Supplementary Note 1 U-Net software

Detailed installation instructions for the installation of the different parts of our software can be found online on the U-Net project page `https://lmb.informatik.uni-freiburg.de/resources/opensource/unet`. There we also provide detailed screencast tutorials on using the U-Net segmentation plugin for segmentation, detection and transfer learning (finetuning).

SN 1.1 Custom caffe_unet backend package

We provide pre-compiled versions of our custom caffe extension for Ubuntu 16.04 with and without GPU support (Supplementary Files). Please pick the package suitable for your system:

- `caffe_unet_package_16.04_gpu_cuDNN.zip`
- `caffe_unet_package_16.04_gpu_no_cuDNN.zip`
- `caffe_unet_package_16.04_cpu.zip`

Each zip-file contains:

- Caffe binaries and required libraries (except CUDA and cuDNN for the GPU versions)

Note that for the GPU versions you need to have the appropriate CUDA libraries installed on your system.

SN 1.2 Memory and running time

We analyzed the running times and GPU memory consumption of the 2D U-Net with and without cuDNN enabled given different input image sizes (Fig. SN1.1). We iteratively increased the input image size and measured the used GPU memory. The measurements were obtained using an nVidia GTX Titan X with 12GB of total memory and almost fit a linear function. Additionally we performed measurements with GPUs with less total memory, but we always observed measurements approximately lying on the curve. Measurements were performed using the caffe MATLAB interface and the nVidia `nvidia-smi` tool on a standard workstation running Ubuntu 16.04.

Use of cuDNN reduces the memory footprint dramatically, therefore we highly recommend to use cuDNN if possible. 3D networks cannot be used without cuDNN. Even with 12GB and cuDNN enabled the maximum input tile shape for a four-stage network with the same number of channels per stage as in the 2D case would be around 140^3px which is below the minimum required tile shape of 188^3px to predict a volume of only 4^3px! Our 3D model allows to train with an input tile shape of $116\times252\times252$px producing output tiles with shape $28\times68\times68$px. Processing a 1024^3px volume with this model requires approximately four hours pure GPU time.

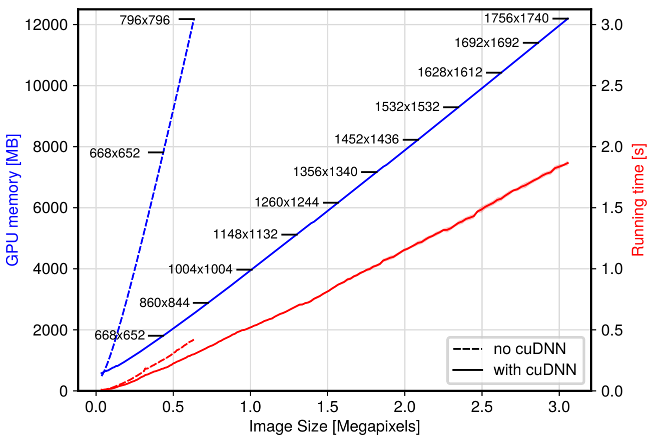


Figure SN1.1: U-Net running time and GPU memory consumption with increasing input tile shape. Measurements were performed on an nVidia GTX Titan X with 12GB DDR5 memory with cuDNN enabled. Running time excludes data transfer between GPU and host memory.

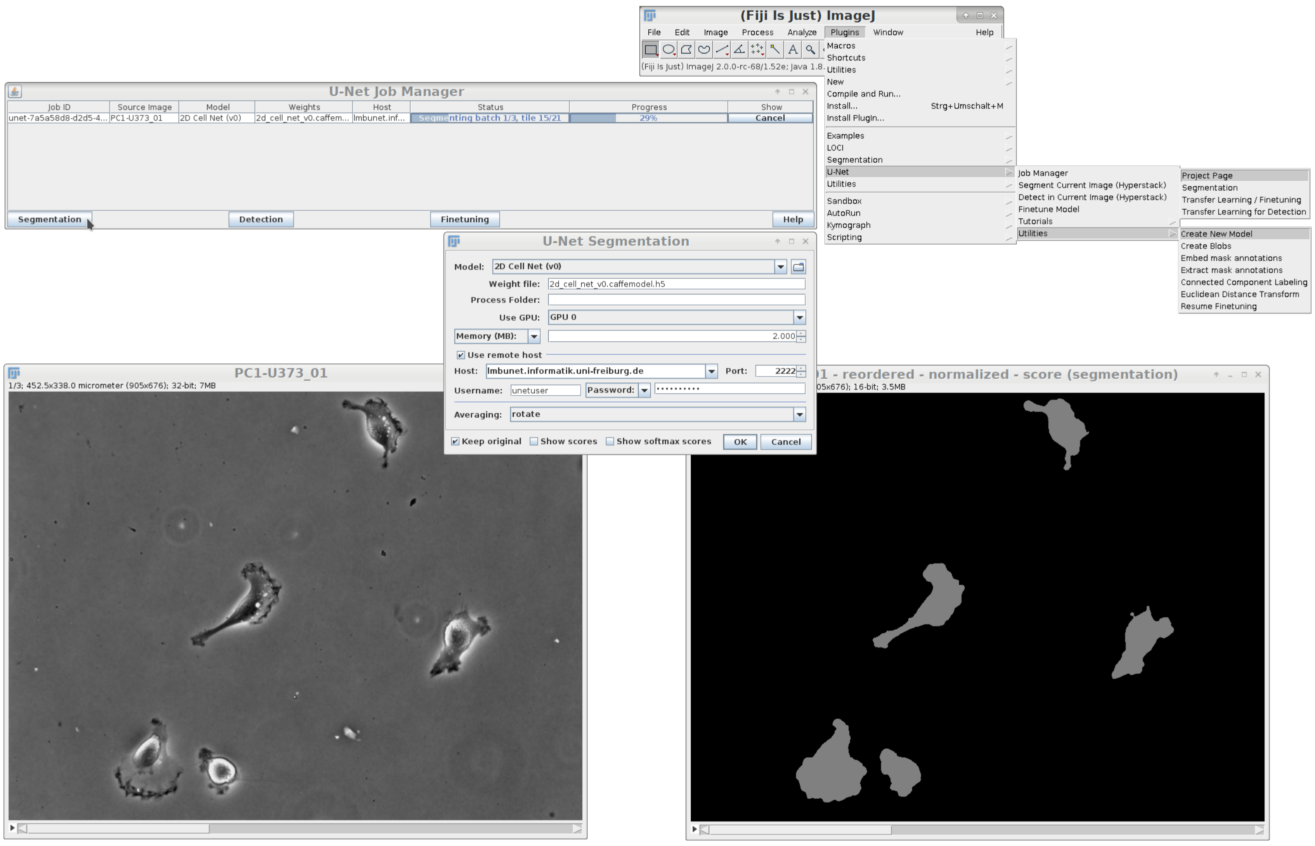
SN 1.3 The U-Net segmentation plugin

The ImageJ/Fiji plugin (Fig. SN1.2) allows intuitive application of the U-Net for detection and segmentation tasks with pre-trained models in standard lab environments within minutes. Any image format accessible through ImageJ or one of its numerous extensions (e.g. contained in Fiji) can be processed. Prerequisite for successful segmentation is correct setup of the element size of the image via the ImageJ Calibration interface. The plugin then takes care of intensity and scale normalization and calls the caffe backend.

U-Net detection and segmentation can be easily embedded into automated image processing pipelines via Macro calls that can be recorded using ImageJ's macro recorder. Created models and their trained parameters are persistently stored to HDF5 files for re-use, refinement and distribution to other labs.

Moreover, the plugin allows to adapt pre-trained models to new datasets via transfer learning when selecting the "Finetuning" task. Opened images with annotation overlays are presented to the user as potential training samples. Training samples can be left out from training and used as validation images for monitoring model evolution during training by means of loss, IoU and F-measure. Transfer learning can be suspended at any time and continued from a saved snapshot. If using a remote server the snapshot can be transferred to another lab computer running the U-Net Segmentation plugin.

The plugin's "Create New Model" function allows more advanced users to create custom U-Net models with variable depth per dimension and number of initial feature channels and train them from scratch on their own data. Experts can also directly provide caffe protocol buffer textfiles (prototxt) containing arbitrary encoder-decoder architectures and corresponding solver settings and convert them to plugin-compliant format using the python script `python/unet/createModeldefH5.py` in the caffe_unet packages.


(Fiji Is Just) ImageJ
File Edit Image Process Analyze Plugins Window Help
(Fiji Is Just) ImageJ 2.0.0-rc-68/1.52e; Java 1.8.
Macros
Shortcuts
Utilities
New
Compile and Run...
Install... Strg+Umschalt+M
Install Plugin...
Examples
LOCI
Segmentation
U-Net
Utilities
Sandbox
AutoRun
Kymograph
Scripting
Job Manager
Segment Current Image (Hyperstack)
Detect in Current Image (Hyperstack)
Finetune Model
Tutorials
Utilities
Project Page
Segmentation
Transfer Learning / Finetuning
Transfer Learning for Detection
Create New Model
Create Blobs
Embed mask annotations
Extract mask annotations
Connected Component Labeling
Euclidean Distance Transform
Resume Finetuning
Segmentation
Detection
Finetuning
Help
PC1-U373_01
1/3; 452.5x338.0 micrometer (905x676); 32-bit; 7MB
01 - reordered - normalized - score (segmentation)
05x676); 16-bit; 3.5MB

U-Net Job Manager

Job ID	Source Image	Model	Weights	Host	Status	Progress	Show
unet-7a5a58d8-d2d5-4...	PC1-U373_01	2D Cell Net (v0)	2d_cell_net_v0.caffem...	lmbunet.inf...	Segmenting batch 1/3, tile 15/21	29%	Cancel

U-Net Segmentation

Model: 2D Cell Net (v0)
Weight file: 2d_cell_net_v0.caffemodel.h5
Process Folder:
Use GPU: GPU 0
Memory (MB): 2.000
[✓] Use remote host
Host: lmbunet.informatik.uni-freiburg.de Port: 2222
Username: unetuser Password: ••••••••••
Averaging: rotate
[✓] Keep original [ ] Show scores [ ] Show softmax scores OK Cancel

Figure SN1.2: The Fiji U-Net Segmentation plugin. Top left: U-Net Segmentation manager; Top right: Fiji interface; Center: U-Net Segmentation parameters dialog; Bottom left: Raw data; Bottom right: segmentation result.

For installation simply check the "U-Net Segmentation" site in the Fiji updater.

Supplementary Note 2 Annotation guide

Proper image annotation is the key for learning algorithms to produce expected results. Given a pre-trained model as *e.g.* the 2D cell segmentation model, we showed that already very few annotated objects or images allow to adapt the model to a new dataset. But in which form do we have to provide these annotations? There are two aspects of annotation: First, what to annotate to obtain good results with only few data and secondly, how to generate and provide the annotations technically. This section will cover both aspects.

SN 2.1 What to annotate?

Be consistent Decide what you want to annotate and prepare a precise annotation protocol before you start annotating. A description like mark all cells with a point, is insufficient, instead write: Place a point marker in the center of the nucleus of every cell. If possible give your annotation protocol to a colleague and ask her to annotate the same data. Compare the annotations using quantitative measures like IoU (segmentation) or F-measure (detection) and discuss discrepancies. If there are systematic differences find a consensus and extend the protocol accordingly.

Especially out-of-focus parts of objects pose a problem, you can decide to guess the boundary of those objects for simple counting tasks or only annotate the part that is in focus which may lead to object fragmentation. Both is not desirable but you have to decide for a clear strategy based on what you want to measure. Ignoring out-of-focus parts does not allow the network to learn how to deal with them and you might get disappointing results.

Quality over quantity For segmentation, always annotate full objects and follow the object contour as accurately as possible. For detection, only annotate positions that can be uniquely localized, *e.g.* line/plane intersections, corners, centers of round structures, ... From our experience it is better to provide few complete high quality annotations than providing a large number of partial or inaccurate annotations.

Capture the whole range of visual appearances The U-Net is quite strong in interpolating between seen object appearances, but it poorly extrapolates to appearances that are very different from what it saw during training. Therefore selecting a proper subset of objects or images capturing the whole range of possible appearances is very important. Objects that only differ by location, orientation or slight deformation can be treated as identical when selecting "good" training samples, because these operations are applied during data augmentation. Do not present only pathological cases, then the network cannot learn the usual appearance. Instead try to keep the ratio of usual and unusual appearances as realistic as possible to avoid a training bias towards rare appearances and classes. *E.g.* if one out of hundred cells looks "somehow odd", annotating only ten normal cells and one odd cell is still fine. But, when annotating only one normal and one odd cell the network might learn that every other cell should look "odd".

The higher the visual variability of either objects or background, the more annotations are required. *I.e.* if you have clean samples without background clutter and your foreground objects are evenly stained with a fluorescence stain, ten cells can already be sufficient. But, *e.g.* in the case of the pollen dataset were particles were automatically collected from polluted air in an urban environment, water droplets, air inclusions, dust particles and other tiny objects produce lots of background clutter. This leads to many false positives which can only be counteracted with more training data and longer training times.

Keep some annotations for validation We highly recommend to use a part of your annotated data for validating your trained model, *i.e.* this part is not used for training. Training without validation set is of course possible, but if you want to know how well your model will perform on new unseen data of the same kind, you must test your model on unseen yet annotated data. Especially with very small amounts of training data, validation on a left-out set is the only way of identifying overfitting. We recommend a split of approximately 75% of your annotated data for training and 25% for validation.

SN 2.2 How to annotate in ImageJ?

The U-Net segmentation plugin supports two kinds of annotations, region of interest (ROI)-based annotations and mask annotations which are prepared in a separate (typically 16-Bit integer) image. In both cases annotations must be associated with the raw data by embedding them as overlay. After embedding, image and annotations form a training sample that is recognized by the plugin. Both, ROI and mask annotations are equally well suited for network training and transfer learning but require slightly different preparation steps.

SN 2.2.1 ROI-based image annotation for instance-aware semantic segmentation

General notes ROI-based annotation is one of ImageJ's core capabilities and is not part of our plugin nor does it require third-party plugins. You manually generate and manipulate ROIs using the regional selection tools: Rectangle, Oval, Ellipse, Selection Brush, Polygon or Freehand. For simple segmentation tasks you can also consider using the Wand (tracing) tool to obtain an initial segmentation. ROIs are managed using the ImageJ ROI Manager. Please read the "Selections" section of the ImageJ User's guide (`https://imagej.nih.gov/ij/docs/guide/146-10.html`) for a comprehensive overview over ROI generation and manipulation. Use ROI manager "Add" to add the current selection to its list of ROIs. If you want to modify an existing ROI, simply select it in the ROI manager, modify the selection using ImageJ's selection tools and click "Update" in the ROI manager to replace the selected ROI by the new selection.

Avoid overlapping ROIs! Behavior is undefined in the case of overlapping ROIs. Internally the plugin turns ROI annotations to mask annotations which only allow for one label per pixel. The masks are generated using the painter's algorithm, *i.e.* ROIs are painted irrespective of whether affected pixels already contained label information. Therefore, the label from the last ROI that is painted will be used. We do not give any guarantees that ROIs are painted in the order they are shown in the ROI manager.

