [6] is used for binary segmentation of membrane and background regions. For each set of images we generate training data by manually labeling 30 cells (membrane contour and corresponding non-membrane region inside). We train a random forest classifier with 200 trees and 2 random features per node. The features used for training are: Gaussian blur, Hessian, Anisotropic diffusion, Lipschitz, Sobel filter,

Difference of Gaussians, Bilateral and Kuwahara. The trained classifier yields probability maps indicating the likelihood of a pixel to be a part of a cell membrane.

We continue processing these probability maps with *Tissue Analyzer* [7], a tool for tracking and segmentation of cells in 2D epithelia. Here, the probability maps are skeletonized, refined, segmented and tracked. This yields images with color-coded cell identities. Note that the segmentation of the ground-truth images is manually curated to correct all visible inconsistencies and thus provide a good baseline. Since segmentation and tracking errors are often hard to identify, the segmentation ground-truth is actually not completely error-free, although it is significantly more accurate than the uncorrected version.

Our test data set is a time-lapse where rather strong photo-bleaching occurs, hence images of later time-points exhibit much worse image quality than earlier ones. Consequently, we also expect segmentation quality to decrease over time, which we quantify with the SEG score. For each frame we compute the SEG score based on the raw and restored images with respect to the ground-truth (see Supp. Fig. 8). The score of the network-based segmentation (*gt/network*) decreases substantially more slowly than the one based on the raw input image (*gt/input*), indicating that the network restoration can meaningfully improve the signal quality for further analysis.

We provide an additional visual representation of the data to support this idea. Based on the SEG score we re-color the cell identity maps such that the matched segments have the same color across all images and the segments without a match are colored black. This helps to visually evaluate the segmentation and tracking quality by estimating the abundance of unmatched (black) segments (Supp. Fig. 8). Overall, when using our CARE-based reconstruction, we find that imaging with 6–10 less exposure time (without changing laser power) has no serious effects on the quality of segmentation and tracking. This potentially allow us to image 6–10 times faster, thus increasing temporal resolution by the same factor.

2.4 Image Restoration with Semi-synthetic Training Data

The main reasons for anisotropy in microscopy data are (i) the inherent axial elongation of the blur-inducing optical point spread function (PSF) of most microscopes, and (ii) the often reduced axial sampling due to considerations regarding image acquisition-time and photo-toxicity. As this anisotropy is intrinsic to acquisitions on most microscopy setups[3], the strategy for training data generation we used before cannot be readily applied (as isotropic ground-truth cannot be acquired). To generate *semi-synthetic* training data in this case, we exploit the fact that cellular and sub-cellular structures in most biological samples are present in diverse orientations, and that the mentioned physical process leading to image anisotropy is well understood and can be modeled in *in-silico*.

Specifically, we proceeded as follows: For a given anisotropic volume g, the PSF of the microscope h and the axial up-sampling factor σ, we first compute for each volume g a corresponding

[3]Notable exceptions are isotropic multi-view light-sheet microscopy [23] or 4π-confocal microscopy [24]

volume p that is *laterally* (i) blurred by a rotated version of the point spread function h and (ii) down-sampled by σ. This process takes high resolution lateral slices (xy) and computationally applies the same distortions usually introduced by a microscope in axial slices (xz and yz). The result are image pairs containing the same structures, but exhibiting once lateral resolution and once axial (poor) resolution. Next, we sample these pairs of *real* and *synthetically generated* image pairs and use those smaller samples to train a network to restore isotropic resolution (Main Figure 3 a). The penultimate step is to apply the resulting neural network model on all lateral slices (xz/yz) of raw image volumes. The isotropically reconstructed signal is finally computed by taking the average of the two corresponding voxel intensities retrieved from two network predictions for reconstructed xz and yz slices.

2.4.1 Drosophila (Restoration of Anisotropic Volumes)

Here we considered live imaging of developing histone-labeled *Drosophila melanogaster* embryos imaged using a light-sheet microscope [2]. Note that this data set was already processed [2], but still exhibited an anisotropic PSF (4x) and axial sub-sampling (5x) that translated into a combined 4–6 fold decrease in axial resolution (Main Figure 3 b). We followed the training data strategy explained before, using a theoretical PSF for the given microscope parameters and the given sub-sampling rate. Training was performed on patches of size 128×128 pixels, sampled from 15 equally spaced time-points during development (between embryo cellularization and germband retraction). The final number of use patches was ~ 10000. The network architecture and used training parameters are shown in Supp. Fig. 15 and Table 1. Main Figure 3 b shows that the network prediction appear isotropic and the distinctive pattern of crowded nuclei can be clearly seen. To quantify the quality of this image restoration, we computed the spectral energy of the signal in the Fourier-domain (Supp. Fig. 11) along the axial and lateral dimension and found that the network restored previously absent frequencies along the axial dimension without altering the spectral components along the lateral dimensions. Specifically, we calculated the *spectral isotropy ratio* Φ of the axial vs lateral spectral energy and found that the restored volume achieved higher spectral isotropy ratio $\Phi = 0.99$ than the raw volumes $\Phi = 0.66$ (Supp. Fig. 11).

Segmentation In addition to visual and spectral inspections, we evaluated our network-based isotropic restoration for improving the accuracy of nuclei segmentation. To that end, we compare the accuracy of the network-based segmentation against that based on images upsampled with bicubic interpolation.

Note that the *Drosophila* images were used were from a developing embryo during the 14^{th} mitotic cycle – roughly three hours after egg laying – just before the onset of gastrulation. The alignment of the embryo with the microscope's detectors defines the relationship between the embryo's body axes and the x, y, and z dimensions of the image. The undersampled spatial dimension orthogonal to the detection plane is assigned to the z image axis and corresponds to the dorsal-ventral body axis of the embryo. The (x, y, z) size of a single voxel in our initial

anisotropic image is $(0.40, 0.40, 2.0)$ μm, and after applying the network the full pixel size of the image is $(528, 1352, 536)$. We crop the image along the anterior-posterior axis, y, taking slices $[600, 800)$, and then again in the middle of the bilaterally-symmetric left-right axis, $x \in [0, 270)$, giving us a region containing approximately 470 nuclei (see Supp. Fig. 12 b).

We used ilastik [18] to create ground-truth nuclei segmentations, training a two-class pixel-wise classifier to separate nuclei from background using all features with kernel width $\sigma \geq 2$, and performing initial segmentations after applying Gaussian blur with $\sigma = 1$. We curated this initial segmentation with a variety of segmentation tools including *SciPy* [19], *spimagine*, and manual, pixel-wise corrections.

We compare the segmentability of network-restored images (*Network*) with bicubically up-sampled fluorescence intensity images (*Bicubic*) using the following simple segmentation pipeline: First, for both Network and Bicubic images we train a pixel-wise random forest (RF) classifier using ilastik with all available 3D features and all kernel widths $\sigma \geq 2$. This is done once using Network images as training data and a second time using the corresponding Bicubic images.[4] To ensure a fair comparison, we use the same set of approx. 2000 labeled pixels to train each classifier. Then, for both Network and Bicubic images we:

1. Use the appropriate pre-trained RF to generate a probability map of the image
2. Threshold the probability map across a range of values
3. Label connected components of pixels above the threshold as individual nuclei
4. Score the proposed segmentation against manually generated ground truth

To compare the quality of the proposed segmentations, we match nuclei in the threshold-based segmentation and ground-truth images and compute the *fraction of unmatched nuclei* as a measure of segmentation error (denoted by *ERROR*, see Eq. (2.5) below). The matching is created from the proposed segmentation and ground-truth as follows:

1. Build a directed, bipartite graph between nuclei in the proposed segmentation and the ground-truth with an edge from segment A to segment B if (and only if) more than 50% of the voxel volume of A falls within the volume of B. Note that this is the same matching criterion as used in the SEG segmentation score.
2. Identify A and B as a matching pair if (and only if) A points to B and B points to A.
3. Compute the fraction of unmatched nuclei as

$$\text{ERROR} = \frac{\textsc{UnmatchedSeg} + \textsc{UnmatchedGt}}{\text{GT}}\,, \tag{2.5}$$

   where UnmatchedSeg denotes the number of nuclei in the proposed segmentation with no match in the ground-truth, UnmatchedGt denotes the number of nuclei in the ground-truth with no match in the proposed segmentation, and GT denotes the number of nuclei in the ground-truth.

[4] Note that the RF used in the creation of the ground-truth is *separate* from the ones used for generating the Network and Bicubic trial segmentations. No data from either the Network or Bicubic test images was included in training the RFs used in the test segmentations.

We then search over a range of threshold values (see Supp. Fig. 13) to minimize this error measure for both Network and Bicubic images, and the optimal segmentations are compared visually in Supp. Fig. 12 a. The network-restored images produce more accurate segmentations at every threshold level, and have a better optimal *ERROR* value of 0.21% as compared to 1.7% obtained with Bicubic images. Additionally, when considering correctly matched nuclei, Network produces visually more accurate shapes, with fewer holes and smoother less ragged boundaries. Finally, note the tendency of Bicubic segmentation to make mistakes by fusing nuclei in the vertical direction along the axis of upsampling *despite* the efforts to separate these nuclei via the pixel-wise random forest classifier.

2.4.2 Retina (Restoration of Anisotropic Volumes)

Here we apply CARE to multi-channel acquisitions of the developing retina of *Danio rerio* (zebrafish) embryos (see Main Figure 3 c). This system is an important model of vertebrate organ formation, displaying diverse phenomena of cell migration and coordinated morphological tissue changes [25, 26]. The analysis of cell specific migration patterns relies on the correct tracking and segmentation of individual cells, both directly affected by the attainable volumetric resolution. The study of spatial neuron reorganization during optical cup formation, for example, is currently only possible for sparsely labeled subsets of cells [27]. Since the anisotropy is a prime cause for automated segmentation methods to fail, our method is an important step towards solving such issues and will help to analyze ubiquitously labeled tissues.

