nature methods

ARTICLES

https://doi.org/10.1038/s41592-020-01018-x

Check for updates

Cellpose: a generalist algorithm for cellular segmentation

Carsen Stringer, Tim Wang, Michalis Michaelos and Marius Pachitariu ✉

Many biological applications require the segmentation of cell bodies, membranes and nuclei from microscopy images. Deep learning has enabled great progress on this problem, but current methods are specialized for images that have large training datasets. Here we introduce a generalist, deep learning-based segmentation method called Cellpose, which can precisely segment cells from a wide range of image types and does not require model retraining or parameter adjustments. Cellpose was trained on a new dataset of highly varied images of cells, containing over 70,000 segmented objects. We also demonstrate a three-dimensional (3D) extension of Cellpose that reuses the two-dimensional (2D) model and does not require 3D-labeled data. To support community contributions to the training data, we developed software for manual labeling and for curation of the automated results. Periodically retraining the model on the community-contributed data will ensure that Cellpose improves constantly.

Quantitative cell biology requires measurements of multiple cellular properties such as shape, position, RNA expression and protein expression[1]. To assign these properties to individual cells, one must first segment an image volume into cell bodies, usually based on a cytoplasmic or membrane marker[2–8]. This step can be straightforward when cells are sufficiently separated from one other, for example in monodispersed cultures in vitro. However, in many tissues, cells are tightly packed together and difficult to separate.

Most methods for cell body segmentation make trade-offs between flexibility and automation. These methods range from fully manual labeling[9], to user-customized pipelines involving a sequence of image transformations with user-defined parameters[2,8,10,11], to fully automated methods based on deep neural networks with parameters estimated on large training datasets[4,5,7,12–14]. Fully automated methods have many advantages, such as reduced human effort, increased reproducibility and better scalability to big datasets from large screens. However, these methods are typically trained on specialized datasets, and do not generalize well to other types of data, requiring new human-labeled images to achieve good performance on any one image type.

Recognizing this generalization problem in the context of nuclear segmentation, a recent Data Science Bowl challenge amassed a dataset of varied images of nuclei from many different laboratories[15]. Methods trained on this dataset can generalize far more widely than those trained on data from a single laboratory. The winning methods from the challenge were based on established computer vision algorithms such as the Mask R-CNN model[16,17]. Following the competition, the dataset generated further progress, with other methods such as Stardist and nucleAIzer being developed[18,19].

In this work, we followed the approach of the Data Science Bowl team to collect and manually segment a large dataset of cell images from a variety of microscopy modalities and fluorescent markers. We had no guarantee that we could replicate the success of the Data Science Bowl, because cells have highly diverse shapes and a single model may not be able to segment all these shapes. Previous approaches did not perform well on this dataset, and we therefore developed a new model with better expressive power called Cellpose. We next describe the architecture of Cellpose and the new training dataset, and then proceed to show performance benchmarks on test data, as well as an extension to 3D. The Cellpose package with a graphical user interface (Extended Data Fig. 1) was implemented in Python 3 using open-source packages[20–27]. Cellpose is freely available and can be installed locally or tested online at www.cellpose.org.

Results

Model design. Classical segmentation approaches based on the watershed algorithm[28,29] use the grayscale values of an image to create a topological map, whose basins represent the segmented regions. These methods work well when the segmented objects have intensity profiles that decay smoothly from the center, so that the watershed forms a single basin. However, many types of cell form multiple intensity basins, usually due to the nuclear exclusion of the fluorescent marker and its inhomogeneous distribution along cell borders and protuberances. These issues motivated us to construct an intermediate representation of an object that does form a single smooth topological basin (Fig. 1a,b).

We generated topological maps through a process of simulated diffusion that used the ground-truth masks drawn by a human annotator (Fig. 1a). A neural network was then trained to predict the horizontal and vertical gradients of the topological maps, as well as a binary map that indicates if a given pixel is inside or outside of regions of interest (ROI) (Fig. 1c,d). On a test image, the neural network predicted the horizontal and vertical gradients, which formed vector fields. By following the vector fields through a process known as gradient tracking[29], all pixels belonging to a given cell can be routed to its center. Thus, by grouping together pixels that converge to the same point, we recover individual cells and their precise shapes (Fig. 1e,f and Methods). The cell shapes were further refined by removing pixels that were predicted by the neural network to be outside of cells.

The neural network that predicts the spatial gradients was based on the general U-Net architecture[3], which downsamples the convolutional maps several times before upsampling in a mirror-symmetric fashion (Fig. 1d). On the upsampling pass, the

HHMI Janelia Research Campus, Ashburn, VA, USA. ✉e-mail: pachitarium@janelia.hhmi.org

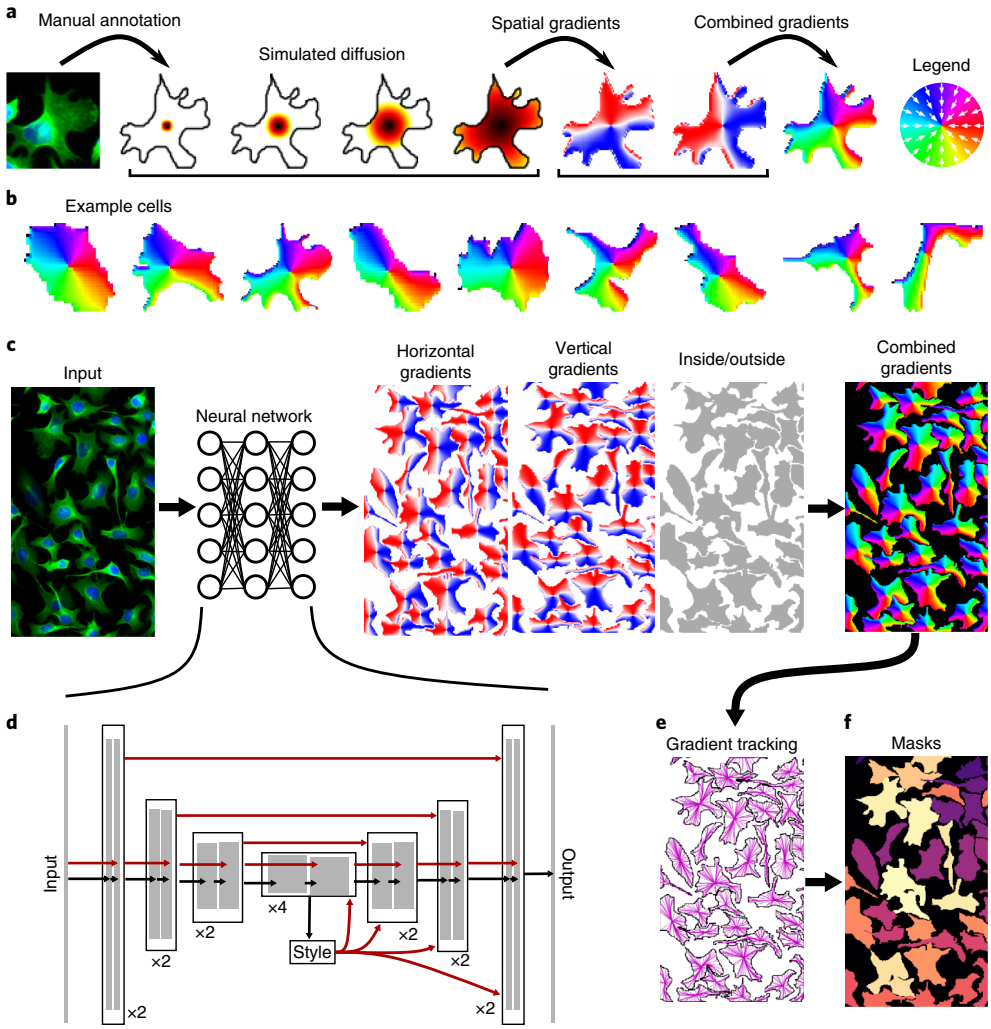


Fig. 1 | Model architecture. a, Procedure for transforming manually annotated masks into a vector flow representation that can be predicted by a neural network. A simulated diffusion process starting from the center of the mask is used to derive spatial gradients that point toward the center of the cell, potentially indirectly around corners. The *x* and *y* gradients are combined in a single direction from 0° to 360°. **b**, Example spatial gradients for cells from the training dataset. **c**, A neural network is trained to predict the horizontal and vertical gradients, as well as whether a pixel belongs to any cell. The three predicted maps are combined into a gradient vector field. **d**, The details of the neural network that contains a standard backbone U-Net[3] to downsample and then upsample the feature maps, with skip connections between layers of the same size and global skip connections from the image styles, computed at the lowest resolution, to all successive computations. **e**, At test time, the predicted gradient vector fields are used to construct a dynamical system with fixed points whose basins of attraction represent the predicted masks. Informally, every pixel 'tracks the gradients' toward their eventual fixed point. **f**, All the pixels that converge to the same fixed point are assigned to the same mask.

U-Net 'mixes in' the convolutional maps from the downsampling pass. This mixing is typically done by feature concatenation, but we opted for direct summation to reduce the number of parameters. We also replaced the standard building blocks of a U-Net with residual blocks, which have been shown to perform better, and doubled the depth of the network as typically done for residual networks[30]. In addition, we used global average pooling on the smallest convolutional maps to obtain a representation of the 'style' of the image (for similar definitions of style, see refs. [19,31,32]). We anticipated that images with different styles might need to be processed differently, and therefore fed the style vectors into all the upsampling stages on an image by image basis. These modifications to the U-Net architecture improved performance significantly (Extended Data Fig. 2a–d).

We use several test time enhancements to further increase the predictive power of the model: test time resizing (Extended Data Fig. 3), ROI quality estimation, model ensembling and image tiling (Methods). The run time performance of the algorithm is shown in Supplementary Table 1.

Training dataset. We collected images from a variety of sources, primarily via internet searches for keywords such as 'cytoplasm', 'cellular microscopy', 'fluorescent cells' and so on. Some of the websites with hits shared the images as educational material, promotional material or news stories, while others contained datasets used in previous scientific publications (for example on OMERO[33,34]). The dataset consisted primarily of fluorescently labeled proteins that localized to the cytoplasm with or without 4,6-diamidino-2-phenylindole- (DAPI-) stained nuclei in a separate channel ($n=316$ images). We also included images of cells from brightfield microscopy ($n=50$) and images of membrane-labeled cells ($n=58$). Finally, we included a small set of images from other types of microscopy ($n=86$), and a small set of nonmicroscopy images that contained large numbers of repeated objects such as fruits, rocks and jellyfish ($n=98$). We hypothesized

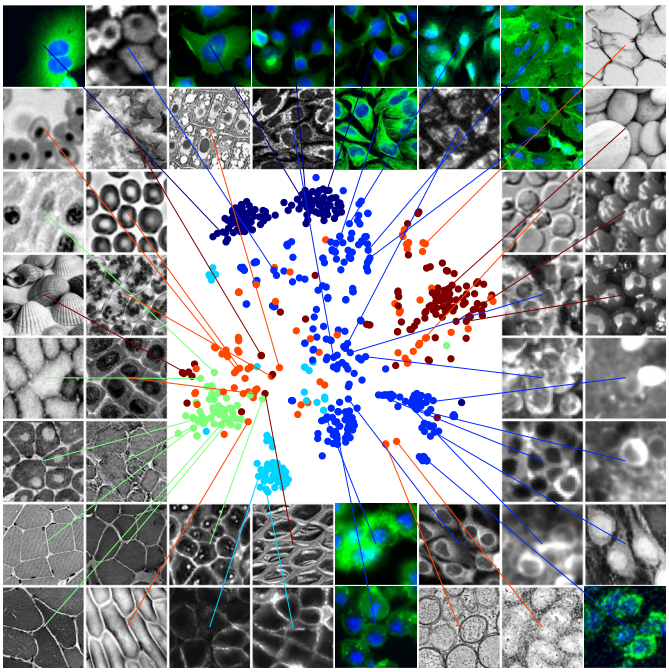


Fig. 2 | Visualization of the diverse training dataset. The styles from the network for all the images in the cytoplasm dataset were embedded using *t*-SNE. Each point represents a different image. Legend is as follows: dark blue, Cell Image Library; blue, cytoplasm; cyan, membrane; green, nonfluorescent cells; orange, microscopy other and red, nonmicroscopy. Randomly chosen example images are shown around the *t*-SNE plot. Images with a second channel marking the nucleus are displayed in green/blue.

that the inclusion of such images in the training set would allow the network to generalize more widely and more robustly.