Class label information is inferred from the ROI name. There is one special case: If you name a ROI **"ignore"**, pixels in the selected region will not contribute to the training. There are two use cases: First, the image contains many very similar instances and a subset already covers all possible appearances, then select only a part of the image for annotation and turn the rest into one big ignore ROI. Secondly, if you are not able to provide accurate annotations in a region, you can tell the network to ignore it. However, this has been taken with a grain of salt, because these regions would in fact be most valuable for network training. Try to provide annotations wherever possible even if they are not perfectly pixel-accurate. This gives you control over how the network will address problematic regions.

2D foreground/background segmentation Create one ROI per object. We recommend to start with a simple shape as *e.g.* an ellipse (Right-click "Oval" selection for its alternatives) for round cells and add fine details using the "Selection Brush" tool (Another alternative of the "Oval" tool). The selection brush adds to the current selection when painting from its inside and subtracts from the selection when painting from its outside. When you are satisfied with the segmentation, add it to the ROI manager (Hit "Ctrl+T" or click "Edit→Selection→Add to Manager"). Continue with the next object until all foreground objects in the image are annotated. Finally, turn the selections in the ROI manager into an Image Overlay by clicking "Image→Overlay→From ROI manager". Uncheck "Labels are classes" during transfer learning.

2D multi-class segmentation Create one ROI per class and object. Sequentially add new ROIs to the ROI manager and rename them to the object class name, *e.g.* cell, cytoplasm, nucleus, membrane, ... Don't use class names like "organelle-1", "organelle-2", since the "-<number>" part will be stripped when parsing the class name. After annotating all objects, turn the selections in the ROI manager into an Image Overlay by clicking "Image→Overlay→From ROI manager". Check "Labels are classes" during transfer learning.

3D foreground/background or multi-class segmentation We recommend to annotate only non-consecutive selected slices, at least one showing upper cell boundaries, one showing central cuts and another one for lower cell boundaries. More annotated slices help, but increase annotation effort and difficulty. Especially in the case of consecutive slices it is very hard to annotate object boundaries such that they lead to smooth surfaces in depth. If only sparse annotations are given the network will usually learn to produce smooth 3D segmentations which is most often desired. For slices you don't want to annotate, select the whole slice (Hit "Ctrl+A" or click Edit→Selection→Select all) add it to the ROI manager and rename it to "ignore". Process all other slices as described in the 2D case and sequentially add new ROIs to the ROI manager for each object. Change the ROI names to distinguish object instances. ROI names must start with the class name, *e.g.* cell, cytoplasm, nucleus, membrane, ..., followed by a hash ("#") symbol and a unique instance number. Trailing parts of the ROI name containing only numbers or dashes are ignored, *e.g.* ROI label "cell#5-123-456" will be translated to class "cell" instance "5" and everything following the first dash being ignored. Make sure that annotations belonging to the same object have the same instance number. After annotating all objects, turn the selections in the ROI manager into an Image Overlay by clicking "Image→Overlay→From ROI manager". Check "Labels are classes" during transfer learning.

SN 2.2.2 ROI-based image annotation for multi-class detection

2D Instead of the regional ROI tools, use the Multi-Point tool (alternative under "Point" tool). For each class, add all locations to the multi-point-ROI and add the ROI to the ROI manager (Hit "Ctrl+T" or click "Edit→Selection→Add to Manager"). Change the ROI names to their class label, *e.g.* GFP, RFP, Colocalization, ... Don't use class names like "marker-1", "marker-2", since the "-<number>" part will be stripped when parsing the class name. Especially in the case of multi-point selections we recommend to use different colors for the different classes during annotation. The marker color can be changed using the ROI properties in the ROI manager. You can edit multi-point-ROIs by selecting them in the ROI manager. Then click into the image to add new points, drag points to new locations or delete points by clicking them with pressed Ctrl-key. After annotating all objects, turn the selections in the ROI manager into an Image Overlay by clicking "Image→Overlay→From ROI manager". Check "Labels are classes" during transfer learning.

3D Process all slices of your image as described in the 2D case.

SN 2.2.3 Mask-based image annotation for instance-aware semantic segmentation

Mask-based image annotation is the most common way of annotating images. Annotated data in public repositories usually come with mask annotations, due to their simplicity and the possibility to generate mask images with many popular image processing tools. ImageJ's core support for generating mask images is comparably poor because painting of integer labels is not supported by the UI of core ImageJ. Here a third-party plugin, the Segmentation Editor (Plugins→Segmentation→Segmentation Editor) comes to the rescue. Please refer to the tutorial of the authors of the plugin for a detailed description on how to use it (https://imagej.net/Segmentation_Editor). The segmentation editor opens a second window that shows the generated segmentation masks live.

Important remark: The U-Net segmentation plugin treats all pixels with value zero as "ignore" regions, pixels with value one are background, pixels with values greater than one are foreground objects. This differs from publicly available annotations and the Segmentation Editor which have no notion of ignored regions. There, a value of zero indicates background and values greater than zero mark foreground objects. You can still train the U-Net with these annotations, but have to add a value of 1 to all slices of the mask image using "Process→Math→Add..." before adding the masks as Overlay to the raw image. Please make sure that annotations from a public source are fully annotated, especially in the ISBI Cell Tracking Challenge this is not guaranteed. In order not to introduce false negatives during training add ignore regions over cells without annotation.

2D foreground/background segmentation Rename the first class (Exterior) to "ignore" and change its color to black. Rename the second class (Interior) to "background" and change its color to mid gray. Select the part of the image you want to annotate (all if you want to annotate the whole image) and add the selected region to the background class. Now use the ImageJ ROI tools to annotate the cells. Give each cell a unique label for instance-aware semantic segmentation. When all cells are annotated, click "OK" and embed the mask annotations into the raw image using "Plugins→U-Net→Utilities→Embed mask annotations" and uncheck "Labels are classes" during transfer learning.

2D multi-class segmentation Rename the first class (Exterior) to "ignore" and change its color to black. Rename the second class (Interior) to "background" and change its color to mid gray. Add new labels for each class you want to annotate. Select the part of the image you want to annotate (all if you want to annotate the whole image) and add the selected region to the background class. Now use the ImageJ ROI tools to annotate the objects of the different classes and add them to the corresponding class mask. When all objects are annotated, click "OK" and embed the mask annotations into the raw image using "Plugins→U-Net→Utilities→Embed mask annotations" and check "Labels are classes" during transfer learning. Instance information cannot be encoded in this case, because the label is needed for the class information. Ensure that a gap of at least one pixel remains between different instances.

SN 2.3 Saving annotations

After embedding the annotations into the raw image you can store the prepared training samples as TIFF. ImageJ will store both raw image and annotations to the same file, so they can be easily transfered and used again later. Don't use other formats like PNG or JPEG, otherwise your annotations will be "burnt" into the raw image and you won't be able to extract them!

Supplementary Note 3 Experimental validation on real-world applications

SN 3.1 Cell detection, classification, and counting

Cell counting is typically achieved by manually marking the cell/nucleus centers on a small image set and statistically evaluating the resulting counts. In screening experiments with control populations and several replicates, manual annotation is time consuming and often results are reported only for a selected (ideally representative) subset of the recorded data.

The U-Net was tested in two such screening experiments – one co-localization experiment in the context of optogenetics and one proliferation quantification experiment in the context of wound healing. In both cases, a hand-annotated dataset was created and a subset of this was used to train the U-Net. After training, the U-Net processed all remaining images of the dataset for model evaluation.

SN 3.1.1 Co-localization analysis in optogenetics (2D)

Sample preparation In order to histologically quantify the opsin expression as part of a typical optogenetic experiment, we stereotactically injected 5 wild type rats at several different sites with three different constructs (Tab. SN3.1). These constructs included the membrane channel channelrhodopsin-2 (ChR2)[15], which activates neurons when driven with blue light and the chloride pump enhanced Natronomonas pharaonis halorhodopsin (eNpHR3.0)[16], which inhibits spiking when driven with yellow or green light. The coexpression of opsin and fluorescent protein leads to strong labeling not only of the cell soma but also dendrites and axons, rendering the detection of opsin expressing cell bodies difficult. Thus, we added a 2A self-cleaving peptide (2AP)[17] in one of the constructs, in order to separate the expression of opsin and fluorescent protein, which simplifies the detection of opsin expressing cell bodies. Our promoter choices included human Synpasin (hSyn), which has been implicated in the regulation of neurotransmitter release at synapses, particularly at glutamatergic and GABAergic synapses[18], and has been described to label excitatory as well as inhibitory neurons, and Ca2+/calmodulin-dependent protein kinase II (CamKII), which is involved in many signaling cascades and has been described as targeting excitatory neurons[19]. As viral vectors, we chose adeno-associated virus (AAV) serotype 2 pseudotyped with serotype 5 (here referred to as AAV5). We injected 1μl of virus at several sites (Tab. SN3.1). After an expression time of 4-5 weeks, we used standard histological techniques to determine expression patterns in cryosections of 40μm. All sections were co-labeled with a monoclonal antibody staining adding a second fluorescent channel in red. Image were recorded with a ZEISS LSM510 Meta mit Axio Imager.Z1 Stativ equipped with a ZEISS LD LCI Plan-Apochromat 25x/0.8 Imm Corr DIC M27 objective using ZEN 2010 B SP1 Version 6.0.0.485. Pixel size: 0.17μm/px.

Approval of animal experimentation Animal experimentation was approved by the Regional Council of Freiburg (G15/11).

Image Annotation and label generation The overall dataset consists of 12 images, each containing a red channel showing the antibody stain and a green channel showing eYFP expression. An expert annotated cells in rectangular regions of all images (Annotator 1). To estimate human annotation variance, two additional experts (Annotator 2 and 3) annotated subsets of these image regions. In total, five images were annotated by all three annotators, two images only by Annotator 1 and 2 and the residual five only by Annotator 1. Experts annotated cells in the red and green channels separately by placing one roughly centered point in nuclei of cells showing signal in the corresponding channel using ImageJ's Multi-Point tool. We denote the generated point sets as G (green channel) and R (red channel). Cells annotated in both channels with a distance of less than $d_{\text{coloc}} = 10$px (3.4 μm) were treated as co-localization. In case of co-localization, both points were removed from their corresponding point sets and their average position added to a third point set C. From these three sets, ground-truth labels were generated using

$$y(\mathbf{x}) := \begin{cases} 0 & \forall \mathbf{p} \in G \cup R \cup C : \|\mathbf{x} - \mathbf{p}\| > r_{\text{disk}} \\ 1 & \exists \mathbf{p} \in G : \|\mathbf{x} - \mathbf{p}\| \leq r_{\text{disk}} \\ 2 & \exists \mathbf{p} \in R : \|\mathbf{x} - \mathbf{p}\| \leq r_{\text{disk}} \\ 3 & \exists \mathbf{p} \in C : \|\mathbf{x} - \mathbf{p}\| \leq r_{\text{disk}} \end{cases}$$

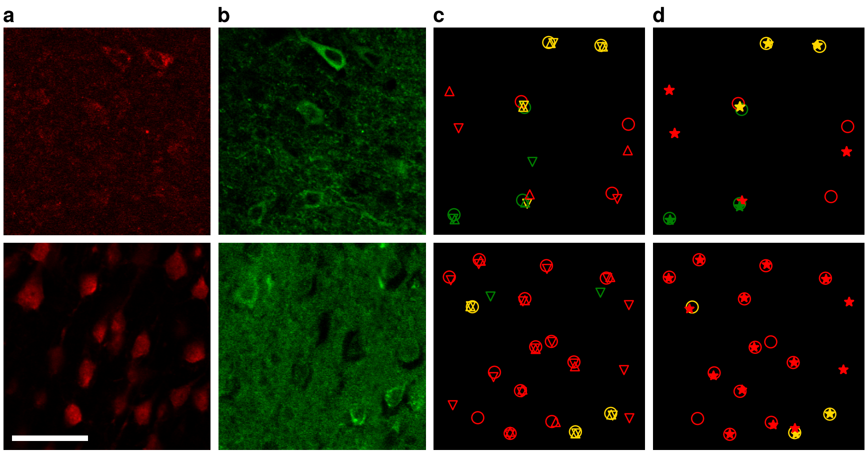
with disk radius $r_{\text{disk}} = 5$px (1.7μm).

Weight computation parameters: $\lambda = 0$ (no instance separation), $v_{\text{bal}} = 0.1$, $\sigma_{\text{bal}} = 0$px, radius of the ignored region around each annotated point $r_{\text{ign}} = 15$px (5.1μm).

Data augmentation We applied random rotation drawn from a uniform distribution in the full 360° range and elastic deformation with random seed displacement vectors drawn from a Gaussian distribution with standard deviation 10px along each dimension. Seed grid point distance: 100px.

Training We trained the U-Net on annotations of annotator 1 and 2 on the subset of images containing annotations from both. We left out the residual five images from training to quantify the performance of U-Net on new images. Training from scratch and transfer learning were performed at original resolution (pixel size: 0.34×0.34 μm/vx).

We trained one network from scratch (U-Net$_s$) and in a second experiment applied transfer learning to adapt the generic



e

Detection method		Reference Annotation Annot. 1	Annot. 2	Annot. 3
Annot. 1		1	0.73	0.63
Annot. 2		0.73	1	0.58
Annot. 3		0.63	0.58	1
U-Net$_s$	rep 1	0.69	0.66	0.56
	rep 2	0.66	0.63	0.55
	rep 3	0.66	0.63	0.56
U-Net$_f$	rep 1	0.62	0.56	0.49
	rep 2	0.62	0.56	0.48
	rep 3	0.61	0.57	0.46

Figure SN3.3: Comparison of manual annotation and automatic detection of opsin-expressing cells. **(a–d)** Top row: Image 8 (see Tab. SN3.1); Bottom row: Image 6. **(a)** Antibody stain (red channel); **(b)** eYFP expression (green channel). **(c)** Comparison of independent annotations of three experts; ◯: Annotator 1; △: Annotator 2; ▽: Annotator 3. Marker color indicates kind of signal. Red and green: Cell shows only signal in the channel with corresponding color; Yellow: co-localization, i.e. cell shows signal in both channels. **(d)** Detection result of the U-Net ($\star$) compared to annotator one (◯). **(e)** Quantitative comparison. Agreement among three independent human experts (Annot. 1–3), a U-Net trained from scratch (U-Net$_s$), and a generic U-Net pre-trained for 2D cell segmentation and adapted to the co-localization detection task (U-Net$_f$). Scores show the average F-measure with matching tolerance $d_{\mathrm{match}} = 5$px (1.7μm) on five test images. Network training was repeated three times (rep 1-3) leading to comparable results. Scale bar: 50μm.

Table SN3.1: Description of the samples used in the optogenetic 2D co-localization experiments.

