Applications of deep learning in single-cell analysis

PhD Thesis

Réka Hollandi

Supervisor: Péter Horváth, PhD

Department of Biochemistry, Biological Research Centre

Doctoral School of Interdisciplinary Medicine,

Faculty of Medicine, University of Szeged

Szeged

2021

Publications

This thesis is based on the following publications:

I. **Hollandi, R.**, Szkalisity, A., Toth, T., Tasnadi, E., Molnar, C., Mathe, B., Grexa, I., Molnar, J., Balind, A., Gorbe, M., Kovacs, M., Migh, E., Goodman, A., Balassa, T., Koos, K., Wang, W., Caicedo, J.C., Bara, N., Kovacs, F., Paavolainen, L., Danka, T., Kriston, A., Carpenter, A.E., Smith, K., Horvath, P. (2020): "nucleAIzer: a parameter-free deep learning framework for nucleus segmentation using image style transfer", *Cell Systems*, Volume 10, Issue 5, 20 May 2020, Pages 453-458.e6

Estimated IF: 8.673, Q1

II. **Réka Hollandi**, Ákos Diósdi, Gábor Hollandi, Nikita Moshkov, Péter Horváth (2020): "AnnotatorJ: an ImageJ plugin to ease hand-annotation of cellular compartments", *Molecular Biology of the Cell*, Vol. 31, No. 20, 2179-2186, https://doi.org/10.1091/mbc.E20-02-0156

Estimated IF: 3.791, Q2

III. Rasse, T.M., **Hollandi, R.**, Horvath, P. (2020): " OpSeF: Open Source Python Framework for Collaborative Instance Segmentation of Bioimages", *Frontiers in Bioengineering and Biotechnology*, 8:558880. doi: 10.3389/fbioe.2020.558880

Estimated IF: 3.644, Q2

IV. Moshkov, N., Mathe, B., Kertesz-Farkas, A., **Hollandi, R.**, Horvath, P. (2020), "Test-time augmentation for deep learning-based cell segmentation on microscopy images." *Sci Rep* 10, 5068. https://doi.org/10.1038/s41598-020-61808-3

Estimated IF: 3.998, Q1

Other publications related to the subject of this thesis:

V. James L. Daly, Boris Simonetti, Carlos Antón-Plágaro, Maia Kavanagh Williamson, Deborah K. Shoemark, Lorena Simón-Gracia, Katja Klein, Michael Bauer, **Reka Hollandi**, Urs F. Greber, Peter Horvath, Richard B. Sessions, Ari Helenius, Julian A. Hiscox, Tambet Teesalu, David A. Matthews, Andrew D. Davidson, Peter J. Cullen, Yohei Yamauchi (2020): "Neuropilin-1 is a host factor for SARS-CoV-2 infection", *Science*, Vol. 370, Issue 6518, pp. 861-865, DOI: 10.1126/science.abd3072

Estimated IF: 41.845, Q1

VI. Andreas Mund, Fabian Coscia, **Réka Hollandi**, Ferenc Kovács, András Kriston, Andreas-David Brunner, Michael Bzorek, Soraya Naimy, Lise Mette Rahbek Gjerdrum, Beatrice Dyring-Andersen, Jutta Maria Bulkescher, Claudia Lukas, Christian Gnann, Emma Lundberg, Peter Horvath, Matthias Mann (2021): "AI-driven Deep Visual Proteomics defines cell identity and heterogeneity", *bioRxiv*, https://doi.org/10.1101/2021.01.25.427969. Manuscript submitted.

Estimated IF: (not applicable)

Cumulative IF: 61.951

Table of Contents

1. Introduction

Biological experiments and medical examinations often include the investigation of biological samples at a cellular level – such as analysing cells or subcellular compartments – in order to draw conclusions and propose a proper diagnosis or treatment. Analysis of *single cells* and more specifically, their phenotyping can lead to much more accurate description of the studied sample than merely examining the sample as an entity e.g. identifying the tumorous region's extension surrounded by healthy tissue in a section. Considering the frequency and occurrence of affected cells as well reveals more subtle details about the progression of deviation (e.g. in case of a tumour) or vitality and may provide valuable insight into the functional differences compared to healthy samples. Hence, we focus on samples at the single-cell level.

Analysing vast amounts of experimental or patient-related data requires appropriate technology both on the acquisition (automated microscopy) and the processing (i.e. analysis software) part. Both can be aided by high-content solutions, such as high-content microscopes and slide scanners intended for the fast and automatic acquisition of samples, and automated (usually also intelligent) software capable of handling such large data. Such software packages often use *deep learning* (DL), a technology spread worldwide relatively recently.

From the early 2010's *deep learning* has been applied in numerous fields of science including computer vision (e.g. object recognition [1] [2] [3] [4] or autonomous driving [5] [6] [7]), language processing (take speech recognition [8] as example), medicine (such as lesion classification [9]) etc. Despite the first appearance of deep neural networks in the 1960s (e.g. perceptron [10], backpropagation [11] [12] [13], multilayer perceptron [14]), widespread application has been limited for decades due to computational complexity and resource requirements until technological advancements like high-performance GPUs (graphical processing units) were made available commercially bringing about the era of deep learning-powered applications worldwide. The underlying methodology is explained in section 1.1.

In this thesis, an assisted annotation tool, *AnnotatorJ* [15], and a so-called image style transfer-based segmentation method, *nucleAIzer* [16], are proposed. Finally, specific applications are presented [17] [18] for the analysis of single cells using the deep learning-based approaches discussed.

1.1. Machine learning

An algorithm's ability to learn the execution of a task either from a set of examples (supervised) or merely features of a collected dataset provided (unsupervised) is referred to as *machine learning* (ML). A specific supervised machine learning objective, classification, might be reached via numerous classical algorithms (such as kNN: *k* nearest neighbour, SVM: support vector machines etc.) using training data as examples with each example comprising of a feature vector corresponding to an object; its analogy in cellular analysis is phenotyping i.e. sorting the cells to distinct phenotypes (e.g. healthy, cancerous, mitotic, apoptotic etc.). Let us briefly discuss how classical machine learning is related to neural networks (NNs) and deep learning (DL, see **Figure 1.1.**) in the following subsections.