We visualized the structure of this dataset by applying *t*-distributed stochastic neighbor embedding (*t*-SNE) to the image styles learned by the neural network (Fig. 2)[35,36]. Images were generally placed in the embedding according to the categories defined above, with the noncell image categories scattered across the entire embedding space. To manually segment these images, we developed a graphical user interface in Python (Extended Data Fig. 1) that relies on PyQt and pyqtgraph[26,27]. The interface enables quick switching between views such as outlines, filled masks and image color channels, as well as easy image navigation such as zooming and panning. A new mask is initiated with a right-click, and gets completed automatically when the user returns within a few pixels of the starting area. This interface allowed human operators to segment 300–600 objects per hour, and is part of the code we are releasing with the Cellpose package. The segmentation algorithm can also be run in the same interface, allowing users to easily curate and contribute their own data to our training dataset.

Within the varied dataset of 608 images, a subset of 100 images were presegmented as part of the Cell Image Library[37]. These two-channel images (cytoplasm and nucleus) were from a single experimental preparation and contained cells with complex shapes. We reasoned that the segmentation methods we tested would not overfit on such a large and visually uniform dataset, and may instead fail due to a lack of expressive power such as an inability to precisely follow cell contours or to incorporate information from the nuclear channel. We therefore chose to also train the models on this dataset alone as a 'specialist' benchmark of expressive power.

Benchmarks. Trained on our entire library of images, the 'generalist' Cellpose model achieved good performance on the test set (Fig. 3). We also trained Cellpose as a specialist model on the Cell Image Library alone ('specialized data', Fig. 4a,b), and compared this to the generalist model (Fig. 4ab). We included in the comparisons several other state-of-the-art methods: Stardist[18], Mask R-CNN[16,17] and two- and three-class U-Net models[3]. Example segmentations for Stardist and Mask R-CNN are shown in Extended Data Figs. 4 and 5. On test images, we matched the predictions of the algorithms to the true masks at different thresholds of matching precision based on the standard intersection over union metric (IoU). We evaluated performance with the average precision metric (AP), computed from the number of true positives, TP, false positives, FP and false negatives, FN, as $AP = \frac{TP}{TP+FP+FN}$. Similar results were obtained using boundary prediction metrics (Extended Data Fig. 6a,b) and the aggregate Jaccard index (Extended Data Fig. 6c,d).

Cellpose substantially outperformed previous methods at all matching thresholds, when trained and tested on the specialized data. For example, at the commonly used IoU threshold of 0.5, Cellpose correctly matched 485 out of 521 total ground-truth ROI and gave only 18 false positives. This corresponded to an average precision of 0.91, compared to 0.76 for Stardist and 0.80 for Mask R-CNN. At higher IoUs, which benchmark the ability to precisely follow cell contours, the relative improvement of Cellpose compared to other methods grew even larger. This analysis thus shows that Cellpose has enough expressive power to capture complicated shapes (Fig. 4d).

However, all models trained on the specialized data alone performed poorly on the full dataset (Fig. 4e), motivating the need for a generalist algorithm. On the specialized data alone (the Cell Image Library), the generalist Cellpose model matched the performance of the specialist model (Fig. 4f). On the generalized data alone, the generalist Cellpose model had an average precision of 0.77 at a threshold of 0.5, significantly outperforming Mask R-CNN and Stardist that had average precisions of 0.61 and 0.61 respectively (Fig. 4g). We also tested if the inclusion of specialized images in the training of the generalist model impaired its performance on generalized images, and found that they in fact helped considerably (Extended Data Fig. 2e). Altogether, these results show that the inclusion of more images in the training set does not saturate the capacity of the neural network, indicating that Cellpose might have even more spare capacity for additional training data, which we hope to identify via community contributions.

We next wanted to determine what types of image and cell are more challenging to segment. For this, we examined the predictions of the models on each of the image categories in our generalized test set (Fig. 5a). Across image categories, Cellpose had better performance on nonfluorescent images of cells, on cell membranes and on the Cell Image Library (Fig. 3f), likely reflecting the higher homogeneity of these image sets. This led us to hypothesize images that contain more variability across cells, for example in cell sizes, would be more difficult to segment. However, we found no relation between segmentation performance and the homogeneity of cell sizes (Fig. 5b). Finally, we asked whether the convexity of a cell influences segmentation performance, by grouping cells into low, medium and high convexity (Fig. 5b). We found that all models performed better on cells with high convexity compared to low convexity (Fig. 5d), and the performance penalty for low versus high convexity was similar across models.

Finally, we assembled a large dataset of images of nuclei, by combining presegmented datasets from several previous studies, including the Data Science Bowl kaggle competition[15]. Because nuclear shapes are simpler and more uniform, this dataset did not have as much variability as the dataset of cells (see the *t*-SNE embedding of the segmentation styles in Extended Data Fig. 7a). Cellpose outperformed the other methods on this dataset by smaller amounts, likely reflecting the reduced difficulty of this segmentation problem (Extended Data Figs. 7b,c and 8).

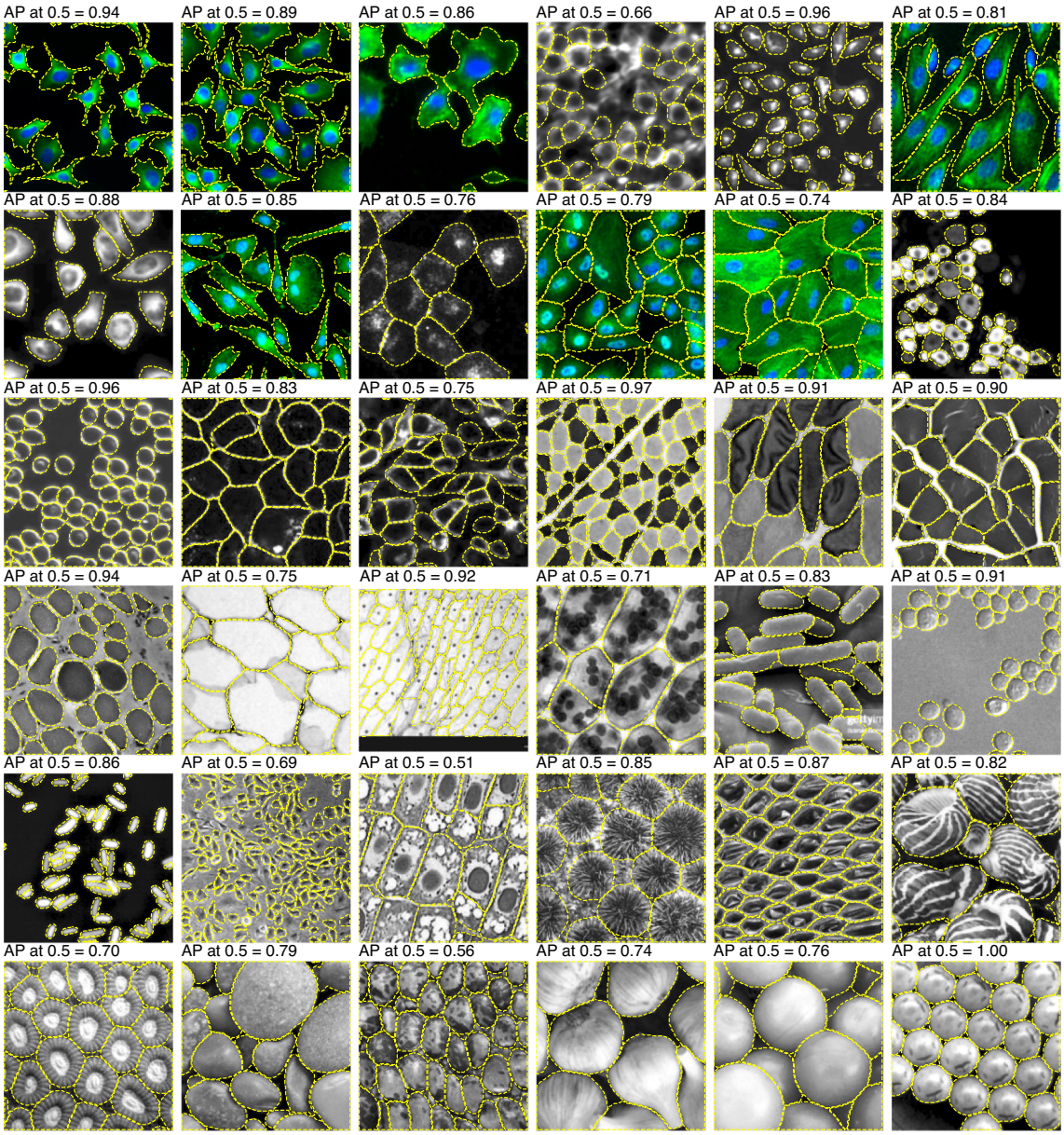


Fig. 3 | Example Cellpose segmentations for 36 test images. The masks predicted by Cellpose are shown with dotted yellow lines. Compare these to the Stardist and Mask-R-CNN in Extended Data Figs. 4 and 5.

3D segmentation. Our last contribution was to generalize Cellpose for 3D segmentation. This task usually requires 3D training data, which is much more difficult to obtain than 2D training data as it requires one 2D segmentation for every slice in the volume. Thus, a cell that spanned 30 pixels, the median diameter in our dataset, would take 30 times longer to manually segment in 3D. An alternative approach would be to extend pretrained, generalist 2D models to 3D by running the neural networks on every 2D slice. The 3D segmentation step that would follow is straightforward for some models such as the U-Nets, where the postprocessing step of finding connected components can be performed similarly in 3D and 2D.

However, this type of extension is not available for models with more complex postprocessing steps, such as Cellpose, Stardist and Mask R-CNN. We therefore designed a new method for extending Cellpose to 3D, using only the trained 2D model and no additional 3D training data. For a test volume, we ran Cellpose on all *xy*, *xz* and *yz* slices independently (Fig. 6a,b). For each point, this procedure generated two estimates of the gradient in *x*, *y* and *z* (that is, six total predictions), which we then averaged together to obtain a complete set of 3D vector gradients. To generate ROI, we then ran the gradient vector tracking step in 3D, followed by a clustering of pixels that converge to the same fixed points (Fig. 6e).

We compared Cellpose3D to the 3D extensions of the 2D U-Net models (Fig. 6e). We also compared against a different 3D approach of 'stitching' together sequential 2D segmentations from each *xy* plane. This approach can be applied to any 2D method, including the generalist Stardist and Mask R-CNN models. Finally, we used ilastik to generate a 3D segmentation pipeline specific for this volume. The parameters of ilastik were chosen manually to give good performance on a subset of the volume that was not used for testing. Like Cellpose3D, ilastik did not require a large 3D training dataset, and thus can be trained and deployed easily by nonexperts.

We evaluated the performance of Cellpose3D and the other models on a manually annotated 3D test volume in which the DNA and RNA were costained to serve as nuclear and cytoplasmic markers, respectively (Fig. 6f). The human annotator found 183 cells,