Image	Vector and virus titer (vg/ml) (green)	Antibodies (red)	Area	Rat	Injected brain area (volume)	Expression time (weeks)
5	hSyn-ChR2-eYFP	NeuN – AlexaFluor647	Stratium	1	Striatum; AP 1; ML 3,4; DV -4,5 (2μl)	5
6	hSyn-ChR2-p2A-eYFP ($1.86 \cdot 10^7$)	NeuN – AlexaFluor647	Cortex	2	M1; AP 1,7; ML 2,5; DV -1,5 (1μl)	4
7	CamKII-eNpHR-eYFP ($5.2 \cdot 10^{12}$)	CamKII – AlexaFluor647	Cortex	3	M1; AP 1,7; ML 2,5; DV -1,5 (1μl)	4
8	CamKII-eNpHR-eYFP ($5.2 \cdot 10^{12}$)	CamKII – AlexaFluor647	Cortex	3	PFC; AP 3,2; ML 0,6; DV -3,5 (1μl)	4
9	CamKII-eNpHR-eYFP ($5.2 \cdot 10^{12}$)	CamKII – AlexaFluor647	Cortex	3	M1; AP 1,7; ML 2,5; DV -1,5 (1μl)	4
10	CamKII-eNpHR-eYFP ($5.2 \cdot 10^{12}$)	CamKII – AlexaFluor647	Cortex	3	M1; AP 1,7; ML 2,5; DV -1,5 (1μl)	4
11	hSyn-ChR2-p2A-eYFP ($9.89 \cdot 10^9$)	PV – AlexaFluor647	Cortex	4	M1; AP 1,7; ML 2,5; DV -1,5 (1μl)	4
12	hSyn-ChR2-p2A-eYFP ($9.89 \cdot 10^9$)	PV – AlexaFluor647	Stratium	4	Stratium; AP 1; ML 3,4; DV -4,5 (1μl)	4
13	hSyn-ChR2-p2A-eYFP ($9.89 \cdot 10^9$)	PV – AlexaFluor647	Cortex	4	M1; AP 1,7; ML 2,5; DV -1,5 (1μl)	4
14	hSyn-ChR2-p2A-eYFP ($9.89 \cdot 10^9$)	PV – AlexaFluor647	Cortex	4	M1; AP 1,7; ML 2,5; DV -1,5 (1μl)	4
17	CamKII-eNpHR-eYFP ($5.2 \cdot 10^{12}$)	CamKII – AlexaFluor647	Cortex	5	PFC; AP 3,2; ML 0,6; DV -3,5 (1μl)	4
19	CamKII-eNpHR-eYFP ($5.2 \cdot 10^{12}$)	CamKII – AlexaFluor647	Cortex	5	M1; AP 1,7; ML 2,5; DV -1,5 (1μl)	4

2D cell segmentation model (Sec. SN 3.2.1) to this detection task (U-Net$_f$).

U-Net$_s$: Input patch size: 508×508 (2 channels); Solver: Stochastic gradient descent (SGD); Base Learning rate: 10^{-3}; Momentum: 0.9; Learning rate decay: 10^{-4} after 150000 iterations; Iterations: 300000.

U-Net$_f$: Input patch size: 508×508 (2 channels); Solver: ADAM[20]; Base learning rate: 10^{-4}; Momentum: 0.9; Momentum2: 0.999; Iterations: 30000.

Results The considered task is challenging due to the large variety in terms of cell shape, size and signal-to-noise ratio (Fig. SN3.3a,b). Inconsistencies between the expert annotations (Fig. SN3.3c) confirm this claim. We quantify annotation/detection agreement by the F-measure with matching tolerance $d_{\text{match}} = 5\text{px}$ (1.7μm) (Fig. SN3.3e). The F-measure ranges from 0 (no agreement) to 1 (perfect agreement) of two annotations (Online Methods). Two annotations within the matching tolerance agree if they share the same class label.

The agreement of U-Net with the experts is comparable to the agreement among experts (Fig. SN3.3). U-Net$_s$ performed a bit better than the generic U-Net$_f$. However, for from scratch training we changed the training schedule to better cope with noisy annotations, while in the transfer learning experiment we used the training parameters from the base-model which resembles the behaviour of the ImageJ plugin.

SN 3.1.2 Clonal analysis of confetti-labeled microglial cells during brain recovery (3D)

To understand the clonal relationship of microglia, or brain immune cells, that rapidly proliferate in response to injury, we analyzed a transgenic mouse model, "Microfetti", in which microglial cells express Confetti colors[21].

Sample preparation Unilateral facial nerve transection was performed in tamoxifen-treated Microfetti mice to induce the expression of Confetti labels in microglial cells. Microglial cell proliferation in response to the lesion occurred in the facial nucleus in the ventral pons. Contralateral facial nucleus was used as healthy control. Mouse brains were fixed at onset of injury (2 d after lesion), peak of injury (14 d after lesion) and recovery (60 d after lesion) for microglial marker IBA-1 immunohistochemistry, nuclear counterstain and confocal imaging as detailed previously[21]. In total, 301 fixed mouse brain sections from 30 animals at different stages of injury progression up to recovery were imaged.

Approval of animal experimentation Animal experimentation was approved by the Regional Council of Freiburg (G13/107).

Image annotation and label generation Semi-automatic cell detection followed by manual quality control[21] were performed for all microglial cells positive for the IBA-1 marker, as well as Confetti-labeled microglia (Fig. SN3.4a). Centers of cell nuclei of microglia cells were annotated in 2D maximum intensity projections along z using a custom MATLAB interface. All channels – a reference channel showing all IBA-1-positive microglia cells and four Confetti channels – were annotated independently. For each annotated 2D location its depth z was automatically estimated as the local maximum along the z-axis. All annotated points were put into separate point sets M (all IBA-1-positive microglia) and $C_1, \dots C_4$ for the different Confetti markers. Point annotations occurring both in M and any Confetti channel (tolerance: $d_{\text{coloc}} = 10\mu$m) were removed from M. Ground truth labels were rendered as spheres in a 3D map encoding six classes using

$$y(\mathbf{x}) := \begin{cases} 0 & \forall \mathbf{p} \in M \cup C_{1,\dots,4} : \|\mathbf{x} - \mathbf{p}\| > r_{\text{sphere}} \\ 1 & \exists \mathbf{p} \in C_1 : \|\mathbf{x} - \mathbf{p}\| \le r_{\text{sphere}} \\ 2 & \exists \mathbf{p} \in C_2 : \|\mathbf{x} - \mathbf{p}\| \le r_{\text{sphere}} \\ 3 & \exists \mathbf{p} \in M : \|\mathbf{x} - \mathbf{p}\| \le r_{\text{sphere}} \\ 4 & \exists \mathbf{p} \in C_3 : \|\mathbf{x} - \mathbf{p}\| \le r_{\text{sphere}} \\ 5 & \exists \mathbf{p} \in C_4 : \|\mathbf{x} - \mathbf{p}\| \le r_{\text{sphere}} \end{cases}$$

with sphere radius $r_{\text{sphere}} = 5\text{vx}$ at voxel size $(0.621, 0.621, 1.5)\mu$m/vx. Clones of microglial cells were identified by clusters of Confetti-labeled cells that were above the 98th percentile range of Monte Carlo simulations[21]. The images were randomly split into a training set containing 80 stacks and a test set containing the remaining 221 stacks.

Weight computation parameters: $\lambda = 0$ (no instance separation), $v_{\text{bal}} = 0.1$, $\sigma_{\text{bal}} = 0\text{vx}$, $r_{\text{ign}} = 15\text{vx}$.

Data augmentation We applied random rotation around the z-axis drawn from a uniform distribution in the full 360° range and elastic deformation with random seed displacement vectors drawn from a Gaussian distribution with standard deviations (10, 10, 0.1)px along corresponding dimensions. Seed grid point distance: 150px.

Training Input patch size: $236 \times 236 \times 100$; Solver: ADAM; Base learning rate: 10^{-5} (fixed schedule); Momentum: 0.9; Momentum2: 0.999; Iterations: 150000.

Results Injury induces proliferation of microglial cells as wound healing response. When randomly modifying some microglial cells to express fluorescent markers, daughter cells (clones) show the same fluorescence expression. Consequently, in areas of high proliferation, we observe clusters of marked cells, while in areas with low proliferation such clustering is absent. When using multiple markers of different colors, the visual impression of colored spots on dark

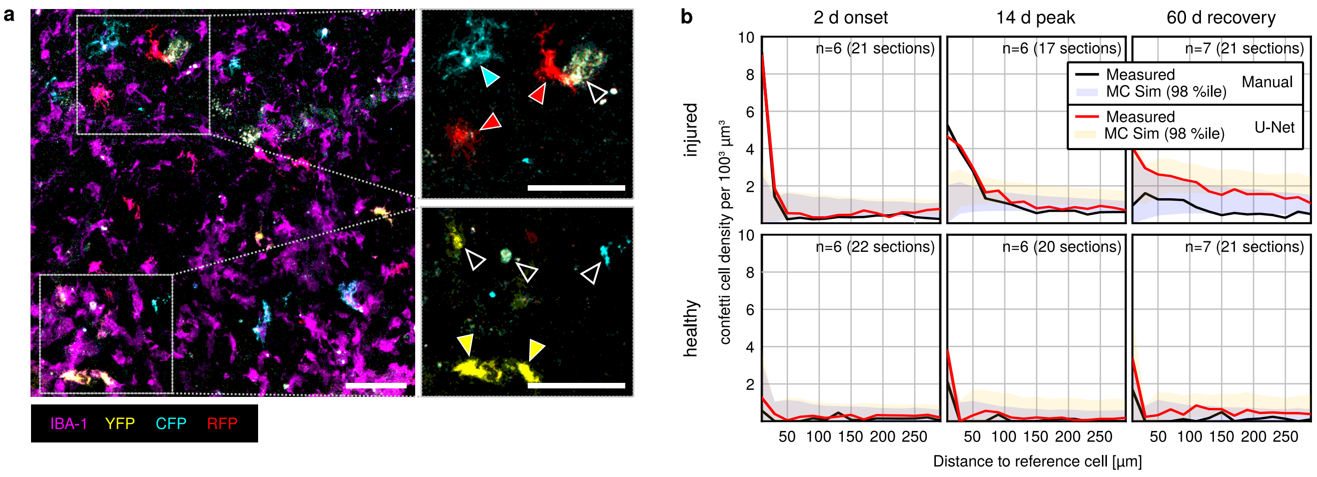


Figure SN3.4: Clonal analysis of Confetti-labeled microglial cells during neurodegeneration and recovery. **(a)** Representative maximum intensity projection of a typical "Microfetti" brain section at peak of neurodegeneration. All microglial cells are marked by IBA-1 (magenta). A small population of microglial cells express Confetti colors, namely membrane-tagged CFP (cyan) and cytoplasmic YFP (yellow) and RFP (red). Higher magnification insets show only Confetti-labeled microglia (filled arrowheads) and background noise from the tissue (open arrowheads). Scalebars: 50μm. **(b)** Clonal relationships of Confetti-labeled microglial cells in injured (top row) and healthy (bottom row) brain tissue at disease onset 2 days (d) after injury, peak of disease (14 d after injury), and recovery (60 d after injury). Bold lines: Cell densities of Confetti-labeled microglia (mean over mice in population); red: based on U-Net detections; black: based on manual annotation. Transparent areas: Monte-Carlo simulations for random microglial proliferation (10000 bootstrap iterations); yellow: based on U-Net detections; blue: based on manual annotation. n: Number of mice in population

background lead to the term Confetti marker. In our experiment we used one tag that marks all microglial cells (IBA-1) and three Confetti markers (YFP, CFP, RFP). Note, that multiple colors are only used to allow multiplexing in order to reduce the number of required images. The same experiment could be performed with only one Confetti color, but would require more images.

We trained U-Net on volumetric image stacks to detect microglial cells and, if expressing a Confetti marker, its color. For each detected Confetti-positive (reference) microglial cell we counted the number of microglial cells containing the same Confetti marker in cylindrical shells with increasing radius and normalized them by the shell volume. In case of clustering, the resulting cumulated distribution shows a peak close to the microglial diameter, indicating response to the injury. As null-hypothesis for significance tests we performed a monte-carlo experiment in which we randomly redistributed the Confetti markers among all microglial cells.

We evaluated only the test set for both U-Net and manual annotations. Both U-Net and the manual analysis came to the same conclusion that Confetti-labeled microglial clusters were only detected in the tissues from onset and peak of injury, but not in healthy and recovery stages (Fig. SN3.4b, intermediate states after 7d and 30d are not shown).

U-Net required a manual annotation effort that was four folds lower than the fully manual analysis.

SN 3.2 Cell segmentation

SN 3.2.1 Segmentation of single cells recorded with various common imaging modalities (2D)

In this section we present our generic 2D cell segmentation model that allows to segment cells of various types recorded with any of the most common light microscopy techniques. For this, we trained one U-Net on 11 different single cell datasets (Fig. SN3.5). The resulting model can be applied to new similar cell datasets out-of-the-box (Sec. SN 3.2.2). Moreover, the resulting model is the basis for easy transfer learning to new 2D segmentation or detection tasks with only few additional annotated training samples (Sec. SN 3.2.3).

Table SN3.2: Statistics for training and test sets for the eleven datasets for 2D cell segmentation. Valid pixels are all pixels that do not belong to an "ignored region", and therefore have an effect on the loss function and its gradient during optimization.

Dataset	**#images** train/test	**#cells** train/test	**#valid pixels** train/test	**Touching**
F1-MSC	51 / 4	228 / 20	24M / 1.6M	No
F2-GOWT1	50 / 4	265 / 122	2.3M / 1M	No
F3-SIM	371 / 10	9194 / 283	10M / 0.3M	No
F4-HeLa	36 / 4	1386 / 566	7M / 4.8M	No
DIC1-HeLa	18 / 6	195 / 59	0.6M / 0.2M	Yes
PC1-U373	34 / 6	211 / 33	21M / 3.7M	No
PC2-PSC	4 / 4	493 / 509	17M / 12M	Yes
PC3-HKPV	26 / 10	133 / 46	15M / 5.4M	Yes
BF1-POL	531 / 125	928 / 235	260M / 61M	No
BF2-PPL	10 / 2	659 / 124	16M / 3.1M	Yes
BF3-MiSp	1 / 1	87 / 97	0.5M / 0.5M	Yes

Sample preparation, selection and annotation Datasets F1-MSC, F2-GOWT1, F3-SIM, F4-HeLa, DIC1-HeLa, PC1-U373 and PC2-PSC are from the ISBI Cell Tracking Challenge 2015[22] (http://celltrackingchallenge.net). Please refer to the individual dataset descriptions at the corresponding challenges for more details. At the challenge in 2015 our U-Net won in DIC1-Hela, F4-HeLa, PC1-U373, and

