ZeroCostDL4Mic: an open platform to simplify access and use of Deep-Learning in Microscopy

Lucas von Chamier[1 *], Romain F. Laine[1,2 *], Johanna Jukkala[3,4], Christoph Spahn[5], Martina Lerche[3], Sara Hernández-Pérez[3,6], Pieta K. Mattila[3,6], Eleni Karinou[7], Seamus Holden[7], Ahmet Can Solak[8], Alexander Krull[9-11], Tim-Oliver Buchholz[9-11], Florian Jug[9-11], Loïc A Royer[8], Mike Heilemann[5], Guillaume Jacquemet[3,4] ✉, and Ricardo Henriques[1,2] ✉

[1]MRC-Laboratory for Molecular Cell Biology, University College London, London, UK
[2]The Francis Crick Institute, London, UK
[3]Turku Bioscience Centre, University of Turku and Åbo Akademi University, Turku, Finland
[4]Faculty of Science and Engineering, Cell Biology, Åbo Akademi University, Turku, Finland
[5]Institute of Physical and Theoretical Chemistry, Goethe-University Frankfurt, Frankfurt, Germany
[6]Institute of Biomedicine, and MediCity Research Laboratories, University of Turku, Finland
[7]Centre for Bacterial Cell Biology, Biosciences Institute, Faculty of Medical Sciences, Newcastle University, UK
[8]Chan Zuckerberg Biohub, San Francisco, CA, USA
[9]Center for Systems Biology Dresden (CSBD), Dresden, Germany
[10]Max Planck Institute for Molecular Cell Biology and Genetics, Dresden, Germany
[11]Max Planck Institute for Physics of Complex Systems, Dresden, Germany
[*]Equal contributing authors

Deep Learning (DL) methods are increasingly recognised as powerful analytical tools for microscopy. Their potential to outperform conventional image processing pipelines is now well established (1, 2). Despite the enthusiasm and innovations fuelled by DL technology, the need to access powerful and compatible resources, install multiple computational tools and modify code instructions to train neural networks all lead to an accessibility barrier that novice users often find difficult to cross. Here, we present ZeroCostDL4Mic, an entry-level teaching and deployment DL platform which considerably simplifies access and use of DL for microscopy. This is achieved by exploiting computational resources provided by Google Colab, a free, cloud-based service accessible through a web browser. ZeroCostDL4Mic allows researchers with little or no coding expertise to quickly train, assess and use popular DL networks. In parallel, it guides researchers to improve their understanding and to experiment with optimising DL parameters and network architectures. Altogether, ZeroCostDL4Mic accelerates the uptake of DL for new users and promotes their capacity to use increasingly complex DL networks.

Microscopy | Deep Learning | Machine Learning | Google Colab | Segmentation | Denoising | Image Restoration | Artificial Labelling

Correspondence: ***r.laine@ucl.ac.uk, guillaume.jacquemet@abo.fi and r.henriques@ucl.ac.uk***

ZeroCostDL4Mic is a collection of self-explanatory Jupyter Notebooks for Google Colab, featuring an easy-to-use graphical user interface (Fig. S1). It complements current community efforts to simplify access to DL in microscopy, e.g. ImJoy (3) or integration projects of DL into Fiji/ImageJ (4–7). It differs from these solutions by providing a single simple interface to install, use and learn about popular DL networks as well as quantitatively assess the performance of the trained models. This is possible by exploiting the free high-performance computational resources provided by Google Colab (Fig. 1a). Using ZeroCostDL4Mic does not require prior knowledge of coding since researchers can, in a few mouse clicks and with the help of an easily accessible workflow, install all needed software dependencies, upload their imaging data and run networks for training and prediction

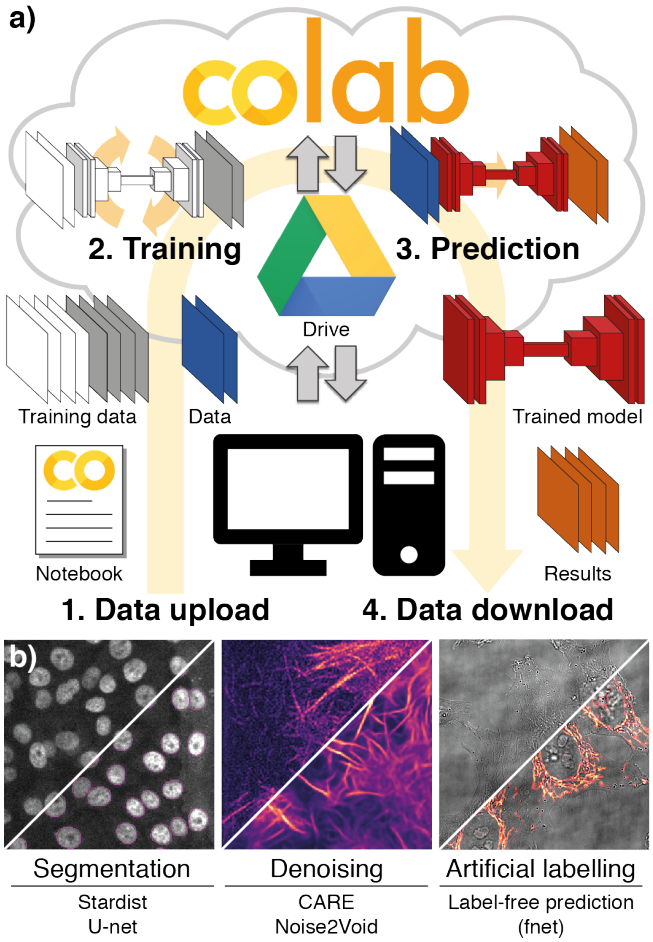


Fig. 1. Overview of ZeroCostDL4Mic a) Workflow of ZeroCostDL4Mic with data transfer via Google Drive and training and prediction on Google Colab. b) Overview of the networks currently implemented in ZeroCostDL4Mic and their functions.

(Video S1). The underlying code is hidden by default but remains accessible, allowing advanced users to explore and edit the programmatic structure of the notebooks. No extra resources beyond a web browser and a Google Drive account are needed, all computations are directly performed on the

cloud using Google's computational capacity.

We created a common and straightforward interface for popular microscopy DL networks: U-net (8, 9), Stardist (5, 6), CARE (7), Noise2Void (10) and Label-free prediction (fnet) (11). Together, they exploit DL to enable numerous important image analysis tasks e.g. image segmentation, denoising, restoration and artificial labelling (e.g. prediction of pseudo-fluorescence images from bright-field images) (Fig. 1b and S2, Video S2). Google Colab provides free but finite resources in terms of remote disk space, RAM capacity and run-time duration (discussed in detail in Supplementary Note 1), which allowed us to efficiently train the various networks on the corresponding training dataset that we provide (Supplementary Table S1). The training only takes a few minutes to a few hours and allowed us to produce the output shown in Fig. 1 and Supplementary Fig. S2 (training times available in Supplementary Table S1). Users can easily test the performance of the trained models (see Supplementary Note 2 and Supplementary Figures S4, S5 and S6) and exploit them on their own data to obtain network predictions. Importantly, ZeroCostDL4Mic lowers the barriers that impede researchers to train neural networks on the most meaningful data for their application. Doing so, it provides an alternative to the current use of sometimes inappropriate pre-trained models, as it is difficult for these to re-capitulate all the possible types of data researchers will want to analyse (Supplementary Fig. S3). Additionally, to identify limitations of the trained models, a quantitative quality control is implemented (Supplementary Fig. S4, S5 and S6) that allows the assessment of model performance (discussed in Supplementary Note 2) and aid in improving model performance (Supplementary Fig. S5).

By bringing previously published methods into a streamlined format that allows easy access and customised use of DL in microscopy, we believe this resource will be an important step towards widening the use of DL approaches beyond the community of computer scientists (further discussed in Supplementary Note 3). In addition, we envision that the template presented here could be used by DL developers to showcase their own network architectures (Supplementary Note 4). This would ensure the rapid dissemination of novel technologies and provide consistent user experience for reproducible and comparative studies of DL approaches.

Availability. ZeroCostDL4Mic is available as Supplemental Software or can be accessed from our GitHub page. This resource is fully open-source, providing users with tutorials, Jupyter Notebooks for Google Colab and many real-life example datasets for training and testing. The example datasets are available for download in Zenodo (links provided in Supplementary Table S1 and our GitHub page).

ACKNOWLEDGEMENTS

First and foremost, we would like to thank Dr Martin Weigert from the Swiss Federal Institute of Technology (EPFL) and Dr Uwe Schmidt from the Max Planck Institute of Molecular Cell Biology and Genetics (MPI-CBG), who pioneered a considerable portion of the technology this work is based on and whose ethos in making Deep Learning more accessible for microscopy helped inspire this work. Dr Schmidt has kindly given key feedback during the preparation of the manuscript. All the network architectures and tasks presented here originate from already published work, having been edited and prepared for Google Colab and to simplify their uptake by novice users. **When using the ZeroCostDL4Mic platform, please cite the original publications associated to each network**.

This work was funded by grants from the UK Medical Research Council (MR/K015826/1) (R.H.) and the Wellcome Trust (203276/Z/16/Z) (R.H.). R.F.L. would like to acknowledge the support of the MRC Skills development fellowship (MR/T027924/1). This work was also supported by grants awarded by the Academy of Finland (to G.J. and P.K.M.), the Sigrid Juselius Foundation (to G.J. and P.K.M.),the University of Turku foundation (to SHP), Åbo Akademi University Research Foundation (CoE CellMech; G.J.) and by Drug Discovery and Diagnostics strategic funding to Åbo Akademi University (G.J.). We thank Dr Aki Stubb for providing us with the raw data used to showcase Noise2Void 2D. The Cell Imaging and Cytometry Core facility (Turku Bioscience, University of Turku, Åbo Akademi University and Biocenter Finland) and Turku Bioimaging are acknowledged for services, instrumentation, and expertise. M.L is supported by Victoriastiftelsen (FI). A.K, T. - O.B and F.J. funded by German Research Foundation (DFG) under the code JU3110/1-1 and German Federal Ministry of Research and Education under the code 01IS18026C, ScaDS2. M.H. and C.S. acknowledge funding by the German Science Foundation (grant nr. SFB1177), C.S. additionally acknowledges support by the European Molecular Biology Organization (short term fellowship 8589). We additionally thank the Chan Zuckerberg Biohub and its donors for funding the work of Loic A. Royer and Ahmet Can Solak. Note: CoLaboratory™ and Google Drive™ are trademarks of Google LLC - ©2018 Google LLC All rights reserved.