For generating training data we acquired 5 volumes of the developing optical cup using a multi-color labeling of nuclei (DRAQ5) and the nuclear envelope (GFP-tagged LAB2B). To test whether reconstruction was possible even for generously spaced z stacks (a large σ), we chose stacks with an axially under-sampling factor of $\sigma = 10.2$. We extracted lateral patches as before and applied the corresponding PSF and subsampling model, yet always keeping the information of both image channels. In total we used ~ 25000 patches of size 128×128. The network architecture is shown in Supp. Fig. 15.

After training, the network learned to jointly predict both channels (see Main Figure 3 c, Supp. Fig. 14 and Supp. Vid. 6). An interesting observation is, that areas in which one channel showed a higher level of distortion, both channels where recovered similarly well. This means that the network leveraged on learned correlations between the channels and uses this cross-channel prior during image reconstruction (see Supp. Fig. 14).

2.4.3 Liver (Restoration of Anisotropic Volumes)

Here we used volumetric images of mouse liver tissue (see Main Figure 3 d). On a tissue level, the liver consists mainly of protein-producing hepatocytes that exchange molecules with complex three dimensional tubular networks[5] which are responsible for transport of blood into, and bile out of the organ [28]. Imaging and analyzing this organ and its complex structure,

[5]These tubular networks are the sinusoidal network and the bile canaliculus.

however, poses two main challenges: (*i*) The sheer size of the tissue ($\sim 1mm^3$) requires very fast, tiled imaging, and (*ii*) building size- and shape-preserving models of structures contained in the tissue require isotropic image resolution. This is particularly true for tubular structures bearing small radii, *e.g.* the bile canaliculi.

As before, we acquired 8 stacks of mouse liver using a multi-color labeling of nuclei (DAPI) and the sinusoidal network (FLK-1). Again we chose a rather large axially under-sampling factor of $\sigma = 8$. Stack-sizes were $\sim 750 \times 750 \times 50$ and we extracted ~ 15000 patches of size 128×128 from the given body of data.

We found that our method, due to it operating on 2D slices, works best for 3D or 2D structures (nuclei, membranes, sheets) and less well for 1D structures, which poses a limitation to certain biological structures. Further, it might be less effective for genuinely anisotropic biological tissues, such as strongly stratified layers of cells with a fixed, global elongation of membranes along one dimension.

2.5 Image Restoration with Synthetic Training Data

We finally consider the case, when neither real biological ground-truth nor the real image degradation process is experimentally accessible, but a synthetic generative model of both the structures and the image degradation process is available. This is true, for example, when using conventional widefield microscopy to image biological structures that are smaller than the diffraction limit but whose physical properties are well known, such as microtubules. Its main idea is to *synthetically* create ground-truth images that are realistic representations of the structures to be examined, and create *synthetically* degraded microscopy images via simulation of the imaging process. Since the creation of realistic ground-truth structures is a complex task in its own right [29–31], this training regime works best in cases when a simulation pipeline is already available or when the structures to be synthesized are not too complex.

2.5.1 INS-1 cells (Structural enhancement beyond the diffraction limit)

To test this approach we first considered the dynamics of insulin secretory granules (SG) and filamentous microtubule-networks (MTs) in insulin-secreting beta-cells (INS-1). The role of MTs in insulin SG exocytosis has been heavily debated recently [32] and high resolution live-cell imaging is crucial for understanding the function of the MT network therein. Due to its resolution limit, widefield imaging only allows for restricted insights into this process. Live-cell super-resolution imaging (*e.g.* SIM), on the other hand, requires high laser power resulting in quick photo-bleaching of the fluorescent signal.

Synthetic tubulin: For creating artificial images of a single microtubule structure, we first simulate two-dimensional trajectories $\{x_n\}_{n\in\mathbb{N}}$ on a pixel grid, with randomly changing orientation and given maximal curvature κ_{max}, mimicking the physical stiffness properties of microtubules (e.g. having a non vanishing persistence length). To render the computation con-

ceptually easier, we will describe a point in the Euclidian plane as given by a complex number $x_n \in \mathbb{C}$ and regard its real and imaginary parts as the corresponding coordinates in $\mathbb{R}^2$. For that we first generate a random walk on the curvatures $\kappa_n \in \mathbb{C}$:

$$\kappa_n = \exp\left(2\pi i \cdot \text{clip}_{\kappa_{max}}[\kappa_0 + \mathcal{W}_n(d\kappa)]\right), \quad \mathcal{W}_n(d\kappa) \sim \sum_i^n \mathcal{N}(0, d\kappa)$$

which we successively sum up to yield velocities and finally the trajectory itself:

$$v_n = v_0 \exp\left(2\pi i \cdot \sum_{i=0}^{n} \kappa_i\right) \qquad\qquad x_n = x_0 + \sum_{i=0}^{n} v_i$$

Many of these single tubules trajectories are then rasterized on a pixel-grid of given pixel-size to yield the final, ground-truth image (Supp. Fig. 17). To synthetically generate the corresponding widefield image, we simulate auto-fluorescence by adding low-frequency perlin noise, blur the result with the PSF of the microscope (we use the theoretical PSF for the respective NA) and add Poisson and Gaussian noise, mimicking camera noise (Supp. Fig. 17).

Synthetic granula: To simulate ground-truth of granular objects, we use clipped perlin noise and whose intensity we vary randomly. The corresponding widefield images are simulated like for tubulin (Supp. Fig. 17).

In total, we created $\sim$ 8000 synthetic patch-pairs of size 128×128. We finally trained a network to invert this degradation process and applied it on the original widefield images (Supp. Fig. 15). As can be seen in Main Figure 4 b and Supp. Fig. 16, this approach allows to restore the tubular signal that are in line with the expected structures, as inferred from the deconvolved result.

2.5.2 HeLa cells (Structural enhancement beyond the diffraction limit)

Here we aim to validate the accuracy of our approach against super-resolution imagery of microtubules. For that, we used a dataset of widefield images of GFP-tagged microtubules, with the corresponding super-resolution images reconstructed via SRRF (super-resolution radial fluctuations [3]). We created training data as described (while adapting the experimental parameters, such as pixel size and PSF), in total $\sim$ 5000 synthetic patch-pairs of size 128×128. We then compared our reconstruction with the output of SRRF. As can be seen in Main Figure 4 c, both super-resolution results and the network reconstruction were able to recover relevant microtubule structures that are indistinguishable in the raw images. It is worth noting, however, that the input of SRRF consisted of 200 consecutive frames of widefield images (not shown), whereas our result is based on a single image obtained from averaging $5 - 10$ widefield frames.

3 Reliability of Image Restoration

It is crucial in science to be able to trust made observations and acquired experimental data. If the reliability of data cannot be established, subsequently drawn conclusions are obviously prone to error. However, the reality of modern microscopy is that image quality often has to be intentionally compromised to be able to capture biological phenomena (as illustrated by the design-space tetrahedron mentioned in the main paper).

Hence, restoring images from incomplete and corrupted observations is important. However, such restored images will not be perfect, and it would be very desirable to provide scientists with a measure of uncertainty. While this has also been the case in the past, very few methods do actually report uncertainty of their results. One reason for this may be that previous methods often made rather weak assumptions about the restored data and thus – while limiting their effectiveness – exhibited failure cases that could more easily be noticed and corrected by scientists.

We believe this is becoming increasingly infeasible with the advent of more powerful data-adaptive methods, such as those used here based on neural networks[6]. These methods learn very effectively how to adapt to the data, thereby making stronger assumptions about the restored images. While this is the reason for their improved restoration quality, it also poses the risk to produce mistakes that look very realistic and cannot be easily detected by the scientist. Although this issue is not restricted to biological images, it is often more important here than in the fields of computer vision and machine learning, where most of these powerful data-adaptive methods are being developed. It can be argued that the bulk of computer vision research is currently driven by standardized benchmarks, where many researchers compete to improve results for an agreed-upon problem on the exact same set of test images. The evaluation protocols for these benchmarks typically do not take prediction uncertainty into account, thereby providing no direct incentives for researchers to develop methods that quantify their own limitations. Nevertheless, there has been some work on assessing the reliability and uncertainty of neural networks, which we adopt and extend in this work.

We are mainly interested in two issues, which we will subsequently discuss in more detail in Sections 3.1 and 3.2, respectively:

1. First, we want to have a measure of uncertainty for every pixel in the image. To that end, we employ models that predict a probability density function for every pixel, instead of just a scalar value, as is common. We empirically validate that our uncertainties are well-calibrated. Apart from being important to the scientist to draw conclusions, the per-pixel distributions of the restored image may be useful in subsequent tasks, such as segmentation or tracking.
2. Second, we want to identify (parts of) images where the results of our trained models may not be trustworthy. This can occur on challenging (*e.g.*, low signal-to-noise ratio)

[6]We use the terms *model* and *(convolutional neural) network* interchangeably in this section.

input images or when a model is applied to an image that is very different compared to the data it was trained on.

We end with a self-contained discussion and validation of the two aforementioned issues in the context of our learned models in Section 3.3. A reader not interested in (technical) details may directly skip to this section.

3.1 Modeling Uncertainty

There can be many sources of uncertainty in practical applications. However, for the purpose of modeling, we think it is useful to distinguish between *aleatoric* and *epistemic* uncertainty [*cf.* 33].

Aleatoric uncertainty refers to intrinsic uncertainties that are inevitable. For example, even if the sample under the microscope does not change, the microscope's camera will never capture the exact same image twice due to the inherently random process of collecting photons. The uncertainty of this noisy observation process cannot be reduced for a given acquisition configuration (*e.g.*, laser power and exposure time). Hence, instead of a single value, we can predict a distribution that captures this intrinsic uncertainty.

Epistemic uncertainty, on the other hand, refers to sources of uncertainty that could be reduced if additional information or observations were available. Concretely, we want to capture the uncertainty of choosing an image restoration model (specified through its parameters). Intuitively, there is high model uncertainty if only little training data is available. This uncertainty can be decreased by collecting additional data.

3.1.1 Aleatoric uncertainty

Let use first consider a typical scenario for image regression with a convolutional neural network (CNN), represented as a function g_θ with model parameters θ. The predicted output image $\hat{y} = g_\theta(x)$ is obtained from input x. Based on training data comprised of input-output images $\{(x^t, y^t)\}_{t=1}^T$, the model parameters are chosen by minimizing a loss functions $L(\theta)$ based on a per-pixel distance function. A common choice is the *mean squared error (MSE)*

$$L_{\text{mse}}(\theta) = \frac{1}{T}\frac{1}{N}\sum_{t=1}^{T}\sum_{i=1}^{N}\big(y_i^t - g_\theta(x^t)_i\big)^2, \tag{3.1}$$

where N denotes the total number of pixels of an image.