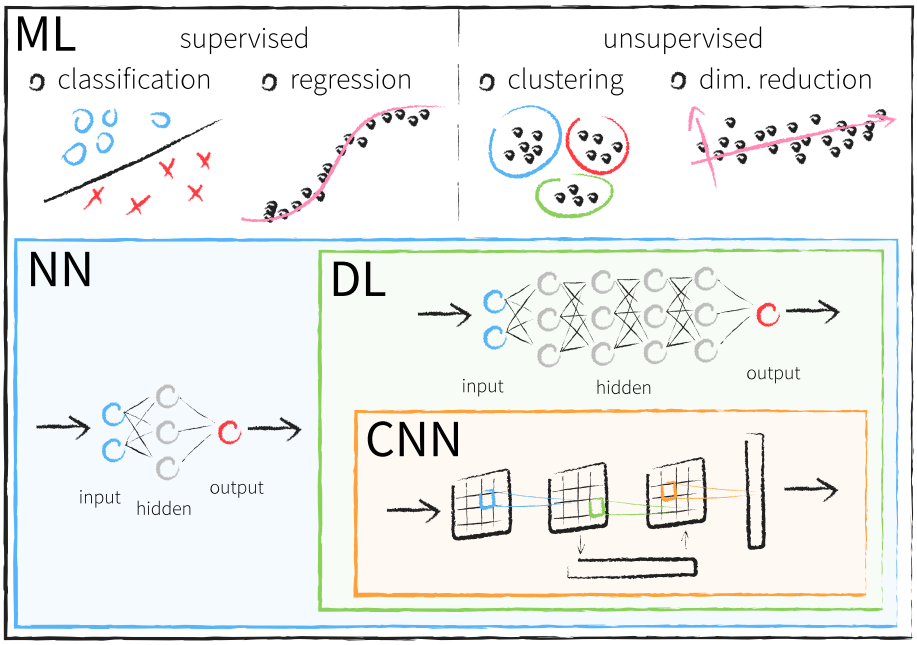


Figure 1.1. Machine learning algorithms overview. ML: machine learning, NN: neural network, DL: deep learning (or DNN: deep neural network), CNN: convolutional neural network.

1.1.1. Neural networks

Modelled after the fundamental structure of the human brain built from connections between its basic processing units, physiological neurons, (artificial) *neural networks* (ANNs) used in computer science are constructed from such *neurons* that execute a single function, the so-called activation function (e.g. a sigmoid), in input- and output layers forming connections between the layers, hence a network [10] [19] [20]. Output vector $\boldsymbol{y}_i$ of the i-th layer is

computed from the activation function f of its input vector $\boldsymbol{x}_i^T$ using connections weighted with vector $\boldsymbol{w}_i$ and additional bias b as $\boldsymbol{y}_i = f(\boldsymbol{w}_i \boldsymbol{x}_i^T + b)$; this output serves as input for the next layer $\boldsymbol{y}_i = \boldsymbol{x}_{i+1}^T$ in a feed-forward network [13] [19]. Shallow NNs have a single hidden layer of neurons.

1.1.2. Deep learning

When information is propagated through multiple hidden layers from the inputs to the output layer, a *deep neural network* (DNN) is constructed [19] [20]. With each additional layer the number of model parameters increases – so does the computational complexity. Consequently, a large training set is needed to achieve desirable performance that significantly surpasses conventional algorithms [20]. Due to DL's increased resource requirements, it is critical to run DNNs on high computational performance-capable hardware such as GPUs [21] or TPUs (tensor processing units) [22] [23].

A specific type of DNNs, deep *convolutional neural networks* (CNNs) [24] [25] [20] are mainly constructed from convolutional layers – applying the standard image processing procedure, convolution, on their input matrix (see **Figure 1.2C**) – to extract features from the input and pass them to downstream layers e.g. pooling (to reduce size) or a fully connected layer (for classification) [19] [20]. Therefore, CNNs can be effectively used to process images, using the images (2D matrices) as input and outputting either an image (e.g. labelled mask in case of segmentation) [26] [27] [28] [29], object description (e.g. bounding box coordinates in object detection) [30] [31] [3] [5] or a class label (in classification) [1] [32] [33] [34].

1.1.3. Connection to image segmentation

Reliability of microscopy image-based single-cell measurements depend on cell *segmentation* i.e. identification of the underlying area (region) on the image corresponding to each cell individually. Classical image processing methods for object segmentation include Otsu thresholding, watershed, propagation, region growing [35] [36] etc. most of which were broadly applied in bioimage analysis (such as CellProfiler [37]) before the DL revolution of segmentation methods available as convenient software solutions. U-Net [26], Mask R-CNN [27], Cellpose [38], StarDist [39] are included but not limited to the list of such methods; they are discussed in more detail in subsection 1.1.4 and section 4.1. Alternatively, the proposed

approach nucleAIzer [16] is suggested (see in section 4.2). See examples of both classical and DL methods on **Figure 1.2**. See also subsection 3.2.1 and **Figure 3.1** for demonstration.

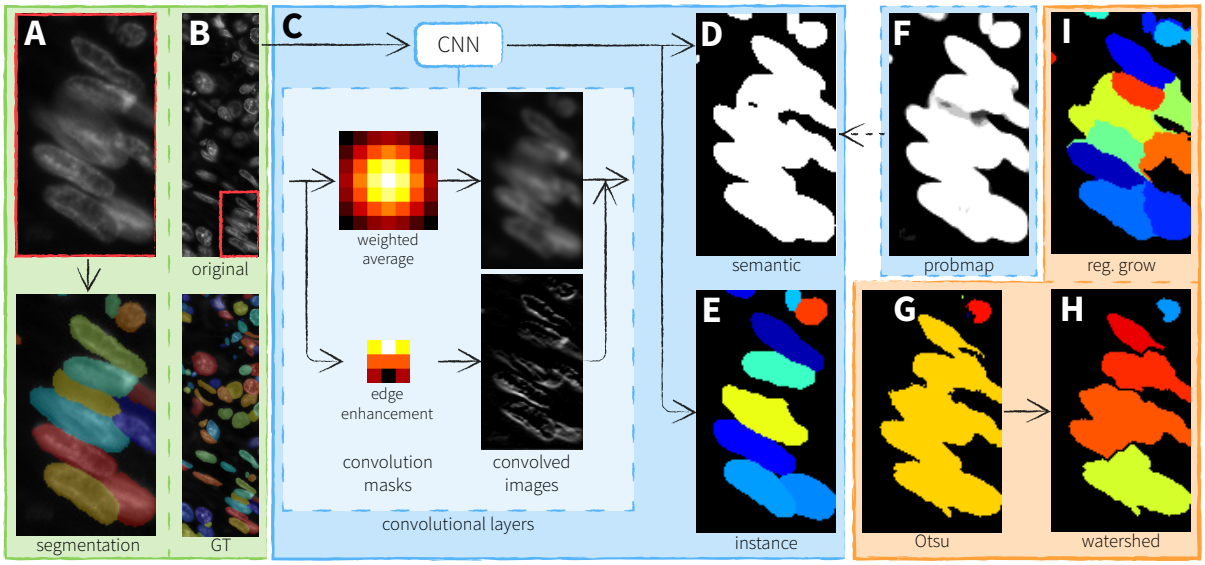


Figure 1.2. Image segmentation methods. A) Crop of a challenging image region with clumped, out-of-focus nuclei according to inset on B marked in red, and its corresponding ground truth segmentation overlayed with random colours. Original image is from DSB test1 [40] (see in subsection 4.3.1). B) Original image and its ground truth, as on A. C) A segmentation CNN whose input is an image-mask pair as on B. Two arbitrary examples are shown for image convolution in a dashed box on input image crop from A: weighted average (Gaussian) and edge enhancement (Sobel); mask values are colour-coded (light: high, dark: low). D) Semantic segmentation of A as a binary image; result of Otsu thresholding F. E) Instance segmentation of A, objects are labelled; result of Mask R-CNN [27]. F) Probability map with floating point values in [0,1]; result of U-Net [26] (e.g. ilastik [41] [42] produces similar maps too). G-I) Classical algorithms: G) Otsu thresholding, H) watershed applied on G, I) region growing from seed points.