EXTENDED AUTHOR INFORMATION

- Lucas von Chamier: 0000-0002-9243-912X; ChamierLucas
- Romain F. Laine: 0000-0002-2151-4487; LaineBioImaging
- Johanna Jukkala: Liehuletti
- Christoph Spahn: 0000-0001-9886-2263; miCHRIScopy
- Martina Lerche: 0000-0002-8203-1120; martina_lerche
- Sara Hernández-Pérez: 0000-0002-0227-1045; sara_mattilalab
- Pieta K. Mattila: 0000-0003-2805-0686; mattilalab
- Eleni Karinou: 0000-0002-1099-4964; Kar__El
- Seamus Holden: 0000-0002-7169-907X; seamus_holden
- Ahmet Can Solak: 0000-0002-1381-8309; _ahmetcansolak
- Alexander Krull: sagzehn
- Tim-Oliver Buchholz: 0000-0001-6953-8915; tibuch_
- Florian Jug: 0000-0002-8499-5812; florianjug
- Loïc A Royer: 0000-0002-9991-9724; loicaroyer
- Mike Heilemann: 0000-0002-9821-3578
- Guillaume Jacquemet: 0000-0002-9286-920X; guijacquemet
- Ricardo Henriques: 0000-0002-2043-5234; HenriquesLab

AUTHOR CONTRIBUTIONS

R.F.L., G.J. and R.H. initiated the research. L.v.C., J.J., T.-O.B., R.F.L. and G.J. wrote source code based on the work of A.K., T.-O.B., F.J. and L.A.R. among others; L.v.C., J.J., C.S., R.F.L. and G.J. performed the image acquisition of the test and example data; L.v.C., J.J., C.S., M.L., S.H.-P., P.K.M., E.K., S.H., A. C. S., A.K., T.-O. B., M.H., R.F.L. and G.J. tested the platform; L.v.C., J.J., C.S., R.F.L., G.J. and R.H. wrote the manuscript with input from all co-authors.

COMPETING FINANCIAL INTERESTS

We provide a platform based on Google Drive and Google Colab to streamline the implementation of common Deep Learning analysis of microscopy data. Despite heavily relying on Google products, we have no commercial or financial interest in promoting and using these products. In particular, we did not receive any compensation in any form from Google for this work. The authors declare no competing financial interests.

Bibliography

1. Chinmay Belthangady and Loic A Royer. Applications, promises, and pitfalls of deep learning for fluorescence image reconstruction. *Nature methods*, pages 1–11, 2019.
2. Lucas von Chamier, Romain F Laine, and Ricardo Henriques. Artificial intelligence for microscopy: what you should know. *Biochemical Society Transactions*, 47(4):1029–1040, 2019.
3. Wei Ouyang, Florian Mueller, Martin Hjelmare, Emma Lundberg, and Christophe Zimmer. Imjoy: an open-source computational platform for the deep learning era. *Nature methods*, 16(12):1199–1200, 2019.
4. Estibaliz Gómez-de Mariscal, Carlos García-López-de Haro, Laurène Donati, Michael Unser, Arrate Muñoz-Barrutia, and Daniel Sage. Deepimagej: A user-friendly plugin to run deep learning models in imagej. *bioRxiv*, page 799270, 2019.
5. Uwe Schmidt, Martin Weigert, Coleman Broaddus, and Gene Myers. Cell detection with star-convex polygons. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 265–273. Springer, 2018.
6. Martin Weigert, Uwe Schmidt, Robert Haase, Ko Sugawara, and Gene Myers. Star-convex polyhedra for 3d object detection and segmentation in microscopy. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 3666–3673, 2020.
7. Martin Weigert, Uwe Schmidt, Tobias Boothe, Andreas Müller, Alexandr Dibrov, Akanksha Jain, Benjamin Wilhelm, Deborah Schmidt, Coleman Broaddus, Siân Culley, et al. Content-aware image restoration: pushing the limits of fluorescence microscopy. *Nature methods*, 15(12):1090–1097, 2018.
8. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

9. Thorsten Falk, Dominic Mai, Robert Bensch, Özgün Çiçek, Ahmed Abdulkadir, Yassine Marrakchi, Anton Böhm, Jan Deubner, Zoe Jäckel, Katharina Seiwald, et al. U-net: deep learning for cell counting, detection, and morphometry. *Nature methods*, 16(1):67–70, 2019.
10. Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. Noise2void-learning denoising from single noisy images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2129–2137, 2019.
11. Chawin Ounkomol, Sharmishtaa Seshamani, Mary M Maleckar, Forrest Collman, and Gregory R Johnson. Label-free prediction of three-dimensional fluorescence images from transmitted-light microscopy. *Nature methods*, 15(11):917–920, 2018.

Online Methods

Cell Culture. U-251 glioma cells were grown in DMEM/F-12 (Dulbecco's Modified Eagle's Medium/Nutrient Mixture F-12; Life Technologies, 10565-018) supplemented with 10 % fetal bovine serum (FCS) (Biowest, S1860). U-251 glioma cells expressing endogenously tagged paxillin-GFP were generated using CRISPR / Cas9 and were described previously (1). MCF10 DCIS.COM (DCIS.COM) lifeact-RFP cells were cultured in a 1:1 mix of DMEM (Sigma-Aldrich) and F12 (Sigma-Aldrich) supplemented with 5% horse serum (16050-122; GIBCO BRL), 20 ng/ml human EGF (E9644; Sigma-Aldrich), 0.5 mg/ml hydrocortisone (H0888-1G; Sigma-Aldrich), 100 ng/ml cholera toxin (C8052-1MG; Sigma-Aldrich), 10 µg/ml insulin (I9278-5ML; Sigma-Aldrich), and 1% (vol/vol) penicillin/streptomycin (P0781-100ML; Sigma-Aldrich). DCIS.COM lifeact-RFP cells were generated using lentiviruses, produced using pCDH-LifeAct-mRFP, psPAX2, and pMD2.G constructs - see (2) for more details. HeLa ATCC cells were seeded on fibronectin-coated 8-well chamberslides (Sarstedt, Germany, 1.5 x 104 cells/well). Cells were grown for 16h at 37°C and 5% CO2 in Dulbecco's modified Eagle's medium containing 4.5 g/l glucose, 10% FBS and 1% L-alanyl-L-glutamine (Thermo Fisher, GlutaMAX). To fix the HeLa cells, we employed a protocol shown to preserve the cytoskeleton and organelles (adapted from (3)). In brief, the culture medium was directly replaced with PHEM buffer containing 3% methanol-free formaldehyde (Thermo Fisher, USA) and 0.2% EM-grade glutaraldehyde (Electron Microscopy Sciences, USA) and incubated the samples for 1 h at room temperature. Cells were washed thrice with PBS, quenched with 0.2% sodium borohydrate in PBS for 7 min and washed again thrice with PBS. A2780 cells were cultured in RPMI 1640 supplemented with 10% FCS. The cells were grown at 37°C in a 5% CO2 incubator.

U-net training dataset. U-net (4) is a deep learning architecture originally developed for segmentation of EM images although it has since been adapted for various other tasks (e.g. CARE, Label-free prediction). Here, we use U-net for segmentation of EM images on the dataset from the ISBI challenge 2012 (5). However, datasets for segmentation tasks can also be created manually. This requires target images which have been segmented by an expert using drawing tools, e.g. in ImageJ/Fiji (6), to draw outlines around the structures of interest. For training in the notebook the source (raw EM image) and target (8-bit mask obtained from expert drawing) images were placed in separate folders, with each source image having a corresponding target image with the same name.

Stardist training dataset. Stardist (7) is a deep learning method that was designed specifically to segment cell nuclei in microscopy images. Training a Stardist network requires matching images of nuclei and of corresponding masks. To generate the nuclei images that we provide as a training dataset, DCIS.COM lifeact-RFP cells were incubated for 2h with 0.5 µM SiR-DNA (SiR-Hoechst, Tetu-bio, Cat Number: SC007) before being imaged live using a spinning-disk confocal microscope. The spinning-disk confocal microscope used was a Marianas spinning disk imaging system with a Yokogawa CSU-W1 scanning unit on an inverted Zeiss Axio Observer Z1 microscope (Intelligent Imaging Innovations, Inc.) equipped with a 20x (NA 0.8) air, Plan Apochromat objective (Zeiss). The corresponding mask images were generated manually in Fiji (6). Briefly, the outlines of each nucleus were drawn using the freehands selection tool and added to the ROI manager. Once all outlines are stored in the ROI manager, the LOCI plugin was used to create a ROI map. The ROI map images were used as the mask images to train Stardist.