It is well-known that training a model to minimize a squared error can be interpreted as maximizing the likelihood of a Gaussian distribution [*e.g.*, 34, 35]. Here, minimizing Eq. (3.1)

is equivalent to maximizing the likelihood

$$\mathcal{L}_{\text{gauss}}^{\text{homoscedastic}}(\theta) = \prod_{t=1}^{T} \prod_{i=1}^{N} p_{\text{gauss}}\big(y_i^t; g_\theta(x^t)_i, \sigma\big) \quad \text{with} \tag{3.2}$$

$$p_{\text{gauss}}(z; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right) \tag{3.3}$$

of a fully-factorized probabilistic graphical model [*cf.* 36] where $g_\theta(x)$ specifies the per-pixel *location* parameters (*i.e.*, means) of univariate Gaussian distributions p_{gauss} with shared *scale* parameter (*i.e.*, standard deviation) σ. Learning θ by minimizing Eq. (3.1) is indeed equivalent to maximizing Eq. (3.2):

$$\arg\max_\theta \mathcal{L}_{\text{gauss}}^{\text{homoscedastic}}(\theta) = \arg\min_\theta -\log \mathcal{L}_{\text{gauss}}^{\text{homoscedastic}}(\theta) \tag{3.4}$$

$$= \arg\min_\theta \sum_{t=1}^{T} \sum_{i=1}^{N} \frac{\big(y_i^t - g_\theta(x^t)_i\big)^2}{2\sigma^2} + \log\sqrt{2\pi\sigma^2} \tag{3.5}$$

$$= \arg\min_\theta L_{\text{mse}}(\theta) \tag{3.6}$$

Since σ is constant and not learned, it does not play a role and can be dropped from the objective function. We could improve on this by also learning σ, but it is crucial to understand that a shared scale parameter σ assumes the same amount of noise or uncertainty for every image pixel (referred to as *homoscedastic* in statistics). However, a more realistic and flexible approach is to use a different σ for every pixel (called *heteroscedastic*), allowing the model to express a tailored uncertainty for every pixel.

Nix and Weigend [34] already proposed over two decades ago to learn a neural network that predicts pixel-adaptive scale parameters $\sigma_\theta(x)$, which has recently been adopted in the context of deep learning [33, 37, 38]. We follow this here and learn CNNs that predict both location $\mu_\theta(x)_i$ and scale $\sigma_\theta(x)_i$ parameters[7] for every pixel with $i = 1, \ldots, N$. Instead of p_{gauss}, we choose a Laplace distribution

$$p_{\text{laplace}}(z; \mu, \sigma) = \frac{1}{2\sigma} \exp\left(-\frac{|z-\mu|}{\sigma}\right), \tag{3.7}$$

which we empirically found to work much better for our image restoration applications[8]. Hence, the output of our CNNs

$$g_\theta(x)_i = p_{\text{laplace}}\big(\mu_\theta(x)_i, \sigma_\theta(x)_i\big) \tag{3.8}$$

[7] We ensure $\sigma_\theta(x)_i > 0$ by using a *softplus* activation function in the respective layer of the CNN.

[8] In principle, we could use any parametric (normalized) probability density function here, given the technical condition that it should be easy to compute and differentiate. For example, Bishop [35] proposed to use a mixture of densities to model multi-modal outputs.

is now a Laplace distribution for every pixel i. We learn the model parameters θ by minimizing the loss function

$$L_{\text{laplace}}(\theta) = \frac{1}{T}\frac{1}{N}\sum_{t=1}^{T}\sum_{i=1}^{N}\frac{|y_i^t - \mu_\theta(x^t)_i|}{\sigma_\theta(x^t)_i} + \log\sigma_\theta(x^t)_i, \tag{3.9}$$

which is equivalent to maximizing the likelihood function

$$\mathcal{L}_{\text{laplace}}^{\text{heteroscedastic}}(\theta) = \prod_{t=1}^{T}\prod_{i=1}^{N} p_{\text{laplace}}\big(y_i^t; \mu_\theta(x^t)_i, \sigma_\theta(x^t)_i\big). \tag{3.10}$$

Equation (3.9) is related to the commonly used *mean absolute error (MAE)* loss, but with the crucial difference that we learn per-pixel scale parameters $\sigma_\theta(x)$, which have already been interpreted by Kendall and Gal [33] as weights that can attenuate the cost of making a mistake in the location parameters $\mu_\theta(x)$. For each pixel individually, the model can essentially interpolate between two extremes to minimize Eq. (3.9): *(1)* ideally, accurate output prediction $\mu_\theta(x)_i \to y_i$ with low uncertainty, *i.e.* small $\sigma_\theta(x)_i \to 0$, and *(2)* high uncertainty, *i.e.* large $\sigma_\theta(x)_i \gg 0$ if an accurate output prediction is not possible, *i.e.* $|y_i - \mu_\theta(x)_i| \gg 0$. If $\sigma_\theta(x)_i$ is large, then the cost of predicting an incorrect value $\mu_\theta(x)_i$ is diminished, which comes at a greater expense for a large value of $\sigma_\theta(x)_i$. On the other hand, the cost for $\sigma_\theta(x)_i$ can be reduced by making it small, which leads to an amplification of a mistake in $\mu_\theta(x)_i$.

Predicting scale $\sigma_\theta(x)$ in addition to location $\mu_\theta(x)$ parameters and training the model with the loss function from Eq. (3.9) are minor changes for typical CNN regression models and can easily be implemented. Please note that we do not need ground truth labels for the scale values $\sigma_\theta(x)$, which allows to train on existing datasets.

After a model has been trained, we can apply it to new images to obtain per-pixel Laplace distributions $g_\theta(x)_i$ according to Eq. (3.8). However, most applications expect a restored image, *i.e.* a single value $\hat{y}_i$ per pixel instead of an entire distribution. To that end, we use Bayesian decision theory [*cf.* 39] to compute *point estimates* from each distribution $g_\theta(x)_i$. Since many common image quality metrics (*e.g.*, RMSE[9] and PSNR[10]) are based on squared errors, we compute each pixel of the restored image via a *minimum mean squared error (MMSE)* estimator, *i.e.* as the expected value of the corresponding Laplace distribution, which is simply its location parameter:

$$\hat{y}_i = \mathbb{E}\big[g_\theta(x)_i\big] = \mu_\theta(x)_i. \tag{3.11}$$

By predicting each pixel of the restored image according to Eq. (3.11), we find that we can achieve similar results in terms of MSE as compared to a model that was specifically trained with an MSE loss (Eq. (3.1)). In other words, we do not suffer a substantial decrease in MSE performance by predicting a distribution instead of directly a scalar value.

[9] Root-mean-square deviation: $\sqrt{\text{MSE}}$

[10] Peak signal-to-noise ratio: $20 \cdot \log_{10}(\text{MAX}_y) - 10 \cdot \log_{10}(\text{MSE})$

3.1.2 Epistemic uncertainty of model parameters

As discussed above, we model aleatoric uncertainty by predicting a probability distribution for every pixel of the restored image, instead of just a scalar value. However, we select our CNN model by choosing a single parameter vector θ that (approximately) minimizes Eq. (3.9), hence we do not model any uncertainty of the model parameters θ. This is less of a problem when training data is abundant (and we assume that Eq. (3.9) can be minimized accurately), resulting in low epistemic uncertainty that may be disregarded. On the other hand, modeling the uncertainty of model parameters can be crucial when little data is available.

To that end, a Bayesian approach uses a distribution over model parameters, which has also been done in the context of neural networks [40, 41]. Unfortunately, making predictions with a Bayesian approach is often difficult and slow, since predictive inference needs to compute expectations over the distribution of parameters. In practice, these expectations are typically approximated by drawing samples from the distribution or with variational inference [*cf.* 42], which uses distributional approximations. Although recent work has proposed more efficient approximate inference methods in the context of deep learning [43, 44] based on extensions of Dropout [45], we do not use them here. Despite being much more practical than earlier Bayesian neural networks, they require changes to the model architecture and the selection of additional hyper-parameters.

Instead, we follow the non-Bayesian approach of Lakshminarayanan *et al.* [37] and simply use an *ensemble* of models to capture some of the epistemic uncertainty with respect to the parameters of the model. To that end, we independently train[11] M identical models on the same data that only differ[12] through their learned parameters θ_m. The per-pixel output of our CNN model is now a Laplace *mixture density* (see Fig. 1 for an example)

$$g_\Theta(x)_i = \frac{1}{M}\sum_{m=1}^{M} p_{\text{laplace}}\big(\mu_{\theta_m}(x)_i, \sigma_{\theta_m}(x)_i\big) \tag{3.12}$$

with $\Theta = \{\theta_m\}_{m=1}^M$, which captures both aleatoric and epistemic uncertainty. In contrast to [37], we do not approximate the ensemble prediction of Eq. (3.12) with a simpler density function for ease of use.

We still use an MMSE estimator to predict each pixel of the restored image, but replace Eq. (3.11) with the expected value of the ensemble mixture, which is also easy to compute as the average of the location parameters of the ensemble members:

$$\hat{y}_i = \mathbb{E}\big[g_\Theta(x)_i\big] = \frac{1}{M}\sum_{m=1}^{M} \mu_{\theta_m}(x)_i. \tag{3.13}$$

In our experiments with available ground-truth for comparison, we find that the image restoration quality obtained from an ensemble is on average typically only slightly better than

[11] In contrast to [37], we do not use adversarial training [46].

[12] Due to random parameter initialization and stochastic gradient-based optimization.

that of the best member of the ensemble. Hence, it may not be crucial to model epistemic uncertainty, which may be due to a sufficient amount of training data available to us. As a result, just using a single model with $M = 1$ is typically good enough to achieve high-quality restoration results. However, we find that using an ensemble is very useful to identify problematic image regions. We will discuss this in Section 3.2.