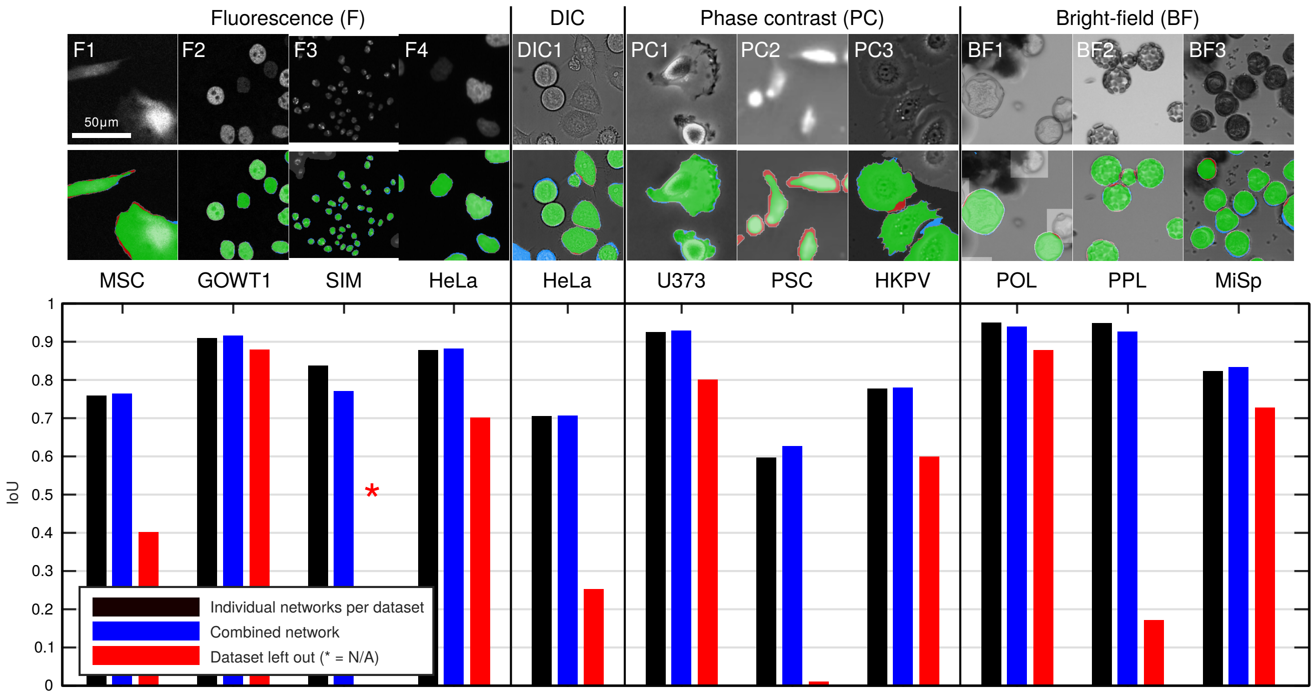


Figure SN3.5: U-Net classification performance on different cell types and imaging modalities. **(Top)** Sample images from the 11 2D cell datasets from fluorescence, brightfield, DIC, and phase contrast microscopy and corresponding segmentations using the U-Net jointly trained on all classes. green: true positive; blue: false positive; red: false negative. Scale is identical for all images. **(Bottom)** Quantitative comparison (IoU) of U-Nets trained for each dataset individually (black bars) and the U-Net trained jointly on all datasets (blue bars). Red bars: Generalization performance of the jointly trained network when leaving out the corresponding dataset from training.

got the second place in F1-MSC and F3-SIM. In the mean time other approaches could achieve better results for some datasets. However, especially our new jointly trained U-Net is among the top three for four datasets. For PC1-U373 it sets a new state-of-the-art (Tab. SN3.3).

F1-MSC (Original name: Fluo-C2DL-MSC) Rat mesenchymal stem cells on a flat polyacrylamide substrate (2D). Provided by Dr. F. Prósper. Cell Therapy laboratory, Center for Applied Medical Research (CIMA), Pamplona, Spain. **Microscope:** PerkinElmer UltraVIEW ERS; **Objective lens:** Plan-Neofluar 10×/0.3 (Plan-Apo 20×/0.75); **Pixel size:** 0.3×0.3 μm (0.3977×0.3977 μm), time step: 20 (30) min; **Images (Train/Test):** 51/4; **Objects (Train/Test):** 228/20.

F2-GOWT1 (Original name: Fluo-N2DH-GOWT1) GFP-GOWT1 mouse stem cells (2D). Provided by Dr. E. Bártová. Institute of Biophysics, Academy of Sciences of the Czech Republic, Brno, Czech Republic. **Microscope:** Leica TCS SP5; **Objective lens:** Plan-Apochromat 63×/1.4 Oil; **Pixel size:** 0.240×0.240μm, time step: 5 min; **Images (Train/Test):** 50/4; **Objects (Train/Test):** 265/122.

F3-SIM (Original name: Fluo-N2DH-SIM+) Simulated nuclei moving on a flat surface (2D). Provided by Dr. V. Ulman & Dr. D. Svoboda. Centre for Biomedical Image Analysis (CBIA), Masaryk University, Brno, Czech Republic (Created using Cytopacq). **Pixel size:** 0.125×0.125μm; **Images (Train/Test):** 371/10; **Objects (Train/Test):** 9194/283.

F4-HeLa (Original name: Fluo-N2DL-HeLa) HeLa cells stably expressing H2b-GFP (2D). Provided by the Mitocheck Consortium. **Microscope:** Olympus IX81; **Objective lens:** Plan 10×/0.4; **Pixel size:** 0.645×0.645μm, time step: 30 min; **Images (Train/Test):** 36/4; **Objects (Train/Test):** 1386/566.

DIC1-HeLa (Original name: DIC-C2DH-HeLa) HeLa cells on a flat glass surface (2D). Provided by Dr. Gert van Cappellen. Erasmus Medical Center, Rotterdam, The Netherlands. **Microscope:** Zeiss LSM 510 Meta; **Objective lens:** Plan-Apochromat 63×/1.4 (oil); **Pixel size:** 0.19×0.19μm, time step: 10 min; **Images (Train/Test):** 18/6; **Objects (Train/Test):** 195/59.

PC1-U373 (Original name: PhC-C2DH-U373) Glioblastoma-astrocytoma U373 cells on a polyacrylimide substrate (2D). Provided by Dr. Sanjay Kumar. Department of Bioengineering, University of California at Berkeley, Berkeley CA, USA. **Microscope:** Nikon; **Objective lens:** Plan Fluor DLL 20×/0.5; **Pixel size:** 0.65×0.65μm; **Images (Train/Test):** 34/6; **Objects (Train/Test):** 211/33.

PC2-PSC (Original name: PhC-C2DL-PSC) Pancreatic Stem Cells on a Polystyrene substrate (2D). Provided by Dr. Tim Becker, Fraunhofer Institution for Marine Biotechnology, Lübeck, Germany. **Microscope:** Olympus ix-81; **Objective lens:** UPLFLN 4XPH; **Pixel size:** 1.6×1.6μm, time step: 10 min; **Images (Train/Test):** 4/4; **Objects (Train/Test):** 493/509.

We used the provided training sets with manual annotations after adding ignore regions in only partially annotated datasets for network training and annotated a representative random subset of the corresponding test sets ourselves using ImageJ's regional ROI tools for evaluation. We selected 2-3 frames per sequence such that they were uniformly distributed across the sequences.

We added custom datasets PC3-HKPV, BF1-POL, BF2-PPL and BF3-MiSp.

PC3-HKPV (D. Saltukoglu and M. Simons): Keratinocytes (courtesy of Leena Bruckner-Tuderman, Department of Dermatology, University Hospital, Freiburg) isolated from human foreskin were transformed with the human papillomavirus oncogenes E6 and E7 for immortalization[23]. **Microscope:** Zeiss Cell Observer microscope controlled by AxioVision and equipped with a cooled charge-coupled device AxioCam Rev3 camera; **Objective lens:** Zeiss EC Plan Neofluar 20×/0.5 phase contrast[24]; **Pixel size:** $0.65\times0.65\mu$m, time step: 1.5h; **Images (Train/Test):** 26/10; **Objects (Train/Test):** 133/46; **Annotation:** We manually selected 36 frames from the recorded sequences. Training and test split was performed, such that training and test frames originate from different sequences. We annotated the images using ImageJ's regional ROI tools. Then ROIs were converted to instance mask images, rescaled to processing resolution, and one pixel wide gaps between touching cell instances added before binarization.

BF1-POL (O. Ronneberger): Airborne pollen grains collected, prepared and recorded using a fully automated pollen monitor[25] which was developed together with the German weather service (Deutscher Wetterdienst) and the Fraunhofer institute, Freiburg, Germany. Aerosolic particles were collected on custom plates containing glycerine gelatine. After heat shock, rehydrated particles settled on the bottom of the plate and image stacks were recorded under green (533nm) LED illumination. **Microscope:** Embedded inverted microscope (Custom design) using a CCD camera (AVT Dolphin F-145B from Allied Vision Technologies) equipped with a 2/3" sensor with 1392×1040 pixels and a cell size of $6.45\times6.45\mu$m; **Objective lens:** 20×/0.8 Zeiss Plan-apochromat; **Pixel size:** $0.3225\times0.3225\times1.5\mu$m; **Images (Train/Test):** 531/125; **Objects (Train/Test):** 928/235; **Annotation:** Spherical particles were detected in 656 volumes using a robust hough-based sphere detector. 2D segmentations in the sharpest layer of each particle were obtained using active contours (snakes). We inspected the resulting segmentations and replaced erroneous segmentations and out-of-focus pollen by rectangular regions enclosing the corresponding objects to be ignored during training. Segments containing objects that are no pollen were treated as background. Then contours were rendered to instance masks, images rescaled to processing resolution, and one pixel wide gaps between touching pollen added before binarization.

BF2-PPL (A. Dovzhenko, S. Walsh, O. Tietz, C. D. Bosco and K. Palme): Tobacco (Nicotiana tabacum) mesophyll protoplasts were isolated and immobilized in alginate hydrogels according to[26]. The globally sharpest image was extracted from a recorded bright-field stack with 1μm z-step using edge-based autofocus. **Microscope:** Automated inverted microscope platform MORE (TILL I.D. GmbH, Planegg, Germany) equipped with an AVT-Stingray F-145 camera (Allied Vision, Germany); **Objective lens:** Zeiss 10×/0.45 objective; **Pixel size:** $0.51\times0.51\mu$m; **Images (Train/Test):** 10/2; **Objects (Train/Test):** 659/124; **Annotation:** We annotated 10 training and 2 test images in ImageJ using the region-based ROI tools. Then ROIs were converted to instance mask images, rescaled to processing resolution, and one pixel wide gaps between touching cell instances added before binarization.

BF3-MiSp (A. Dovzhenko, S. Walsh and K. Palme): Tobacco (N. tabacum) microspores were isolated and cultured according to[27]. Image series acquisition was performed as for BF2-PPL. We incubated the microspores in Carboxyfluorescein diacetate (CFDA) 30 minutes prior to recording. We recorded CFDA accumulation in living microspores using structured illumination fluorescence microscopy. **Microscope:** Automated inverted microscope platform MORE (TILL I.D. GmbH, Planegg, Germany) equipped with an AVT-Stingray F-145 camera (Allied Vision, Germany); **Objective lens:** Zeiss 20×/0.8 objective; **Pixel size:** $0.27\times0.27\times1\mu$m; **Images (Train/Test):** 1/1; **Objects (Train/Test):** 87/97; **Annotation:** The globally sharpest image was extracted from two recorded bright-field stacks using edge-based autofocus and annotated manually using ImageJ's regional ROI tools. Then ROIs were converted to instance mask images, rescaled to processing resolution, and one pixel wide gaps between touching cell instances added before binarization.

Weight computation parameters: $\lambda = 50$, $v_{\text{bal}} = 0.1$, $\sigma_{\text{bal}} = 10$px, $\sigma_{\text{sep}} = 6$px.

Data augmentation We applied random rotation drawn from a uniform distribution in the 360° range and elastic deformation with random seed displacement vectors drawn from a Gaussian distribution with standard deviation 10px along each dimension. Seed grid point distance: 150px.

Training Training was performed on the training images for our $N = 11$ datasets. We normalized the image resolution to pixel extents of 0.5μm $\times$ 0.5μm and the image intensities to $[0, 1]$. We presented single images of the different datasets to the network in an interleaved order to ensure that the network optimizes the segmentation for all datasets in parallel.

Input patch size: 540×540, Solver: ADAM, Base learning rate: 10^{-5} (fixed schedule), Momentum: 0.9, Momentum2: 0.999, iterations: 150000.

Table SN3.3: Ranking of our U-Net variants in the cell tracking challenge leader board w.r.t. the SEG measure (http://celltrackingchallenge.net) (2018/10/16). Only challenge results we participated in are shown. Colored cells: Results of our U-Net variants. Uncolored cells: Results of other participants. Top row: Dataset names. Second row: Total number of participants for each dataset. Rank 1 – Rank 3: Leader board excerpt; Additional rows: Results not among the top three. The corresponding submission IDs are given in the legend.

Dataset	F1-MSC	F3-SIM	F4-HeLa	DIC1-HeLa	PC1-U373	PC2-PSC
	17	18	18	9	10	13
Rank 1	0.645	0.811	🟩 0.903	0.814	🟦 0.924	0.682
Rank 2	🟦 0.617	0.807	🟦 0.902	0.793	0.922	0.665
Rank 3	0.590	0.802	0.900	0.792	🟩 0.920	🟦 0.633
	🟩 0.582	🟩 0.781		🟩 0.776		🟩 0.536
		🟦 0.581		🟦 0.755		

Legend: 🟩 **FR-Ro-GE**: Individual U-Nets (original resolution)[28]
🟦 **FR-Fa-GE**: U-Net trained on all classes (0.5μm/px)

We use drop-out with probability 0.5 when entering and exiting the 'innermost' network layer, *i.e*. the one with the coarsest spatial resolution.

We report segmentation performance as object-wise *intersection over union* (IoU) (Online Methods). IoU evaluates the overlap between the ground truth for each cell and the segmentation computed by the network. A value of 0 corresponds to no overlap at all, a value of 1 to a perfect pixel-by-pixel match. Values above 0.8 indicate a very good segmentation.

SN 3.2.2 Out-of-the-box cell segmentation

Joint training of U-Net on all 11 datasets (Fig. SN3.5 blue bars) yields similar results compared to multiple individual U-Nets trained separately on each of these datasets (Fig. SN3.5 black bars).

A possible use case for the U-Net jointly trained on all 11 datasets is to directly process new cell data that is similar enough to one of the training datasets. In such use case, there is neither a need to collect and annotate any new training data nor to run the training procedure. The images to be analyzed must be loaded in ImageJ and the U-Net plugin will provide the cell segmentations for these images.

To quantitatively compare our results to other approaches, we submitted segmentation results of our jointly trained U-Net to the Cell tracking challenge (http://celltrackingchallenge.net) and let it evaluate on six datasets (Tab. SN3.3).

For all experiments in this work images were rescaled to an isotropic element size of 0.5μm per pixel. For some datasets this halves the original image resolution. Surprisingly, results are still on par with individual U-Nets trained at original image resolution except for the simulated dataset F3-SIM which is of no practical importance.

Note that these results are not directly comparable to the ones in Fig. SN3.5 for which we tested on our own annotated subset of the official test set only. Additionally, SEG measure and our IoU measure are slightly differently defined. Both measure the average intersection over union of the ground truth segmentation of all objects and their corresponding predictions. However, the SEG measure assigns a score of zero for IoU values below 0.5, while our IoU measure does not. Thus the SEG measure is slightly more strict.

To also explore the generalization capabilities of our model, we performed a *leave-one-dataset-out* cross-validation experiment on the 10 non-synthetic datasets, *i.e*. the network was trained on all except one dataset, and the held-out dataset was used for testing (Fig. SN3.5 red bars). The network generalized well to datasets F2-GOWT1, F4-HeLa, PC1-U373, BF1-POL, and BF3-MiSp. Datasets DIC1-HeLa, PC2-PSC, and BF2-PPL showed poor segmentation quality in this use case. This is because datasets PC2-PSC and BF2-PPL differ too much in terms of cell appearance from the other datasets. Dataset DIC1-HeLa is the only DIC dataset. If it was removed from the training set, the network had never seen DIC images; hence, it failed to segment them correctly.

Analysis and treatment of false positive segments

The Object IoU metric does not account for false positive segments, therefore we present an evaluation including false positives here.