Noise2Void training datasets. Noise2Void (8) is a deep learning method that was designed to perform denoising on microscopy images. It takes an unsupervised training approach so it can learn how to denoise a dataset directly from the dataset to be denoised. Therefore, no specific training datasets are required, only noisy images. Even one noisy image is sufficient to train the network. The 2D dataset provided with our notebooks was generated by plating U-251 glioma cells expressing endogenously tagged paxillin-GFP on fibronectin-coated poly-acrylamide gels (stiffness 9.6 kPa) (1). Cells were then recorded live using a spinning disk confocal microscope equipped with a long working distance 63x (NA 1.15 water, LD C-Apochromat) objective (Zeiss). The 3D dataset provided with our notebooks was generated by recording A2780 ovarian carcinoma cell, transiently expressing Lifeact-RFP (to visualize the actin cytoskeleton), migration on fibroblast-generated cell-derived-matrices, further see (9) for methods. The cell-derived matrices were labelled using Alexa Fluor 488 recombinant fibronectin and the images acquired using a spinning disk confocal microscope equipped with a 63x oil (NA 1.4 oil, Plan-Apochromat, M27 with DIC III Prism) objective (Zeiss). For both datasets, the spinning disk confocal microscope used was a Marianas spinning disk imaging system with a Yokogawa CSU-W1 scanning unit on an inverted Zeiss Axio Observer Z1 microscope controlled by SlideBook 6 (Intelligent Imaging Innovations, Inc.). Images were acquired using a Photometrics Evolve, back-illuminated EMCCD camera (512 x 512 pixels).

Content-aware image restoration (CARE) training datasets. CARE (10) is a deep learning method capable of image restoration from corrupted bio-images (whether corrupted by noise, artefacts or low resolution for instance). The network allows image denoising and resolution improvement in 2D and 3D images, using supervised training. The specific function of the network is determined by the images provided in the training dataset. For instance, if noisy images are provided as input and high signal-to-noise ratio images are provided as targets, the network will perform denoising. The dataset provided as an example with our notebooks was generated to denoise live-cell structured illumination microscopy (SIM) imaging data. Briefly, DCIS.COM lifeact-RFP cells were plated on high tolerance glass-bottom

dishes (MatTek Corporation, coverslip 1.5) pre-coated first with Poly-L-lysine (10 mg/ml, 1 h at 37°C) and were allowed to reach confluence. Cells were then fixed and permeabilised simultaneously using a solution of 4% (wt/vol) paraformaldehyde and 0.25% (vol/vol) Triton X-100 for 10 min. Cells were then washed with PBS, quenched using a solution of 1 M glycine for 30 min, and incubated with phalloidin-488 (1/200 in PBS; Cat number: A12379; Thermo Fisher scientific) at 4°C until imaging (overnight). Just before imaging using SIM, samples were washed three times in PBS and mounted in vectashield (Vectorlabs). The SIM system used was DeltaVision OMX v4 (GE Healthcare Life Sciences) fitted with a 60x Plan-Apochromat objective lens, 1.42 NA (immersion oil RI of 1.516) used in SIM illumination mode (five phases and three rotations). Emitted light was collected on a front illuminated pco.edge sCMOS (pixel size 6.5 µm, readout speed 95 MHz; PCO AG) controlled by SoftWorx. In the provided dataset, the high signal-to-noise ratio images were acquired from the phalloidin-488 staining using acquisition parameters optimal to obtain high-quality SIM images (in this case, 50 ms exposure time, 10% laser power). In contrast the low signal-to-noise ratio images were acquired from the LifeAct-RFP channel using acquisition parameters more suitable for live-cell imaging (in this case, 100 ms exposure time, 1% laser power). The dataset provided with the 2D CARE notebooks are maximum intensity projections of the collected data.

Label-free prediction (fnet) training dataset. Fnet (11) is a deep learning method which was developed as a tool for label-predictions from unannotated brightfield and EM images. Training fnet in the provided Colab notebook requires 3D stacks in two channels, e.g. fluorescence and transmitted light. The dataset provided for training Label-free prediction notebook was designed to predict a mitochondrial marker from brightfield images. Before acquisition fixed HeLa ATCC cells were permeabilised and blocked using 0.25% TX-100 (Sigma Aldrich, Germany) and 3% IgG-free BSA (Carl Roth, Germany) in PBS for 1.5 h. Cells were labelled for TOM20 using 5 µg/ml rabbit anti-TOM20 primary antibody (sc-11415, Santa Cruz, USA) and 10 µg/ml donkey anti-rabbit-secondary antibody (Alexa Fluor 594 conjugated, A32754, Thermo Fisher, USA) in PBS containing 0.1% TX-100 and 1% BSA for 1.5 h each. Samples were rinsed twice and washed 3x with PBS (5 min) after each incubation step. Image stacks were acquired on a Leica SP8 confocal microscope (Leica Microsystems, Germany) bearing a 63x/1.40NA oil objective (Leica HC PL APO). The pixel size was set to 90 nm in xy-dimensions and 150 nm in z (32 slices) and fluorescence image stacks were recorded using 561 nm laser excitation and collected by a PMT. The corresponding transmitted light image stack was recorded in parallel using a transmitted light PMT. We acquired 25 3D stacks with dimensions of 1024x1024x32. To create the training set each image stack was split into four stacks with dimensions of 512x512x32, giving a dataset of 100 images of which 92 were used for training and 8 unseen for testing and quality control (see Table). The raw data was converted into .tif file format and split into stacks of the respective channels (fluorescence and transmitted light). To prepare a training set, stacks were split into individual folders by channel. To create matched training pairs the signal files (transmitted light) and their respective targets (fluorescence) must be in the same order in their respective folders. It is therefore advisable to number source-target pairs or to give the files the same names. This was done with an ImageJ/Fiji macro (6).

Bibliography

1. Aki Stubb, Romain F Laine, Mitro Miihkinen, Hellyeh Hamidi, Camilo Guzmán, Ricardo Henriques, Guillaume Jacquemet, and Johanna Ivaska. Fluctuation-based super-resolution traction force microscopy. *Nano Letters*, 2019.
2. Guillaume Jacquemet, Ilkka Paatero, Alexandre F Carisey, Artur Padzik, Jordan S Orange, Hellyeh Hamidi, and Johanna Ivaska. Filoquant reveals increased filopodia density during breast cancer progression. *Journal of Cell Biology*, 216(10):3387–3403, 2017.
3. Wesley R Legant, Lin Shao, Jonathan B Grimm, Timothy A Brown, Daniel E Milkie, Brian B Avants, Luke D Lavis, and Eric Betzig. High-density three-dimensional localization microscopy across large volumes. *Nature methods*, 13(4):359–365, 2016.
4. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
5. Ignacio Arganda-Carreras, Srinivas C Turaga, Daniel R Berger, Dan Cireşan, Alessandro Giusti, Luca M Gambardella, Jürgen Schmidhuber, Dmitry Laptev, Sarvesh Dwivedi, Joachim M Buhmann, et al. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in neuroanatomy*, 9:142, 2015.
6. Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, et al. Fiji: an open-source platform for biological-image analysis. *Nature methods*, 9(7): 676–682, 2012.
7. Martin Weigert, Uwe Schmidt, Robert Haase, Ko Sugawara, and Gene Myers. Star-convex polyhedra for 3d object detection and segmentation in microscopy. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 3666–3673, 2020.
8. Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. Noise2void-learning denoising from single noisy images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2129–2137, 2019.
9. Riina Kaukonen, Guillaume Jacquemet, Hellyeh Hamidi, and Johanna Ivaska. Cell-derived matrices for studying cell proliferation and directional migration in a complex 3d microenvironment. *nature protocols*, 12(11):2376, 2017.
10. Martin Weigert, Uwe Schmidt, Tobias Boothe, Andreas Müller, Alexandr Dibrov, Akanksha Jain, Benjamin Wilhelm, Deborah Schmidt, Coleman Broaddus, Siân Culley, et al. Content-aware image restoration: pushing the limits of fluorescence microscopy. *Nature methods*, 15(12):1090–1097, 2018.
11. Chawin Ounkomol, Sharmishtaa Seshamani, Mary M Maleckar, Forrest Collman, and Gregory R Johnson. Label-free prediction of three-dimensional fluorescence images from transmitted-light microscopy. *Nature methods*, 15(11):917–920, 2018.

Supplementary Note 1: Capabilities of Google Colab

The Google Colab platform offers free and easy access to a Graphical Processing Unit (GPU) and Tensor Processing Unit (TPU), which enables users to train many networks on bespoke training datasets, as well as running predictions on unseen data. In practice, for each notebook session, Google Colab assembles a virtual machine with allocated RAM, disk space and access to GPU/TPU. Although these resources are free, they are finite. Consider this limitation when exploiting Google Colab for Deep Learning training and predictions. In our experience, we consistently found that GPU acceleration provided faster computation than TPU for the networks and datasets presented. Therefore, we focused our attention on GPU accessibility and performance below. In the following sections, we discuss how to handle these resources in order to perform efficient training.

1.1. Google Drive storage. When using a free Google Drive account to perform training, the user will have access to 15 GB of data storage that can be accessed by any Google Colab notebook. All training and test datasets, plus the output of the training, need to fit within this 15 GB limit. We have shown, however, that this is sufficient to efficiently train all of the networks with the datasets that we provide (Table S1). Additional storage space can also be purchased from Google.

1.2. Remote RAM capacity. A 12.72 GB RAM limit currently exists for the free GPU or TPU provided by Google Colab. The amount of RAM required to execute the code is determined by the size of the training data, the number and sizes of patches/batches. Exceeding this RAM limit can cause the notebook to crash when initialising the networks. For the datasets we have tested, it was always possible to train the networks with the currently available RAM. It is important to note that large datasets (e.g. datasets made of large 3D stacks) may reach or exceed the RAM capacity when using a large number of patches/batches.

1.3. Time-outs.

12-hour time-out. The time taken to train a network sufficiently is primarily determined by the size of the training dataset, the number of steps/epochs/patches, and the efficiency of the underlying code. Google Colab currently offers 12 hours of GPU/TPU access after which remote data loaded into the virtual machine will be lost, a limit primarily enforced to prevent cryptocurrency-mining. For users, this means that if training has not completed by the 12h limit, all progress may be lost if no intermediate steps were saved. This constraint can be problematic when networks need to be trained over many epochs to reach high performance, often necessary for large datasets. We have found that Label-free prediction (fnet) typically requires long training times (Table S2), compared to the other networks tested here. Users can circumvent this issue by saving network weights in a Google Drive folder regularly. In the Label-free prediction (fnet) notebook, checkpoints are automatically saved, which allows the user to continue training later from such a checkpoint after a potential time-out.