1.1.4. Related work

CNNs used in this thesis mainly include Mask R-CNN [27]: an object detection- and instance segmentation network (see **Figure 1.2E**), U-Net [26]: a pixel classification network typically used in semantic segmentation (see **Figure 1.2F-D**) and finally pix2pix [43]: a generative adversarial network (GAN) [44] for image-to-image translation (see **Figures 4.2B, 4.4, S2-3**).

Mask R-CNN [27] is an extension of its predecessor, Faster R-CNN [45] appended with a mask branch to yield segmentation masks inside the detected bounding boxes. The most important additions are 1) the RoIAlign layer for feature extraction which is quantization-free as opposed

to RoIPool of the other and 2) separating mask and class prediction by predicting a mask for each class, respectively, without classification and using the ROI classification branch to classify them. The network has a ResNet backbone for feature extraction, while the Faster R-CNN head is extended with a fully convolutional mask prediction branch.

U-Net [26] was published as specifically intended for bioimage segmentation due to the generally occurring shortage of labelled data for training; it can be trained on a few dozens of images. In contrast, most CNNs expect thousands or tens of thousands of training samples. U-Net's encoder consisting of convolutional layers is responsible for understanding image context via down-sampling with max pooling layers, while the decoder for localization using up-sampling. They form a U-shaped fully convolutional network with skip connections for more precise localization.

GANs are constructed from two fundamental parts that compete during training: the generator attempts to output data nearly indistinguishable from real input while its opposition is set out to identify the artificially generated samples; the latter is called the discriminator [44]. Our selected GAN, pix2pix [43] performs image-to-image translation, that is it learns to transform the input images to a set of targets and outputs artificial images displaying the defined pattern (style).

1.2. Biological background

Personalized medicine has the potential of becoming a future application in various uncurable diseases [46]. The most crucial step of such treatment development is the validation of efficiency on the target cell types, preferably at a single-cell level. Studying the effect of treatment may be performed on either cell cultures or tissue sections from patients; single-cell phenotyping is required for reliable and accurate results in each case.

To visualize the expression of certain protein products (e.g. those affected by the currently studied drug compound) or more generally the functional integrity of the cells, different kinds of labelling techniques can be applied. Most frequently, fluorescent dyes are used to label certain proteins or cellular compartments (such as the nucleus, cell membrane, cytoskeletal proteins like actin or tubulin, lipid droplets or virus particles in infection studies etc.). Typically, DAPI or Hoechst (both emit blue fluorescent light upon appropriate excitation) is used to label

the nucleus, e.g. Alexa Fluor dyes[1] are usually for cytoskeleton, while expressed proteins are often tagged with GFP (green fluorescent protein) or mCherry (red). However, fluorescent labelling often prevents the examination of living cells (except for expressed fluorescent proteins) as the dye conjugation to the target biological structure may cause perturbations or even cell death.

Live-cell assays are usually conducted in a label-free manner (i.e. brightfield microscopy without staining) which proposes challenges to image processing of the sample. Brightfield staining is conventionally applied to tissue sections, most frequently haematoxylin and eosin (H&E), or immunohistochemical (IHC) markers are used to highlight the target structures. We will concentrate on the software-guided analysis of microscopy images acquired from the samples and highlight the different challenges corresponding to each sample type discussed.

1.2.1. Cell cultures

Culturing is a convenient technique in high-content screening experiments as it allows testing of various compounds and treatments in several dosages simultaneously on multi-well plates, possibly on different cell lines. Patient-derived cells can also be cultured in case of clinical trials or medical treatment.

Preparing and maintaining the cell cultures requires experience e.g. to avoid contamination of the medium or mixing of reagents. Each well of a multi-well plate can be used as a separate experimental environment enabling the examination of numerous configurations in a large-scale manner which is ideal for e.g. drug screens or genome knockout experiments[2].

Regarding the analysis of such samples, after undergoing microscopy, the acquired images are analysed with appropriate software; image processing is generally more straightforward in case of cell culture images compared to tissues (especially when using fluorescent labelling, due to negligible background signal).

[1] Alexa Fluor dyes are referred to by numbers close to their excitation wavelength; the colour of emitted fluorescent spectrum is listed here: 488 – green, 568 – orange, 594 – red, 647 – far-red etc.
[2] Each gene in a certain pathway – signalling or metabolic – is knocked out (turned off) one by one, and the cellular response is studied. It can be used to understand the function of a gene in its pathway or its lethality.

1.2.2. Tissue sections

Frequently used in pathology, tissue sections of the target organ are visually inspected to identify affected regions (e.g. tumorous vs healthy). Samples might originate from patients or model organisms and are primarily stained with H&E.

As for their software-guided image analysis, the intrinsic 3D nature of the few microns-thick sections might cause difficulties: cells or nuclei located above or below the given focal plane appear out-of-focus and occluded by surrounding tissue which renders accurate segmentation a challenge. However, also arising from the biological structure of the sample, cells imaged in tissue resemble their true appearance more faithfully than in 2D culture[3], therefore provide further information on the underlying cellular processes.

1.3. Microscopy

Even though only label-free or brightfield-stained (such as H&E) samples might be observed with it, conventional light microscopy suitable to view live cells, cultures, fixed samples or tissue sections is still the most commonly used type of microscopy in cell biology laboratories. Additionally, high-content screening facilities and pharmaceutical industry are equipped with fluorescent (screening) microscopes as well allowing better distinguishability of labelled compartments on separate channels. Furthermore, electron microscopy (EM) can also be used to study subcellular processes at a high resolution without labels, while phase-contrast or differential interference contrast (DIC) techniques may also yield volume information of the objects relative to the bulk sample e.g. in live tissue; 3D spheroids can be visually sliced with light-sheet microscopy. Each method has their own advantages and intended application spectra; however, analysis of the produced images might face various obstacles. Target objects on images acquired in a label-free manner are more difficult to distinguish from the background (culturing dish or adjacent tissue), while fluorescent samples might suffer bleaching[4] or – together with brightfield stains – aggregates can cause artefacts.

[3] we note that 3D cell cultures i.e. spheroids or organoids offer a more appropriate reconstruction of the target cell type or organ than 2D cultures, however, in this thesis we do not concentrate on 3D cultures

[4] when exhaustingly excited, fluorescent molecules undergo conformational change and cannot be further excited to emit signal

2. Aims

We set out to demonstrate how large training data can be generated and used to train image segmentation models. To reach our goals, we conducted two studies: 1) how to efficiently create annotated data sets and 2) how to train robust and accurate instance segmentation DNNs. Subsequently, we investigated how our proposed solutions can be used in real projects and briefly discuss their findings.

The main points of this thesis are as follows.