We observe that small impurities in the background are sometimes erroneously classified as cells. The reason is that training data contained almost no background clutter leaving the network with only very few training examples of non-cell objects that should be classified as background. More training data would also increase the amount of "training clutter" and allow the network to learn to distinguish cells from clutter. However, in many cases a simple post-processing of the segmentation allows to reach a similar result without additional annotations.

We compared the area statistics from predictions of U-Net jointly trained on all 11 datasets, with the area statistics from predictions of individually trained U-Nets, i.e. one U-Net per dataset (Fig. SN3.6). Statistics are very comparable, except for dataset BF1-POL, where the individually trained U-Net produces considerably less false positives. This is due to non-pollen objects that were labeled as background, but are visually similar to cells from other datasets.

Most false positive segments are substantially smaller than cells in the respective datasets and can be identified using a simple area threshold. We decided to set this threshold individually for each dataset based on the area statistics of the labeled cells in the training set (Fig. SN3.6). A suitable threshold is the break-even point at a recall of 0.91 with a precision of 0.94 which is obtained at a value such that 95% of the cells in the training set are accepted. See Sec. SN 3.2.3 for individual threshold values per dataset.

SN 3.2.3 Transfer Learning

The U-Net encodes its knowledge about the appearance of cells in the weights of its filters. Due to the joint training on

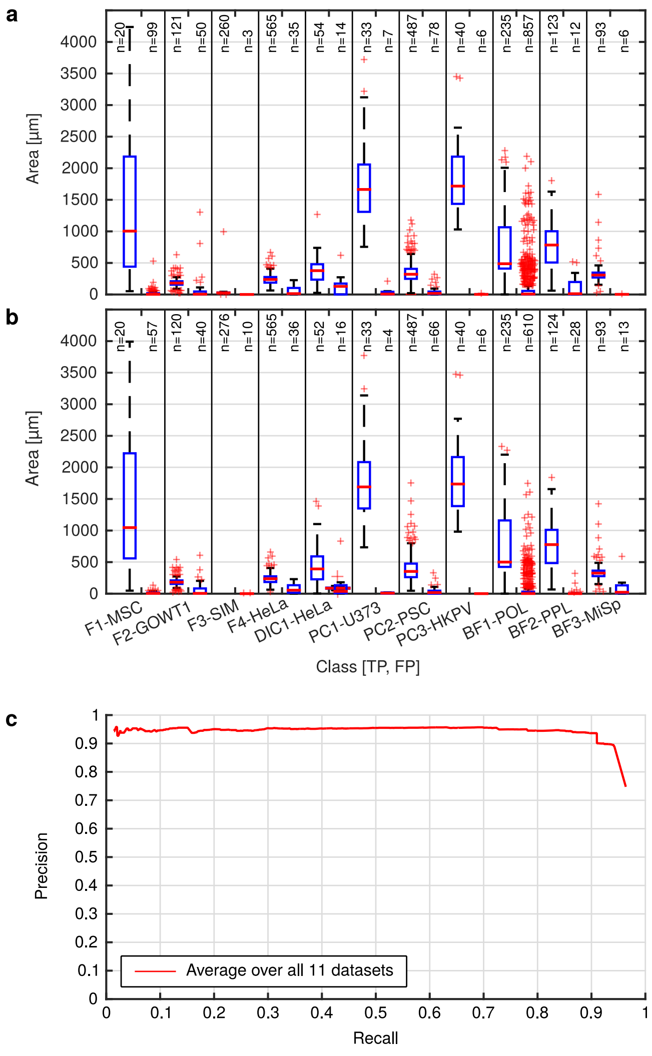


Figure SN3.6: Comparison of area-distributions of accepted and rejected segments generated using **(a)** the U-Net trained jointly on all datasets and **(b)** multiple U-Nets trained individually for each dataset. Blocks: Cell datasets; Boxplots in each block: (left) accepted segments when comparing to ground truth, (right) rejected segments when comparing to ground truth. For each boxplot: central mark: median; Box: 25th to 75th percentiles; whiskers extend to the most extreme data points within 1.5× IQR; crosses: outliers. **(c)** Precision-Recall diagram for different area thresholds in the post-processing of segmentation candidates. n indicates the number of true positive / false positive segments respectively.

different datasets, it has learned about the appearance variation of cells in general, but only within the range seen during training. For cell appearances outside that range, U-Net offers the possibility to be adapted by *transfer learning* on few annotated images of the new dataset. The provided ImageJ interface automatically prepares and hands over training images and their annotations to the training software. Thanks to the adaptation on task-specific data, U-Net is expected to yield much more reliable results than in the use case without transfer learning.

To verify this claim, the models from the leave-one-out experiment were finetuned on subsets of the training set of the held-out dataset. We measured the segmentation quality for different training set sizes and for different numbers of transfer learning iterations. We distinguish between touching cells (DIC1-HeLa, PC2-PSC, PC3-HKPV, BF2-PPL, BF3-MiSp) and non-touching cells (F1-MSC, F2-GOWT1, F4-HeLa, PC1-U373, BF1-POL). For non-touching cells it was possible to annotate only some of the cell instances in the images. Depending on the dataset, one image contained between 5 and 120 cells. Touching cells require at least one fully annotated image for training. Therefore, we note the number of labelled images instead of the number of cells for those datasets.

The segmentation performance for datasets with non-touching cells was measured after transfer learning (Fig. SN3.7a,d). Experiments were performed with a subset of 1, 10, 100, and with all available annotated cells. For comparison, the performance without transfer learning (0 annotated cells) is also shown. We report the mean IoU and its variance over ten repetitions of each experiment with a randomly picked subset of annotated cells. The quality generally improved as more training samples were added, yet for some datasets the quality saturated already after very few samples. The same analysis was done for the datasets with touching cells (Fig. SN3.7b,e). Already a single training image increased the segmentation quality to an acceptable level above 0.5 IoU on all datasets.

A critical issue of deep learning in case of few training samples is overfitting, where the network learns to represent the training samples but its performance on new samples decreases. Since transfer learning uses very few samples, we measured after how many iterations the network started to overfit (Fig. SN3.7c). Up to 1000 iterations, the performance increased or saturated even in cases with very few training samples. After 1000 iterations, overfitting effects became visible in cases with very few annotated cells (10 cells).

Transfer learning must be stopped before severe overfitting effects occur. The number of stopping iterations for a certain number of annotated samples can be derived roughly from the above results (Fig. SN3.7c). More accurate, dataset-specific stopping times can be obtained with few additional annotated validation images, which are not used as training samples but for measuring and monitoring the performance of the model during training.

We trained U-Net for $600k$ iterations and evaluated the model every $10k$ iterations on the test sets of the eleven datasets (Fig. SN3.8). Peak performance was reached at $150k$ training iterations. After more than $150k$ iterations training loss steadily decreased which indicates slight overfitting to the training set.

To show the benefit of a pre-trained model, we compared training times and segmentation performance when training from scratch (Fig. SN3.9) or adapting a pre-trained U-Net using transfer learning (Fig. SN3.7). Training from scratch requires at least $10 - 50k$ iterations for satisfactory performance, which is one order of magnitude more than in the transfer learning setting.

Transfer learning details for single datasets This section summarizes the performed *leave-one-out* experiments. We first analyze the *out-of-the-box* performance, *i.e.*

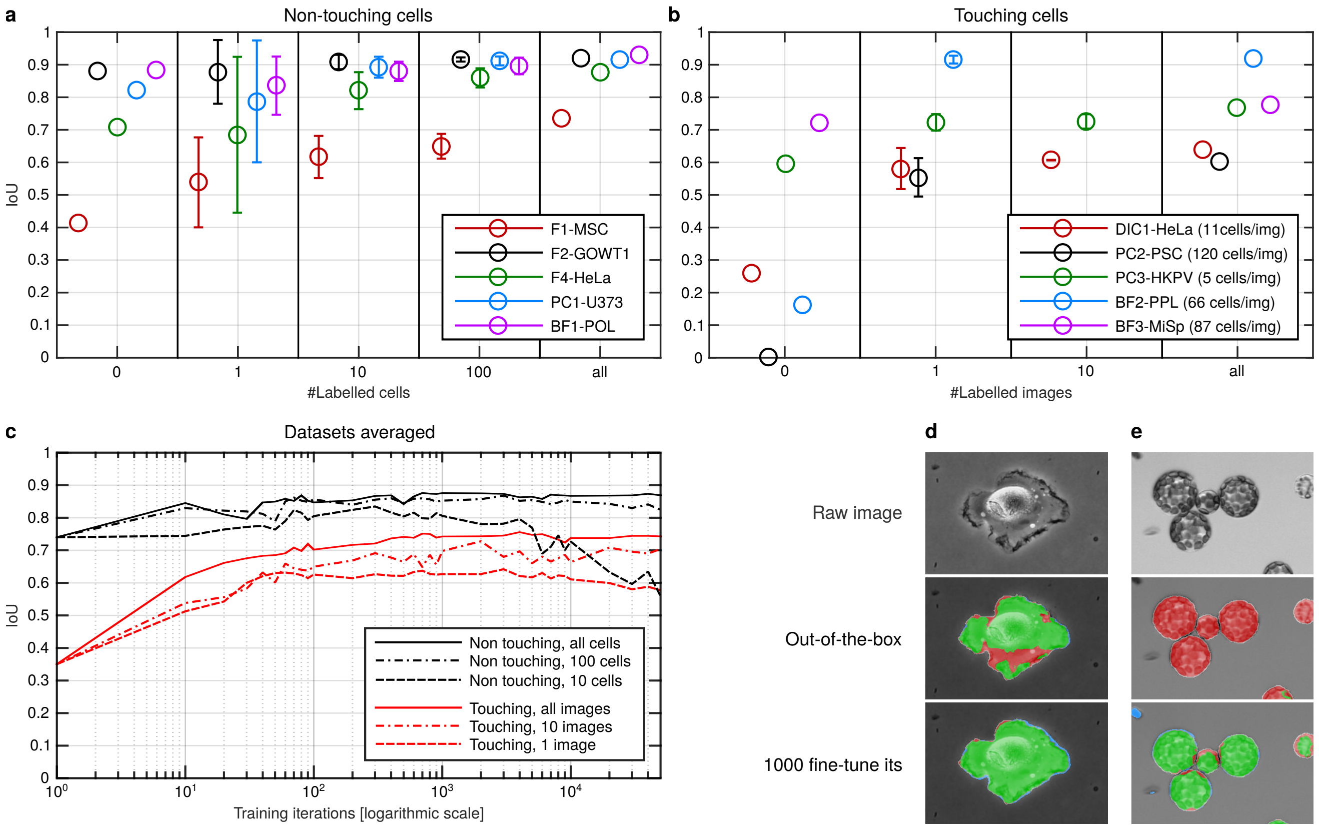


Figure SN3.7: Leave-one-dataset-out transfer learning results. **(a–b)** Segmentation quality after 1000 iterations of transfer learning for different numbers of annotated cells. Circles show the average IoU for 10 training runs with randomly chosen training data. Error bars: standard deviation. Column '0': performance of U-Net without transfer learning; Column 'all': all available training data were used for transfer learning; Missing values indicate too few training images (PC2-PSC: 4 training images, BF2-PPL: 10 training images, BF3-MiSp: 1 training image). **(c)** Segmentation quality evolution during transfer learning over training iterations. **(d–e)** Qualitative examples for the non-adapted U-Net (out-of-the-box) and the U-Net after transfer learning. Green: true positive; blue: false positive; red: false negative. **(d)** PC1-U373. **(e)** BF2-PPL.

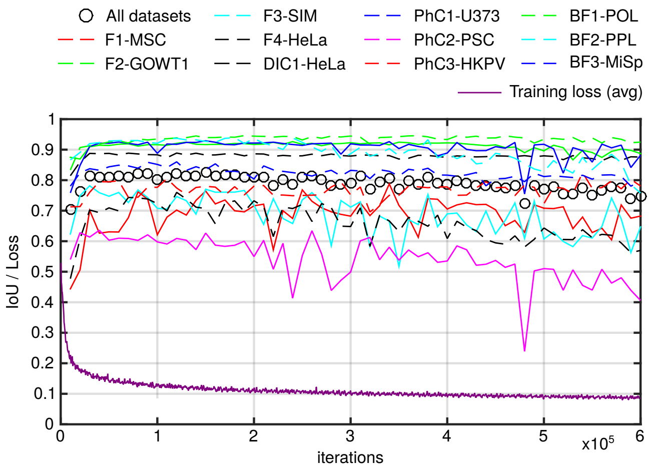


Figure SN3.8: Quantitative evaluation of U-Net model evolution during training. **curves at top**: Average object IoU for the eleven classes evaluated every 10000 iterations. **Curve at bottom:** Running average filtered training loss with a window size of 1001 iterations.

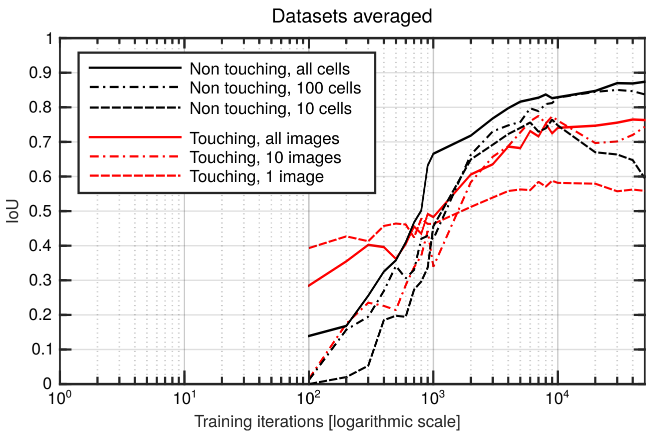


Figure SN3.9: Segmentation performance for 50k iterations when training from scratch measured as object-wise IoU, averaged over touching- and non-touching datasets.

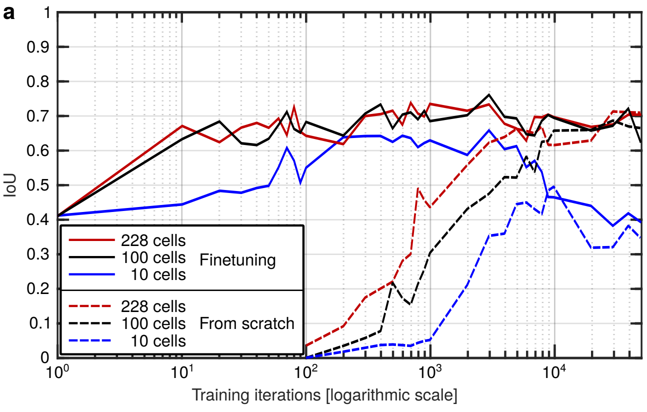
how accurate an unseen dataset can be segmented if the network was trained on ten different cell datasets (Tab. SN3.2). For this, we trained ten U-Nets, each leaving out one of the ten real 2D cell datasets (transfer to simulated dataset F3-SIM is of no practical relevance) for 200k iterations.

We then investigate the effect of *transfer learning* on the left-out dataset with different amounts of annotated cells or images as follows: For datasets with non-touching cells, *i.e.* single cell instances are separated by background, we randomly select 10, 100, or all available cells from the pool of labeled cell instances. The remaining cell instances are masked out using ignore regions, such that they have no effect on the loss function during optimization. For datasets with touching cells we randomly select 1, 10, or all available *images* from the respective dataset because we need annotated clusters of cells for training cell instance separation. For touching datasets with less than 11 training images, we skipped the training set size with 10 images.