Log-out if idle. Google Colab may disconnect significantly earlier than after 12 hours if it detects an interruption of user interaction or network training. Usually, this time-out happens after 30 to 90 minutes of idleness in our hands, i.e. code not running or lack of user interaction with the Google Colab interface. When the log-out occurs, local variables, installed packages and data stored outside any mounted drive are deleted. Hence, if the log-out occurs before training a model, cells setting up parameters for training such as paths or hyperparameters may have to be reset before training. With all of our notebooks, the models are automatically saved in Google Drive upon training completion, meaning that long training sessions do not have to be attended.

1.4. Unreliable GPU access. Google Colab does not guarantee access to a GPU, as the number of current users may be larger than the number of available devices. It is not clear how access to a GPU is regulated, but it may be determined by traffic to the Google Colab servers. If no GPUs are available, Colab will offer to run the notebook using either a TPU, CPU (without acceleration) or as a local runtime (i.e. on the machine of the user). It should be noted though that these alternatives can be significantly slower than Google Colab GPU access. However, GPU access usually becomes available again from within a few hours to a day.

1.5. GPU type. Google Colab uses different GPUs which currently include Nvidia K80, T4, P4 and P100 (as of March 2020). The user cannot decide which GPU will be available when using the notebook. According to the Google Colab FAQ, this is due to limitations in the provision of a free service to users which makes certain types of GPU unavailable at the time a notebook is used. In practice, this should not affect the performance of the networks in the notebooks. However, it may affect the speed at which networks can be trained and used. Therefore, users might encounter variability in training times as a consequence. To find out which type of GPU Google Colab is using, the user can play the first cell in each notebook which will give information on GPU access and type.

1.6. How to mitigate the limitations of Google Colab. Several steps can be taken by users of ZeroCostDL4Mic to mitigate Google Colab limitations. Regarding the 12h training limitation, we encourage users to change the number of epochs and training parameters so that the training takes less than 12h. This is possible with all of the networks we provide. In the case

of Label-free prediction (fnet), in order to accommodate the typically large number of steps necessary to train the network efficiently, we provide the option to continue the training from a saved training checkpoint in case of a time-out. This allows us to start from a pre-trained model and accumulate many rounds of steps. For the log-out if idle issue, Google Colab cells can be played all at once (or a subset at once). In this case, the activated cells will run one after the other. This can be useful to ensure that the runtime does not disconnect until the completion of all the analytical steps and the user data saved in Google Drive.

ZeroCostDL4Mic is aimed to be an entry point to learn about Deep Learning methods where users can quickly train networks with their data. While ZeroCostDL4Mic is entirely free to use, many of its limitations (associated with the Google Colab platform) can be mitigated with small financial investments. For instance, the 15 GB Google Drive limitation can be bypassed by purchasing more Google Drive storage from Google. In addition, Google is rolling out a Google Colab Pro version that offers faster GPUs, longer runtimes and access to more RAM. If these intermediate options are still not sufficient, ZeroCostDL4Mic notebooks can also be easily adapted to run on the user's own computer by connecting Google Colab to a local runtime. This option allows the user to access their local files and resources (local GPU) directly from the Google Colab notebooks. This, of course, requires the users to invest in a powerful workstation.

Supplementary Note 2: Quality Control on trained models

The reliable implementation of Deep Learning methods depends on a careful evaluation of model's output performance. This step is crucial to avoid using models which produce low-quality images and artefacts, especially when these may not be easily identifiable by eye. ZeroCostDL4Mic implements quality control metrics to compare models trained with different hyperparameters (Fig. S5). These metrics can thus help the user improve the models created. The notebooks provided contain a section fully dedicated to evaluating the performance of trained models (Fig. S6). This section is divided into two parts:

- Inspection of the loss function over the number of epochs trained.
- Evaluation of image quality metrics by comparing the model predictions against a ground truth equivalent.

The first performance metrics shown to users are loss curves for model training and validation (Fig. S4a). They allow users to determine if the model started over-fitting to data during training which can be identified by an increasing divergence between the validation and training loss (Figure S4). This divergence is expected if the model learns details of the training set which prevent it from generalising to unseen data, a common problem for deep neural networks (1). However, even a model which fits well to the validation data can produce unwanted results when used on unseen data which would ultimately make such models unreliable. These issues can be further identified by comparing the predictions from unseen data to the equivalent ground-truth data. The metrics used in individual notebooks vary to reflect the differences in the type of data these models operate with. For networks producing a grey-scale image, e.g. CARE, Noise2Void, U-net and Label-free prediction, the metrics used are SSIM (Structural similarity) and RSE (Root Square Error). For networks producing a binary or semantic image, e.g. StarDist, the metric used is IoU (intersection over union).

2.1 Structural Similarity Index (SSIM). First introduced by Zhou *et al.* (2), SSIM is calculated as:

$$SSIM(X,Y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_{2)}}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad \textbf{(S1)}$$

where X and Y are the images to be compared, μ_x and μ_y the mean pixel values, σ_x^2 and σ_y^2 the variance of pixel values and σ_{xy} the covariance between the pixel values in the images. C_1 and C_2 are constants introduced to avoid instability for small denominators, and are defined as:

$$C_1 = (K_1 L)^2 \quad \textbf{(S2)}$$

$$C_2 = (K_2 L)^2 \quad \textbf{(S3)}$$

where L is the bit-depth of the images to be compared (for 16-bit images used for the quality assessment in the notebooks L =65536) and K_1=0.01 and K_2=0.03, as suggested by the authors of SSIM (2). To calculate the SSIM on a consistent dynamic range, images are normalised to values between 0 and 1, first by percentile normalisation on all data (source, target and prediction). Then, both source and prediction are further normalised by linear regression compared to the target.

The percentile normalisation was performed as follows:

$$I_{i,j}^{norm} = \frac{I_{ij} - I_{99.9}}{I_{99.9} - I_{0.1}} \quad \textbf{(S4)}$$

where $I_{i,j}^{norm}$ represents the normalised intensity value at pixel (i,j), I_{ij} the intensity to be rescaled, $I_{99.9}$ the value of the pixel which lies in the 99.9th percentile of pixel values in the image and $I_{0.1}$ the pixel value of the pixel in 0.1th percentile of pixel values in the image. This percentile-based normalisation (instead of using minimal and maximal pixel values) is aimed to prevent the influence of *dead* or *hot* pixels which are common in microscopy images and may distort the useful dynamic range of the image upon normalisation.

The linear regression normalisation is done the same way as in (3), based on least square minimisation, defined as:

$$(\alpha_o, \beta_o) = \underset{\alpha,\beta}{argmin} \sum_{i,j} (GT_{i,j} - (\alpha I_{i,j} + \beta))^2 \quad \textbf{(S5)}$$

where GT is the ground truth image, I the predicted image, $GT_{i,j}$ and $I_{i,j}$ the respective pixels in the ground truth and prediction and α and β the parameters which rescale the prediction to the dynamic range of the ground-truth.

The normalised image is then calculated as follows:

$$I^{norm} = \alpha_o I + \beta_o \quad \textbf{(S6)}$$

After normalisation, the pixel values above 1 and below 0 are thresholded to 1 and 0, respectively.

The SSIM map is calculated by estimating Eq. S1 on a local window around the pixel of interest. The window size is set to 11x11 pixels and a Gaussian weighting function of 1.5 pixels standard deviation. The similarity map obtained displays areas with high and low similarity in different colours, enabling inspection of areas in the image where the model performs better or worse.

Values for SSIM can vary between -1 and 1, with 1 indicating a perfect structural content agreement between the two images. Therefore a well-performing model will produce high values of mSSIM and an SSIM map with nearly uniform values close to 1 across the image. This is demonstrated in Supplementary figure S4b where we compare the prediction of a trained CARE network with the expected ground-truth of the denoising task. The almost uniform SSIM map and a SSIM index above 0.9 signify a high level of similarity between the prediction and the ground-truth. It also shows a clear improvement to the SSIM map and index calculated between ground truth and the noisy input image which suggests that the network did improve the input image.

The SSIM index over the whole image (mSSIM) is simply calculated by calculating the average SSIM value over the whole image. This value allows an overall quantitative comparison of performance between different images or network parameters.

In the notebook, the SSIM map is calculated between ground-truth against model prediction and between ground-truth against source image (when appropriate). This way, the metric can be used comparatively to judge the improvement of the prediction over the source image. In practice, SSIM values rarely fall below 0, which would represent inverted structures in the image, as explained by the author of SSIM here. For this reason, we truncate SSIM values to a range between 0 and 1 on the similarity maps.
S4

2.2. Root Squared Error (RSE). The RSE is obtained by calculating the unsigned difference between two images on a pixel-per-pixel basis. By calculating the RSE between the prediction and the target image (ground-truth), this error map shows areas of high and low errors in different colours, providing a second metric for inspecting local artefacts.

$$RSE_{i,j} = \sqrt{(GT_{i,j} - I_{i,j}^{norm})^2} \tag{S7}$$

Similarly to SSIM we also provide an image-wise metric as a normalised root mean squared error (NRMSE) between the images defined as

$$NRMSE = \sqrt{\sum_{i,j} (GT_{i,j} - I_{i,j}^{norm})^2} \tag{S8}$$

The RSE map and NRMSE reflect the amount of discrepancy between two images. In this case good performance is indicated by low values for these metrics. A perfect agreement with the ground truth image will lead to RSE = 0 across the image and NMRSE = 0.