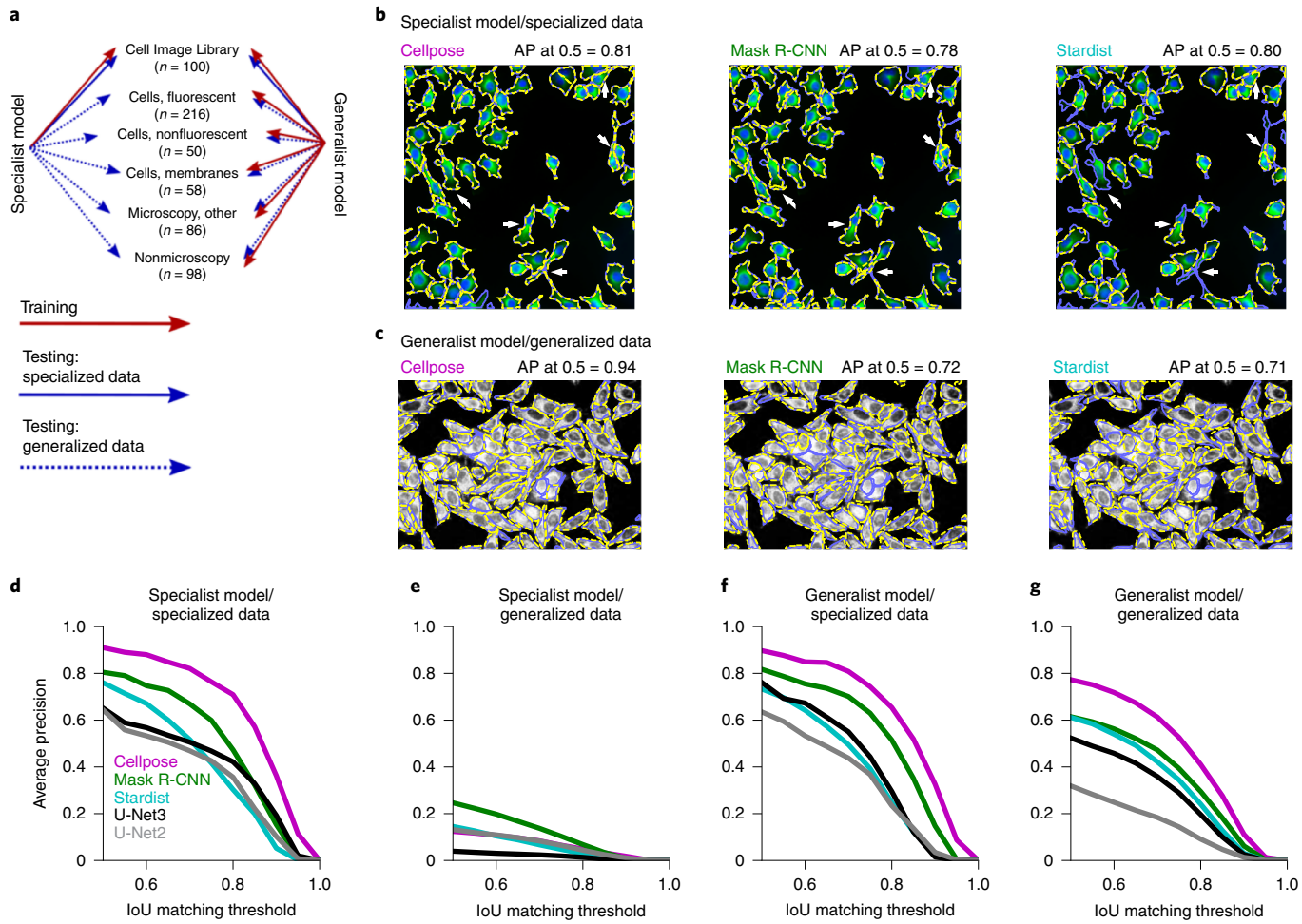


Fig. 4 | Segmentation performance of specialist and generalist algorithms. a, Illustration of the training and testing data for specialist and generalist algorithms. We refer to the Cell Image Library dataset as a 'specialist dataset' due to the uniformity of its 100 images. **b**, Example test image segmentation for Cellpose, Mask R-CNN and Stardist when trained as specialist models. The ground truth is shown as a blue line, while the model predictions are shown with a dotted yellow line. **c**, Example test image segmentation for the same models trained as generalist models. **d**, Quantified performance of Cellpose, Mask R-CNN, Stardist, U-Net3 and U-Net2 models, trained as specialists and tested on specialized data ($n = 11$ test images). The IoU threshold quantifies the match between a predicted mask and a ground-truth mask, with 1 indicating a pixel-perfect match and 0.5 indicating as many correctly matched pixels as there were missed and false positive pixels. The average precision score is computed from the proportion of matched and missed masks. To gain an intuition for the range of these scores, refer to the reported values in **b**,**c** as well as Fig. 3. **e**, Performance of the specialist models on generalized data ($n = 57$ test images). **f**,**g**, Same as **d**,**e** for generalist models.

while Cellpose3D predicted 172 cells, with 17 false positives at an IoU threshold of 0.5 (Fig. 6g, Videos 1 and 2). At all IoU thresholds, Cellpose3D outperformed the ilastik pipeline. Cellpose3D also outperformed the 2D-stitched versions of Cellpose and other models by a substantial margin (Fig. 6g). Finally, Cellpose3D outperformed the U-Net extensions to 3D.

Discussion

Here we introduced Cellpose, a generalist model that can segment many types of cell without requiring parameter adjustments, new training data or further model retraining. Cellpose uses two main innovations: a reversible transformation from training set masks to vector gradients that can be predicted by a neural network and a large segmented dataset of varied images of cells. In addition, multiple smaller improvements to the basic approach led to a significant cumulative performance increase: we developed methods to use an image 'style' for changing the neural network computation on an image by image basis, to validate segmented ROI, to average the predictions of multiple models, to resize images to a common object size and to average model predictions in overlapping tiles. Finally, we introduced a new method for 3D cell segmentation that reuses the 2D model and does not require new 3D-labeled data.

Our approach to the segmentation problem allows anyone to contribute new training data to improve Cellpose for themselves and for others. We encourage users to contribute up to a few manually segmented images of the same type, which we will use to periodically retrain a single generalist Cellpose model for the community. Cellpose has high expressive power and high capacity, as shown by its ability to segment cells with complex protuberances such as elongated dendrites, and even noncell objects such as rocks and jellyfish. We therefore expect Cellpose to successfully incorporate new training data that has a passing similarity to an image of cells and consists of collections of similar-looking objects. Our initial tests indicate that the inclusion in the training set of unusual, nonbiological images especially helps for generalization to out-of-sample images. However, generalizing to out-of-sample data is a notoriously difficult problem for neural networks (for example, Fig. 4e), and more work will be needed to best take advantage of new types of training data.

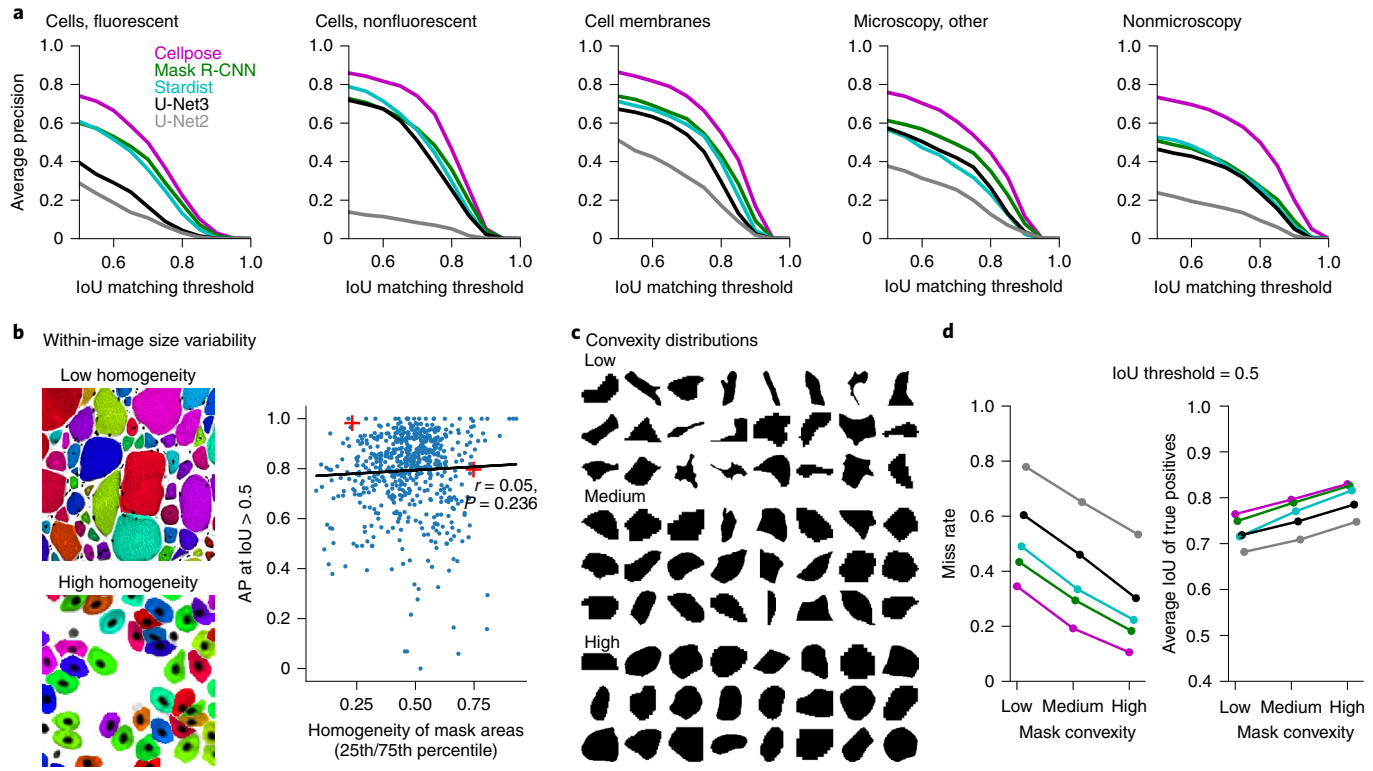


Fig. 5 | Model performance across image types and ROI statistics. a, Average precision for generalist models on subcategories of images from the test set ($n = 17, 5, 9, 13, 13$ test images). **b**, Left panels, example images with low and high homogeneity of ROI sizes, with ground truth masks in random colors. Right panel, Cellpose per-image performance as a function of size homogeneity. Red crosses indicate the example images from the left panels. **c**, Random samples of ROI masks divided into three subsets by convexity percentile. **d**, Left panel, fraction of missed ROI at an IoU threshold of 0.5 for generalized data ($n = 57$ test images). Right panel, average IoU of true positive ROI at IoU ≥ 0.5 for generalized data ($n = 57$ test images).

While our goal here was to develop generalist algorithms for 2D and 3D segmentation, it is also possible to train Cellpose on specialized types of data, such as cryo-EM images[38], provided that a large training dataset exists. Similarly, a 3D version of Cellpose could be trained directly on a 3D ground-truth dataset, which may result in better performance[39]. Other extensions of Cellpose are possible, for example to cell tracking[40], which could be addressed by adding a 'temporal gradient' dimension to the spatial gradient dimensions. Finally, further algorithmic improvements could be targeted toward low convexity cells, which were the least well segmented by all algorithms (Fig. 5d). Combined with new histology methods such as spatial transcriptomics[41], Cellpose has the potential to aid and enable new approaches in quantitative single-cell biology that are reproducible and scalable to large datasets.

Online content

Any methods, additional references, Nature Research reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41592-020-01018-x.

Received: 3 April 2020; Accepted: 11 November 2020;
Published online: 14 December 2020

References

1. Boutros, M., Heigwer, F. & Laufer, C. Microscopy-based high-content screening. *Cell* **163**, 1314–1325 (2015).
2. Sommer, C., Straehle, C., Koethe, U. & Hamprecht, F. A. Ilastik: interactive learning and segmentation toolkit. *IEEE International Symposium on Biomedical Imaging*, 230–233 (2011).
3. Ronneberger, O., Fischer, P. & Brox, T. U-Net: convolutional networks for biomedical image segmentation. Preprint at *arXiv* 1505.04597 (2015).
4. Apthorpe, N. et al. Automatic neuron detection in calcium imaging data using convolutional networks. *Advances in Neural Information Processing Systems* **29**, 3270–3278 (2016).
5. Guerrero-Pena, F. A. et al. Multiclass weighted loss for instance segmentation of cluttered cells. In *Proc. 2018 25th IEEE International Conference on Image Processing (ICIP)* 2451–2455 (IEEE, 2018).
6. Xie, W., Noble, J. A. & Zisserman, A. Microscopy cell counting and detection with fully convolutional regression networks. *Comput. Methods Biomech. Biomed. Eng. Imaging Vis.* **6**, 283–292 (2018).
7. Al-Kofahi, Y., Zaltsman, A., Graves, R., Marshall, W. & Rusu, M. A deep learning-based algorithm for 2-D cell segmentation in microscopy images. *BMC Bioinformatics* **19**, 1–11 (2018).
8. Berg, S. et al. ilastik: interactive machine learning for (bio)image analysis. *Nat. Methods* **16**, 1226–1232 (2019).
9. Schindelin, J. et al. Fiji: an open-source platform for biological-image analysis. *Nat. Methods* **9**, 676–682 (2012).
10. McQuin, C. et al. Cellprofiler 3.0: next-generation image processing for biology. *PLoS Biology* **16**, e2005970 (2018).
11. Carpenter, A. E. et al. Cellprofiler: image analysis software for identifying and quantifying cell phenotypes. *Genome Biol.* **7**, R100 (2006).
12. Chen, J. et al. The Allen cell structure segmenter: a new open source toolkit for segmenting 3D intracellular structures in fluorescence microscopy images. Preprint at https://www.biorxiv.org/content/10.1101/491035v1 (2018).
13. Funke, J., Mais, L., Champion, A., Dye, N. & Kainmueller, D. A benchmark for epithelial cell tracking. *Proceedings of the European Conference on Computer Vision* https://doi.org/10.1007/978-3-030-11024-6_33 (2018).
14. Yi, J. et al. Object-guided instance segmentation for biological images. Preprint at https://arxiv.org/abs/1911.09199 (2019).
15. Caicedo, J. C. et al. Nucleus segmentation across imaging experiments: the 2018 data science bowl. *Nat. Methods* **16**, 1247–1253 (2019).
16. He, K., Gkioxari, G., Dollár, P. & Girshick, R. Mask R-CNN. Preprint at https://arxiv.org/abs/1703.06870 (2018).
17. Abdulla, W. Mask r-cnn for object detection and instance segmentation on keras and tensorflow (GitHub, 2017); https://github.com/matterport/Mask_RCNN