3.1.3 Validation and visualization

We just discussed above how we address the issue of uncertainty, *i.e.* we train an ensemble of independent models $g_\Theta(x)$ that together yield for each pixel a probability distribution in form of a mixture model (Eq. (3.12)). Furthermore, we obtain the restored image $\hat{y}$ via Eq. (3.13) by computing the per-pixel expected values of the mixture distributions.

A quantitative evaluation of an image restoration method is typically done by assessing the quality of the restored image, for example by simply computing the MSE between the ground truth value y and the predicted restored image $\hat{y}$. While we also do that (*e.g.*, Supp. Fig. 2), we want to go beyond that and also validate that our predictive uncertainties are well-*calibrated*, *i.e.* that the distributions $g_\Theta(x)_i$ are accurate.

One common measure that actually takes the whole distribution into account is the likelihood function that we already introduced in Eq. (3.10) in the context of model training. Learning the model parameters by maximizing the likelihood *should* yield a well-calibrated model. To check this after training, we could then evaluate the likelihood[13] of our model ensemble

$$\mathcal{L}(\Theta) = \prod_{s=1}^{V} \prod_{i=1}^{N} \frac{1}{M} \sum_{m=1}^{M} p_{\text{laplace}}\big(y_i^s; \mu_{\theta_m}(x^s)_i, \sigma_{\theta_m}(x^s)_i\big) \tag{3.14}$$

based on a validation set of image pairs $\{(x^s, y^s)\}_{s=1}^V$. While likelihoods are very useful to do *relative* comparisons between several models, they do not provide an *absolute* calibration measure that is easy to interpret. Also note that instead of the likelihood, the same discussion applies to any other *proper scoring rule* [*cf.* 47].

To obtain an absolute measure of calibration, we take an alternative approach by adopting and extending methods from the context of classification [*cf.* 48–51]. The main idea is that the calibration of a classifier can be evaluated by comparing the probability assigned to the predicted class (for a given input) with the frequency of success based on knowledge of the true class for each input.

More concretely, let $\{(x^s, y^s)\}_{s=1}^V$ denote a set of (validation) data in the context of classification, where each label $y^s \in \mathcal{C}$ belongs to a finite set of discrete classes $\mathcal{C} = \{c_1, c_2, \ldots\}$. Consider a classifier $\phi(z; x)$ that assigns a (normalized) probability mass $\phi(z = c; x)$ to each class $c \in \mathcal{C}$ for an input x. As is typical, the class is predicted as $\hat{y} = \arg\max_c \phi(z = c; x)$ with the associated probability $\hat{r}^s = \phi(z = \hat{y}; x)$.

[13] Log-likelihood computed in practice, *i.e.* the logarithm of Eq. (3.14).

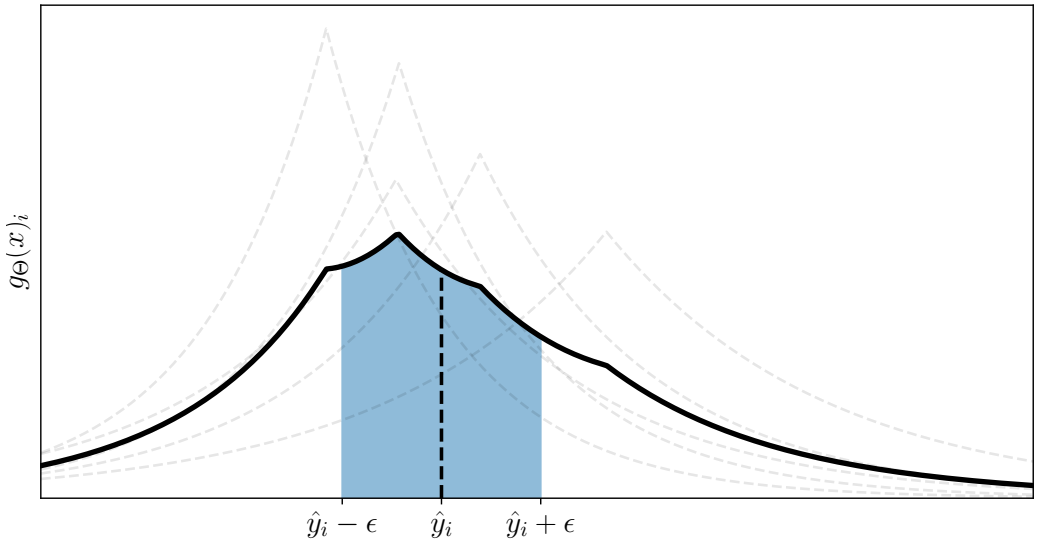


Figure 1: Example of a mixture of Laplace probability distributions $g_\Theta(x)_i$ *(black, solid)*, obtained as the average of $M = 5$ individual probability densities $g_{\theta_m}(x)_i$ *(gray, dashed)*. Further shown are the expected value $\hat{y}_i = \mathbb{E}\big[g_\Theta(x)_i\big]$ of the mixture *(black, dashed)* and the area under the mixture distribution for the interval $[\hat{y}_i - \epsilon, \hat{y}_i + \epsilon]$ *(blue, shaded)*.

We can now denote the *accuracy* $\frac{1}{V}\sum_{s=1}^{V} \mathbf{1}[y^s = \hat{y}^s]$ as the fraction of cases where the classifier successfully predicted the correct class[14], and refer to *confidence* $\frac{1}{V}\sum_{s=1}^{V} \hat{r}^s$ as the average of the probabilities associated with each of the predicted classes. Hence, accuracy and confidence should be similar if the classifier is well-calibrated.

Furthermore, we can compare confidence and accuracy more granularly for subsets of data with different average confidences. To that end, all samples x^s are first put into different groups based on an ordered partition of their associated prediction probabilities $\hat{r}^s$, for example with $K = 10$ groups and thresholds $\tau = [0.0, 0.1, \ldots, 0.9, 1.0]$ with group k being all samples with $\hat{r}^s \in (\tau_k, \tau_{k+1}]$. Accuracy and confidence are then computed separately for the samples of each group and can be plotted against each other (*cf.* Supp. Figs. 18 and 19), which is called a *reliability diagram* [48, 49]; the classifier is well-calibrated if the resulting curve is close to the identity function. Furthermore, we can define an *expected calibration error (ECE)* [50], as the (weighted) average of the absolute differences between accuracy and confidence for each group.

Unfortunately, the discussion from above cannot directly be applied to our models that predict continuous probability density functions. We now discuss how we adapt the concepts of accuracy and confidence to make them work for continuous probability distributions in the context of pixel-wise regression.

To make the following exposition more readable, let us assume without loss of generality that we only have one large N-pixel validation input image x and its corresponding ground truth y, instead of many smaller validation samples as used above. Given a model ensemble parameterized by Θ, let $q_i = g_\Theta(x)_i$ denote the predicted mixture density (Eq. (3.12)) and $\hat{y}_i$ the restored image (Eq. (3.13)) for pixel i.

[14] $\mathbf{1}[P]$ denotes the Iversion bracket that is 1 if P is true and 0 if P is false.

Accuracy as defined above cannot be used, since a pixel $\hat{y}_i$ of the restored image will never *exactly* match the ground truth y_i, *i.e.* $\hat{y}_i \neq y_i$ for all $i = 1, \dots, N$. But we can modify the definition of success to those cases where $y_i \in A_i^\epsilon$ falls into an *interval* $A_i^\epsilon = [\hat{y}_i - \epsilon, \hat{y}_i + \epsilon]$ around $\hat{y}_i$ with $\epsilon > 0$. Based on this, we can define the associated probability *mass*[15] for such an event (shown as shaded region in Fig. 1) as

$$\hat{r}_i^\epsilon = \mathbb{P}\big(Y_i \in A_i^\epsilon\big) = \int_{\hat{y}_i - \epsilon}^{\hat{y}_i + \epsilon} q_i(z)\, dz, \tag{3.15}$$

where $Y_i \sim q_i$ denotes the random variable associated with pixel i of the restored image. With these adaptations, we can now formally define accuracy and confidence for our regression model as

$$\text{accuracy}(S, \epsilon) = \frac{1}{|S|} \sum_{i \in S} \mathbf{1}\big[y_i \in A_i^\epsilon\big] = \frac{1}{|S|} \sum_{i \in S} \mathbf{1}\big[y_i \in [\hat{y}_i - \epsilon, \hat{y}_i + \epsilon]\big] \tag{3.16}$$

$$\text{confidence}(S, \epsilon) = \frac{1}{|S|} \sum_{i \in S} \hat{r}_i^\epsilon \qquad = \frac{1}{|S|} \sum_{i \in S} \int_{\hat{y}_i - \epsilon}^{\hat{y}_i + \epsilon} q_i(z)\, dz, \tag{3.17}$$

where $S \subseteq G$ denotes a subset of all pixel indices $G = \{1, \dots, N\}$ and $\epsilon > 0$ determines the size of the interval around each $\hat{y}_i$. To obtain a reliability diagram with K groups, we first need to specify bin edges (thresholds) τ_k with $\tau_k < \tau_{k+1}$, and extremal points $\tau_1 = 0, \tau_{K+1} = 1$. For $k = 1, \dots, K$ we then plot $\text{confidence}(S_k^\epsilon, \epsilon)$ against $\text{accuracy}(S_k^\epsilon, \epsilon)$ with $S_k^\epsilon = \big\{i \in G \colon \hat{r}_i^\epsilon \in (\tau_k, \tau_{k+1}]\big\}$ denoting the pixel indices for group k. We additionally find it informative to plot $\text{confidence}(S_k^\epsilon, \epsilon)$ versus the fraction of pixels $|S_k^\epsilon|/N$ that belong to group k, which is essentially a (normalized) histogram of $\hat{r}^\epsilon$. Finally, the expected calibration error (ECE) is obtained as the weighted average of the absolute differences of accuracy and confidence

$$\text{ECE}(\epsilon) = \sum_{k=1}^{K} \frac{|S_k^\epsilon|}{N} |\text{accuracy}(S_k^\epsilon, \epsilon) - \text{confidence}(S_k^\epsilon, \epsilon)|, \tag{3.18}$$

where the weights are simply the fractions of pixels that belong to each group.