1) AnnotatorJ, an annotation tool implemented as an ImageJ/Fiji plugin is proposed to help manual object annotation on images via DL.
2) nucleAIzer, a DL instance segmentation pipeline is discussed that utilizes image style transfer to include new experimental data in training.

3. Annotation

Regarding the topic of this thesis *annotation* refers to a collection of processes (manual or semi-automatic) resulting in labelled data retrieved from images suitable to train a neural network to perform a task. Depending on the task several types of annotation may be distinguished such as objects' position, category (class), extension (region) represented by either their centroid, enclosing rectangle (bounding box), label, or list of corresponding points (pixels i.e. segmentation); see 3.2.1. and **Figure 3.1**.

As DL models' performance is dependent on the training data size and quality it is vital to have a sufficiently large and reliable dataset at our disposal prior to training [23]. Such a dataset might be retrieved from publicly available sources for general tasks such as scene segmentation (see e.g. the Cityscapes dataset [47] of traffic video frames that can be used to train autonomous driving systems) or object detection (like Facebook's face recognition). However, medical (and more specifically, patient-related) and biological data cannot be shared in such a straightforward manner due to ethical constraints; even though several public datasets exist of various tissue scans or specific object annotations of e.g. nucleus (see the Broad Bioimage Benchmark Collection [48] [40] [49]). To emphasize its significance in diagnosis, several worldwide challenges have been conducted to push the state of the art in cell segmentation (e.g. DSB 2018 (Kaggle) [40] [50], annual ISBI [51], MoNuSeg 2018 (MICCAI) [52] or MoNuSAC 2020 (ISBI) [53]), classification (e.g. Histopathologic Cancer Detection 2019 (Kaggle) [54], ISBI 2019 [55] or [53]) and tracking (e.g. ISBI 2012 [56]). An expert-annotated dataset was released with each of them positively contributing to potential future methods and applications.

3.1. Annotation strategies

We conducted a thorough comparative analysis of existing annotation software solutions [15] and concluded that most suffer disadvantages either in usage restrictions (whether they are open source, cross-platform, consider sensitive data handling etc.), free availability or supported functionality (specifically focusing on assistance of annotation). A review table from [15] is provided as **Supplementary Table 1**. These software packages usually allow multiple types of annotation such as point, ellipse, rectangle, polygon, free-hand drawing and paint brush strokes. Some operate as a web-service, hence regarding the biological or medical samples discussed in

this thesis locally installable ones are preferred (thus no data is ought to be shared outside the laboratory).

An arbitrary selection of such methods is displayed herein including but not limited to LabKit [57], Diffgram [58] and CytoMine [59] [60]. LabKit is an ImageJ [61] [62] [63] plugin recommended for annotation by the authors of StarDist [39]; it allows trainable labelling and consequently automatic pixel classification similar to ilastik [41] [42] and Trainable Weka Segmentation [64] (also an ImageJ plugin). Certain pieces of software integrate DL models to decrease annotation time and effort; see Diffgram [58] with Tensorflow [65] support and both local and online versions as an example, or CytoMine [59] [60] offering a web interface for annotation with various options and DL model support alongside its local version for general image processing tasks.

3.2. AnnotatorJ

We developed an annotation tool – AnnotatorJ [15] – in the broadly applied bioimage analysis software, ImageJ/Fiji [61] [62] [63] as a plugin to ease usage for end-users in a potentially familiar environment, allow integration into processing pipelines and trivial modification by developers in the image processing community. AnnotatorJ can be used to annotate objects on 2D images (see supported annotation types in subsection 3.2.1. and **Figure 3.1**) and offers assistance in its *Contour assist* function (see in 3.2.2.) using a pre-trained U-Net model-based solution that initializes the object boundaries, thus merely minor subsequent fine-tuning is required by the user to create annotations. This approach allows increased efficiency – both in terms of sparing considerable effort from the expert drawing contours, and faster annotations; see subsection 3.3.1. for detailed results.

Not only can AnnotatorJ become beneficial in bioimage analysis, it may also be employed in general annotation tasks such as crafting datasets of casual object types on images e.g. vehicles, signs and pedestrians in traffic to train self-driving cars' object recognition algorithms (see 3.3.3. and **Figure 3.7** for demonstration and further examples).

3.2.1. 2D object annotation on images

Depending on the given task various kinds of annotations can be constructed on a 2D image. Typically, object detection or recognition (in networks like YOLO [5] [66] or Faster R-CNN

[45]) expects bounding box coordinates of the target object class, while segmentation can be realised in two distinct manners: *instance* (such as Mask R-CNN [27], StarDist [39] or CellPose [38]) or *semantic* (e.g. U-Net). The former defines individual objects as distinct segmented regions – therefore overlapping or adjacent objects can be separated – while the latter does not store such information but the distinct classes of regions; see examples on **Figure 3.1**. Corresponding export data formats can be extracted with the supplemented exporter plugin, *AnnotatorJExporter*.

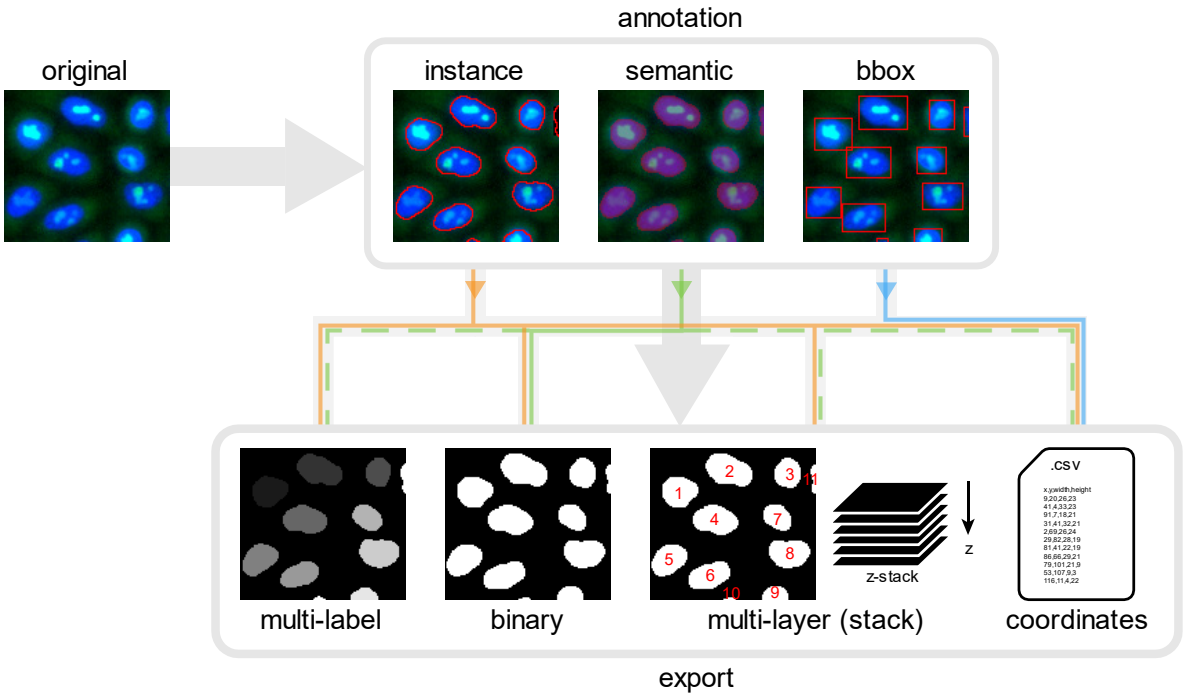


Figure 3.1. Annotation and export types supported in AnnotatorJ. A sample fluorescent cell culture image is annotated in red representing the three annotation types (instance, semantic and bounding box – the latter denoted as 'bbox') identifying the nuclei (stained in blue). Possible export types are connected with colour-coded lines for each annotation type, respectively. Dashed lines indicate that even though conversion to an export type is possible, results might not contain what the given export type was intended for (e.g. touching objects on a semantic annotation image exported as coordinates might contain multiple objects merged as one since they cannot be separated). Figure is adapted from [15].