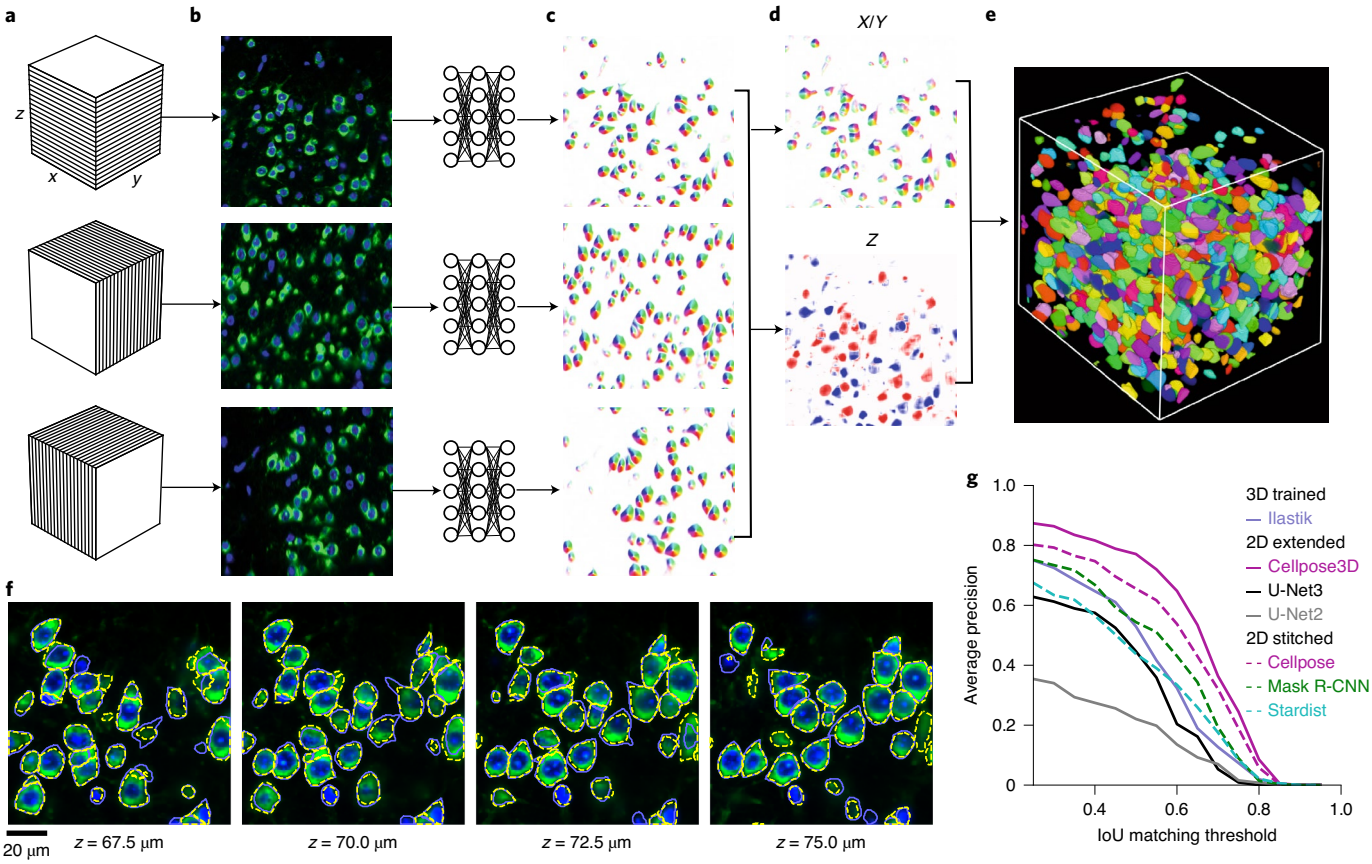


Fig. 6 | Segmentation in 3D without 3D-labeled data. a, A 3D volume is sliced into *xy*, *xz* and *yz* sections. **b**, Example frames for each type of section. **c**, Each 2D frame is passed through the Cellpose prediction network (trained on generalized data, Fig. 4f,g), generating predicted 2D gradient maps. **d**, The six predicted gradient maps (two each for *x*, *y* and *z*) are pairwise averaged and combined into a single 3D gradient vector field *xyz*. Shown are the combined *xy* gradient map as well as the *z* gradient map, for a single *xy* slice through the volume. **e**, The 3D gradient maps are used to create pixel dynamics toward the 3D sinks of the gradient vector field, and the pixels are grouped into 3D masks if they converge to the same sink. **f**, Example *xy* slices with ground-truth (blue) and predicted (dotted yellow) masks at different depths in the *z* stack. **g**, Average precision at different IoU thresholds of Cellpose3D compared to other models ($n = 183$ cells in test volume).

18. Schmidt, U., Weigert, M., Broaddus, C. & Myers, G. Cell detection with star-convex polygons. *International Conference on Medical Image Computing and Computer-Assisted Intervention* 265–273 (2018).
19. Hollandi, R. et al. A deep learning framework for nucleus segmentation using image style transfer. Preprint at https://www.biorxiv.org/content/10.1101/580605v1 (2019).
20. Van Rossum, G. & Drake, F. L. *Python 3 Reference Manual* (CreateSpace, 2009).
21. Harris, C. R. et al. Array programming with numpy. *Nature* **585**, 357–362 (2020).
22. Jones, E. et al. SciPy: open source scientific tools for Python (SciPy, 2001); http://www.scipy.org/
23. Lam, S. K., Pitrou, A. & Seibert, S. Numba: a llvm-based python jit compiler. In *Proc. Second Workshop on the LLVM Compiler Infrastructure in HPC* 7 (2015); https://doi.org/10.1145/2833157.2833162
24. Bradski, G. The OpenCV Library. *Dr. Dobb's J. Softw. Tools* **25**, 120–125 (2000).
25. Chen, T. et al. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. Preprint at https://arxiv.org/abs/1512.01274 (2015).
26. Summerfield, M. *Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming* (Pearson Education, 2007).
27. Campagnola, L. Scientific graphics and GUI library for Python; http://pyqtgraph.org/
28. Beucher, S. & Meyer, F. in *Mathematical Morphology in Image Processing* (ed. Dougherty, E. R.) 433–481 (Marcel Dekker, 1993).
29. Li, G. et al. Segmentation of touching cell nuclei using gradient flow tracking. *J. Microsc.* **231**, 47–58 (2008).
30. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition* 770–778 (2016).
31. Gatys, L. A., Ecker, A. S. & Bethge, M. Image style transfer using convolutional neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition* 2414–2423 (2016).
32. Karras, T., Laine, S. & Aila, T. A style-based generator architecture for generative adversarial networks. *Proceedings of the IEEE conference on computer vision and pattern recognition* 4401–4410 (2019).
33. OMERO. Image data resource (IDR, 2020); https://idr.openmicroscopy.org/cell/
34. Williams, E. et al. Image data resource: a bioimage data integration and publication platform. *Nat. Methods* **14**, 775–781 (2017).
35. Maaten, Lvd & Hinton, G. Visualizing data using t-SNE. *J. Machine Learning Res.* **9**, 2579–2605 (2008).
36. Pedregosa, F. et al. Scikit-learn: Machine learning in python. *J. Machine Learning Res.* **12**, 2825–2830 (2011).
37. Yu, W., Lee, H. K., Hariharan, S., Bu, W. Y. & Ahmed, S. Ccdb:6843, *Mus musculus*, neuroblastoma. *Cell Image Library* https://doi.org/10.7295/W9CCDB6843
38. Bai, X.-c, McMullan, G. & Scheres, S. H. How cryo-em is revolutionizing structural biology. *Trends Biochem. Sci.* **40**, 49–57 (2015).
39. Weigert, M., Schmidt, U., Haase, R., Sugawara, K. & Myers, G. Star-convex polyhedra for 3D object detection and segmentation in microscopy. Preprint at https://arxiv.org/abs/1908.03636 (2019).
40. Ulman, V. et al. An objective comparison of cell-tracking algorithms. *Nat. Methods* **14**, 1141–1152 (2017).
41. Ståhl, P. L. et al. Visualization and analysis of gene expression in tissue sections by spatial transcriptomics. *Science* **353**, 78–82 (2016).

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© The Author(s), under exclusive licence to Springer Nature America, Inc. 2020

Methods

The Cellpose code library is implemented in Python 3 (ref. [20]), using numpy, scipy, numba, opencv and the deep learning package mxnet[21–25]. The graphical user interface additionally uses PyQt and pyqtgraph[26,27]. The figures were made using matplotlib and jupyter-notebook[44,45].

Datasets. *Animals.* For data collected by us, all experimental procedures were approved by the Janelia IACUC (Institutional Animal Care and Use Committee) at Janelia Research Campus, Howard Hughes Medical Institute. All mice were housed at 21 °C ± 2 °C and humidity 30–70% in a 12:12 light-dark cycle.

Specialized data. This dataset consisted of 100 fluorescent images of cultured neurons with cytoplasmic and nuclear stains obtained from the Cell Image Library[37]. Each image has a manual cytoplasmic segmentation. We corrected some of the manual segmentation errors where touching neurons were joined, and removed any overlaps in the segmentation so that each pixel in the image was assigned to a maximum of one cytoplasmic mask. A small number of dendrites were cut off by this process, but the bulk of their cell bodies was unmodified.

Generalized data. The full dataset included the 100 images described above, as well as 508 additional images from various sources detailed below. All these extra images were fully manually segmented by a single human operator (M.M.). Of these images, 69 were reserved for testing.

Two hundred and sixteen of the images contained cells with fluorescent cytoplasmic markers. We used BBBC020 from the Broad Bioimage Benchmark Collection[46], which consisted of 25 images of bone-marrow derived macrophages from C57BL/6 mice. We also used BBBC007v1 image set v.1 (ref. [47]), which consisted of 15 fluorescent images of *Drosophila melanogaster* Kc167 cells with stains for DNA and actin. We imaged in vivo mouse cortical and hippocampal cells expressing GCaMP6 using a two-photon microscope, and included eight images (mean activity) of such data (two C57BL/6 female mice 3–8 months old). We also used ten images from confocal imaging of mouse cortical neurons with cytoplasmic and nuclear markers from a similar dataset to that which we used for 3D segmentation. The other images in this set were obtained through Google image searches.

Fifty of the images were taken with standard brightfield microscopy. There were four images shared via OMERO[33] of pancreatic stem cells on a polystyrene substrate taken using a light microscope[48]. The other images in this set were obtained through Google image searches.

In 58 of the images, the cell membrane was fluorescently labeled. We used the Micro-Net image set, which consisted of 40 fluorescent images of mouse pancreatic exocrine cells and endocrine cells with a membrane marker (Ecad-FITC) and a nuclear marker (DAPI)[49]. Because the nuclear marker did not appear in focus with the membrane labels, we discarded this channel and resegmented all images according to the membrane marker exclusively. The other images in this set were obtained through Google image searches.

Eighty-six of the images were other types of microscopy sample that were either not cells or cells with atypical appearances. These images were obtained through Google image search.

Ninety-eight of the images were nonmicroscopy images of repeating objects. These images were obtained through Google image search, and include images of fruits, vegetables, artificial materials, fish and reptile scales, starfish, jellyfish, sea urchins, rocks, sea shells and so on.

Nucleus data. This dataset consisted of 1,139 images with manual segmentations from various sources, 111 of which were reserved for testing. We did not segment any of these images ourselves.

We used image set BBBC038v1 (ref. [15]), available from the Broad Bioimage Benchmark Collection. For this image set, we used the unofficial corrected manual labels provided by Konstantin Lopuhin[50].

We used image set BBBC039v1 (ref. [15]), which consists of 200 fluorescent images of U2OS osteosarcoma cells with the Hoechst stain. Some of these images overlap with the BBBC038v1, so we only used the 157 unique images.

We used image set MoNuSeg, which consists of 30 H&E stained histological images of various human organs[51]. Because the images contain many small nuclei (roughly 700 per image), we divided each of these images into nine separate images. We inverted the polarity of these images so that foreground nuclear pixels had higher intensity values than the background, which is more similar to fluorescence images.