Note that our adaptations cause accuracy and confidence, and thus also expected calibration error (Eq. (3.18)) and reliability diagrams, to depend on the size of interval A_i^ϵ as defined by ϵ. We will evaluate ECE and reliability diagrams for several values of ϵ, since our models should be calibrated regardless of the size of A_i^ϵ.

Visualization. A calibration diagram, as described above, is one way to visually inspect if the predictive uncertainty of our model is on average well-calibrated. However, we now want to discuss how to display per-pixel results from our models. Of course, one can simply look at the restored image $\hat{y}$, as is possible with any other restoration approach. Instead, we want to

[15] Eq. (3.15) can be computed in closed-form based on the *cumulative distribution functions* of the individual ensemble members. Alternatively, numerical approximations are feasible since all distributions are univariate.

focus on specific ways to visualize the predictive uncertainty, since directly showing an entire probability distribution per pixel is infeasible.

One approach is to display $\hat{r}^\epsilon$ according to Eq. (3.15), where each pixel $\hat{r}_i^\epsilon = \text{confidence}(\{i\}, \epsilon)$ denotes the model's confidence that the correct value will be inside the interval A_i^ϵ. Of course, one may choose any other interval depending on the application. In the main paper, we have for a lineplot also shown per-pixel (confidence) intervals where $\hat{r}_i^\epsilon$ is fixed (*e.g.*, at 90%) and the interval is depicted that contains this probability mass.

A standard measure of uncertainty is *entropy*, which we can also compute in our case (*cf.* Section 3.2.1) for $g_\Theta(x)_i$ of Eq. (3.12). However, we believe that looking at the per-pixel entropy as an image does not convey an intuitive understanding of the predictive uncertainty (*cf.* Supp. Fig. 20). Instead, we draw *samples* from our model to obtain possible instances of restored images, which is typically easy due to drawing each pixel independently from a mixture of "nice" distributions (such as Laplace distributions in our case). Importantly, we propose to arrange several samples of restored images as a movie to get an intuitive sense of the per-pixel variability (and thus uncertainty) of our model prediction (see Supp. Vid. 9).

3.2 Identifying Problematic Image Regions

When sufficient training data is available, there may be little reason to model epistemic uncertainty to achieve good results *on average*. However, epistemic uncertainty can help to identify problematic image regions that were under-represented or missing in the training data, where the prediction of the trained CNNs may not be trustworthy as a consequence.

As discussed in Section 3.1.2, we capture epistemic uncertainty by using an ensemble of identical and independently trained CNNs. This is feasible because training neural networks is not globally optimal, *i.e.* learned model parameters typically vary due to random initialization and stochastic gradient-based optimization methods. As a result, the predictions of the individual ensemble members are typically *slightly* different in most regions, but can be *very* different in some regions. We want to identify these latter problematic regions, which should be treated with caution. We typically found these to be especially challenging image regions (*e.g.*, with low signal-to-noise ratio), or when we applied trained CNNs to so-called *out-of-distribution* images, *i.e.* images that are not sampled from the same distribution as the training data.

We identify problematic image regions by measuring the *disagreement* of the network ensemble, which we explain below. Note that strong disagreement does indicate cause for concern, but the reverse is not necessarily true since all networks could simply make the same or similar mistakes.

3.2.1 Ensemble disagreement

Please note that the discussion in the remainder of this section applies to mixture ensembles of arbitrary distributions, not just Laplace mixture densities (Eq. (3.12)) as we use in this work. We first need to select a measure to compare probability distributions. The most

common choice is the *Kullback-Leibler-divergence* (KL-divergence), which we also use here. To quantify the per-pixel disagreement between an ensemble of models, we compute the average KL-divergence between the individual model distributions and the distribution of the mixture model. Note that [37] already used ensemble disagreement like this, but only for qualitative comparisons in the context of image classification.

The KL-divergence between two distributions $p(z)$ and $q(z)$ is defined as

$$D_{\mathrm{KL}}(p\|q) = \int p(z) \log \frac{p(z)}{q(z)}\, dz = \underbrace{\Big[-\int p(z) \log q(z)\, dz\Big]}_{H(p,q)} - \underbrace{\Big[-\int p(z) \log p(z)\, dz\Big]}_{H(p)} \tag{3.19}$$

with $H(p, q)$ being the cross-entropy between p and q, and $H(p)$ the entropy of p. With that, we can now define a disagreement score as described above as

$$\mathcal{D}(x)_i = \frac{1}{M} \sum_{m=1}^{M} D_{\mathrm{KL}}\big(g_{\theta_m}(x)_i \| g_\Theta(x)_i\big) \tag{3.20}$$

with $g_\theta(x)_i$ and $g_\Theta(x)_i$ as defined in Eqs. (3.8) and (3.12), respectively. For better understanding, we can reformulate Eq. (3.20) as

$$\mathcal{D}(x)_i = \frac{1}{M} \sum_{m=1}^{M} H\big(g_{\theta_m}(x)_i, g_\Theta(x)_i\big) - H\big(g_{\theta_m}(x)_i\big) \tag{3.21}$$

$$= H\Bigg(\underbrace{\tfrac{1}{M} \textstyle\sum_{m=1}^{M} g_{\theta_m}(x)_i}_{g_\Theta(x)_i}, g_\Theta(x)_i\Bigg) - \frac{1}{M} \sum_{m=1}^{M} H\big(g_{\theta_m}(x)_i\big) \tag{3.22}$$

$$= H\big(g_\Theta(x)_i\big) - \frac{1}{M} \sum_{m=1}^{M} H\big(g_{\theta_m}(x)_i\big) \tag{3.23}$$

to see that it corresponds to the difference between the ensemble entropy and the average entropy of the individual ensemble members.

In the context of image classification, [37] has shown that ensemble entropy is higher on out-of-distribution (OD) samples as compared to in-distribution (ID) samples (*i.e.* from the same distribution as the training data). However, this is not transferable to our scenario where the entropy is very dependent on the actual image content. Concretely, our models are typically much more confident in their predictions on "background" regions[16], *i.e.* exhibit lower entropy, as compared to "foreground" pixels[17]. This means the ensemble entropy on a background pixel of an OD image is typically lower than the entropy of a foreground pixel of an ID image (see Supp. Fig. 20). Instead of just using ensemble entropy, the advantage of the disagreement score as defined in Eq. (3.20) is that it has a normalizing effect due to subtracting the average entropy of the individual ensemble members (Eq. (3.23)). It is thus much less

[16] *I.e.*, predicted pixels $\hat{y}_i \approx 0$, where the minimal intensity value is 0.
[17] *I.e.*, predicted pixels $\hat{y}_i \gg 0$, where the minimal intensity value is 0.

sensitive to the intrinsic predictive uncertainty. Note that this property is likely also beneficial for classification problems, especially with unbalanced class distributions, where the predictive entropies of different classes may vary strongly.

Bounds. We just discussed the normalizing effect of the disagreement score as compared to just using the entropy of the ensemble. In fact, we can show that Eq. (3.20) is bounded from above. Note that it is naturally bounded from below by zero, since it is an average of non-negative KL-divergences. To make the following derivation more readable, we denote the predictive distribution for pixel i of a single model (Eq. (3.8)) as $q_m^i = g_{\theta_m}(x)_i$ and the distribution of an ensemble (Eq. (3.12)) as $q^i = g_\Theta(x)_i$. From Eqs. (3.8), (3.12) and (3.19) follows that

$$D_{\mathrm{KL}}(q_m^i \| q^i) = \int q_m^i(z) \log \frac{q_m^i(z)}{\frac{1}{M}\sum_{m'=1}^M q_{m'}^i(z)}\, dz \tag{3.24}$$

$$= \log M + \int q_m^i(z) \underbrace{\Big[\log q_m^i(z) - \log \sum\nolimits_{m'=1}^M q_{m'}^i(z)\Big]}_{\leq 0}\, dz \tag{3.25}$$

$$\leq \log M, \tag{3.26}$$

since all q_m^i are non-negative density functions and the logarithm is monotonic, *i.e.*

$$\log q_m^i(z) \quad \leq \quad \log\Big[q_m^i(z) + \sum\nolimits_{\substack{m'=1 \\ m' \neq m}}^M q_{m'}^i(z)\Big] \quad = \quad \log \sum\nolimits_{m'=1}^M q_{m'}^i(z). \tag{3.27}$$

Therefore, the disagreement score from Eq. (3.20) is bounded from above and below:

$$0 \quad \leq \quad \mathcal{D}(x)_i = \frac{1}{M}\sum\nolimits_{m=1}^M D_{\mathrm{KL}}\big(q_m^i \| q^i\big) \quad \leq \quad \log M. \tag{3.28}$$

Note that $\mathcal{D}(x)_i = 0$ if all distributions q_m^i are identical, and $\mathcal{D}(x)_i = \log M$ if there is no intersection between the support of all q_m^i, *i.e.* $\forall m, m'$ with $m \neq m'$: $\mathrm{supp}(q_m^i) \cap \mathrm{supp}(q_{m'}^i) = \emptyset$.

Based on Eq. (3.20) and its bound from above, we define a *normalized disagreement* score

$$\widehat{\mathcal{D}}(x)_i = \frac{1}{\log M}\mathcal{D}(x)_i = \frac{1}{M \log M}\sum_{m=1}^M D_{\mathrm{KL}}\big(g_{\theta_m}(x)_i \| g_\Theta(x)_i\big), \tag{3.29}$$

which is bounded from above by one, independent of the size of the ensemble, *i.e.* $0 \leq \widehat{\mathcal{D}}(x)_i \leq 1$ for all $M \geq 2$. We show normalized disagreement scores in Supp. Fig. 20. Note that Eq. (3.29) generally applies to ensembles of arbitrary sizes and member distributions (discrete and continuous).