Additionally, classes can be assigned to each annotated object in instance or bounding box annotation types using the *Class mode* of AnnotatorJ (by selecting its checkbox in the main window of the plugin). Detailed description of all available functionality as well as install

instructions are supplemented to [15] in its corresponding GitHub repository's documentation (https://github.com/spreka/annotatorj/blob/master/AnnotatorJ_documentation.pdf).

Annotation options can be set either in the plugin by 1) selecting checkboxes in the main window to e.g. edit existing contours, overlay additional annotations, add new objects automatically or use *Class mode* or *Contour assist* (see 3.2.2. for details) etc., 2) adjusting parameters in the options window (via button '…') for e.g. brush size or *Contour assist* settings, or 3) in the config file *AnnotatorJconfig.txt* (located by default in */plugins/models/* in the ImageJ folder) including e.g. model path and defaults such as class list for saving, annotation type and colour etc.

3.2.2. Contour assist function

Semi-automatic annotation is feasible in the plugin via its *Contour assist* function for which either a classical image processing algorithm (region growing [35] [36]) or a DL network, U-Net [26] may be selected, and the given object's contour is initialized with it so that the user would need only to perform minor adjustment to finalize the annotation. Our aim is to avoid fully automatic object annotation by requiring a final manual step to fine-tune, accept or reject the algorithm-yielded boundaries to ensure user- (in bioimage annotation, expert-) curated data is created – as ground truth (GT) – and is used to train subsequent models. The effect automatically created GT data – i.e. fetched from an algorithm – may have on training is demonstrated in [16] and section 4.3. comparing segmentation performance of models trained on expert-annotations against automatic masks, showing increase for the former.

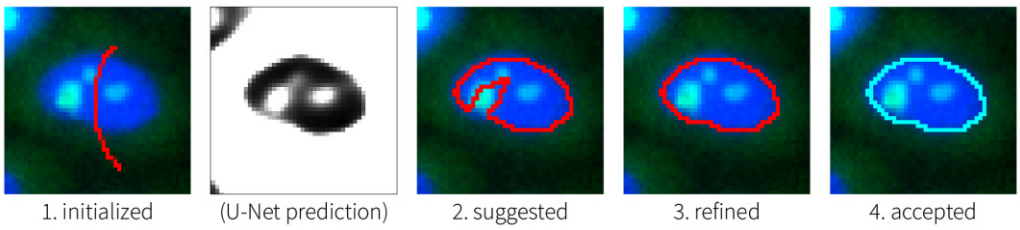


Figure 3.2. Contour assist mode. The images show contour suggestion steps in order. 1) an approximate initialization of the object contour is performed by the user via as little as a line drawn over the object (in red), 2) based on the U-Net prediction a contour is suggested automatically within a bounding box defined by the initial contour, 3) the user manually refines the contour if needed and 4) the contour is added to the annotated objects using shortcuts; see details in the software documentation. Figure is partially adapted from [15]. See also the video supplementary in [15].

The figure above (**Figure 3.2**) and the video supplementary in [15] shows how the suggested contour (initialization) in *Contour assist* mode is corrected by the user, thus mostly the uncertain or challenging image regions require tender attention from the annotator, while rather obvious regions such as distinct objects clearly visible on the background can be easily and quickly labelled. Therefore, minimizing user interaction needed to create an annotation, the process is far less tedious and considerably faster: see evaluation in subsection 3.3.1.

The process of contour suggestion – considering the use of U-Net as the underlying method as presented on **Figure 3.2** – is performed as follows. Firstly, a region in which contour suggestion is desired to happen automatically must be marked by the user; this is the initialization step that might be performed by a roughly drawn line across the object. Its purpose is to define a bounding box extended to x-by- x pixels in width and height around the initial contour, where x is defined in the plugin configuration file or in the options. Subsequently, the selected method (U-Net) is used to determine pixels in this bounding box whose probability of belonging to the target class (object) is larger than a weighted threshold t to yield a suggested contour around such a defined region of pixels. This contour is displayed on the image for minor correction by the user – using brush editing according to ImageJ standard – to align the contour precisely to the object boundaries as the user sees fit. Finally, the revised contour can be accepted or rejected (even without edit) via shortcuts in the plugin. Suitable tools are automatically selected.

When the classical region growing [35] [36] algorithm is selected for *Contour assist* a more precise initialization is expected to define the seed within the object which the algorithm would extend following its constraints ideally towards the true edge pixels of the object.

The major bottleneck of DL model-assisted semi-automatic annotation is the need for prior existence of an appropriate pre-trained model in the target object class. Obviously, such a model can be trained on publicly available datasets that exist for several applications (as introduced in the beginning of chapter 3); however, reliability cannot be guaranteed *per se*, e.g. in the PSB 2015 dataset [67] (based on TCGA [68] [69]) storing image region annotation masks corresponding to specific cellular phenotypes on histological images only polygons are marked, consequently preventing accurate image segmentation model training that would rely on pixel-precise regions. Notably, the extracted regions still hold valuable information of the target class and may be used to train e.g. an object detection method. On the other hand, as the case for

several published datasets [70] [40] where curation was performed by multiple experts simultaneously on the same data averaging was applied in cases when the experts disagreed – in e.g. the exact boundary pixels, or whether to include an image region as an object of the target class against out-of-focus regions or debris resembling real objects –, an obvious GT might be difficult to find. We also experienced this phenomenon as explained in 3.3.2. comparing annotators.

Additionally, users can train their own custom models and use them in AnnotatorJ; see subsection 3.2.3. for details.

3.2.3. Implementation details

ImageJ [61] [62] [63] was chosen as the implementation framework (written in Java) for this plugin due to its broad user base among both image analysts and developers. The AnnotatorJ plugin can be installed via ImageJ/Fiji's built-in updater with the following credentials: update site name is *AnnotatorJ*, URL is https://sites.imagej.net/Spreka/.