A comparison between prediction against ground-truth and source against ground-truth are shown side-by-side to give an indication of how much the image has improved (i.e. brought closer to the ground-truth than the source) by the trained model. Similarly to the SSIM metric, the output of the RSE metric is shown in Supplementary figure S4b where we compare the prediction of the trained CARE network with the expected ground-truth. The RSE map is almost uniformly dark and the NRMSE score is low suggesting a good match between ground truth and prediction. The comparison with the same metrics calculated between the noisy input and the ground truth also shows how the CARE prediction improved the image by reducing the errors seen in the RSE map and by lowering the value of the NRMSE compared the the input image.

2.3 Intersection over union - IoU. For the segmentation model StarDist, we use the intersection over union (IoU) metric, commonly used for segmentation performance evaluation (4). To test how well the model segments an input image the ground-truth mask is compared to the predicted segmentation mask by dividing the number of pixels shared between both masks by the total number of pixels in the union of the two masks:

$$IoU = \frac{I \cap GT}{I \cup GT} \tag{S9}$$

where I represents the predicted image and GT the ground-truth image.

The IoU metric quantifies the percentage of overlap between the target mask and the prediction output. Therefore, scores closer to 1 mean better performance of the trained model. How this metric is used for quality control is shown in Supplementary figure S4c. The overlay of ground truth (target) and prediction can act as a direct readout of the IoU score. Here, the white areas represent where the signal predicted by the Stardist network overlaps with the target, therefore agreeing with the ground-truth. A large proportion of white areas in the overlay suggests a good agreement between ground truth and prediction which is also reflected by a high IoU score of over 0.8.

While using these metrics as quality control for the models cannot prevent errors or artefacts to occur in the prediction, they can characterise the performance of the model. This can be exploited to improve model performance as shown in Supplementary figure S5. The figure shows an example of how the IoU score in Stardist can aid in identifying a set of training parameters which can accelerate how quickly the model reaches its top performance. Here, we show how the number of epochs, number of steps and batch size can affect the IoU score. Additionally here, the number of nuclei found in the image also provides a way to assess the model performance, when compared to the number of nuclei identified from the ground-truth (target) mask.

Supplementary Note 3: When in doubt, always retrain! A supplementary discussion

The primary focus of ZeroCostDL4Mic is to provide a straightforward and free platform to aid novice users in using Deep Learning in microscopy. A vital component of this platform is the capacity to simplify network training, which remains a significant difficulty in the field. Because it can be difficult to train Deep Learning networks (in time, resources and skills), a number of labs are taking the approach of providing pre-trained network models that can be used to process imaging data. Both DeepImageJ (5) and CSBDeep (3) are excellent examples. This tends to create model "zoos" - databases of pre-trained models loadable into software packages such as DeepImageJ and CSBDeep. Pre-trained models can be very powerful, but they can also be very specific to the microscopes and sample types used in their training and may lead to erroneous results when applied to a different dataset type than that they were trained on (6). Figure S3 shows how using an inappropriate model can lead to erroneous results in the prediction.

It is, however, not uncommon for researchers to apply inappropriate pre-trained models to their data, which unfortunately lead to inaccurate but often visually pleasing results (1, 7). **Given this issue, it becomes critical for researchers to have the option to train models using their own specific data in order to produce high-fidelity and reliable results**. ZeroCostDL4Mic allows researchers to access the resources needed to train many Deep Learning networks, in an accessible and educational manner. By default, the code is hidden and only the user interface is present but it is very easy to view and edit the code via the notebook page itself (by double-clicking on the cell of interest), allowing for interactive testing and learning of the different code sections, if necessary. This offers a unique and powerful way for users to explore the code structure and build more confidence and understanding in the workflow of the analysis. We believe ZeroCostDL4Mic takes a significant step to increase the uptake of Deep Learning tools by microscopy users and highlights the need for all Deep Learning users to understand and tackle the limitations of the approach.

Supplementary Note 4: Future perspectives for ZeroCostDL4Mic

Together with the help of the wider research community, we expect to grow the number of networks available in ZeroCostDL4Mic quickly. We are currently working in collaboration with the Royer laboratory to add Noise2Self (8) to this repertoire. We also expect ZeroCostDL4Mic to become a standard which users can use to evolve existing networks, adapting them to data analysis tasks optimised for their specific image processing problems. To this end, we will further keep developing ZeroCostDL4Mic, continuously adapting it to novel Deep Learning paradigms. This focus will incorporate the capacity to maintain compatibility with rapidly evolving Deep Learning libraries and the ability to export pre-trained models which could be used in either DeepImageJ, CSBDeep or other prediction engines.

Supplementary Tables

Network	Data Type	# of files	Image Sizes	Image Type	Comments
U-net	Training - Images	28	512x512	EM 8-bit TIFF	ISBI or here
U-net	Training - Masks	28	512x512	Binary 8-bit TIFF	ISBI or here
U-net	Test - Images	2	512x512	EM 8-bit TIFF	ISBI or here
U-net	Test - Masks	2	512x512	EM 8-bit TIFF	ISBI or here
Stardist (2D)	Training - Images	45	1024x1024	Fluo 16-bit TIF	This paper, and here
Stardist (2D)	Training - Masks	45	1024x1024	Object-labelled 8-bit TIF	This paper, and here
Stardist (2D)	Test - Images	2	1024x1024	Fluo 16-bit TIF	This paper, and here
Stardist (2D)	Test - Masks	2	1024x1024	Object labelled 8-bit TIF	This paper, and here
Stardist (2D)	Test - Stacks	4	1024x1024x86	Fluo 16-bit TIF	This paper, and here
Stardist (3D)	Training - Images	27	128x128x64	Fluo 16-bit TIF	Martin Weigert et al , 2020 (4) and here
Stardist (3D)	Training - Masks	27	128x128x64	Fluo 16-bit TIF	Martin Weigert et al, 2020 (4) and here
Stardist (3D)	Test - Images	3	128x128x64	Fluo 16-bit TIF	Martin Weigert et al, 2020 (4) and here
Stardist (3D)	Test - Images	3	128x128x64	Fluo 16-bit TIF	Martin Weigert et al, 2020 (4) and here
N2V (2D)	Training - Images	1	512x512	Fluo 16-bit TIFF	Stubb et al, 2020 (9), and here
N2V (2D)	Test - Images	22	512x512	Fluo 16-bit TIFF	Stubb et al, 2020 (9), and here
N2V (3D)	Training - Images	1 (Actin) +1 (Fibronectin)	512x512x13	Fluo 16-bit TIFF	Kaukonen et al, 2017 (10) (Actin and fibronectin datasets), and here
N2V (3D)	Test - Images	48 (Actin) +48 (Fibronectin)	512x512x13	Fluo 16-bit TIFF	Kaukonen et al, 2017 (10) (Actin and fibronectin datasets), and here
CARE (2D)	Training – Low SNR images	22	1024x1024	SIM fluo (MIP from 3D stack) 32-bit TIFF	This paper (Filopodia dataset - Maximum projection), and here
CARE (2D)	Training – High SNR images	22	1024x1024	SIM fluo (MIP from 3D stack) 32-bit TIFF	This paper (Filopodia dataset - Maximum projection), and here
CARE (2D)	Test – Low SNR images	2	1024x1024	SIM fluo (MIP from 3D stack) 32-bit TIFF	This paper (Filopodia dataset - Maximum projection), and here
CARE (2D)	Test – High SNR images	2	1024x1024	SIM fluo (MIP from 3D stack) 32-bit TIFF	This paper (Filopodia dataset - Maximum projection), and here
CARE (3D)	Training – Low SNR images	22	1024x1024x33	SIM fluo 32-bit TIF	This paper (Filopodia dataset 3D - stack), and here
CARE (3D)	Training – High SNR images	22	1024x1024x33	SIM fluo 32-bit TIF	This paper (Filopodia dataset 3D - stack), and here
CARE (3D)	Test – Low SNR images	2	1024x1024x33	SIM fluo 32-bit TIF	This paper (Filopodia dataset 3D - stack), and here
CARE (3D)	Test – High SNR images	2	1024x1024x33	SIM fluo 32-bit TIF	This paper (Filopodia dataset 3D - stack), and here
Label-free prediction (fnet)	Training - Brightfield	92	512x512x32	Bright-field confocal 8-bit TIF	This paper, and here
Label-free prediction (fnet)	Training – Fluo (mitochondrial marker)	92	512x512x32	Fluo confocal 8-bit TIF	This paper, and here
Label-free prediction (fnet)	Test - Brightfield	8	512x512x32	Bright-field confocal 8-bit TIF	This paper, and here
Label-free prediction (fnet)	Test – Fluo (mitochondrial marker)	8	512x512x32	Fluo confocal 8-bit TIF	This paper, and here

Table S1. Overview of the available datasets used for training the networks. In all cases of supervised network training (all networks provided here apart from Noise2Void), the test datasets consist of the last two files generated for training. These were set aside for testing and are not part of the training dataset.

Network Name	# of epochs	# of steps	Batch size	Image dimensions	# of images	# of patches, patch size, patch height	GPU type	Time for training (using these settings)
U-net	200	6	4	1024x1024	28	n.a.	Tesla P100-PCIE-16GB	8 min
Stardist (2D)	200	12	2	1024x1024	72	n.a.	Tesla K80	170 min
Stardist (3D)	400	14	2	128x128x64	27	n.a.	Tesla P4	400 min
N2V (2D)	100	61	128	512x512	22	392, 64	Tesla T4	17 min
N2V (3D)	100	133	128	512x512x13	48	392, 64, 8	Tesla P100-PCIE-16GB	240 min
CARE (2D)	50	31	64	1024x1024	22	100, 80	Tesla P100-PCIE-16GB	3 min
CARE (3D)	50	62	64	1024x1024x33	22	200, 80, 8	Tesla P4	90 min
Label-free prediction (fnet)	n.a.	50000	4	512x512x32	92	n.a.	Tesla P4	8h20min

Table S2. Hyperparameters used to train the networks, GPU types allocated and corresponding training times. Some hyperparameters are hard-coded in the networks or calculated from the other parameters. n.a.: not applicable.