We also used the image set from ISBI 2009, which consists of 97 fluorescent images of U2OS cells and NIH3T3 cells[52].

3D data. The tissue was acquired from the cortex of a 10-week-old C57BL/6 male mouse. We used a DNA stain (DAPI) and an RNA stain (fluorescent oligonucleotide probe against the 28S ribosomal RNA) to label the nucleus and cytoplasm, respectively, for a volume of cortical neurons. The tissue was first expanded by a factor of two and digested to remove all proteins[53,54]. The tissue was imaged on a Zeiss Z1 lightsheet microscope using the ZEN Microscope Software. A small cube (roughly $190 \times 190 \times 190\,\mu m^3$) was manually labeled by a human annotator in the Cellpose graphical user interface. We allowed the annotator to potentially skip 'easy' planes while drawing a cell, with an algorithm automatically interpolating the shapes of cells in those planes.

Auxiliary vector flow representation. Central to the Cellpose model is an auxiliary representation that converts the cell masks to two images or 'maps' of the same size as the original image. These maps are similar to the vector fields predicted by tracking models such as OpenPose[55], which point from object parts to other object parts and are used in object tracking and coarse segmentation. In our case, the vector fields lead all pixels within a cell toward its center. The gradients do not necessarily point directly toward the center of the cell[56], because that direction might intersect cell boundaries. Instead, the gradients are designed so that locally they translate pixels to other pixels inside cells, and globally over many iterations translate pixels to the eventual fixed points of the gradient vector field. These fixed points are in turn chosen by us to be the cell centers. Since all pixels from the same cells end up at the same cell center, it is easy to recognize which pixels should be grouped together into ROI, by assigning the same cell label to pixels with the same fates. The task of the neural network is to predict these gradients from the raw image, which is potentially a highly nonlinear transformation. Note the similarity to gradient flow tracking methods, where the gradients are computed directly as derivatives of the raw image brightness[29].

To create a vector flow field with the properties described above, we turned to a heat diffusion simulation. We define the 'center' of each cell as the pixel in a cell that was closest to the median values of horizontal and vertical positions for pixels in that cell. In the heat diffusion simulation, we introduce a heat source at the center pixel, which adds a constant value of one to that pixel's value at each iteration. Every pixel inside the cell gets assigned the average value of pixels in a 3×3 square surrounding it, including itself, at every iteration, with pixels outside of a mask being assigned to zero at every iteration. In other words, the boundaries of the cell mask are 'leaky'. This process gets repeated for N iterations, where N is chosen for each mask as twice the sum of its horizontal and vertical range, to ensure that the heat dissipates to the furthest corners of the cell. The distribution of heat at the end of the simulation approaches the equilibrium distribution. We use this final distribution as an energy function, whose horizontal and vertical gradients represent the two vector fields of our auxiliary vector flow representation. We obtained similar, but very slightly worse, performance by using other definitions of 'center' such as the 2D medoid, and other ways to generate an energy function such as the solution of the Poisson equation[57].

Deep neural network. The input to the neural network was a two-channel image with the primary channel corresponding to the cytoplasmic label, and the optional secondary channel corresponding to nuclei, which in all cases was a DAPI stain. When a second channel was not available, it was replaced with an image of zeros. Raw pixel intensities were linearly scaled for each image so that the 1 and 99 percentiles corresponded to 0 and 1.

Downsampling pass. The neural network was composed of a downsampling pass followed by an upsampling pass, as typical in U-Nets[3]. Both passes were composed of four spatial scales, with each scale composed of two residual blocks and each residual block composed of two convolutions with filter size 3×3, as is typical in residual networks[30]. This resulted in four convolutional maps per spatial scale, followed by maximum pooling to downsample the feature maps. Each convolutional map was preceded by a batchnorm + relu operation, in the order recommended by He et al.[58]. The skip connections were additive identity mappings for the second residual block at each spatial scale. For the first residual block, we used 1×1 convolutions for the skip connections, as in the original residual networks[30], because these convolutions follow a downsampling or upsampling operation where the number of feature maps changes. In mathematical notation, the following operations were performed in each layer on the downsampling pass:

$$\begin{aligned} \mathbf{x}'_t &= \mathcal{D}_{2\times 2}(\mathbf{x}_{t-1}) \\ \mathbf{x}^*_t &= \mathcal{F}(\mathcal{F}(\mathbf{x}'_t)) + \mathcal{P}_{1\times 1}(\mathbf{x}'_t) \\ \mathbf{x}_t &= \mathcal{F}(\mathcal{F}(\mathbf{x}^*_t)) + \mathbf{x}^*_t, \end{aligned}$$

where D_{2x2} indicates the downsampling operation, $\mathcal{F}$ combines the batchnorm, relu and convolution steps, and $\mathcal{P}_{1\times 1}$ is the 1×1 convolution. Different applications of $\mathcal{F}$ are understood to have different parameters, which were fit by gradient descent.

Image style. In between the downsampling and upsampling steps we computed an image style[31], defined as the global average pool of each feature map[32], resulting in a 256-dimensional feature vector for each image. To account for differences in cell density across images, we normalized the feature vector to a norm of one for every image. This style vector was passed as input to the residual blocks on the upsampling pass, after projection to a suitable number of features equal to the number of convolutional feature maps of the corresponding residual block, as described below. In mathematical notation, the style vector $\mathbf{s}^*$ was computed as follows from $\mathbf{x}_4$:

$$\begin{aligned} \mathbf{s}[k] &= \textstyle\sum_{ij} \mathbf{x}_4[k,i,j] \\ \mathbf{s}^* &= \mathbf{s}/||\mathbf{s}|| \end{aligned}$$

where k indexes the feature maps, and i, j are indices for the horizontal and vertical dimensions of the convolutions.

Upsampling pass. On the upsampling pass, we followed the typical U-Net architecture, where the convolutional layers after an upsampling operation take as input not only the previous feature maps, but also the feature maps from the equivalent level in the downsampling pass[3]. We depart from the standard feature concatenation in U-Nets and combine these feature maps additively on the second out of four convolutions per spatial scale[59]. The last three convolutions, but not the first one, also had the style vectors added after a suitable linear projection to match the number of feature maps and a global broadcast to each position in the convolution maps. In mathematical notation, each layer in the upsampling pass performed the following operations:

$$\begin{aligned}\mathbf{z}'_t &= \mathcal{U}_{2\times 2}(\mathbf{z}_{t+1})\\ \mathbf{z}^*_t &= \mathcal{G}(\mathbf{s}^*, \mathcal{F}(\mathbf{z}'_t) + \mathbf{x}_t) + \mathcal{P}_{1\times 1}(\mathbf{z}'_t)\\ \mathbf{z}_t &= \mathcal{G}(\mathbf{s}^*, \mathcal{G}(\mathbf{s}^*, \mathbf{z}^*_t)) + \mathbf{z}^*_t,\end{aligned}$$

where $\mathcal{U}_{2\times 2}$ is the upsampling operation and was skipped for computing $\mathbf{z}_4$ from x_4, while $\mathcal{F}$ is the same operation from the downsampling step and $\mathcal{G}$ additionally adds a dense, broadcasted projection from the style vector:

$$\mathcal{G}(\mathbf{x})[k,i,j] = \mathcal{F}(\mathbf{x})[k,i,j] + \mathcal{P}(\mathbf{s}^*)[k].$$

Finally, the last convolutional map on the upsampling pass was given as given as input to a 1×1 layer of three output convolutional maps. The first two of these were used to directly predict the horizontal and vertical gradients using an L2 loss. The third output map was passed through a sigmoid and used to predict the probability that a pixel is inside or outside of a cell with a cross-entropy loss. To match the relative contributions of the L2 loss and cross-entropy loss, we multiplied the gradients by a factor of five. In mathematical notation, the final outputs $\mathbf{y}$ are computed as:

$$\mathbf{y} = \mathcal{F}(\mathbf{z}_1)$$

and the cost function for Cellpose was:

$$\text{Cost} = ||\mathbf{y}_0 - 5\mathbf{H}||^2 + ||\mathbf{y}_1 - 5\mathbf{V}||^2 + \text{L}(\sigma(\mathbf{y}_2), \mathbf{P})$$

where y_k subselected the feature map k, $\mathbf{H}$ and $\mathbf{V}$ were the ground-truth horizontal and vertical flow fields, $\mathbf{P}$ was the inside/outside binary map, L was the binary cross-entropy loss and $\sigma(x) = 1/(1+\exp(-x))$ was the sigmoid function.

We built and trained the deep neural network using mxnet[25].

Benchmarks for submodules. We verified that it was beneficial to make these architectural changes over the more standard U-Net, by comparing the performance of different models with various elements disabled. To obtain a standard U-Net from our architecture, we needed to apply three changes: (1) disabling the style; (2) replacing residual blocks with standard convolutions and (3) replacing feature addition with concatenation on the downsampling pass. We first checked the performance penalty from each change. Both the style and residual blocks were beneficial to Cellpose, because disabling them reduced the performance of the model substantially (Extended Data Fig. 2a,b). The feature addition step was able to replace concatenation without loss of performance, even though it reduced the number of parameters substantially (Extended Data Fig. 2a,b). Finally, all three changes combined led to the standard U-Net architecture, which was also significantly worse than the Cellpose architecture (Extended Data Fig. 2a,b). The performance differences from disabling each feature were approximately additive (Extended Data Fig. 2a–d).

Training. The networks were trained for 500 epochs with stochastic gradient descent with a learning rate of 0.2, a momentum of 0.9, batch size of eight images and a weight decay of 0.00001. The learning rate was started at zero and annealed linearly to 0.2 over the first ten epochs to prevent initial instabilities. The value of the learning rate was chosen to minimize training set loss (we also tested 0.01, 0.05, 0.1 and 0.4). At epoch 400, the learning rate annealing schedule was started, reducing the learning rate by a factor of two every ten epochs. For all analyses in the paper we used a base of 32 feature maps in the first layer, growing by a factor of two at each downsampling step and up to 256 feature maps in the layer with the lowest resolution. Separate experiments on a validation set held out from the main training dataset confirmed that increases to a base of 48 feature maps were not helpful, while decreases to 24 and 16 feature maps hurt the performance of the algorithm.

Image augmentations for training. On every epoch, the training set images are randomly transformed together with their associated vector fields and pixel inside/outside maps. For all algorithms, we used random rotations, random scalings and random translations, and then sampled a 224×224 image from the center of the resultant image. For Cellpose, the scaling was composed of a scaling to a common size of cells, followed by a random scaling factor between 0.75 and 1.25. For Mask R-CNN and Stardist, the common size resizing was turned off because these methods cannot resize their predictions. Correspondingly, we used a larger range of scale augmentations between 0.5 and 1.5. Note that Mask R-CNN additonally uses a multi-scale training model based on the size of the objects in the image.

The random rotations were uniformly drawn from 0° to 360°. Random translations were limited along x and y to a maximum amplitude of $(l_x - 224)/2$ and $(l_y - 224)/2$, where l_y and l_x are the sizes of the original image. This ensures that the sample is taken from inside the image. Rotations, but not translations and scalings, result in changes to the direction of vectors. We therefore rotated the mask flow vectors by the same angles the images were rotated.

Mask recovery via gradient flow tracking. The output of the neural network after tiling is a set of three maps: the horizontal gradients, the vertical gradients and the pixel probabilities. The next step is to recover the masks from these. First, we threshold the pixel probability map and only consider pixels above a threshold of 0.5. For each of these pixels, we run a dynamical system starting at that pixel location and following the spatial derivatives specified by the horizontal and vertical gradient maps. We use finite differences with a step size of one. Note that we do not renormalize the predicted gradients, but the gradients in the training set have unit norm, so we expect the predicted gradients to be on the same scale. We run 200 iterations for each pixel, and at every iteration we take a step in the direction of the gradient at the nearest grid location. Following convergence, pixels can be easily clustered according to the pixel they end up at. For robustness, we also extend the clusters along regions of high-density of pixel convergence. For example, if a high-density peak occurs at position (x, y), we iteratively agglomerate neighboring positions that have at least three converged pixels until all the positions surrounding the agglomerated region have less than 3 pixels. This ensures success in some cases where the deep neural network is not sure about the exact center of a mask, and creates a region of very low gradient where it thinks the center should be.