Computational considerations. Obtaining the disagreement score of Eq. (3.20) requires to compute KL-divergences between each ensemble member q_m^i and the ensemble mixture distri-

bution $q^i = \frac{1}{M} \sum_m q_m^i$. This can be challenging in many cases[18], since closed-form expressions are generally not available for $D_{\mathrm{KL}}(q_m^i \| q^i)$. As an alternative, we use Eq. (3.23) to evaluate disagreement, which needs the entropies of the ensemble mixture $H(q^i)$ and each ensemble member distribution $H(q_m^i)$. Unfortunately, computing entropies of mixture models is also not easy. Instead of computing bounds on the mixture entropy [*e.g.*, 52], we decided to numerically approximate $H(q^i)$, which is possible for univariate distributions. However, since we have to do this for every pixel of an image (which can be millions), we wrote an efficient GPU-based implementation. Note that the entropy $H(q_m^i)$ of each ensemble member is often easy to compute (as for the Laplace distribution in our case), because "nice" distributions with known expressions for common properties (such as entropy and KL-divergence) are typically chosen for the individual ensemble members.

3.3 Discussion and Empirical Validation

As explained in Section 3.1, our approach to modeling uncertainty is that we use an ensemble of independently trained convolutional neural networks that each predict a probability distribution $g_\theta(x)_i$ for every pixel i based on input image x. The predictions from all networks are combined as a mixture of probability distributions to yield the final prediction $g_\Theta(x)$ of the ensemble. To obtain the restored image $\hat{y}$, we compute the per-pixel expected value of each mixture distribution, *i.e.* $\hat{y}_i = \mathbb{E}\big[g_\Theta(x)_i\big]$. In particular, we use ensembles comprised of $M = 5$ networks that are trained via maximum likelihood to predict a separate Laplace distribution for every pixel, overall yielding a mixture of Laplace distributions (*cf.* Fig. 1).

While we evaluate the quality of the restored image $\hat{y}$ with common measures (NRMSE[19] and SSIM [16]), we here also validate that the per-pixel distributions $g_\Theta(x)$ are accurate, *i.e.* well-*calibrated*. To that end, given a set of validation data[20] where the ground-truth is known, for every pixel we first compute the probability that its value will fall into an interval of a certain size around $\hat{y}_i$ (denoted as *confidence*, illustrated as shaded region in Fig. 1). If the true value is indeed contained in the interval, we can count this as a *success*. The per-pixel distributions are well-calibrated if the *accuracy*, *i.e.* the fraction of successfully predicted pixels, is close to the average confidence. Furthermore, we compare confidence and accuracy more granularly for subsets of pixels within a certain confidence range. For each of these subsets, we can then plot average confidence *vs.* accuracy, which gives rise to the *reliability diagrams* shown in Supp. Figs. 18 and 19. The network ensemble is well-calibrated if the plotted curve is close to the identity function. Additionally, we measure the *expected calibration error (ECE)*, which is essentially the average (weighted) difference of the plotted curve to the identity function.

[18] To avoid these computational issues, we may alternatively define disagreement as the average KL-divergence between all pairs of ensemble members. While each comparison in terms of KL-divergence would then typically be easy to compute, the amount of comparisons grows quadratically with the size of the ensemble. Furthermore, such an alternative disagreement score would be unbounded, since the KL-divergence between any two ensemble members could be infinite if their support does not overlap.

[19] Normalized root-mean-square error

[20] Sampled from the same distribution as the training data, but *not* been used for training the networks.

Lastly, note that we validate calibration for intervals $[\hat{y}_i - \epsilon, \hat{y}_i + \epsilon]$ of various sizes as defined by ϵ.

To interpret the results, note that all images are normalized such that the bulk of all pixels (approx. 95%) is between zero and one. Supp. Figs. 18 and 19 show that our network ensembles exhibit low expected calibration error, in most cases below 3%. It is occasionally only larger for the very small interval with $\epsilon = 0.01$ (*i.e.* being roughly within $\pm 1\%$ of the dynamic intensity range). When inspecting the reliability diagrams, we find that the confidence-accuracy curves sometimes have a slight "S"-shape, we means that the predicted probabilities are slightly too high for lower-confidence pixels and slightly too low for higher-confidence pixels. Only for $\epsilon = 0.01$, the ensemble trained for isotropic reconstruction of *Danio rerio* retinas exhibits rather strong under-confidence. Supp. Figs. 18 and 19 also show that large calibration errors can be avoided when using an ensemble ($M > 1$), as compared to just a single network ($M = 1$).

In addition to this quantitative evaluation, we want to qualitatively show predictive uncertainty. A common way to measure the uncertainty of a distribution is via its *entropy* (see Supp. Fig. 20 for an example). However, we believe that looking at the per-pixel entropies as an image does not convey an intuitive understanding of uncertainty. Instead, we draw *samples* from our model to obtain possible instances of restored images and arrange several samples of restored images as a movie, which allows to quickly gauge per-pixel uncertainty (see Supp. Vid. 9).

We now turn to the topic of identifying situations where the predictions of our network ensembles may not be trustworthy. To that end, we compute the per-pixel *disagreement* of the network ensemble[21], which we normalize to take on values between zero (no disagreement at all) and one (maximal disagreement). If there is high disagreement, it intuitively means that not all individual network predictions can be correct (see Figure 5 in the main paper).

After training, we find that all of our ensembles exhibit reasonably low average disagreement on *in-distribution* (ID) images, *i.e.* images that come from the same distribution as the data they were trained on. However, when we apply a network ensemble to *out-of-distribution* (OD) images, *i.e.* images that are not sampled from the distribution of training images, we often find much higher disagreement. Hence, disagreement can sometimes identify when a network ensemble is applied to (regions of) OD images. In these cases, the prediction of the ensemble can be wrong because such data was likely not, or only rarely, seen during training. An example is shown in Supp. Fig. 20, where we applied two separate network ensembles, trained for Planaria (*Schmidtea mediterranea*) and Tribolium (*Tribolium castaneum*), respectively, to both an ID and OD image.

[21] As average Kullback-Leibler divergence between each network distribution and the ensemble mixture.

4 Sample Preparation and Imaging Procedures

Please see Supp. Tab. 1 for an overview.

4.1 Planaria

Planaria (*Schmidtea mediterranea*) were cultured at 20° C in planarian water (1.6 mM NaCl, 1 mM $CaCl_2$, 1 mM $MgSO_4$, 0.1 mM $MgCl_2$, 0.1 mM KCl, 1.2 mM $NaHCO_3$) and fed with organic bovine liver paste. To label planarian nuclei, *S. mediterranea* approximately 2-3 mm in size and starved for 2 weeks were stained for 15 hours in planarian water supplemented with 2x RedDot1 (Biotium, Cat No.: 40060) and 1% (v/v) DMSO. For training data acquisition, planaria were euthanized with 5% w/v N-Acetyl-L-cysteine (Sigma, Cat No. A7250) in PBS and subsequently fixed in 4% w/v paraformaldehyde in PBS (Carl Roth GmbH, Cat No.: 0335.2). For time lapse recordings RedDot1 stained planaria were anesthetized for 1 hr with 0.019% w/v Linalool (Sigma, Cat No. L2602) prior mounting and this Linalool supplementation was maintained throughout the course of the live imaging experiments. A 5 min incubation in 0.5% w/v pH neutralized N-Acetyl-L-cysteine was used to remove the animal's mucus prior mounting. For imaging, fixed or live animals were mounted in refractive index matched 1.5 % agarose (50% w/v Iodixanol) to enhance signal quality at higher imaging depths as previously described in [53]. The spinning disc confocal microscopy setup for imaging live planaria has been described in detail in [53]. In brief, a 30x NA 1.05 silicon oil immersion objective mounted on an Olympus IX83 stand was used. RedDot1 was excited with a 640 nm laser and a 685/40 bandpass filter was used for emission detection.

4.2 Tribolium

The EFA::nGFP transgenic line of *Tribolium castaneum* was used for imaging embryonic development with labeled nuclei [54]. All imaging was done on the Zeiss 710 multiphoton laser scanning microscope using the 25x multi immersion objective. The embryos were imaged either under oil or water. The beetles were reared and embryo collection was done according to standard protocols [55].

4.3 Flywing

Fruitfly datasets consist of in vivo images of the dorsal side of the Ecad::GFP labelled pupal wing recorded with a fluorescent spinning disk microscope.

Flies were raised under 25° C at standard conditions. Along with the Ecad::GFP, some of the samples carried an additional mutation in one of the PCP proteins, which didn't affect the expression of Ecad::GFP. Pupae were collected and prepared for imaging as described in [56].

Images were acquired with a spinning disk microscope equipped with a Zeiss AxioObserver stand, Zeiss LCI Plan-Neofluar 63x, 1.3 NA Imm Corr objective, Zeiss AxioCam MR3 camera (2x2 binning) and a Yokogawa CSU-X1 spinning disk unit. Raw images were acquired with 20−

−40 ms exposure time and 0.37 mW laser power (measured at the sample plane). Groundtruth images were acquired with 120ms exposure time and the same laser power settings.

4.4 Liver

Mice livers were fixed and cut into 100 μm slices. Cell nuclei were stained with DAPI, and the sinusoidal networks with flk-1 antibody. The samples were imaged using a Zeiss 63x glycerol immersion objective (1.3 NA) in an upright Zeiss LSM 780 NLO multiphoton laser-scanning microscope equipped with Gallium arsenide phosphide detectors. DAPI was excited at 780 nm by a Chameleon Ti-Sapphire 2-photon laser. Other dyes were excited by single photon using 488 (phalloidin), 561 (cd13) and 633 (flk-1) laser lines. We acquired 2x2 tile images with 10% of overlap. Every z-tack took 25 minutes to be imaged (total time 1 hr 40 min).

C57BL/6JOlaHsd mice were purchased from Charles River Laboratory. Livers were fixed through transcardial perfusion with 4% paraformaldehyde and post-fixed overnight at 4°C with the same solution. Vibratome sections (thickness 100 μm) were stained with anti-CD13 (Novus, cat NB100-64843, rat, 1/500), anti-flk1 (R&D system, cat AF644, goat, 1/200), phalloidin-488 (LIFE technologies, cat A12379, 1/150) and DAPI (LIFE technologies, cat D1306, 1 μg/ml). Tissue slices were optically cleared by a modified version of SeeDB [57].

All procedures were performed in compliance with German animal welfare legislation and in pathogen-free conditions in the animal facility of the MPI-CBG, Dresden, Germany. Protocols were approved by the Institutional Animal Welfare Officer (Tierschutzbeauftragter) and all necessary licenses were obtained from the regional Ethical Commission for Animal Experimentation of Dresden, Germany (Tierversuchskommission, Landesdirektion Dresden).