Supplementary figures

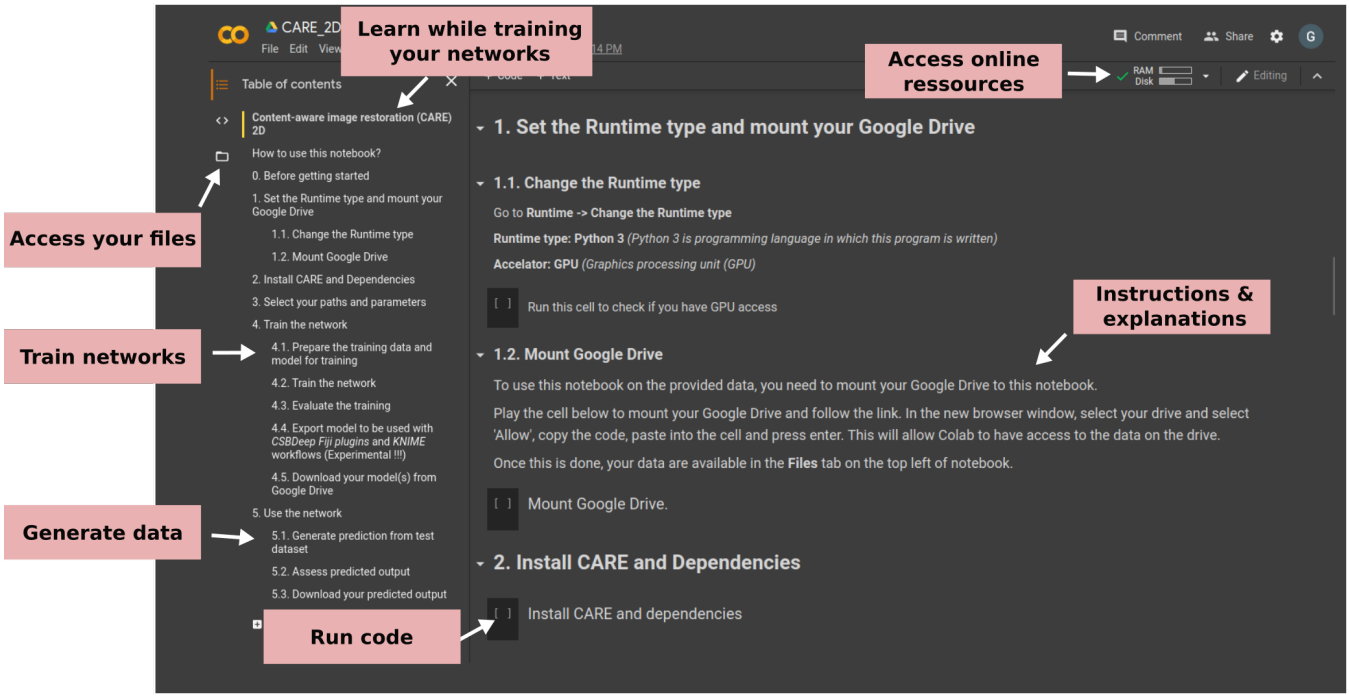


Fig. S1. Graphical user interface (GUI) of the ZeroCostDL4Mic notebooks. The layout of the notebook and quick access to the different sections is available on the left panel. The user has access to the files present on their Google Drive.

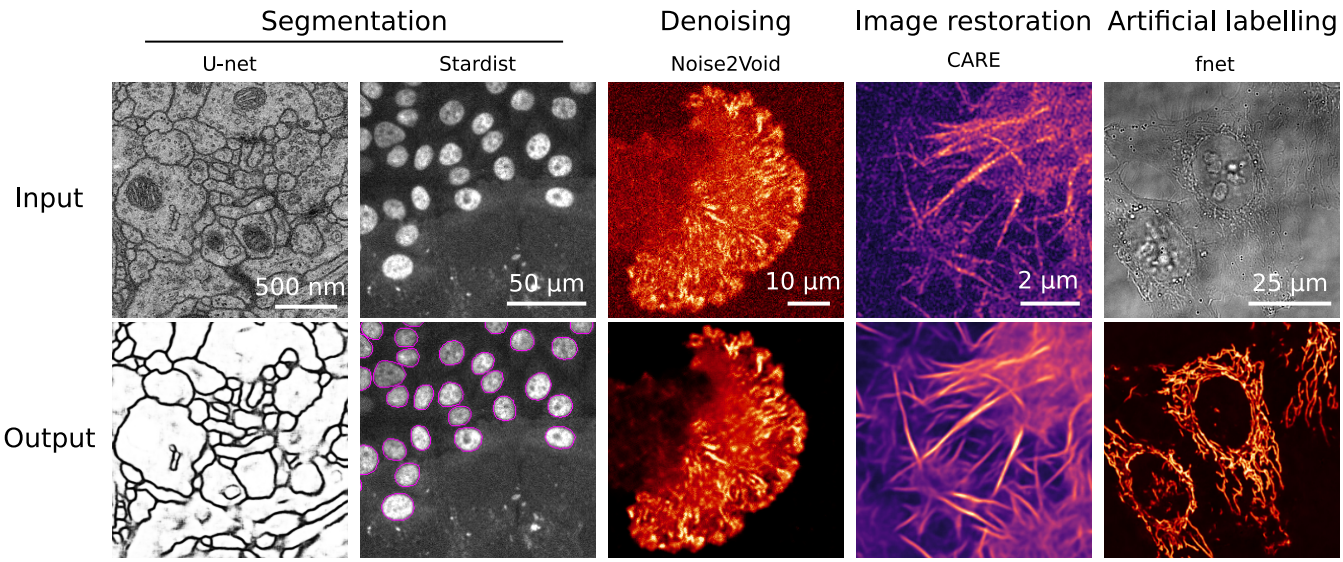


Fig. S2. Overview of the possible tasks to perform with ZeroCostDL4Mic and the corresponding networks. Overview of the networks currently implemented in ZeroCostDL4Mic and their tasks.

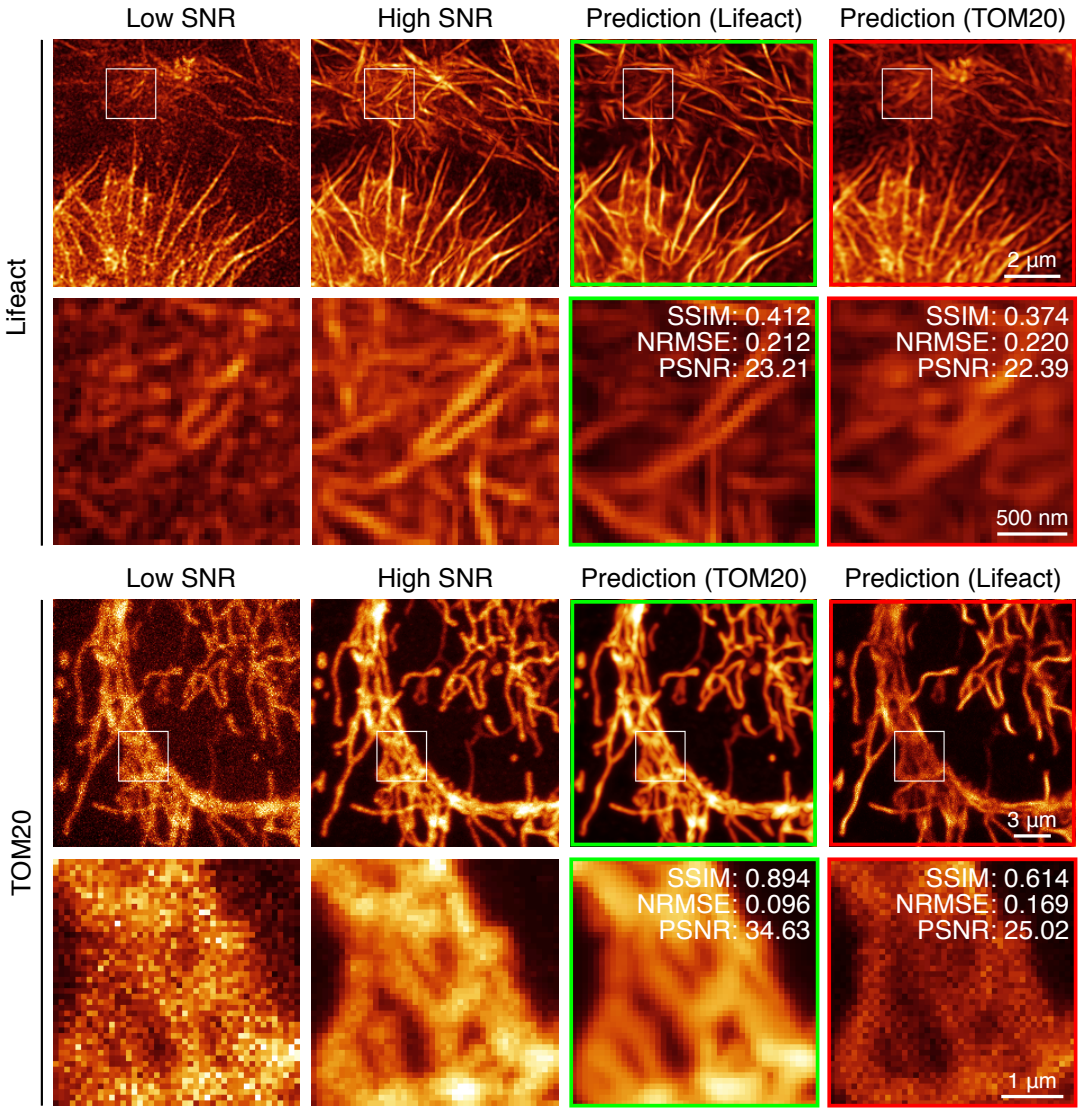


Fig. S3. Of the importance of training models on suitable data. Top: Filopodia (Lifeact) test data and the predictions obtained from a CARE 3D model either trained on similar Lifeact dataset (green box) or on TOM20 dataset (red box). **Bottom**: Mitochondria (TOM20) test data and the predictions obtained from a CARE 3D model trained on similar TOM20 dataset (green box) or on Lifeact dataset (red box).