We investigate first, the minimum amount of training data necessary to achieve satisfactory results, and secondly the saturation point, *i.e.* when adding more training data does not lead to an increase in performance. Both of these values strongly depend on the complexity of the examined cell dataset.

For every dataset we give a quantitative analysis of the evolution of the segmentation performance by means of the object IoU, corresponding recall and precision and one qualitative example (Fig. SN3.10–SN3.19). Note that the leftmost point in each line plot is the *out-of-the-box* performance. We also compare to a training from scratch with randomly initialized network weights. For the qualitative examples we picked the image patch with the highest fraction of foreground pixels from the test set.

a



b

# cells	# iterations 0	100	1k	10k
10	(0.35,0.30)	(0.65,0.57)	(0.75,0.71)	(0.55,0.52)
100	(0.35,0.30)	(0.80,0.76)	(0.80,0.76)	(0.95,0.90)
228	(0.35,0.30)	(0.75,0.71)	(0.85,0.81)	(0.85,0.85)

c

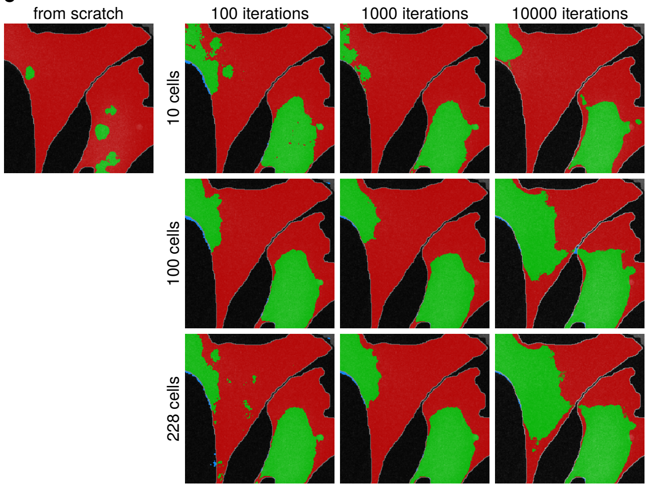
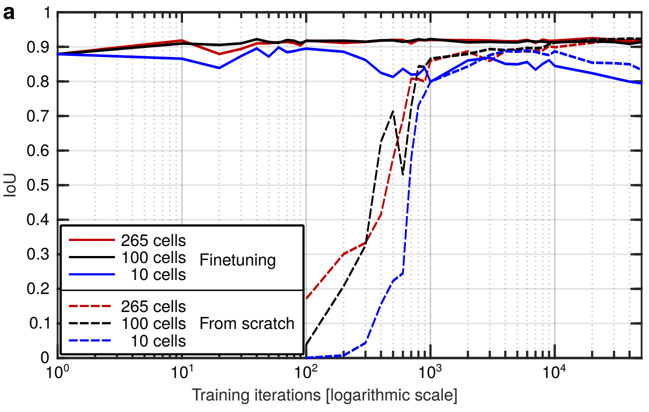


Figure SN3.10: F1-MSC; Segmentation performance after transfer learning with different numbers of cells. **(a)** IoU for transfer learning with different numbers of annotated cells compared to training from scratch; **(b)** (recall, precision) for $T = 827$ pixels which equals 0.95 of training set accepted; Segments with IoU < 0.5 are considered false positives; **(c)** Qualitative example. Green: true positive; red: false negative; blue: false positive.

F1-MSC, Non Touching (Fig. SN3.10) The **out-of-the box performance** is quite low with an IoU of ∼0.4. This is most likely due to the low signal in the raw data, the unusually large and complex structure of the cells, and their uniqueness in appearance compared to the other ten datasets in our training set. **Transfer learning** benefits from many annotated instances while the peak performance is reached after about 3k training iterations with IoU 0.75. For longer training times an overfitting to the training set can be observed when applying transfer learning with 10 labeled cells only. With only few annotated cells, training times of less than 1000 iterations yield higher peak performance compared to a training from scratch.

a



b

# cells	0	100	1k	10k
		# iterations		
10	(0.98,0.94)	(0.97,0.96)	(0.89,0.89)	(0.92,0.91)
100	(0.98,0.94)	(0.99,0.98)	(0.99,0.98)	(0.98,0.98)
265	(0.98,0.94)	(0.99,0.98)	(0.99,0.98)	(0.98,0.97)

c

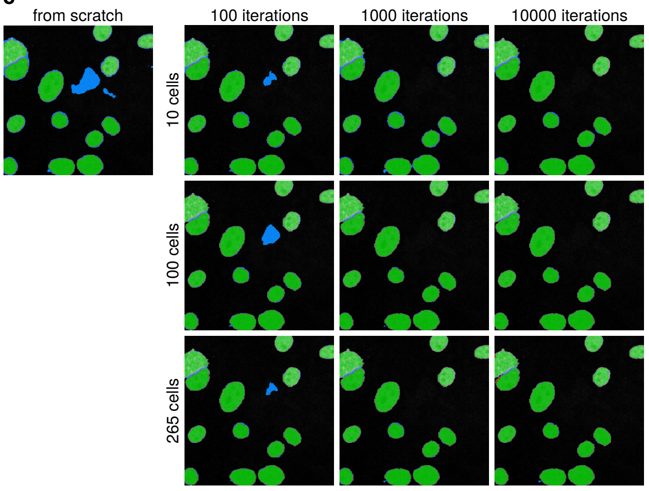
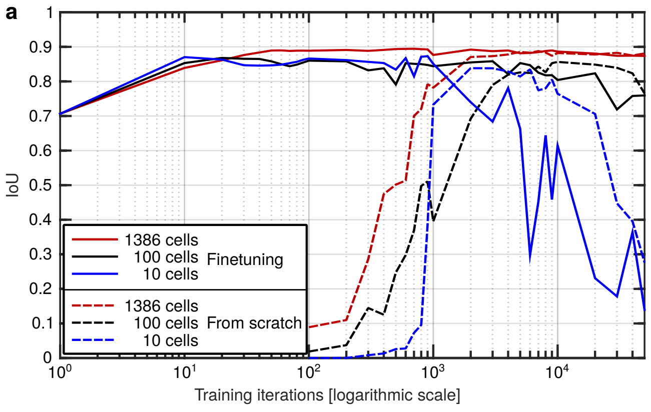


Figure SN3.11: F2-GOWT1; Segmentation performance after transfer learning with different numbers of cells. **(a)** IoU for transfer learning with different numbers of annotated cells compared to training from scratch; **(b)** (recall, precision) for $T = 496$ pixels which equals 0.95 of training set accepted; Segments with IoU < 0.5 are considered false positives; **(c)** Qualitative example. Green: true positive; red: false negative; blue: false positive.

F2-GOWT1, Non Touching (Fig. SN3.11) The **out-of-the box performance** is quite high with an IoU of 0.88 due to the visual similarity to F3-SIM and F4-HeLa. **Transfer learning** leads to marginal improvements. For longer training times slight overfitting to the training set can be observed when applying transfer learning with 10 labeled cells only.

a



b

# cells	0	100	1k	10k
		# iterations		
10	(0.79,0.78)	(0.96,0.96)	(0.95,0.95)	(0.69,0.69)
100	(0.79,0.78)	(0.96,0.96)	(0.94,0.94)	(0.90,0.90)
1386	(0.79,0.78)	(0.99,0.99)	(0.99,0.98)	(0.99,0.98)

c

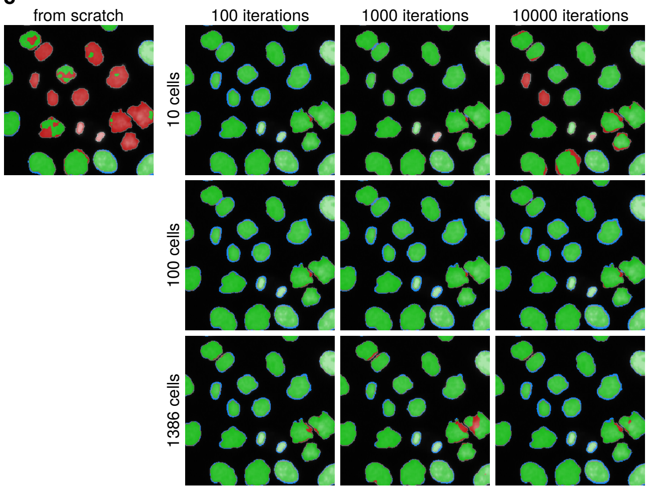
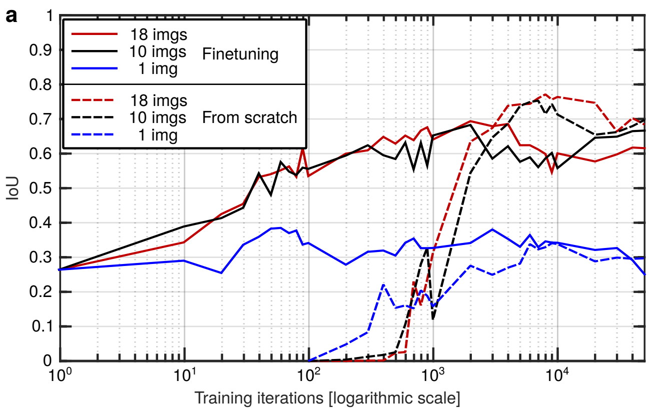


Figure SN3.12: F4-HeLa; Segmentation performance after transfer learning with different numbers of cells. **(a)** IoU for transfer learning with different numbers of annotated cells compared to training from scratch; **(b)** (recall, precision) for $T = 379$ pixels which equals 0.95 of training set accepted; Segments with IoU < 0.5 are considered false positives; **(c)** Qualitative example. Green: true positive; red: false negative; blue: false positive.

F4-HeLa, Non Touching (Fig. Fig. SN3.12) The **out-of-the box performance** is acceptable with an average IoU of ∼0.7. Some of the brighter cells are missed but captured cells show high IoU. **Transfer learning** leads to an improvement to an IoU of ∼0.9 for >1000 training examples. However, already 10 labeled examples lead to an IoU of 0.85 after 100 training iterations. An overfitting to the training set can be observed after more than 1000 training iterations.



b

# imgs	# iterations 0	100	1k	10k
1	(0.29,0.28)	(0.39,0.37)	(0.37,0.35)	(0.37,0.36)
10	(0.29,0.28)	(0.63,0.58)	(0.80,0.76)	(0.63,0.62)
18	(0.29,0.28)	(0.58,0.56)	(0.76,0.74)	(0.68,0.66)

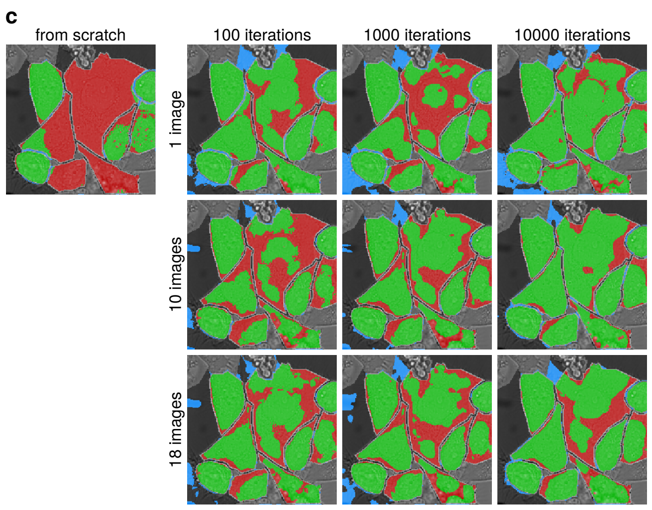
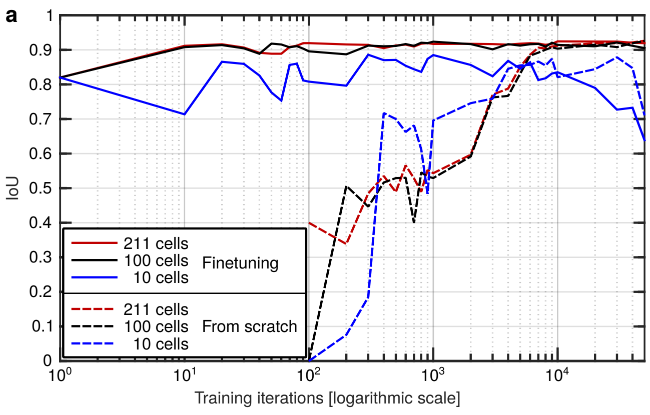


Figure SN3.13: DIC1-HeLa; Segmentation performance after transfer learning with different numbers of images. **(a)** IoU for transfer learning with different numbers of annotated cells compared to training from scratch; **(b)** (recall, precision) for $T = 701$ pixels which equals 0.95 of training set accepted; Segments with IoU < 0.5 are considered false positives; **(c)** Qualitative example. Green: true positive; red: false negative; blue: false positive.



b

# cells	# iterations 0	100	1k	10k
10	(0.97,0.97)	(0.97,0.91)	(1.00,1.00)	(0.91,0.86)
100	(0.97,0.97)	(1.00,1.00)	(1.00,1.00)	(1.00,1.00)
211	(0.97,0.97)	(1.00,1.00)	(1.00,1.00)	(1.00,1.00)

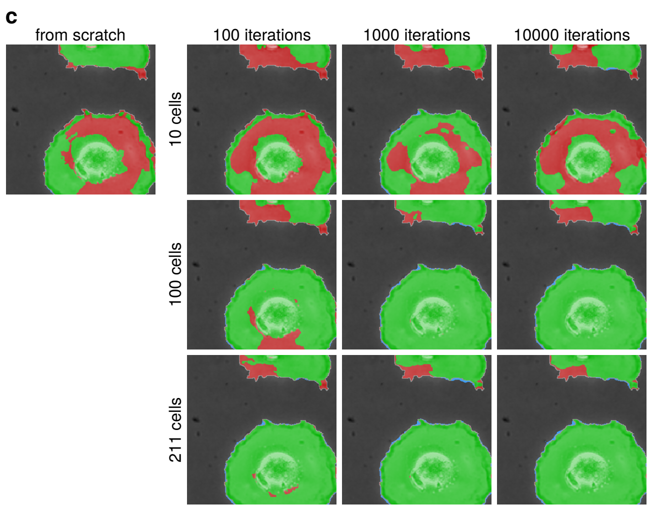
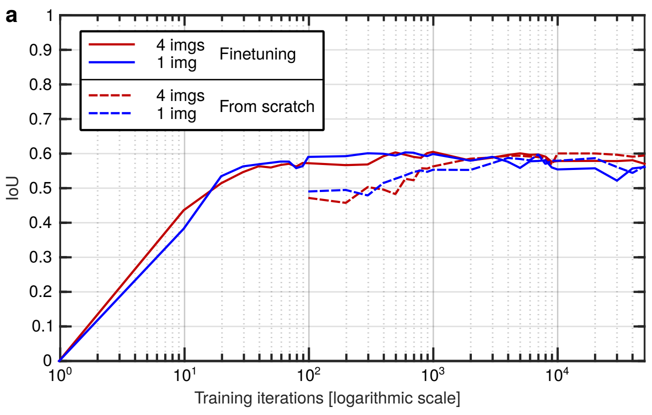


Figure SN3.14: PC1-U373; Segmentation performance after transfer learning with different numbers of cells. **(a)** IoU for transfer learning with different numbers of annotated cells compared to training from scratch; **(b)** (recall, precision) for $T = 2303$ pixels which equals 0.95 of training set accepted; Segments with IoU < 0.5 are considered false positives; **(c)** Qualitative example. Green: true positive; red: false negative; blue: false positive.