4.5 Retina

Zebrafish imaging experiments were performed with a transgenic zebrafish line Tg(bactin:eGFP-LAP2B) that labels the nuclear envelope. This transgenic line consists of a rat LAP2B gene (Lamina-associated polypeptide 2 beta) tagged with GFP under a constitutive promoter. The adult zebrafish were maintained and bred at 26°C. Embryos were raised in E3 medium at 28.5°C and treated with 0.2 mM 1-phenyl-2-thiourea (Sigma-Aldrich) around 8 h postfertilization (hpf) onward to delay pigmentation. All animal work was performed in accordance with European Union directive 2011/63/EU as well as the German Animal Welfare Act. 24 hpf zebrafish embryos were fixed overnight in 4% paraformaldehyde (Sigma-Aldrich). They were permeabilized with 0.25% trypsin and incubated with a far-red DNA stain (DRAQ5, Thermo Fisher Scientific) for 2 days at 4°C. For imaging on Spinning disk confocal SD4 (Andor Revolution WD Borealis Mosaic), the stained embryos were mounted into 0.6% agarose (E3 media) in glass-bottom dishes (MatTek Corporation). The Spinning disk confocal SD4 is composed by scanning head Yokogawa CSU-W1 (4000rpm) with two pinhole discs (50 μm and 25 μm) (Andor Technology). Images were captured with an Olympus UPLSAPO 60x 1.3 SIL objective

and Andor iXon 888 Ultra with fringe suppression. The microscope was operated through the software Andor iQ version 3.4.1. A z stack around 70 μm thick was acquired with 1-μm steps.

4.6 Drosophila

All input stacks were provided by the authors of [2]. Details on sample preparation and imaging can be found in Supplemental Notes of [2].

4.7 INS-1 cells

INS-1 cells were cultured and transiently transfected with pEG-hIns-SNAP as previously described [32]. The cells were labeled with 1 μM SNAP-Cell 505-Star (NEB) and simultaneously with 1 μM SiR-tubulin (Spirochrome) for 1 h. After washing cells were imaged with the DeltaVision OMX (GE) with an Olympus Plan ApochromatN 60x oil objective with a numerical aperture of 1.42 at 37°C and with 5% CO_2. Time-lapse movies were acquired in wide-field mode with 50 ms exposure-time and 10% fluorescence intensity for each channel resulting in a final speed of 2 frames per second (fps). Deconvolution was done with the SoftWorkx software package.

4.8 HeLa cells

For live-cell imaging of GFP-tagged microtubules, HeLa cells stably expressing H2B-mCherry/-mEGFP-α-tubulin [58] were grown in DMEM containing 10% FBS, 100 U/ml penicillin and 100mg/ml streptomycin at 37° C with 5% CO_2 in a humidified incubator. Before imaging cells were seeded onto #1.5 glass bottom 35mm u-Dish (ibidi GmbH). Imaging was performed on a Zeiss Elyra PS.1 inverted microscope at 37° C and 5% CO_2 in TIRF mode with a 100x TIRF objective (Plan-APOCHROMAT 100x /1.46 Oil, Zeiss) and additional 1.6x magnification with 488nm laser illumination at an on-sample intensity of $< 10 W/cm^2$. HeLa cells stably expressing H2B-mCherry/mEGFP-α-tubulin were kindly provided by Buzz Baum, UCL.

5 Deployment of Image Restoration Models

Our goal is to make our work accessible to as many people as possible. Since we aim at image data restoration for life scientists, our strategy is to deploy our trained neural networks within software packages that are commonly used for biological image and data analysis. Two very popular, open source, and freely available platforms are Fiji [59] and Knime [60]. All networks for this publication where trained on a Linux workstation using Python scripts requiring Keras [61] and a TensorFlow [62] backend. All scripts will be made available on GitHub. A set of trained networks are saved and made publicly available for their integration and use from within Fiji and Knime.

Frameworks for deep learning became very powerful in recent years. Still, the deployment of trained networks poses multiple problems for which solutions are not yet readily available: (*i*) Training and inference of neural networks usually needs to be performed on graphics hardware (GPUs) in order to be sufficiently performant. Not every computer running Fiji or Knime has a sufficiently powerful GPU available, (*ii*) most deep learning frameworks rely on CUDA when utilizing GPU computations. CUDA only works on Nvidia GPUs, further reducing the number of machines on which our networks can be efficiently deployed. Finally, (*iii*) even the latest and most powerful deep learning frameworks do not support the Windows and/or MacOS operating systems very well. For example, the installation of TensorFlow requires several manual steps and can currently not easily be automated. Consequently, installing Fiji or Knime is presently not enough to use our restoration pipelines specifically, or previously trained neural networks generally.

We provide Fiji plugins and Knime workflows for four example applications: (*i*) 3D Denoising of *Tribolium castaneum* embryos (Sections 2.3.2 and 4.2) and *Schmidtea mediterranea* flatworms (Sections 2.3.1 and 4.1), (*ii*) Surface projection of developing wings of *Drosophila melanogaster* (Sections 2.3.3 and 4.3), (*iii*) Isotropic reconstruction of *Danio rerio* retinas (Sections 2.4.2 and 4.5), and (*iv*) Deconvolution of microtubules in INS-1 cells (Sections 2.5.1 and 4.7). In addition to the plugins and workflows, we provide example data, detailed installation instructions, as well as step-by-step tutorials on how to use our tools. Please visit http://csbdeep.bioimagecomputing.com for gaining access to tutorials, data, and code.

In the following, we discuss the deployment of our networks with Fiji and Knime. Additionally, we show how the resulting software packages can be used.

5.1 Fiji Integration

Fiji is a very common platform for analyzing biological image data [63]. It is equipped with plenty of easily accessible plugins that can be installed using the Fiji Updater. Our goal was to develop plugins that enable to run trained network models on image data opened in Fiji.

There are two ways to use networks trained with Keras/TensorFlow from within Fiji, which is itself written in Java. First we only used Keras functionality that can be stored as Tensor-

Flow models, leaving us with the problem of running TensorFlow networks from within Java. A Java-based deep learning framework, Deeplearning4j [64], is capable of running saved TensorFlow models. Unfortunately, not all functionality we needed was implemented at the time of publication[22], leaving us no choice than to use TensorFlow's own Java API [65]. Also, Google's TensorFlow Java API is in an early development phase and does currently not follow their own API stability guide. They do, however, cover our minimal requirements already, enabling us to launch TensorFlow model from within Fiji. With collaborators we developed a library to interact between TensorFlow and ImgLib2 [66], the basic library for image data in Fiji. This library provides a functional, open, and reusable interface for plugins that intend to use TensorFlow [67].

When one of our plugins is started for the first time, it downloads the appropriate network model and stores it locally. In then maps the input image, which will be adequately normalized, to the input tensor of the network. Depending on the size of the input image, the data is tiled and processed in chunks to fit into the available (GPU) memory. The results of the respective image tiles will then be reassembled into one big image or volume before being presented to the user.

From within the Fiji Updater, our plugins can be installed by selecting the update site called 'CSBDeep' (see also [68]). The installed plugins can then be found in the Fiji menu under *Plugins* ▷ *CSBDeep* (see also Supp. Fig. 21).

5.1.1 Extending the available functionality

Our examples fulfill the purpose of demonstrating the functionality of our general content-aware image restoration approach. Still, the goal is to reconstruct diverse data for many potentially very different research projects. Once a new network model is trained by using an altered version of our training scripts, we foresee two ways to integrate this model into Fiji.

We offer a Fiji plugin called *Generic Network*. This plugin allows to process image data with any given TensorFlow model. The model must be saved in TensorFlow's *SavedModelBundle* format and before it can be started one has to make the mapping of image to input tensor explicit within the plugin. Additionally, one can choose the desired image normalization parameters and image tiling options. This generic model launcher enables the access to neural networks from within Fiji for the entire community, strongly encouraging the creation of functionality that utilizes deep learning.

In case a trained model turns out to be useful and of general enough interest to justify the creation of a Fiji plugin in its own right, our ImageJ-TensorFlow bridge [67] and the open source CSBDeep demo plugins can be taken as a basis and blueprint for others.

[22]Deeplearning4j turned out to be incompatible as it does not support upsampling and deconvolution layers, which are necessary for all our network topologies.

5.2 KNIME Integration

In addition to FIJI, we also offer full KNIME integration. For each example network available in FIJI we created a KNIME workflow similar to the one shown in Supp. Fig. 22.

We load and run our trained networks using the KNIME *Python extension*, allowing to run arbitrary Python scripts. After loading a raw image file from disk, the data is normalized and prepared for the trained network. Then, in the node called 'Model Execution', the image to be reconstructed is subdivided into tiles of adequate sizes, fed to the network, and finally reassembled to the overall result (see Supp. Fig. 22).

The major drawback of this deployment strategy is that an adequate Python environment, including a TensorFlow installation, has to be made available to KNIME. In the future, KNIME will offer better TensorFlow support, using the TensorFlow Java API mentioned above.

We provide detailed instructions on how to install KNIME and how our workflows can be used in the CSBDeep wiki (https://github.com/mpicbg-csbd/CSBDeep/wiki). There, you can also find example data.