Quality Control Metrics

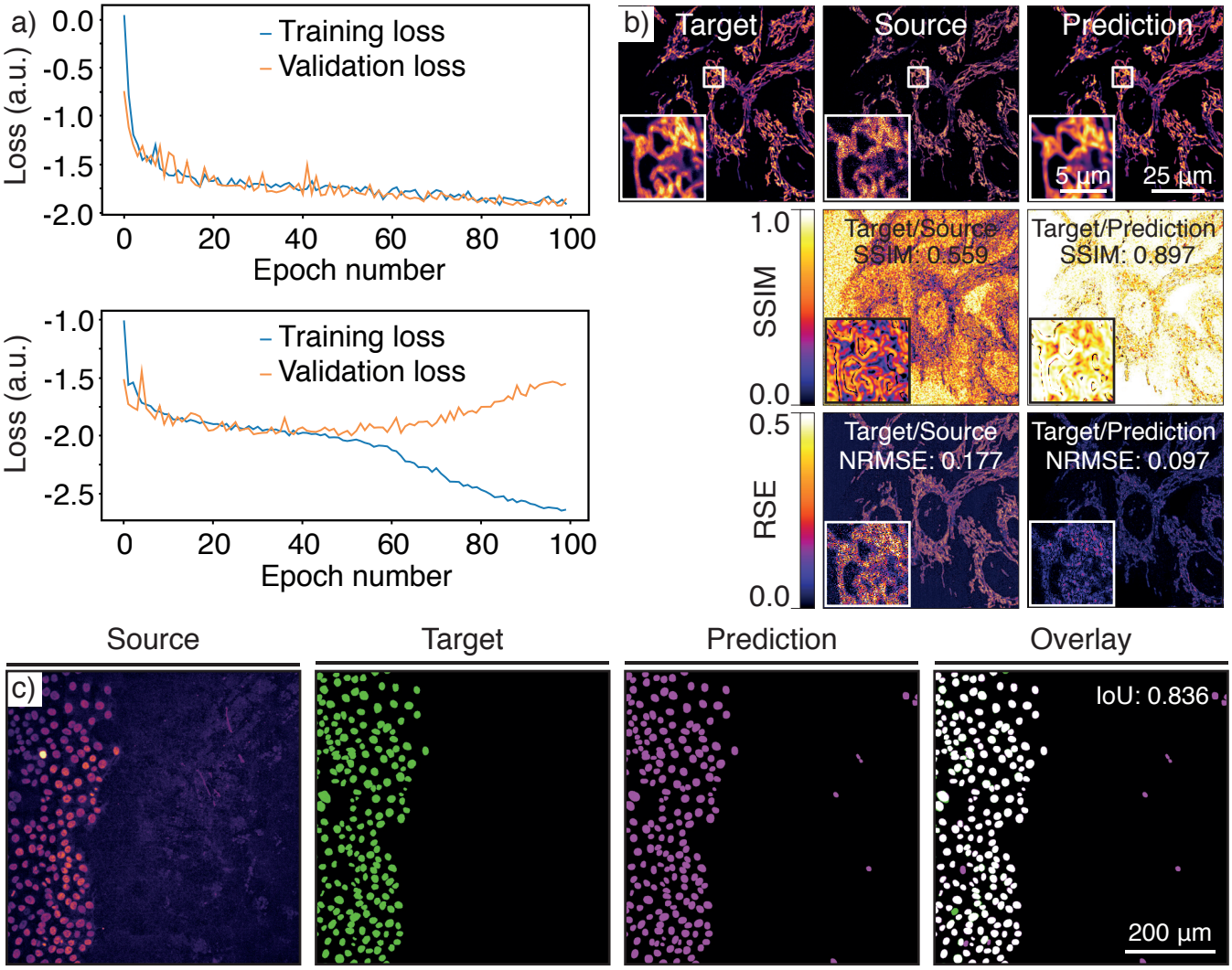


Fig. S4. Quality Control of trained models: a) Overfitting models: Graphs showing training loss and validation loss curves of a Stardist network with different hyperparameters. Upper panel shows a good fit of the model to unseen (validation) data, the lower panel shows an example of a model that overfits to the training dataset. **b) RSE and SSIM maps:** An example of quality control for CARE denoising model performance. **c) IoU maps:** An example of quality control metrics for a Stardist segmentation result.

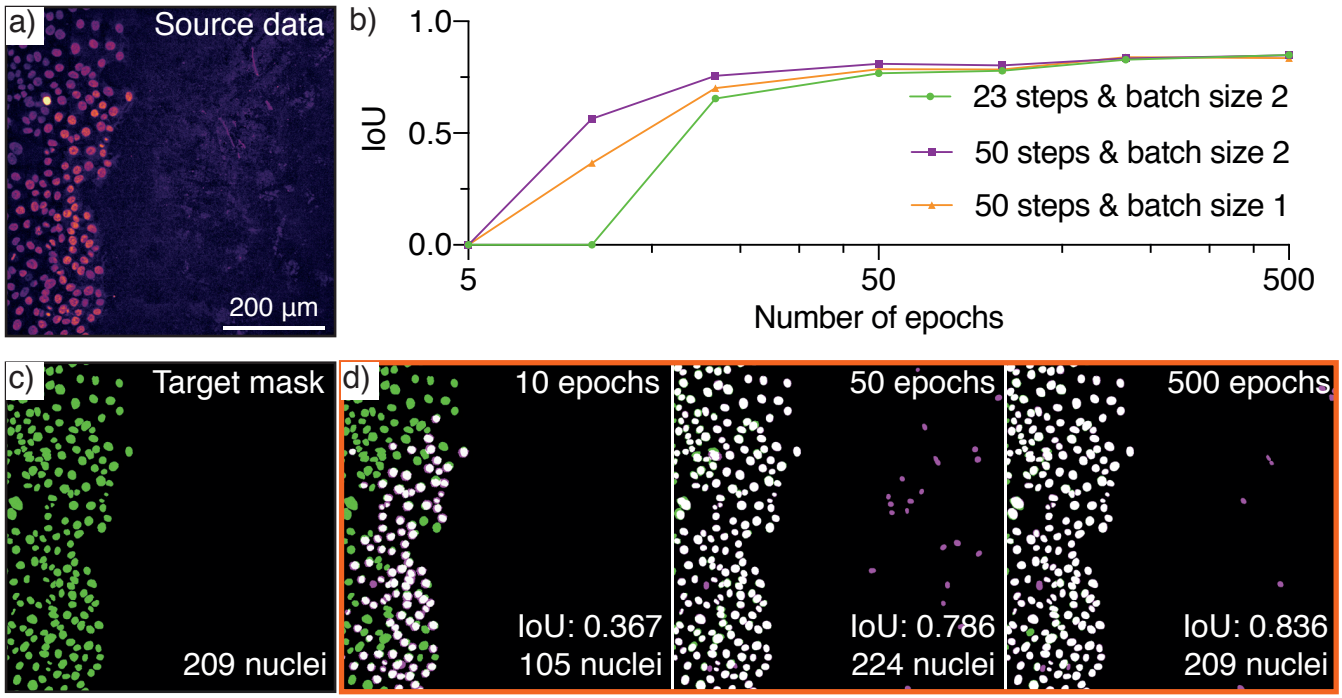


Fig. S5. Model performance vs. training parameters. This figure displays an example of how adjusting parameters (batch size, number of steps and number of training epochs) can affect model performance in Stardist nuclear segmentation. It is also a demonstration how assessing trained models in the ZeroCostDL4Mic notebooks can inform hyperparameter choices and improve predictions.