DIC1-HeLa, Touching (Fig. SN3.13) DIC1-HeLa is one of the most challenging datasets. Cells are touching, and their appearance is very heterogenous. The **out-of-the box performance** is low with an IoU of ∼0.25. **Transfer learning** improves the IoU to ∼0.7 for ∼2k training iterations with at least 10 annotated images. For this dataset we observed a significantly higher performance when training from scratch with a peak IoU of 0.8 at ∼10k training iterations. Image features learnt from other imaging modalities seem not to transfer well to DIC. Low recall mainly stems from non-separated cell instances, low precision from contaminations in the background.

PC1-U373, Non Touching (Fig. SN3.14) The **out-of-the box performance** is good with an IoU of ∼0.8 due to the visual similarity to PC3-HKPV. **Transfer learning** leads to an improved IoU of ∼0.9 after 100 training iterations and at least 20 annotated cells. With at least 100 annotated cells, no overfitting to the training set can be observed. More annotated cells do not give an advantage in terms of IoU.

a



b

# imgs	0	100	1k	10k
1	(0.00,0.00)	(0.75,0.75)	(0.78,0.78)	(0.69,0.69)
4	(0.00,0.00)	(0.73,0.73)	(0.78,0.78)	(0.74,0.74)

c

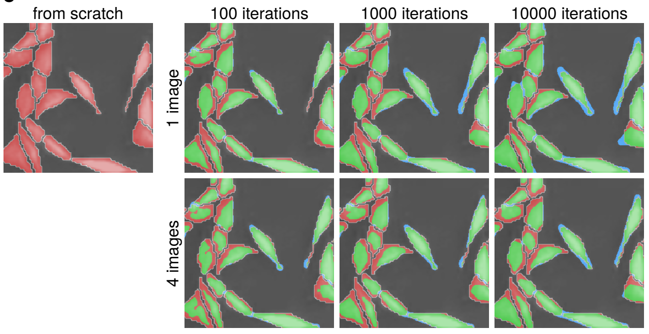
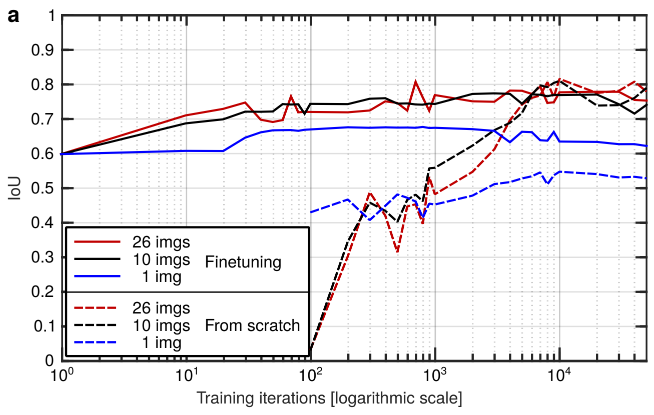


Figure SN3.15: PC2-PSC; Segmentation performance after transfer learning with different numbers of images. **(a)** IoU for transfer learning with different numbers of annotated cells compared to training from scratch; **(b)** (recall, precision) for $T = 780$ pixels which equals 0.95 of training set accepted; Segments with IoU < 0.5 are considered false positives; **(c)** Qualitative example. Green: true positive; red: false negative; blue: false positive.

PC2-PSC, Touching (Fig. SN3.15) This is the only dataset with **out-of-the box performance** of 0. **Transfer learning** quickly leads to an improvement to an IoU of ∼0.6 after 100 training iterations. 1 annotated image seems to be enough to capture the variability of our test set. The relatively low IoU compared to the other cell datasets is due to cell clusters that were not correctly segmented. The area of many cells is underestimated which is a consequence of the artificially introduced background ridges during training.

a



b

# imgs	0	100	1k	10k
1	(0.57,0.57)	(0.74,0.74)	(0.76,0.76)	(0.72,0.72)
10	(0.57,0.57)	(0.80,0.80)	(0.83,0.83)	(0.85,0.85)
26	(0.57,0.57)	(0.78,0.78)	(0.83,0.83)	(0.85,0.85)

c

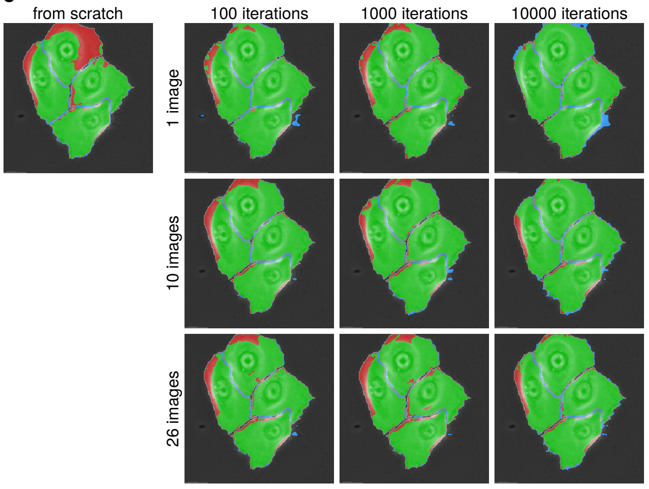
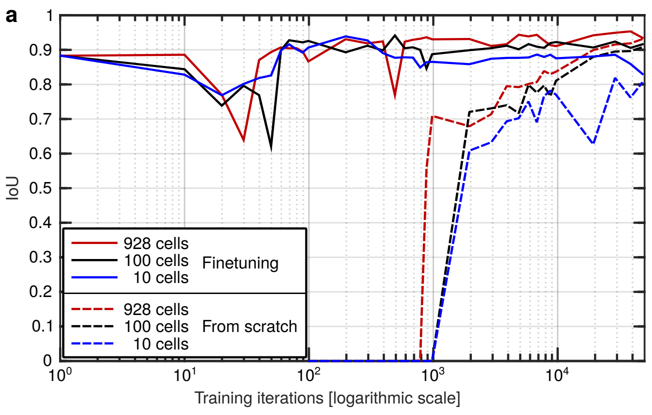


Figure SN3.16: PC3-HKPV; Segmentation performance after transfer learning with different numbers of images. **(a)** IoU for transfer learning with different numbers of annotated cells compared to training from scratch; **(b)** (recall, precision) for $T = 4511$ pixels which equals 0.95 of training set accepted; Segments with IoU < 0.5 are considered false positives; **(c)** Qualitative example. Green: true positive; red: false negative; blue: false positive.

PC3-HKPV, Touching (Fig. SN3.16) The **out-of-the box performance** has an IoU of 0.6 which is mostly due to cell clusters that were not properly separated into their individual cells. **Transfer learning** improves the IoU to a value of ∼0.75 after 1k training iterations. 10 annotated images seems to be enough to capture the variation in visual appearance of cells in our test set.



b

	# iterations			
# cells	0	100	1k	10k
10	(0.95,0.32)	(1.00,0.63)	(0.91,0.77)	(0.92,0.82)
100	(0.95,0.32)	(1.00,0.60)	(0.95,0.90)	(0.97,0.88)
228	(0.95,0.32)	(0.91,0.78)	(0.99,0.77)	(0.95,0.67)

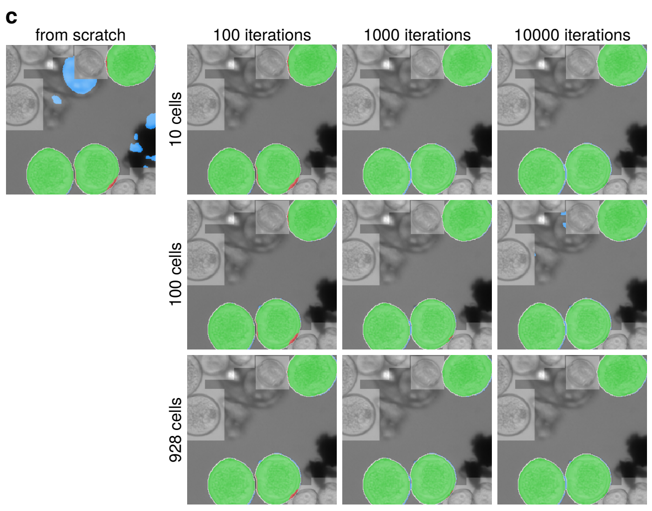
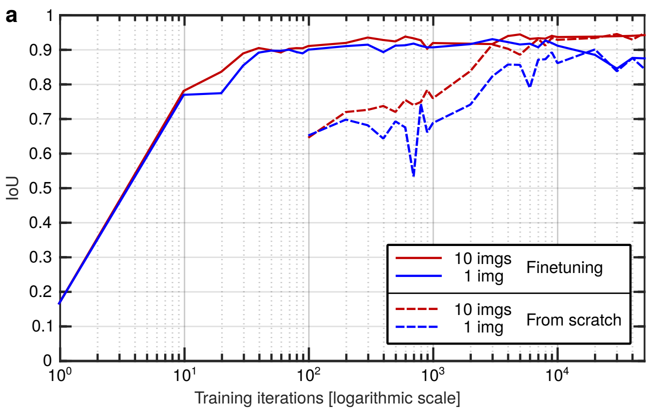


Figure SN3.17: BF1-POL; Segmentation performance after transfer learning with different numbers of cells. **(a)** IoU for transfer learning with different numbers of annotated cells compared to training from scratch; **(b)** (recall, precision) for $T = 1535$ pixels which equals 0.95 of training set accepted; Segments with IoU < 0.5 are considered false positives; **(c)** Qualitative example. Green: true positive; red: false negative; blue: false positive.

BF1-POL, Non Touching (Fig. SN3.17) The **out-of-the box performance** is very good with an IoU of ∼0.88. **Transfer learning** improves the IoU marginally to a value of ∼0.9 after 100 training iterations for 100 annotated objects and to a value of ∼0.94 after 1000 training iterations for 928 annotated objects. Background clutter produces many false positives. With all available 928 labeled cells, the **precision** is considerably lower after 10k iterations of transfer learning compared to training with 100 examples. The 100 randomly selected cells seem to better reflect the statistics of the test set. Precision benefits from longer training times in every setting. For this dataset with its difficult background, our choice of weights that give 10× more weight to the loss for foreground pixels is not optimal.



b

	# iterations			
# imgs	0	100	1k	10k
1	(0.16,0.16)	(0.97,0.94)	(0.98,0.95)	(0.98,0.96)
10	(0.16,0.16)	(0.98,0.96)	(0.98,0.95)	(0.99,0.97)

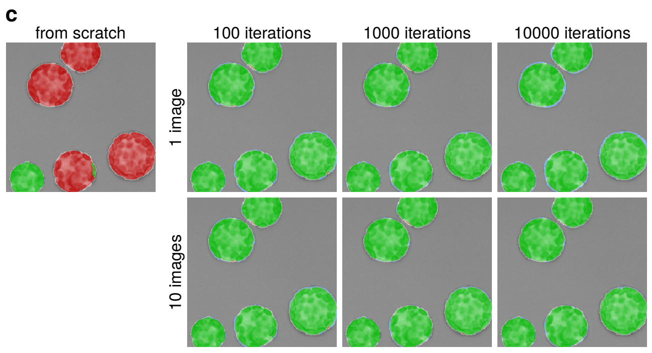
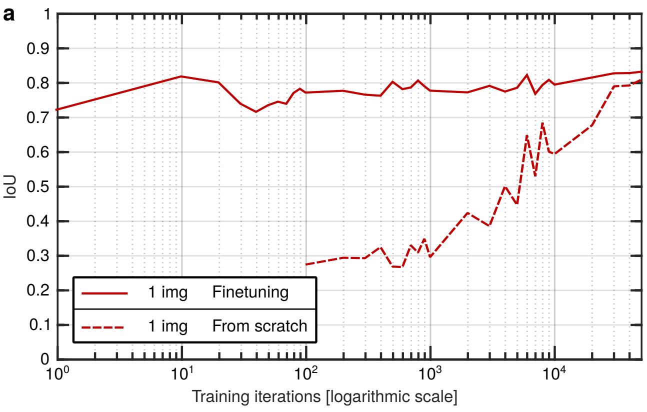


Figure SN3.18: BF2-PPL; Segmentation performance after transfer learning with different numbers of images. **(a)** IoU for transfer learning with different numbers of annotated cells compared to training from scratch; **(b)** (recall, precision) for $T = 887$ pixels which equals 0.95 of training set accepted; Segments with IoU < 0.5 are considered false positives; **(c)** Qualitative example. Green: true positive; red: false negative; blue: false positive.

BF2-PPL, Touching (Fig. SN3.18) The **out-of-the box performance** is low with an IoU of <0.2. **Transfer learning** improves the IoU quickly to a value of ∼0.9 after 100 training iterations. Only 1 annotated image seems to be enough to capture the variability in the appearance of cells in our test set.

a



b

	# iterations			
# imgs	0	100	1k	10k
1	(0.87,0.85)	(0.86,0.86)	(0.85,0.85)	(0.88,0.88)

c

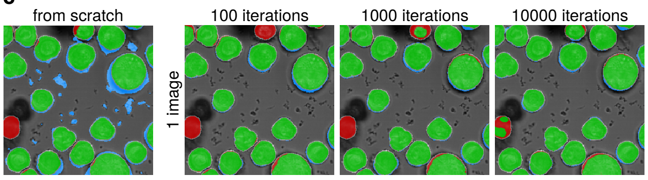


Figure SN3.19: BF3-MiSp; Segmentation performance after transfer learning with different numbers of images. **(a)** IoU for transfer learning with different numbers of annotated cells compared to training from scratch; **(b)** (recall, precision) for $T = 502$ pixels which equals 0.95 of training set accepted; Segments with IoU < 0.5 are considered false positives; **(c)** Qualitative example. Green: true positive; red: false negative; blue: false positive.

BF3-MiSp, Touching (Fig. SN3.19) The **out-of-the box performance** is quite good with an IoU of ~0.72. **Transfer learning** improves the IoU marginally to a value of ~0.8 after 100 training iterations. Beware, that this dataset only contains one image for training and one for testing.

SN 3.2.4 Single cell segmentation in volumetric data (3D)

Image annotation and label generation We used dataset BF3-MiSp as described above for the 3D cell segmentation experiment. Volume datasets were first converted from their native OME-TIFF to Analyze format and then cell instances manually annotated in 3DSlicer using a ball brush with varying diameter (depending on required detail level). Resulting annotations were converted to HDF5 format using ImageJ, and binarized after adding one pixel wide background ridges between individual cell instances to allow instance-aware semantic segmentation.

For the experiments with sparse annotations, all volumetric annotations outside the slices considered for training were ignored by assigning them a pixel-wise weight of zero during loss computation.

Weight computation parameters: $\lambda = 50$, $v_{\text{bal}} = 0.1$, $\sigma_{\text{bal}} = 10\text{vx}$, $\sigma_{\text{sep}} = 6\text{vx}$.