Test time enhancements. We use several test time enhancements to further increase the predictive power of the model: test time resizing, ROI quality estimation, model ensembling and image tiling with overlaps. We describe them briefly in this paragraph and in more detail below. We estimate the quality of each predicted ROI according to the discrepancy between the predicted flows inside that ROI and the optimal, recomputed flows for that mask. We discard ROI for which this discrepancy is large. Model ensembling is performed by averaging the flow predictions of four models with the same architecture trained separately. Image tiling is performed by dividing the image into overlapping tiles of the size of the training set patches (224×224 pixels). We use 50% overlaps for both horizontal and vertical tiling, resulting in every pixel being processed four times. We then combine the results by averaging the four flow predictions at every pixel, multiplied with appropriate taper masks to minimize edge effects.

Test time resizing. While it is easy to resize training set images to a common object size, such resizing cannot be directly performed on test images because we do not know the average size of objects in that image. In practice, a user might be able to quickly specify this value for their own dataset, but for benchmarks we needed an automated object size prediction algorithm. This information is not readily computable from raw images, but we hypothesized that the image style vectors might be a good representation from which to predict the object size. We predicted object sizes in two steps: (1) we train a linear regression model from the style vectors of the training set images, which is a good but not perfect prediction on the test set, and (2) we refine the size prediction on the test set by running Cellpose segmentation after resizing to the object size predicted by the style vectors. Since this segmentation is already relatively good, we can use its mean object size as a better predictor of the true object size. We found that this refined object size prediction reached a correlation of 0.93 and 0.97 with the ground truth on test images, for the cytoplasm and nuclear dataset respectively (Extended Data Fig. 3). We include this algorithm as a calibration procedure that the user can choose to do either on every image, or just once for their entire dataset. We use this object size to resize test images, run the algorithm to produce the three output maps, and then resize these maps back to the original image sizes before running the pixel dynamics and mask segmentation. For all resizing operations we used standard bilinear interpolation from the OpenCV package[24].

Image tiling. Image tiling is performed for two reasons: (1) to run Cellpose on images of arbitrary sizes and (2) to run Cellpose on the same image patch size used during training (224×224). We start with an image of size l_y by l_x and divide into a set of overlapping tiles, which covers the entire image. For the analyses in this paper, we use an overlap of 50% along each dimension, but in the code package we allow the overlap to be specified by the user. We then run Cellpose on all tiles, producing three output maps for each. The output maps are then used to reconstitute the gradient and probability maps for the entire image, tapering each tile around its four edges with a sigmoid taper with bandwidth parameter of 7.5 pixels. Note that with 50% overlap, almost all pixels in the final reassembled image are averages of four Cellpose runs for different tiles.

Mask quality threshold. Note there is nothing keeping the neural network from predicting horizontal and vertical flows that do not correspond to any real shapes

at all. In practice, most predicted flows are consistent with real shapes, because the network was only trained on image flows that are consistent with real shapes, but sometimes when the network is uncertain it may output inconsistent flows. To check that the recovered shapes after the flow dynamics step are consistent with real masks, we recompute the flow gradients for these putative predicted masks and compare them to the ones predicted by the network. When there is a large discrepancy between the two sets of flows, as measured by their mean squared difference, we exclude that mask since it is inconsistent. We cross-validate the threshold for this operation on a validation set held out from the main training dataset, and found that a value of 0.4 was optimal.

3D models. We extend the 2D model to 3D without the need for 3D-labeled data by running the network on various slices of the 3D stack. Slices in *xy* provide *x* and *y* flow information, slices in *xz* provide *x* and *z* flow information and slices in *yz* provide *y* and *z* flow information (Fig. 6a). We average over the two estimates of each flow at each pixel (Fig. 6d). From each slice, we also predict the cell probability and we average across these three estimates for each pixel. We threshold the cell probability at 0.5 and multiply it by the flows. We then use these flows to run the dynamics to create the mask estimates (Fig. 6e). Objects smaller than 10% of the median cell volume (2,000 voxels) were discarded (Fig. 6f,g). The default median diameter was used (30 pixels).

Stitching 2D segmentations into 3D objects. Methods such as Stardist and Mask R-CNN do not have direct extensions to 3D. It was therefore necessary to extend these methods by starting from segmentations of 2D slices throughout the volume. ROI from consecutives slices were marked as 'connected' if they had an IoU ≥ 0.25, a parameter value chosen to maximize the final 3D performance. Using these connections, we then extracted connected components across the *z* dimension as the 3D ROI. As for the other 3D methods, ROI were dropped if they had a volume of less than 10% of the average ground-truth ROI volume.

Using ilastik for 3D segmentation. As a comparison to Cellpose for 3D segmentation, we used ilastik[8] in a two-step process. In the first step, we classify each voxel as 'background,' 'nucleus,' or 'cytoplasm' using a supervised algorithm. After manually annotating a small number of voxels, we verified that the algorithm could classify individual pixels reliably on test data. Additional training data did not help at this step, which is typical for the random forest classifiers used by ilastik[8]. The following features were used for the classification: Gaussian smoothing (sigma = 0, 3, 1, 3.5 and 10), Gaussian gradient magnitude (sigma = 0.7, 1.6 and 5), difference of Gaussians (sigma = 0.7, 1.6 and 5), structure tensor eigenvalues (sigma = 0.7, 1.6 and 5) and Hessian of Gaussian eigenvalues (sigma = 0.7, 1.6 and 5).

Following the per-pixel classification, individual objects (that is, cells) were identified using the hysteresis thresholding method. This step is unsupervised and thus does not require training data. The 'nucleus' and 'cytoplasm' probability maps generated from the first step were used for setting the high (that is, 'core') and low (that is, 'final') thresholds, respectively. Before thresholding, an isotropic Gaussian blur was applied (sigma of 1) and thresholds of 0.85 and 0.4 were chosen. Finally, objects smaller than 10% of the median cell volume (2,000 voxels) were discarded (same threshold we used for Cellpose).

Benchmarking. We compared the segmentations performed by Cellpose to segmentations performed by Stardist[18], Mask R-CNN[16,17] and U-Nets[3,5].

Training two- and three-class U-Nets. To train U-Nets[3], we followed the strategies from ref. [5]. The two-class U-Net simply predicted pixels as inside/outside of cells, while the three-class U-Net predicted a third category of boundary pixels. Boundary pixels were defined as being less than *N* pixels away from border pixels and inside the cell, where *N* was adjusted on a per-image basis as 10% of the average ROI diameter. We used the standard U-Net architecture in all cases[3]. At test time, we made ROI predictions by thresholding the 'inside' probability map and extracting all connected regions. For the three-class U-Net, we then added the boundary pixels to the nearest connected region[5]. To extend U-Nets to 3D, we applied them on 2D slices and then ran the connected region step on the entire concatenated 3D volume.

Training Stardist and Mask R-CNN. Stardist and Mask R-CNN were trained on the same training sets as Cellpose for the 'specialized,' 'generalized' and 'nuclei' datasets. 12% of the training set was used for validation for each algorithm. Stardist and Mask R-CNN were trained for 1,000 epochs. The learning rates were optimized to reduce the training error—this resulted in a learning rate of 0.0007 for Stardist and a learning rate of 0.001 for Mask R-CNN. For Mask R-CNN, TRAIN_ROIS_PER_IMAGE was increased to 300, MAX_GT_INSTANCES to 200 and DETECTION_MAX_INSTANCES to 400. Mask R-CNN was initialized with the pretrained 'imagenet' weights. All other parameters and learning schedules for both algorithms were kept to their default values.

Quantification of segmentation quality. We quantified the predictions of the algorithms by matching each predicted mask to the ground-truth mask that is most similar, as defined by the IoU. Then we evaluated the predictions at various levels of IoU; at a lower IoU, fewer pixels in a predicted mask have to match a corresponding ground-truth mask for a match to be considered valid. The valid matches define the true positives, TP, the masks with no valid matches are false positives, FP, and the ground-truth masks that have no valid match are false negatives, FN. Using these values, we computed the standard average precision metric (AP) for each image:

$$\mathrm{AP} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}.$$

The average precision reported is averaged over the average precision metric for each image in the test set. This is equivalent to using the 'matching_dataset' function from the Stardist package with the 'by_image' option set to True[60].

We also quantified the segmentation quality using boundary prediction metrics[42] and the Aggregated Jaccard Index[43]. See the respective papers for the exact definitions.

Quantification of ROI properties. As a measure of convexity of each ROI we used the solidity index[61], which is equal to the ratio between the area of the ROI and the area of its convex hull. Each ground-truth ROI was matched at an IoU of 0.5 and the fraction of missed ROI for each convexity percentile was reported in Fig. 5d, together with the average IoU of the matched ROI. Note that this analysis ignores false positives.

Within-image size variability. For each image, we computed the 25th and 75th percentiles of the distribution of pixel counts for all ROI in that image. The ratio of these numbers is between 0 and 1, and can be used as a measure of homogeneity, with a high number indicating a narrow distribution of sizes. We obtained a single homogeneity index per image for every image in our dataset and compared it with the segmentation performance of that image from the corresponding fold in the cross-validation where that image was a test image.

Statistical analyses. We performed Wilcoxon two-sided signed-rank tests in Extended Data Fig. 6 ($n = 11$ and 57 test images).

Ethics declaration. All experimental procedures were approved by the Janelia IACUC at Janelia Research Campus, Howard Hughes Medical Institute.

Reporting Summary. Further information on research design is available in the Nature Research Reporting Summary linked to this article.

Data availability

The manually segmented cytoplasmic dataset is available at www.cellpose.org/dataset and https://doi.org/10.25378/janelia.13270466.

Code availability

The reviewed version of Cellpose is available as Supplementary Software. The code, graphical user interface and updated versions are available at www.github.com/mouseland/cellpose. To test out the model directly on the web, visit www.cellpose.org. Note that the test time augmentations and tiling, which improve segmentation, are not performed on the website to save computation time.

References

42. Arbelaez, P., Maire, M., Fowlkes, C. & Malik, J. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **33**, 898–916 (2010).
43. Kumar, N. et al. A dataset and a technique for generalized nuclear segmentation for computational pathology. *IEEE Trans. Med. Imag.* **36**, 1550–1560 (2017).
44. Hunter, J. D. Matplotlib: a 2D graphics environment. *Comput. Sci. Eng.* **9**, 90–95 (2007).
45. Kluyver, T. et al. Jupyter notebooks-a publishing format for reproducible computational workflows. *ELPUB* 87–90 (2016).
46. Ljosa, V., Sokolnicki, K. L. & Carpenter, A. E. Annotated high-throughput microscopy image sets for validation. *Nat. Methods* **9**, 637–637 (2012).
47. Jones, T. R., Carpenter, A. & Golland, P. Voronoi-based segmentation of cells on image manifolds. *International Workshop on Computer Vision for Biomedical Image Applications* 535–543 (2005).
48. Rapoport, D. H., Becker, T., Mamlouk, A. M., Schicktanz, S. & Kruse, C. A novel validation algorithm allows for automated cell tracking and the extraction of biologically meaningful parameters. *PLoS ONE* **6**, e27315 (2011).
49. Raza, S. E. A. et al. Micro-Net: a unified model for segmentation of various objects in microscopy images. *Med. Image Anal.* **52**, 160–173 (2019).
50. Lopuhin, K. kaggle-dsbowl-2018-dataset-fixes (GitHub, 2018); https://github.com/lopuhin/kaggle-dsbowl-2018-dataset-fixes
51. Kumar, N. et al. A multi-organ nucleus segmentation challenge. *IEEE Trans. Med. Imag.* **39**, 1380–1391 (2019).

52. Coelho, L. P., Shariff, A. & Murphy, R. F. Nuclear segmentation in microscope cell images: a hand-segmented dataset and comparison of algorithms. *IEEE International Symposium on Biomedical Imaging: From Nano to Macro* 518–521 (2009).
53. Chen, F., Tillberg, P. W. & Boyden, E. S. Expansion microscopy. *Science* **347**, 543–548 (2015).
54. Chen, F. et al. Nanoscale imaging of RNA with expansion microscopy. *Nat. Methods* **13**, 679–684 (2016).
55. Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E. & Sheikh, Y. OpenPose: realtime multi-person 2D pose estimation using part affinity fields. Preprint at *arXiv* 1812.08008 (2019).
56. Neven, D., Brabandere, B. D., Proesmans, M. & Gool, L. V. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 8837–8845 (2019).
57. Gorelick, L., Galun, M., Sharon, E., Basri, R. & Brandt, A. Shape representation and classification using the poisson equation. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**, 1991–2005 (2006).
58. He, K., Zhang, X., Ren, S. & Sun, J. Identity mappings in deep residual networks. Preprint at *arXiv* 1603.05027 (2016).
59. Chaurasia, A. & Culurciello, E. LinkNet: exploiting encoder representations for efficient semantic segmentation. In *Proc. 2017 IEEE Visual Communications and Image Processing (VCIP)* 1–4 (IEEE, 2017).
60. Schmidt, U. & Weigert, M. StarDist—object detection with star-convex shapes (GitHub, 2019); http://github.com/mpicbg-csbd/stardist
61. Mingqiang, Y., Kidiyo, K. & Joseph, R. A survey of shape feature extraction techniques. *Pattern Recog.* **15**, 43–90 (2008).