a) Inspection of the loss function

b) Error mapping and metrics

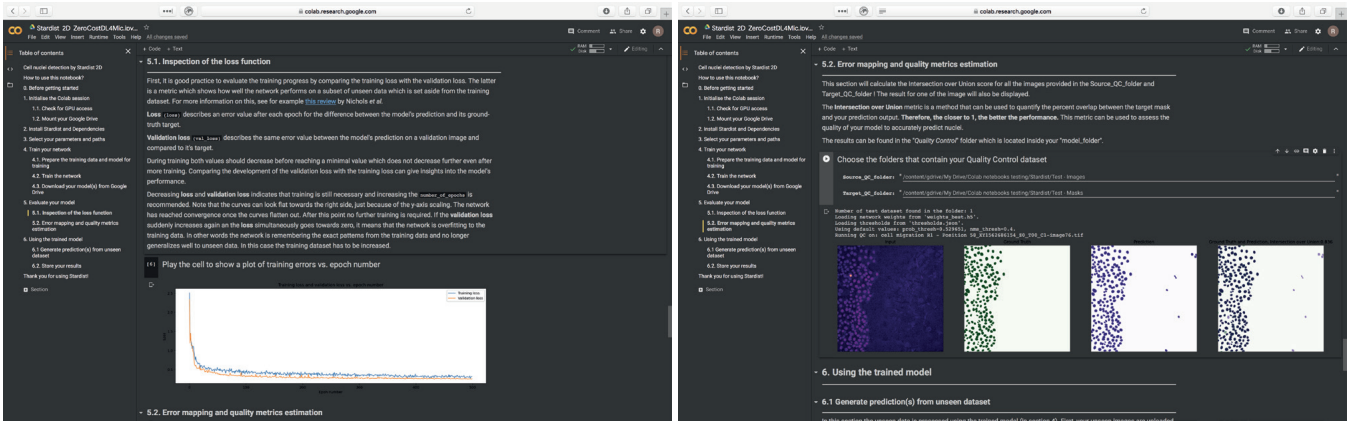
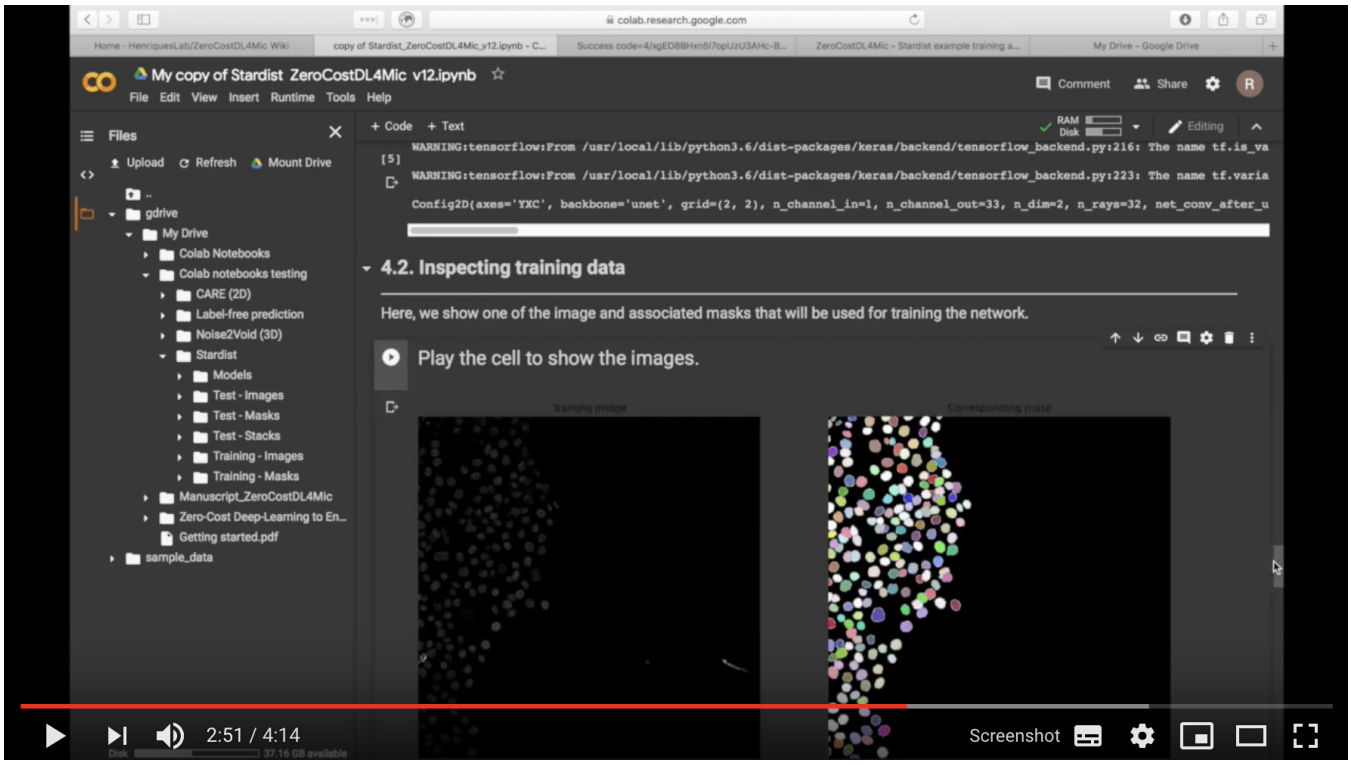
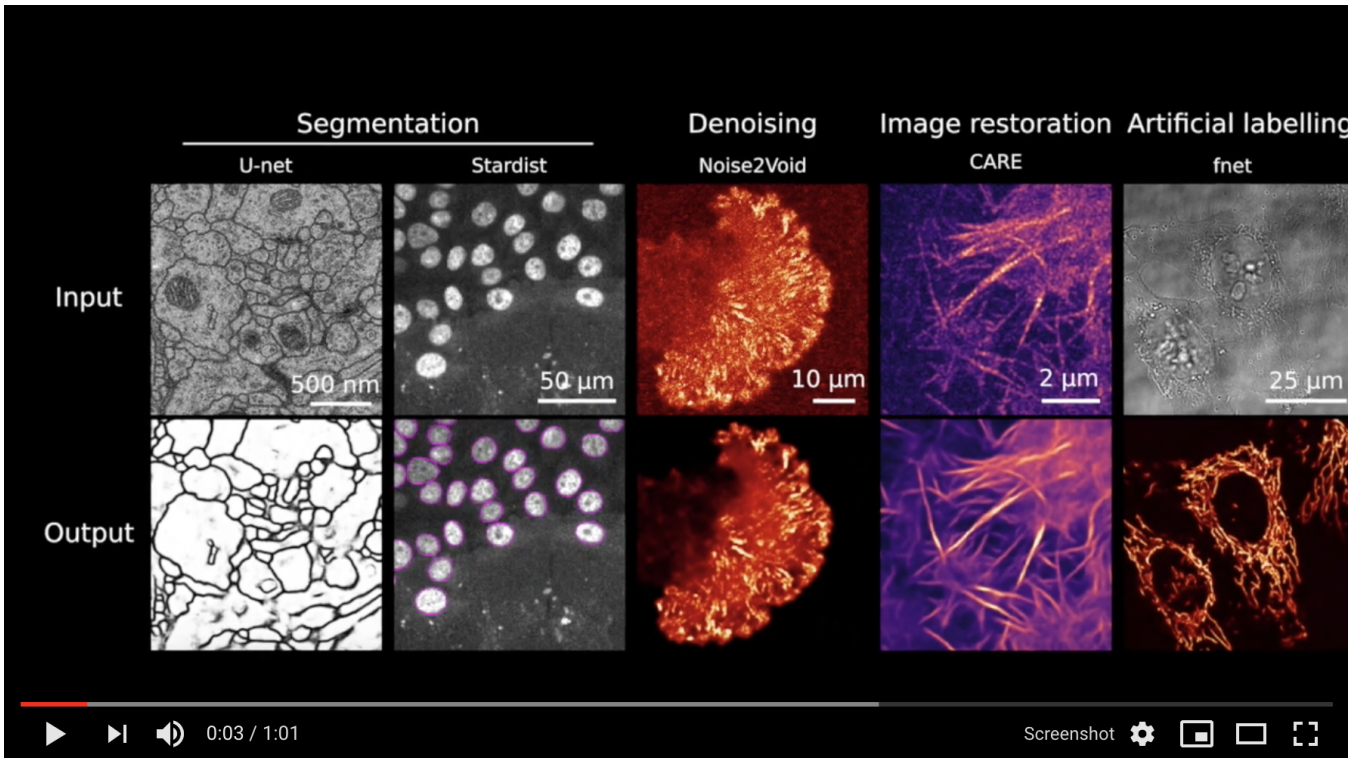


Fig. S6. Quality Control in the ZeroCost4DL notebooks: Screenshot of the two quality control steps performed in the Stardist notebook. These quality control sections are available in all of the notebooks that we provide.

Supplementary Videos



Video S1. Full run through of the workflow to obtain the notebooks and the provided test datasets as well as a common use of the notebook. YouTube link: https://youtu.be/I46tDyLO8QQ.



Video S2. Representative results obtained from the provided test dataset. YouTube link: https://youtu.be/hh2I5xJH67k.

Supplementary Bibliography

1. Chinmay Belthangady and Loic A Royer. Applications, promises, and pitfalls of deep learning for fluorescence image reconstruction. *Nature methods*, pages 1–11, 2019.
2. Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.
3. Martin Weigert, Uwe Schmidt, Tobias Boothe, Andreas Müller, Alexandr Dibrov, Akanksha Jain, Benjamin Wilhelm, Deborah Schmidt, Coleman Broaddus, Siân Culley, et al. Content-aware image restoration: pushing the limits of fluorescence microscopy. *Nature methods*, 15(12):1090–1097, 2018.
4. Martin Weigert, Uwe Schmidt, Robert Haase, Ko Sugawara, and Gene Myers. Star-convex polyhedra for 3d object detection and segmentation in microscopy. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 3666–3673, 2020.
5. Estibaliz Gómez-de Mariscal, Carlos García-López-de Haro, Laurène Donati, Michael Unser, Arrate Muñoz-Barrutia, and Daniel Sage. Deepimagej: A user-friendly plugin to run deep learning models in imagej. *bioRxiv*, page 799270, 2019.
6. Leonhard Möckl, Anish R Roy, and WE Moerner. Deep learning in single-molecule microscopy: fundamentals, caveats, and recent developments. *Biomedical Optics Express*, 11(3):1633–1661, 2020.
7. Lucas von Chamier, Romain F Laine, and Ricardo Henriques. Artificial intelligence for microscopy: what you should know. *Biochemical Society Transactions*, 47(4):1029–1040, 2019.
8. Joshua Batson and Loic Royer. Noise2self: Blind denoising by self-supervision. *arXiv preprint arXiv:1901.11365*, 2019.
9. Aki Stubb, Romain F Laine, Mitro Miihkinen, Hellyeh Hamidi, Camilo Guzmán, Ricardo Henriques, Guillaume Jacquemet, and Johanna Ivaska. Fluctuation-based super-resolution traction force microscopy. *Nano Letters*, 2019.
10. Riina Kaukonen, Guillaume Jacquemet, Hellyeh Hamidi, and Johanna Ivaska. Cell-derived matrices for studying cell proliferation and directional migration in a complex 3d microenvironment. *nature protocols*, 12(11):2376, 2017.