Alternatively, it can be built from source using maven [71] following instructions provided in the GitHub repository's readme (https://github.com/spreka/annotatorj#building-from-source) or used as a standalone pre-built version from the releases (https://github.com/spreka/annotatorj/releases) where a U-Net model pre-trained on heterogeneous nucleus data can also be found.

AnnotatorJ uses DL4J [72] for the U-Net model implementation in Java which allows any custom user model trained in the Keras [73] framework (whether in Python or DL4J) to be loaded in the plugin. Consequently, any target class of object can be efficiently trained in AnnotatorJ (see also subsection 3.3.3).

U-Net might be substituted with two alternative classical image processing algorithms to aid contour initialization: region growing [35] [36] and active contours [74]. While such classical algorithms promise faster execution on conventional computers, their accuracy is significantly lower. Even though DL predictions may be accelerated with GPU implementation [21] (provided optionally in the plugin via DL4J, configurable with maven), the default CPU version is preferred in most use-cases to allow usability on potentially any computer device without a dedicated GPU which is the typical case for the primary target audience of biologists.

Easy and convenient integration in a further DL framework, OpSeF (Open Segmentation Framework) [75] is also possible via its data structure support in AnnotatorJ using the plugin *AnnotatorJImportOpSef*. This plugin treats images and masks created in OpSeF as simulated z-stacks to allow handling of 3D-like image data, and imports/exports annotations and masks as such. Masks to import might be results of an instance segmentation algorithm (like StarDist [39], included in OpSeF) or also carry class information which is preserved upon import to AnnotatorJ. Simulated slices of the stack can be scrolled alongside their corresponding ROIs displayed as overlays (following ImageJ ROI visualization) while objects of different classes are identified by their contour colours. Masks exported from AnnotatorJ are saved to separate files by classes (see the documentation for technical details) allowing re-import and further processing in OpSeF.

3.3. Results

We tested AnnotatorJ with regards to annotation time and accuracy on the following four distinct image sets[5]; example images are displayed on **Supplementary Figure S1**.

i) nucleus target object class on a variety of cell culture and histology images acquired in brightfield or fluorescence microscopy
ii) cytoplasm target object class on an image set similar to i)
iii) cytoplasm target object class on an electron microscopy (EM) image set from ISBI 2012 [56]
iv) car target object class on an arbitrary selection from the Cityscapes dataset [47]

The plugin was evaluated according to annotation time (see in subsection 3.3.1) and contour accuracy (details are given in subsection 3.3.2) on these test image sets. We asked experts with biological image annotation experience to participate in the tests and annotate the test images twice: firstly, with the *Contour assist* mode of AnnotatorJ, then without it only enabling the automatic adding function. We ensured the comparability of these pairwise experiments by asking each annotator to perform both trials on their own computer (with the same hardware

[5] sets i) and ii) contain images provided as courtesy from Kerstin Elisabeth Dörner (ETH Zürich, Switzerland), Andreas Mund (University of Copenhagen, Denmark and Max Planck Institute, Germany), Viktor Honti (BRC, Hungary) and Hella Bolck (University Hospital Zürich, Switzerland)

parameters). We found that for an optimal performance 8 GB of RAM is needed – which is feasible even in conventional laptops (as of writing this thesis).

3.3.1. Annotation time efficiency

Annotation time was measured in the plugin on an object-level for all tests. **Figure 3.3** displays mean annotation times in ms on test sets *i-iv* in the pairwise test (with or without *Contour assist*). We concluded that with *Contour assist* annotation times were significantly decreased for test sets *i*,*iii* and *iv* on average (6 times out of 9); see our two-sample t-test statistics in **Table 3.1**. We note that test set *ii* (cytoplasm images) contained objects we found the most difficult and time-consuming to annotate as overlaps are frequently present and membrane borders can often be barely separable from surrounding tissue structures (in case of histopathology images).

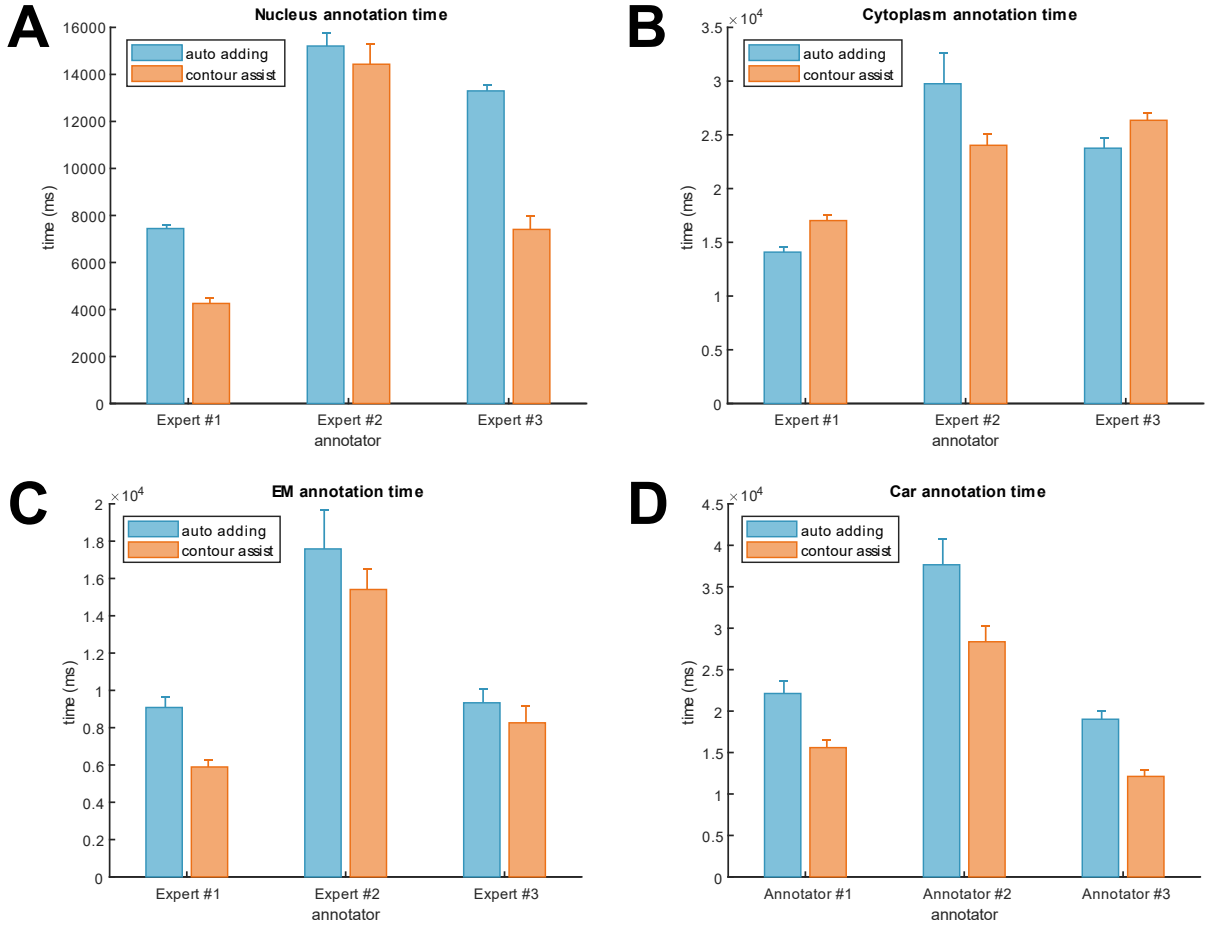


Figure 3.3. Annotation times on test sets *i-iv* comparing *Contour assist* mode with another useful function of AnnotatorJ (*automatic adding* of objects). Times were measured per object on each image then averaged for all objects and all images in the given test set. Error bars show SEM (standard error of the mean: standard deviation

divided by the square root of the number of elements). Panels A-D correspond to test sets *i-iv*, respectively. Figure is partially adapted from [15].