Acknowledgements

This research was funded by the Howard Hughes Medical Institute at the Janelia Research Campus. We thank P. Tillberg and members of the Tillberg laboratory for advice related to expansion microscopy, and W. Sun for sharing calcium imaging data from mouse hippocampus. We also thank S. Saalfeld and J. Funke for useful discussions.

Author contributions

C.S. and M.P. designed the study. T.W. acquired data. M.M. manually segmented images. C.S., M.P. and T.W. performed data analysis. C.S. and M.P. wrote the manuscript, with feedback from T.W.

Competing interests

The authors declare no competing interests.

Additional information

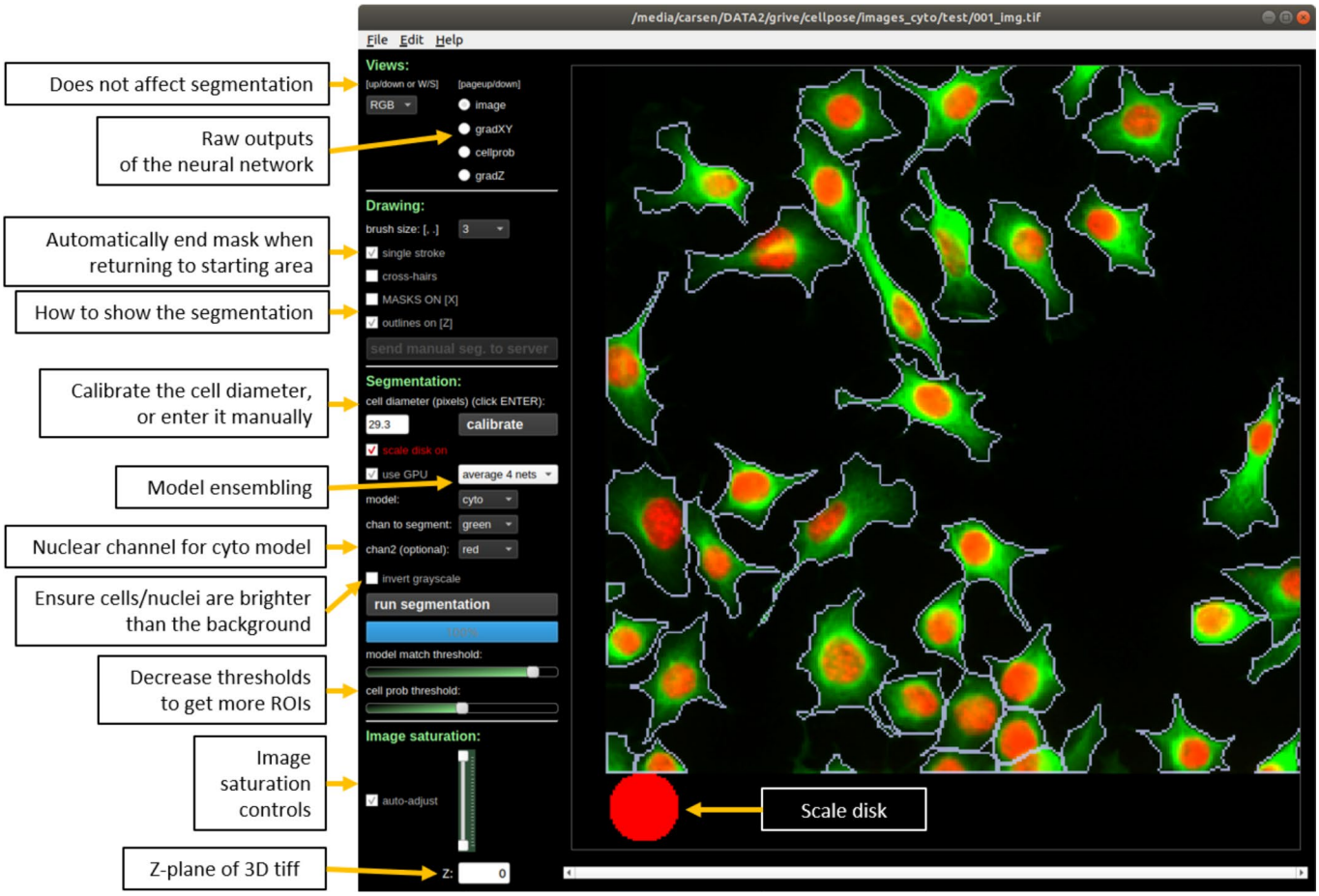
Extended data is available for this paper at https://doi.org/10.1038/s41592-020-01018-x.

Supplementary information is available for this paper at https://doi.org/10.1038/s41592-020-01018-x.

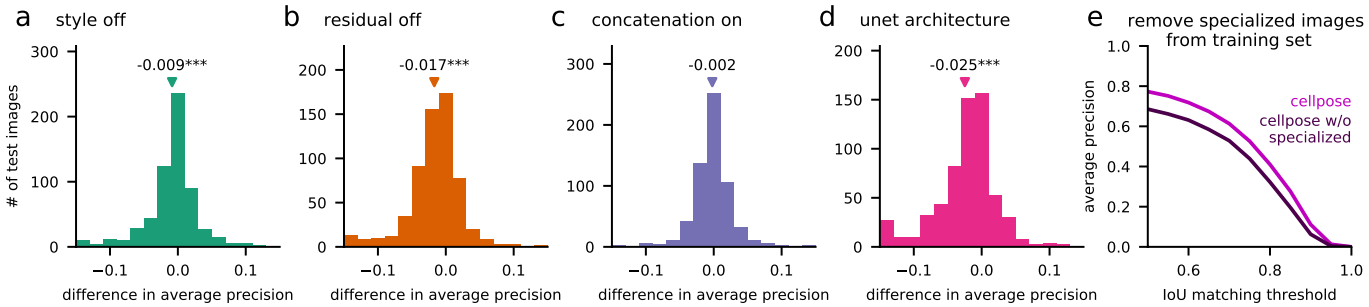
Correspondence and requests for materials should be addressed to M.P.

Reprints and permissions information is available at www.nature.com/reprints.

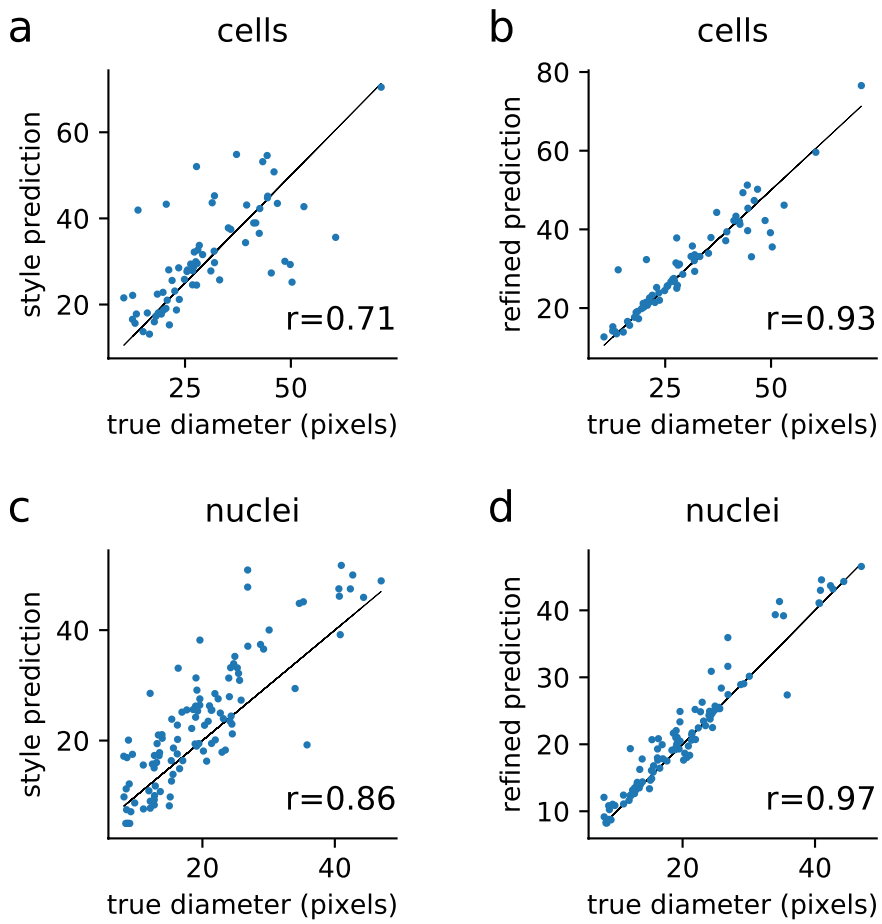
Peer review information Rita Strack was the primary editor on this article and managed its editorial process and peer review in collaboration with the rest of the editorial team.



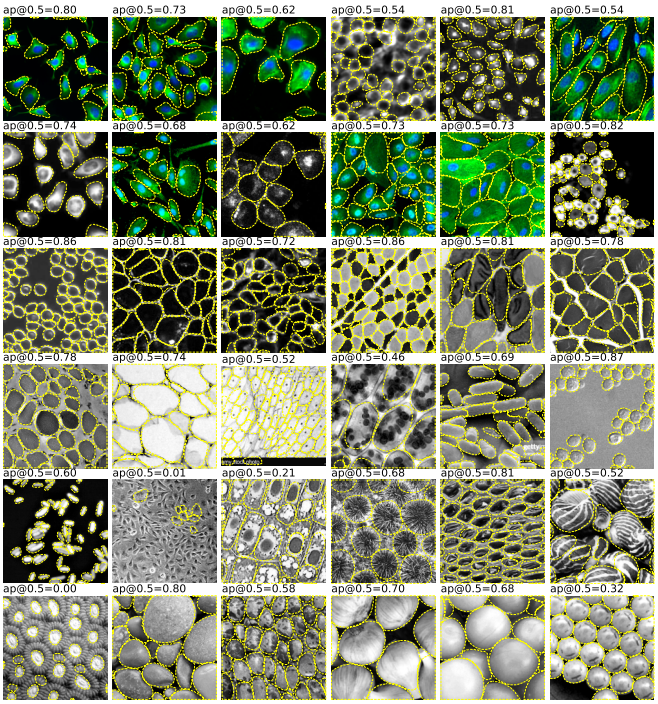
Extended Data Fig. 1 | Graphical User Interface (GUI). Shown in the GUI is an image from the test set, zoomed in on an area of interest, and segmented using Cellpose. The GUI serves two main purposes: 1) easily run Cellpose 'out-of-the-box' on new images and visualize the results in an interactive mode; 2) manually segment new images, to provide training data for Cellpose. The image view can be changed between image channels, cellpose vector flows and cellpose predicted pixel probabilities. Similarly, the mask overlays can be changed between outlines, transparent masks or both. The size calibration procedure can be run to estimate cell size, or the user can directly input the cell diameter, with an image overlay of a red disk shown as an aid for visual calibration. Dense, complete manual segmentations can be uploaded to our server with one button press, and the latest trained model can be downloaded from the drop-down menu.



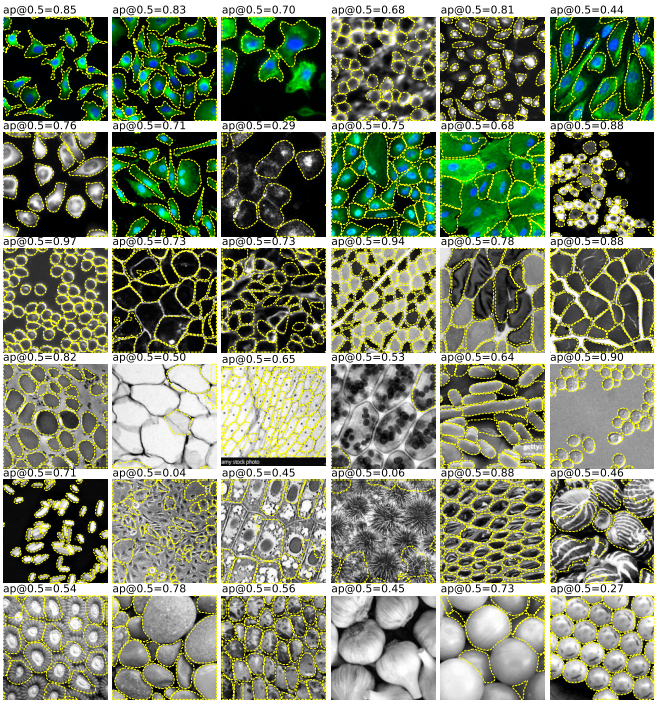
Extended Data Fig. 2 | Effect of network architecture and training set exclusions. a-d, Each analysis compares average precision relative to the standard 4-network average, full Cellpose architecture. The following changes were made: **a**, removed the style vectors; **b**, replaced residual blocks with standard convolutional layers; **c**, Replaced addition with concatenation on the upsampling pass; **d**, standard Unet architecture (in other words **a**, **b** and **c** combined). **e**, Performance on generalized data for the 'generalist' Cellpose model, and a version of the model trained without the 'specialized' data.