References

1. Blasse, C. *et al.* PreMosa: extracting 2D surfaces from 3D microscopy mosaics. *Bioinformatics,* btx195 (2017).
2. Royer, L. A. *et al.* Adaptive light-sheet microscopy for long-term, high-resolution imaging in living organisms. *Nature biotechnology* **34,** 1267–1278 (2016).
3. Gustafsson, N. *et al.* Fast live-cell conventional fluorophore nanoscopy with ImageJ through super-resolution radial fluctuations. *Nature Communications* **7** (2016).
4. Maška, M. *et al.* A benchmark for comparison of cell tracking algorithms. *Bioinformatics* **30,** 1609–1617 (2014).
5. Ronneberger, O., Fischer, P. & Brox, T. *U-net: Convolutional networks for biomedical image segmentation* in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2015).
6. Arganda-Carreras, I. *et al.* Trainable Weka Segmentation: a machine learning tool for microscopy pixel classification. *Bioinformatics,* btx180 (2017).
7. Aigouy, B., Umetsu, D. & Eaton, S. in *Drosophila: Methods and Protocols* (ed Dahmann, C.) 227–239 (Springer New York, New York, NY, 2016). ISBN: 978-1-4939-6371-3.
8. Mao, X.-J., Shen, C. & Yang, Y.-B. Image restoration using convolutional auto-encoders with symmetric skip connections. *arXiv:1606.08921* (2016).
9. McCann, M. T., Jin, K. H. & Unser, M. Convolutional Neural Networks for Inverse Problems in Imaging: A Review. *IEEE Signal Processing Magazine* **34,** 85–95 (2017).
10. Rivenson, Y. *et al.* Deep Learning Microscopy. *arXiv:1705.04709* (2017).
11. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86,** 2278–2324 (1998).
12. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521,** 436–444 (2015).
13. Krizhevsky, A., Sutskever, I. & Hinton, G. E. *Imagenet classification with deep convolutional neural networks* in *Advances in neural information processing systems* (2012), 1097–1105.
14. He, K., Zhang, X., Ren, S. & Sun, J. *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification* in *IEEE Conference on Computer Vision and Pattern Recognition* (2015), 1026–1034.
15. Kingma, D. & Ba, J. Adam: A method for stochastic optimization. *arXiv:1412.6980* (2014).
16. Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* **13,** 600–612 (2004).

17. Buades, A., Coll, B. & Morel, J.-M. *A non-local algorithm for image denoising* in *IEEE Conference on Computer Vision and Pattern Recognition* **2** (2005), 60–65.

18. Sommer, C., Straehle, C., Koethe, U. & Hamprecht, F. A. *Ilastik: Interactive learning and segmentation toolkit* in *Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on* (2011), 230–233.

19. Jones, E., Oliphant, T., Peterson, P., *et al. SciPy: Open source scientific tools for Python* http://www.scipy.org. 2001–.

20. Aigouy, B. *et al.* Cell flow reorients the axis of planar polarity in the wing epithelium of Drosophila. *Cell* **142,** 773–786 (2010).

21. Etournay, R. *et al.* Interplay of cell dynamics and epithelial tension during morphogenesis of the Drosophila pupal wing. *Elife* **4,** e07090 (2015).

22. Guirao, B. *et al.* Unified quantitative characterization of epithelial tissue development. *Elife* **4,** e08519 (2015).

23. Chhetri, R. K. *et al.* Whole-animal functional and developmental imaging with isotropic spatial resolution. eng. *Nature Methods* **12,** 1171–1178 (Dec. 2015).

24. Hell, S. W., Lindek, S., Cremer, C. & Stelzer, E. H. Confocal microscopy with an increased detection aperture: type-B 4Pi confocal microscopy. *Optics letters* **19,** 222–224 (1994).

25. Sidhaye, J. & Norden, C. Concerted action of neuroepithelial basal shrinkage and active epithelial migration ensures efficient optic cup morphogenesis. *eLife* **6,** e22689 (2017).

26. Kwan, K. M. *et al.* A complex choreography of cell movements shapes the vertebrate eye. *Development* **139,** 359–372 (2012).

27. Icha, J., Kunath, C., Rocha-Martins, M. & Norden, C. Independent modes of ganglion cell translocation ensure correct lamination of the zebrafish retina. *Journal of Cell Biology* **215,** 259–275 (2016).

28. Morales-Navarrete, H. *et al.* A versatile pipeline for the multi-scale digital reconstruction and quantitative analysis of 3D tissue architecture. *Elife* **4,** e11214 (2015).

29. Rajaram, S., Pavie, B., Hac, N. E., Altschuler, S. J. & Wu, L. F. SimuCell: a flexible framework for creating synthetic microscopy images. *Nature Methods* **9,** 634–635 (2012).

30. Venkataramani, V., Herrmannsdorfer, F., Heilemann, M. & Kuner, T. SuReSim: simulating localization microscopy experiments from ground truth models. *Nature Methods* **13,** 319–325 (2016).

31. Svoboda, D. & Ulman, V. Mitogen: A framework for generating 3D synthetic time-lapse sequences of cell populations in fluorescence microscopy. *IEEE transactions on medical imaging* **36,** 310–321 (2017).

32. Ivanova, A. *et al.* Age-dependent labeling and imaging of insulin secretory granules. *Diabetes* **62,** 3687–3696 (2013).

33. Kendall, A. & Gal, Y. *What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?* in *Advances in Neural Information Processing Systems 30 (NIPS)* (2017).

34. Nix, D. A. & Weigend, A. S. *Estimating the Mean and Variance of the Target Probability Distribution* in *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on* **1** (1994), 55–60.

35. Bishop, C. M. *Mixture Density Networks* tech. rep. (Aston University, Birmingham, 1994).

36. Koller, D. & Friedman, N. *Probabilistic Graphical Models: Principles and Techniques* (MIT Press, 2009).

37. Lakshminarayanan, B., Pritzel, A. & Blundell, C. *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles* in *Advances in Neural Information Processing Systems 30 (NIPS)* (2017).

38. Tanno, R. *et al. Bayesian Image Quality Transfer with CNNs: Exploring Uncertainty in dMRI Super-Resolution* in *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (2017).

39. Berger, J. O. *Statistical Decision Theory and Bayesian Analysis* (Springer New York, 1985).

40. Neal, R. M. *Bayesian Learning for Neural Networks* PhD thesis (University of Toronto, 1995).

41. MacKay, D. J. C. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation* **4,** 448–472 (1992).

42. Wainwright, M. J. & Jordan, M. I. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning* **1,** 1–305 (2008).

43. Gal, Y. & Ghahramani, Z. *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning* in *33rd International Conference on Machine Learning (ICML)* (2016).

44. Kingma, D. P., Salimans, T. & Welling, M. *Variational Dropout and the Local Reparameterization Trick* in *Advances in Neural Information Processing Systems 28 (NIPS)* (2015).

45. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* **15,** 1929–1958 (2014).

46. Goodfellow, I. J., Shlens, J. & Szegedy, C. *Explaining and Harnessing Adversarial Examples* in *International Conference on Learning Representations (ICLR)* (2015).

47. Gneiting, T. & Raftery, A. E. Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American Statistical Association* **102,** 359–378 (2007).

48. DeGroot, M. H. & Fienberg, S. E. The Comparison and Evaluation of Forecasters. *Journal of the Royal Statistical Society. Series D (The Statistician)* **32,** 12–22 (1983).

49. Niculescu-Mizil, A. & Caruana, R. *Predicting Good Probabilities With Supervised Learning* in *22nd International Conference on Machine Learning (ICML)* (2005).

50. Naeini, M. P., Cooper, G. F. & Hauskrecht, M. *Obtaining Well Calibrated Probabilities Using Bayesian Binning* in *29th AAAI Conference on Artificial Intelligence* (2015).

51. Guo, C., Pleiss, G., Sun, Y. & Weinberger, K. Q. *On Calibration of Modern Neural Networks* in *34th International Conference on Machine Learning (ICML)* (2017).

52. Kolchinsky, A. & Tracey, B. D. Estimating Mixture Entropy with Pairwise Distances. *arXiv:1706.02419* (2017).

53. Boothe, T. *et al.* A tunable refractive index matching medium for live imaging cells, tissues and model organisms. *eLife* **6** (2017).

54. Sarrazin, A. F., Peel, A. D. & Averof, M. A segmentation clock with two-segment periodicity in insects. *Science* **336,** 338–341 (2012).

55. Brown, S. J. *et al.* The Red Flour Beetle, Tribolium castaneum (Coleoptera): A Model for Studies of Development and Pest Biology. *Cold Spring Harbor Protocols* **2009,** pdb.emo126–pdb.emo126 (Aug. 2009).

56. Classen, A.-K., Aigouy, B., Giangrande, A. & Eaton, S. Imaging Drosophila pupal wing morphogenesis. *Drosophila: Methods and Protocols,* 265–275 (2008).

57. Ke, M.-T., Fujimoto, S. & Imai, T. SeeDB: a simple and morphology-preserving optical clearing agent for neuronal circuit reconstruction. *Nature neuroscience* **16,** 1154–1161 (2013).

58. Mchedlishvili, N. *et al.* Kinetochores accelerate centrosome separation to ensure faithful chromosome segregation. *J Cell Sci* **125,** 906–918 (2012).

59. Schindelin, J., Rueden, C. T., Hiner, M. C. & Eliceiri, K. W. The ImageJ ecosystem: An open platform for biomedical image analysis. *Molecular Reproduction and Development* **82,** 518–529. ISSN: 1098-2795 (2015).

60. Berthold, M. R. *et al.* KNIME - the Konstanz Information Miner: Version 2.0 and Beyond. *ACM SIGKDD Explorations Newsletter* **11,** 26–31 (Nov. 2009).

61. Chollet, F. *Keras* https://github.com/fchollet/keras. 2015.

62. Abadi, M. *et al. TensorFlow: A system for large-scale machine learning* in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation. Savannah, Georgia, USA* (2016).

63. Schindelin, J. *et al.* Fiji: An Open-Source Platform for Biological-Image Analysis. **9,** 676–82 (June 2012).

64. Team, D. D. *Deeplearning4j: Open-source distributed deep learning for the JVM, Apache Software Foundation License 2.0.* http://deeplearning4j.org. 2017.

65. TensorFlowers. *TensorFlow for Java* https://github.com/tensorflow/tensorflow/tree/master/tensorflow/java. 2017.

66. Pietzsch, T., Preibisch, S., Tomančák, P. & Saalfeld, S. ImgLib2—generic image processing in Java. *Bioinformatics* **28,** 3009–3011 (2012).

67. Rueden, C., Shankar, A., Yang, S. & Dietz, C. *ImageJ + TensorFlow integration layer* https://github.com/imagej/imagej-tensorflow. 2017.

68. Schmidt, D., Wilhelm, B. & Jug, F. *CSBDeep in Fiji* https://github.com/mpicbg-csbd/CSBDeep/wiki/CSBDeep-in-Fiji. 2017.