annotator	**test set**			
	nucleus	cytoplasm	EM	cars
#1	**2.11e-20******	**0.001679****	**1.48e-05******	**0.002958****
#2	0.738399	0.978379	0.362086	**0.028521***
#3	**1.18e-39******	**0.018930***	0.381501	**5.09e-05******

Table 3.1. Two-sample t-test evaluation of annotation time efficiency using *Contour assist* mode. For each test set we tested the null hypothesis that annotation times measured with either *Contour assist* or *automatic adding* function have equal means at $\alpha = 0.05$ significance level, times were considered on a per object level. In each cell p-values are displayed; asterisk notation is as follows: *: p<0.05, **: p<0.01, ***: p<0.001, ****: p<0.0001; significant difference is displayed in bold. Mean annotation times usually decreased using *Contour assist*, we mark cases with increased time in grey. We note that even though paired t-test seems to have been applicable here, the number of annotated objects did not match (see Table 3.2), hence we chose two-sample t-test.

annotator	**test set**			
	nucleus	cytoplasm	EM	cars
#1	344 | 351	365 | 359	140 | 159	54 | 55
#2	402 | 392	427 | 419	182 | 187	57 | 57
#3	309 | 340	307 | 359	155 | 169	51 | 50
GT	389	401	164	56

Table 3.2. Total number of objects in each test image set. Annotated object numbers presented are as: Contour assist | auto adding.

Independent experts were asked to create ground truth (GT) annotations (masks) prior to these statistical tests. **Table 3.2** shows the object numbers found by annotators in each test set in *Contour assist* and *automatic adding* modes compared to GT; a considerable difference is always found for any given expert even on the same test image set – especially prominent in the most obvious case of cars – showing both intra- and inter-expert deviances (see also in 3.3.2) focusing merely on detected object numbers.

3.3.2. Accuracy evaluation

The same experiments as in subsection 3.3.1 were used to evaluate the accuracy of annotations. Intersection over union (IoU) was chosen as an evaluation metric for contour accuracy

compared to ground truth masks. IoU score is computed as the intersection area divided by the area of the union of the predicted object and its corresponding ground truth object; the best-matching object pairs were selected. This score can be calculated at several thresholds t, the range [0.5,0.95] by 0.05 steps is used in this thesis, by counting the number of true positive (TP), false positive (FP) and false negative (FN) objects; see equation 3.1. An object is TP if its IoU score is greater than the current threshold t. An arbitrarily small $= 10^{-40}$ is added to the denominator for numerical stability. The formula resembles that of precision (equation 3.2), it may be considered as a modified mean average precision, and is also referred as Jaccard index (equation 3.3 where x corresponds to the prediction pixels and y to the ground truth). This metric was also used in chapter 4; we followed the definition given in the Data Science Bowl (DSB) 2018 [40] [50] nucleus segmentation competition as represented on **Figure 3.4** below.

$$IoU\ score(t) = \frac{TP(t)}{TP(t) + FP(t) + FN(t) +} \tag{3.1}$$

$$precision(t) = \frac{TP(t)}{TP(t) + FP(t) +} \tag{3.2}$$

$$Jaccard\ index(x, y) = \frac{|x \cap y|}{|x \cup y|} = \frac{|x \cap y|}{|x| + |y| - |x \cap y|} \tag{3.3}$$

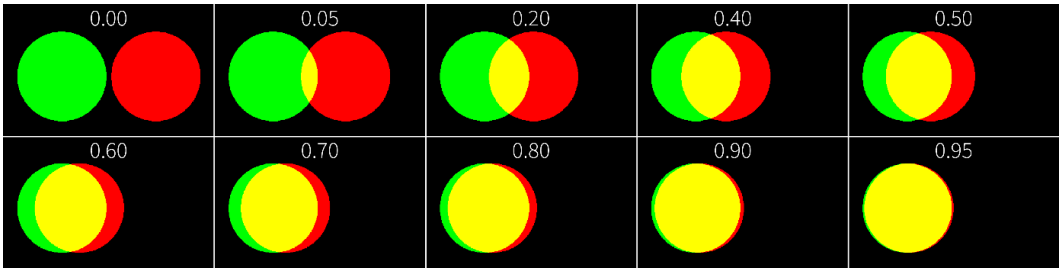


Figure 3.4. IoU scores. Formulation matches that of DSB 2018 as in equation 3.1. Visual examples represent several IoU scores; the green circle is the ground truth, the red one is the prediction while yellow marks the intersection. An IoU score of 0.0 means no overlap, while 1.0 means total overlap with the ground truth. Even at a threshold of 0.5 (see top right example) the overlapping area of the objects yields a relatively fine detection, while at IoU>0.9 only a few pixels differ.

We computed an IoU score for each test image in each set at the ten selected thresholds, averaged them for each image to yield a single IoU score per image, then calculated the mean over each image set (see **Figure 3.5**). Since image numbers did not change between the pairwise experiments (using or not using *Contour assist*), paired t-tests were applied for all image sets,

paired by annotators. We found significant difference for only one of the experts in test set *ii*; asterisk notation is used on **Figure 3.5** as in **Table 3.1**. Test set *ii* (cytoplasm images) was expected to yield larger accuracy difference as these images were the most difficult due to complicated background, crowded areas and overlapping objects resulting in uncertainty. Our results confirm that the accuracies showed larger inter-expert than intra-expert differences (see **Fig. 3.6**), proving that AnnotatorJ can be efficiently and accurately used to create annotations.