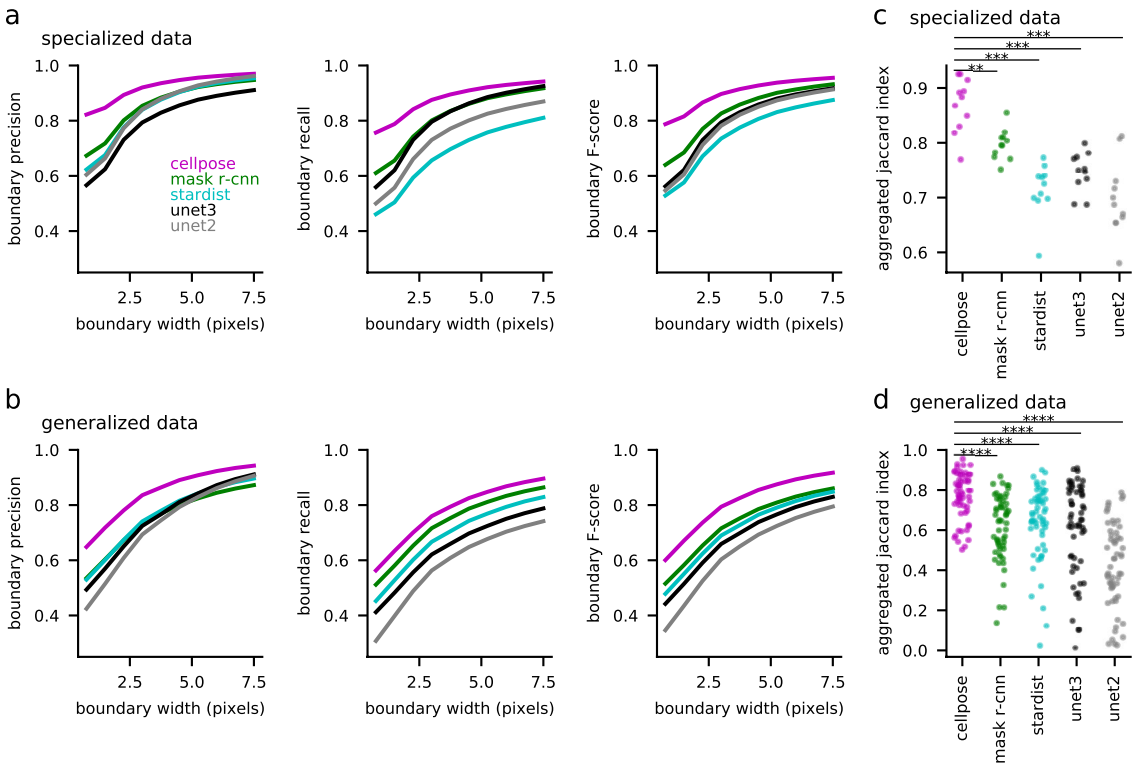
Extended Data Fig. 3 | Prediction of median object diameter. a, The style vectors were used as linear regressors to predict the diameter of the objects in each image. Shown are the predictions for 68 test images (specialized and generalized data together), which were not used either for training cellpose or for training the size model. **b**, The refined size predictions are obtained after running cellpose on images resized according to the sizes computed in **a**. The median diameter of resulting objects is used as the refined size prediction for the final cellpose run. **cd**, Same as **ab** for the nucleus dataset.



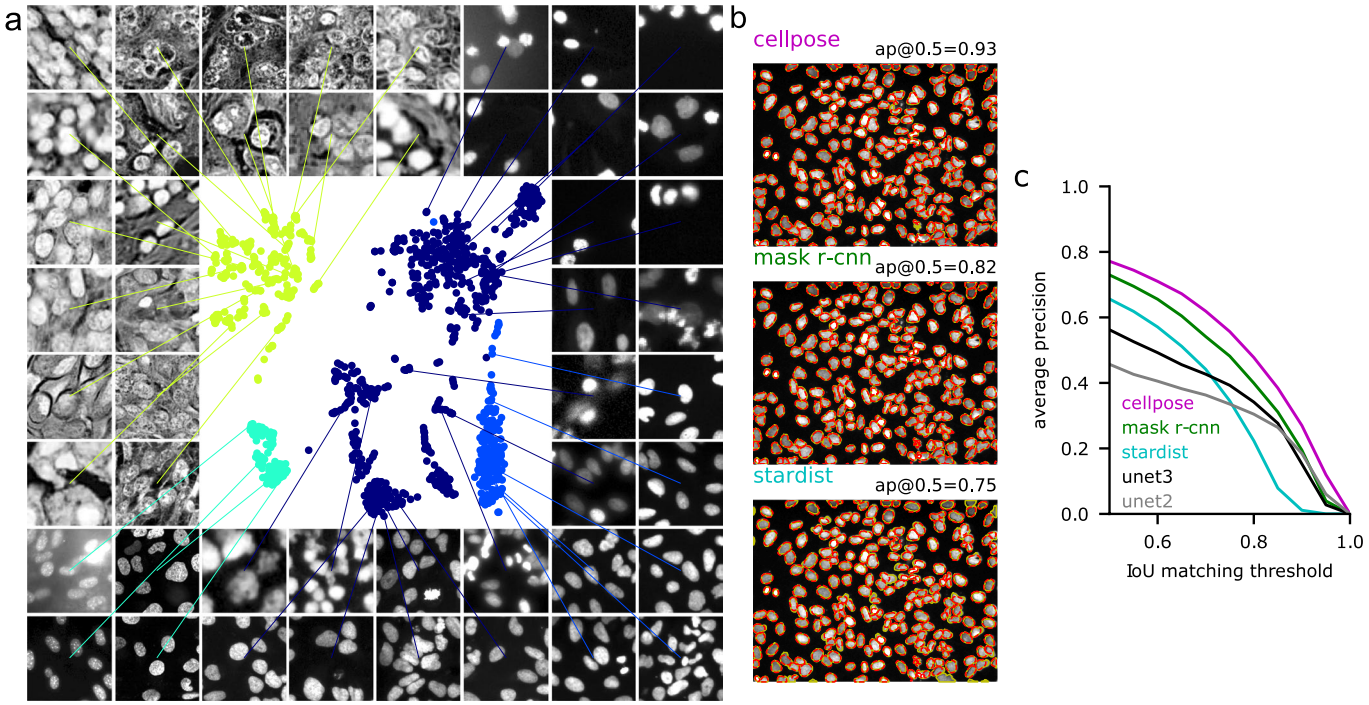
Extended Data Fig. 4 | Example Stardist segmentations. Compare to Fig. 4.



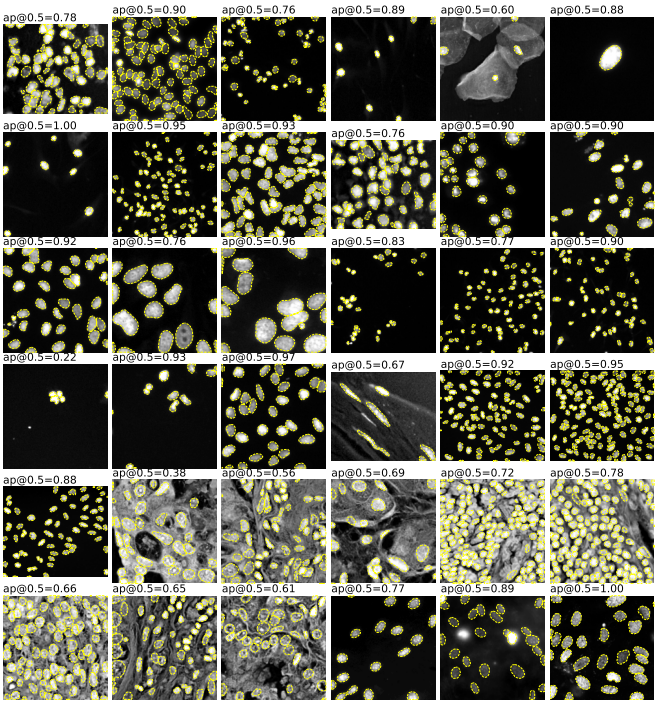
Extended Data Fig. 5 | Example Mask R-CNN segmentations. Compare to Fig. 4.



Extended Data Fig. 6 | Other performance metrics. ab, Boundary prediction metrics (precision, recall, F-score[42]) for: **a**, specialized and **b**, generalized data. All images of masks were first resized to an average ROI diameter of 30 pixels so that a common boundary width can be used across images. **cd**, Aggregated Jaccard Index[43] for: **c**, specialized (n=11 test images) and **d**, generalized data (n=57 test images). **, $p < 0.01$; ***, $p < 0.001$; ****, $p < 0.0001$; Wilcoxon two-sided signed-rank test.



Extended Data Fig. 7 | Benchmarks for dataset of nuclei. a, Embedding of image styles for the nuclear dataset of 1139 images, with a new Cellpose model trained on this dataset. Legend: dark blue = Data Science Bowl dataset, blue = extra kaggle, cyan = ISBI 2009, green = MoNuSeg. **b**, Segmentations on one example test image. **c**, Accuracy precision scores on test data for Cellpose, Mask R-CNN, Stardist, unet3, and unet2 on n=111 test images.



Extended Data Fig. 8 | Example cellpose segmentations for nuclei. The predicted masks are shown in dotted yellow line.

nature research

Corresponding author(s): Marius Pachitariu

Last updated by author(s): Nov 1, 2020

Reporting Summary

Nature Research wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Research policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed	
☐	☒	The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☒	☐	A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☐	☒	The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐	A description of all covariates tested
☒	☐	A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☐	☒	A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☐	☒	For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐	For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐	For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☐	☒	Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	Imaging of tissue in 3D was performed on a Zeiss Z1 lightsheet microscope using the ZEN Microscope Software v9.2.0.0.
Data analysis	The cellpose code uses Python 3 and the numpy, scipy, numba, opencv, and mxnet packages. The cellpose graphical interface uses pyqt and pyqtgraph. All cellpose code is at http://github.com/mouseland/cellpose. We compared cellpose segmentation performance to stardist v0.3.6 (https://github.com/mpicbg-csbd/stardist), Mask R-CNN v2.1 (https://github.com/matterport/Mask_RCNN), and Ilastik (https://www.ilastik.org) segmentation performance. The figures in the paper were made using matplotlib v3.1.3, jupyter v1.0.0, and cellpose v0.1.0.1.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Research guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A list of figures that have associated raw data
- A description of any restrictions on data availability

The manually segmented cytoplasmic dataset is available at www.cellpose.org/dataset.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	No sample-size calculation was performed because this is a computational study.
Data exclusions	No data was excluded from the study.
Replication	Replication not applicable because this is a computational study.
Randomization	Random allocation of images into training and testing sets.
Blinding	Blinding not applicable to study because this is a computational study.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology and archaeology
☐	☒ Animals and other organisms
☒	☐ Human research participants
☒	☐ Clinical data
☒	☐ Dual use research of concern

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging

Animals and other organisms

Policy information about studies involving animals; ARRIVE guidelines recommended for reporting animal research

Laboratory animals	3 C57BL/6 mice (male and female), from 2-10 months old, housed at 70F +/- 2 and humidity 30%-70% in a 12:12 light-dark cycle.
Wild animals	*Provide details on animals observed in or captured in the field; report species, sex and age where possible. Describe how animals were caught and transported and what happened to captive animals after the study (if killed, explain why and describe method; if released, say where and when) OR state that the study did not involve wild animals.*
Field-collected samples	*For laboratory work with field-collected samples, describe all relevant parameters such as housing, maintenance, temperature, photoperiod and end-of-experiment protocol OR state that the study did not involve samples collected from the field.*
Ethics oversight	All experimental procedures were approved by the Janelia IACUC (Institutional Animal Care and Use Committee) at Janelia Research Campus, Howard Hughes Medical Institute.

Note that full information on the approval of the study protocol must also be provided in the manuscript.