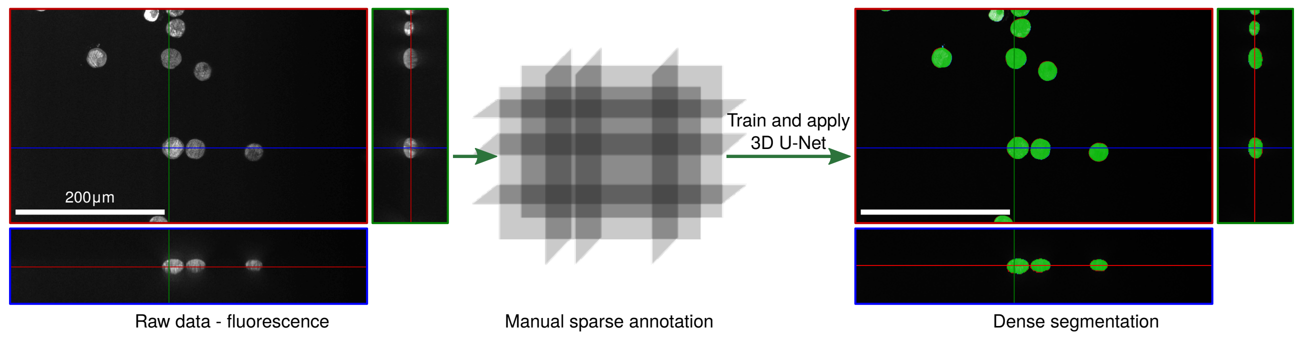
Data augmentation We applied random rotation around the z-axis drawn from a uniform distribution in the full 360° range and elastic deformation with random seed displacement vectors drawn from a Gaussian distribution with standard deviation 1px along each dimension. Seed grid point distance: 150px. We chose a very small magnitude for the elastic deformations to avoid unrealistic distortions of the almost perfectly round protoplasts while still randomizing the otherwise structured noise introduced by image interpolation.

Training Input patch size: 236×236×100, Solver: ADAM, Base learning rate: 10^{-5} (fixed schedule), Momentum: 0.9, Momentum2: 0.999, iterations: 46000.

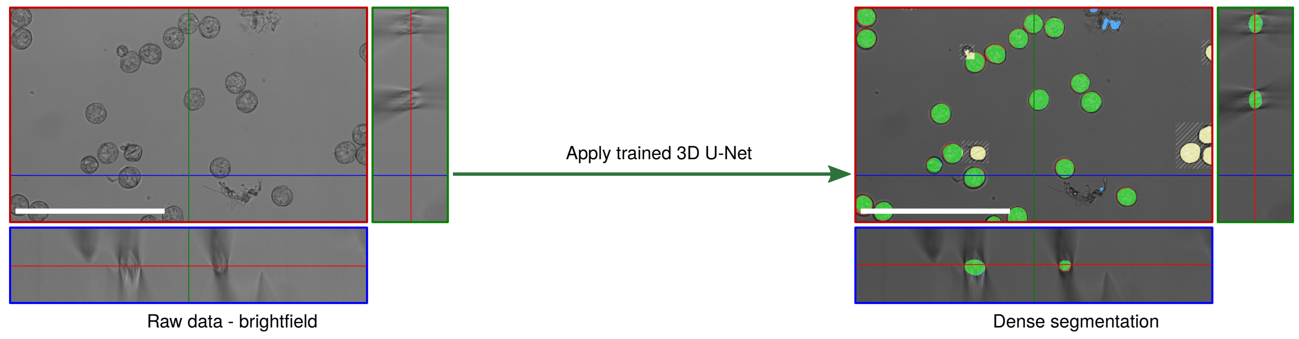
Results In our experiments we used a total of 14 manually annotated volumetric images. For 8 of these volumes, two aligned channels were available: one from brightfield microscopy and one from structured illumination fluorescence microscopy. The other 6 volumes contained only the brightfield channel.

Segmentation of 3D brightfield data is challenging due to the effects of the point spread function (psf) of the optical system. Especially in z-direction (perpendicular to the imaging plane), psf-induced conical structures dominate the actual cell boundaries. Light refraction leads to non-linear effects – so-called caustics – that often fool human observers and traditional segmentation approaches. Thus, on the aligned data, the fluorescence channel was used for annotation. Datasets without fluorescence channel were annotated to the best of the annotator's knowledge in order to produce plausible cell shapes.

a



b



c

Annotation	Channel	Test sample1	Test sample2	Average
Full	brightfield	0.74	0.87	0.80
	fluorescence	0.89	0.89	0.89
Sparse	brightfield	0.84	0.90	0.87
	fluorescence	0.84	0.86	0.85
Semi-automatic	brightfield			0.82
	fluorescence			0.87

Figure SN3.20: Volumetric segmentation with U-Net. **(a)** Example for semi-automatic volumetric segmentation based on only few annotated slices. **(b)** Example for fully-automatic segmentation on challenging brightfield data. **(Left)** Orthogonal cuts through the raw images; colored boxes and lines indicate the cut positions; **(right)** segmentation results overlaid over the raw data: true positive (green), false positive (blue), false negative (red), network prediction in areas without ground truth (yellow). Dashed areas: no ground truth annotation available (see text). **(c)** Quantitative evaluation of 3D cell segmentation by means of IoU for three scenarios. Full: Segmentation of unseen datasets after training on full-volume annotations; Sparse: Segmentation of unseen datasets after training on 2D slice annotations; Semi-Automatic: Full-volume segmentation from sparse 2D slice annotation in the same sample.

Segmentation of unseen datasets after training with dense annotation. We trained a U-Net on the dense annotations of 12 of the 14 brightfield samples and another U-Net on 6 of the 8 fluorescence samples (Fig. SN3.20c Full) and quantified the segmentation performance on the two left-out datasets of the respective imaging modality. The network learned to segment the cells based on the fluorescence channel without problems as expected. Also on the much harder brightfield channel, the network produced surprisingly accurate segmentations with an average IoU of 0.8.

Segmentation of unseen samples after training with sparse annotation. Since manual annotation of volumetric datasets is particularly tedious, we repeated the above experiment, but this time we used annotations of only 8 slices in each sample volume (Fig. SN3.20a, Fig. SN3.20c Sparse). Interestingly, the results obtained with this sparsely annotated training data were as good as those with densely annotated data.

Semi-automated segmentation. In a third experiment, we analyzed the capability of the U-Net to generate a volumetric segmentation of a 3D image from sparse plane annotations. For this, we trained the network from scratch on a sample with only 8 annotated slices. After training we applied the network to the whole sample to produce a dense volumetric segmentation. We repeated the experiment for all samples of the training set (12 for brightfield and 6 for fluorescence data) (Fig. SN3.20c Semi-automatic). Without the use of any pre-trained models, U-Net was able to extend the annotation from 8 slices to a full volumetric segmentation with an average IoU greater than 0.8.

SN 3.2.5 Neurite segmentation in volumetric EM data (3D)

As a last proof-of-principle experiment, we applied the U-Net to the segmentation of neurites in electron microscopy (EM) image stacks.

Image annotation and label generation We trained on the publicly available SNEMI (http://brainiac2.mit.edu/SNEMI3D/) training set, consisting of one annotated EM volume of size $1024 \times 1024 \times 100$ vx (voxel extents: $6 \times 6 \times 30$nm) and tested on the corresponding test image stack.

Weight computation parameters: $\lambda = 50$, $v_{\text{bal}} = 0.1$, $\sigma_{\text{bal}} = 10$px, $\sigma_{\text{sep}} = 6$px, inter-instance ridge spacing: 3px.

Data augmentation We applied random rotation around the z-axis drawn from a uniform distribution in the full 360° range and elastic deformation with random seed displacement vectors drawn from a Gaussian distribution with standard deviation 20px along each dimension. Seed grid point distance: 150px.

Training We trained a four-stage U-Net as in the 3D cell segmentation example, but performed only 2D operations on the first and second resolution levels to reach approximately cubic voxels ($24 \times 24 \times 30$nm) at third level. Input patch size: $300 \times 300 \times 72$, Solver: ADAM, Base learning rate: 10^{-5} (fixed schedule), Momentum: 0.9, Momentum2: 0.999, iterations: 74000.

Results Neurite segmentation is particularly hard for the U-Net for several reasons: First, the structures to segment (soma and neurites) show a wide range of sizes and shapes. Large structures, like soma, exceed the receptive field of the network and would require sub-sampling of the volume, however, sub-sampling precludes segmentation of thin neurites. Secondly, cells are forming a tissue and are thus densely packed. Adding ridges with extra-weights, as we did for the 2D segmentation example, works fine in the case of occasionally touching instances, but solving the problem of dense tissue segmentation in 3D with a pure semantic segmentation approach will very likely fail. In a semantic-segmentation-sense mis-classifying one voxel as cell instead of background is a tiny error, but this one pixel may merge two instances, leading to a large topological error. The probability of such merging is proportional to the area of the interfaces between neighboring cells.

Zeng et al.[29] train an ensemble of semantic segmentation networks on pre-processed challenge annotations. With tuned watershed post-processing they reach results close to human performance.

Our goal is to show the pure U-Net performance with as few engineering as possible. Therefore, we trained on the raw annotations and did not perform any post-processing on the binary output labels. We assigned different colors to the connected components of the output for instance visualization (Fig. SN3.21b).

As expected, large cells merge with neighboring cells due to mis-classifications in their interface area, which could be corrected manually or with automated post-processing. However, most of the thinner neurites are already properly segmented without any post-processing.

A simple extension is to use an ensemble of U-Net models and only accept a voxel as foreground if all models agree. This introduces a bias that treats voxels with low confidence as background and thus improves instance separation. In practice we simply perform segmentation with different snapshots from one training and randomly picked the models after iteration 44k, 50k and 74k (Fig. SN3.21c). We combined them using the minimum of the binary segmentation outputs. With this simple trick that can be easily applied using ImageJ's image calculator, almost all instances are properly separated.

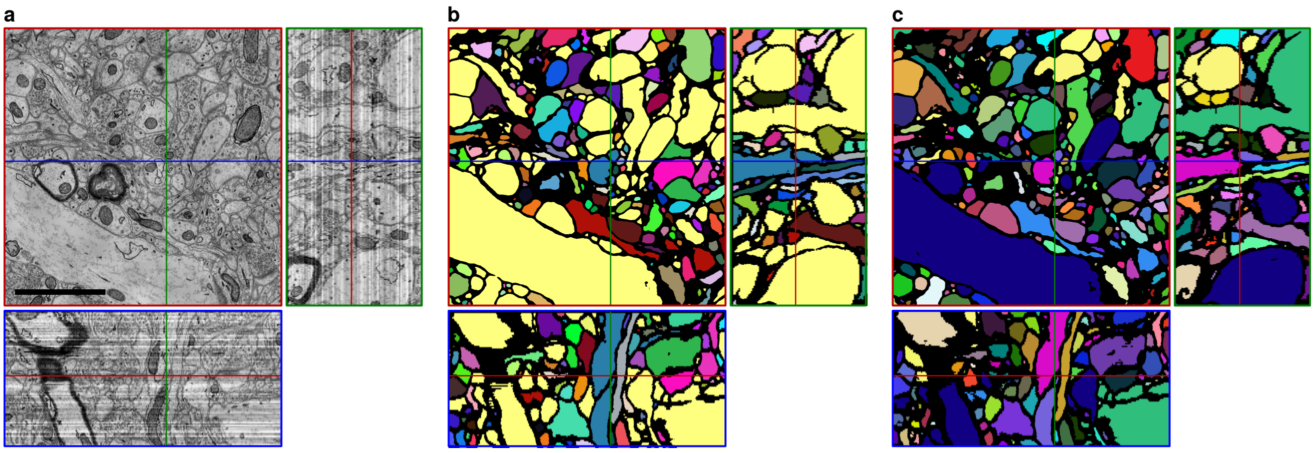


Figure SN3.21: Segmentation result on the SNEMI test volume. (**a**) Raw volume. (**b**) Segmentation result using a single U-Net model. (**c**) Segmentation result using the intersection of the outputs of an ensemble of three U-Net models. Training was repeated three times leading to comparable results. Panels show orthographic cuts through the SNEMI test volume. Colors indicate instance labels. Scalebar: 2μm.

Discussion Segmentation performance would further increase if the distinction of instances would be directly encoded into the objective function. Another possibility is to transform the label space to allow better instance separation[30], however this comes at the cost of higher memory consumption.

References

15. Boyden, E. S., Zhang, F., Bamberg, E., Nagel, G. & Deisseroth, K. Millisecond-timescale, genetically targeted optical control of neural activity. *Nature Neuroscience* **8,** 1263–1268 (2005).
16. Gradinaru, V., Thompson, K. R. & Deisseroth, K. eNpHR: a Natronomonas halorhodopsin enhanced for optogenetic applications. *Brain Cell Biology* **36,** 129–139 (2008).
17. Liu, Z. *et al.* Systematic comparison of 2A peptides for cloning multi-genes in a polycistronic vector. *Scientific Reports* **7,** online: No. 2193 (2017).
18. Bogen, I. L., Haug, K. H., Roberg, B., Fonnum, F. & Walaas, S. I. The importance of synapsin I and II for neurotransmitter levels and vesicular storage in cholinergic, glutamatergic and GABAergic nerve terminals. *Neurochemistry International* **55,** 13–21 (2009).
19. Dittgen, T. *et al.* Lentivirus-based genetic manipulations of cortical neurons and their optical and electrophysiological monitoring in vivo. *Proceedings of the National Academy of Sciences* **101,** 18206–18211 (2004).
20. Kingma, D. P. & Ba, J. L. *Adam: A Method for Stochastic Optimization* in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)* (2015).
21. Tay, T. L. *et al.* A new fate mapping system reveals context-dependent random or clonal expansion of microglia. *Nature Neuroscience* **20,** 793–803 (2017).
22. Ulman, V. *et al.* An objective comparison of cell-tracking algorithms. *Nature Methods* **14,** 1141–1152 (2017).
23. Kaur, P. & McDougall, J. K. Characterization of Primary Human Keratinocytes Transformed by Human Papillomavirus Type 18. *Journal of Virology* **62,** 1917–1924 (1988).
24. Saltukoglu, D. *et al.* Spontaneous and electric field-controlled front-rear polarization of human keratinocytes. *Molecular Biology of the Cell* **26,** 4373–4386 (2015).
25. Rösch, P. *et al.* On-Line Monitoring and Identification of Bioaerosols. *Analytical Chemistry* **78,** 2163–2170 (2006).
26. Dovzhenko, A., Bergen, U. & Koop, H. .-.-U. Thin-alginate-layer technique for protoplast culture of tobacco leaf protoplasts: Shoot formation in less than two weeks. *Protoplasma* **204,** 114–118 (1998).
27. Touraev, A. & Heberle-Bors, E. Microspore Embryogenesis and In Vitro Pollen Maturation in Tobacco. *Plant Cell Culture Protocols* **111,** 281–291 (1999).
28. Ronneberger, O., Fischer, P. & Brox, T. *U-Net: Convolutional Networks for Biomedical Image Segmentation* in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)* **9351** (2015), 234–241.
29. Zeng, T., Wu, B. & Ji, S. DeepEM3D: approaching human-level performance on 3D anisotropic EM image segmentation. *Bioinformatics* **33,** 2555–2562 (2017).
30. Böhm, A., Ücker, A., Jäger, T., Ronneberger, O. & Falk, T. *ISOO$_{DL}$: Instance segmentation of overlapping biological objects using deep learning* in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)* (2018), 1225–1229.