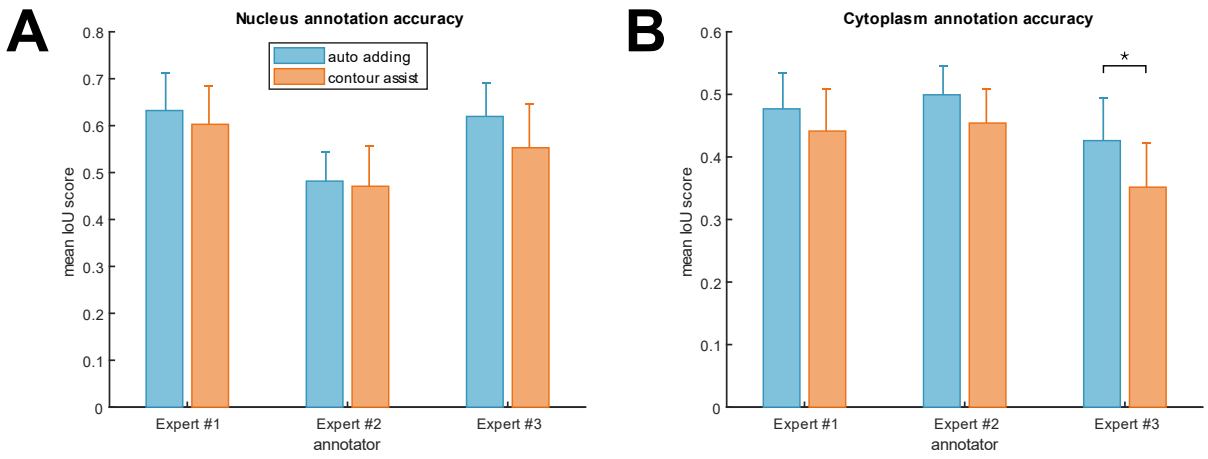


Figure 3.5. Annotation accuracies according to mean IoU scores. Scores for images were averaged in each test set. Error bars show SEM, colours and notations are as in Figure 3.3. Legend is omitted for panel B for better visibility and is the same as in A. Panels A-B correspond to test sets *i*-*ii*, respectively. Asterisk notation corresponds to p-values of paired t-tests, significance is shown as in Table 3.1. Figure is partially adapted from [15].

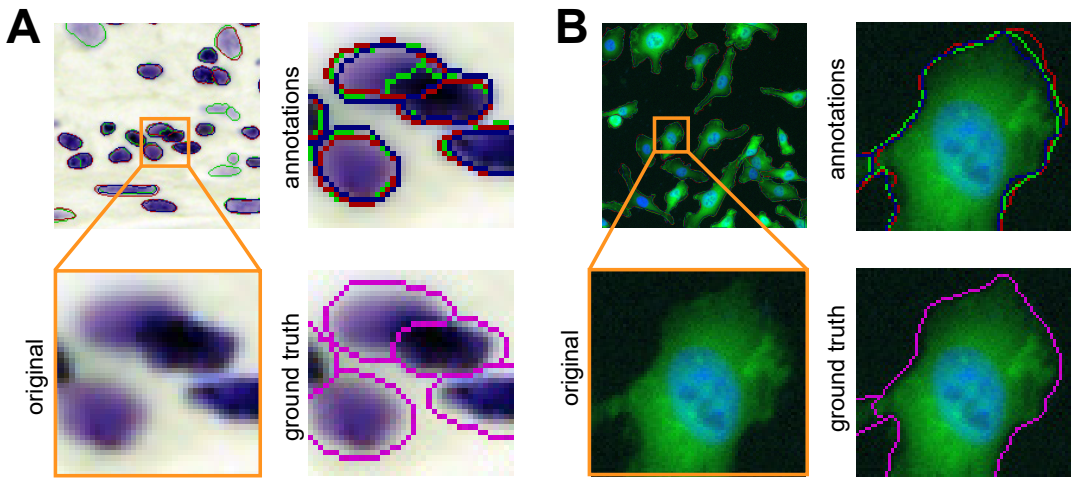


Figure 3.6. Visual representation of difference between annotators. Inset shows a zoomed region of the original image outlined in orange, the corresponding expert annotations (red, green and blue) and ground truth (magenta) are overlayed as contours. A) Example from the nucleus test set, B) example from the cytoplasm test set. Figure is partially adapted from [15].

3.3.3. Perspectives

We demonstrate the wide range of potential applications AnnotatorJ can be used on test sets *iii* and *iv*; **Figure 3.7** presents custom models trained on image modalities vastly different from sets *i* and *ii*.

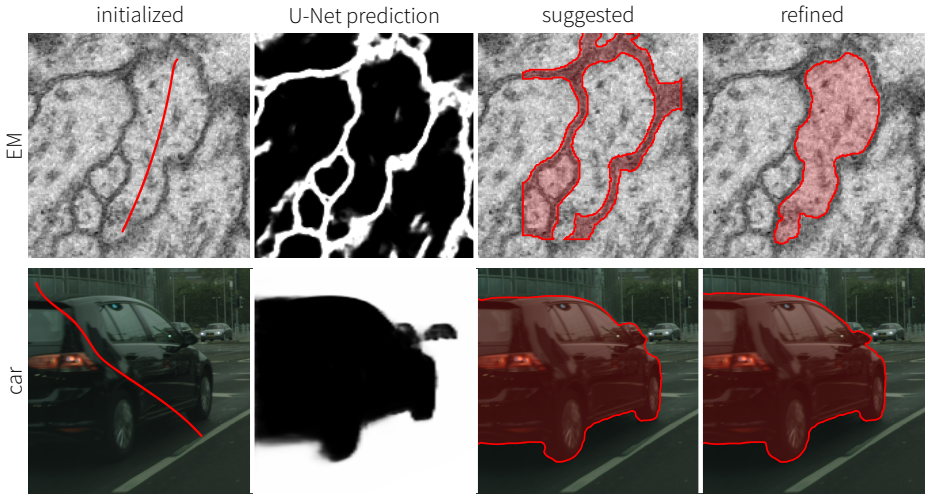


Figure 3.7. General object class annotation assisted in AnnotatorJ. Rows show examples from the EM (*iii*) and car (*iv*) test sets, respectively. Steps are as in Figure 3.2. (refined contours are also accepted). The EM example contour suggested by U-Net was inverted using 'Ctrl'+'u' key combination in the plugin, then refined with the brush. Contour outlines were visually highlighted in Adobe Illustrator for better visibility.

Furthermore, various household items (like cutlery, cup, laptop, furniture etc. as in the COCO dataset [76]), or targets in automated robotic sorting systems (e.g. visual quality control of products on an industrial pipeline [77]) can be easily annotated for free and used to train custom models – unlike most commercially available solutions (including e.g. Lionbridge.AI [78] or Hive [79]) that operate in a project management manner and charge the user for outsourced annotations.

As for future improvement perspectives, further DL models could be integrated in the plugin. Additionally, it could be extended to incorporate and utilize active learning[6] for the efficient sparse annotation[7] of challenging object types, further decreasing annotation time and effort.

[6] specific machine learning algorithms present such queries to the user for labelling that they are the most uncertain about to allow fine-tuning of the class-separating function
[7] annotating only a narrow selection of all the objects present

4. Image segmentation with deep learning

In the previous chapter we discussed how important it is to have reliable training data in order to train our deep learning models accurately, and we suggested AnnotatorJ [15] as an annotation tool to create such datasets efficiently (see **Figure 4.1**). Subsection 3.2.1. and **Figure 3.1.** show the main types of annotation (and segmentation) as well as what kind of deep learning models can be trained with them. *Instance segmentation* is particularly useful in cellular analysis since single cell-level measurements require object-based identification of cells on microscopy images so that downstream analysis may retrieve the accurate phenotypic profile of the sample.

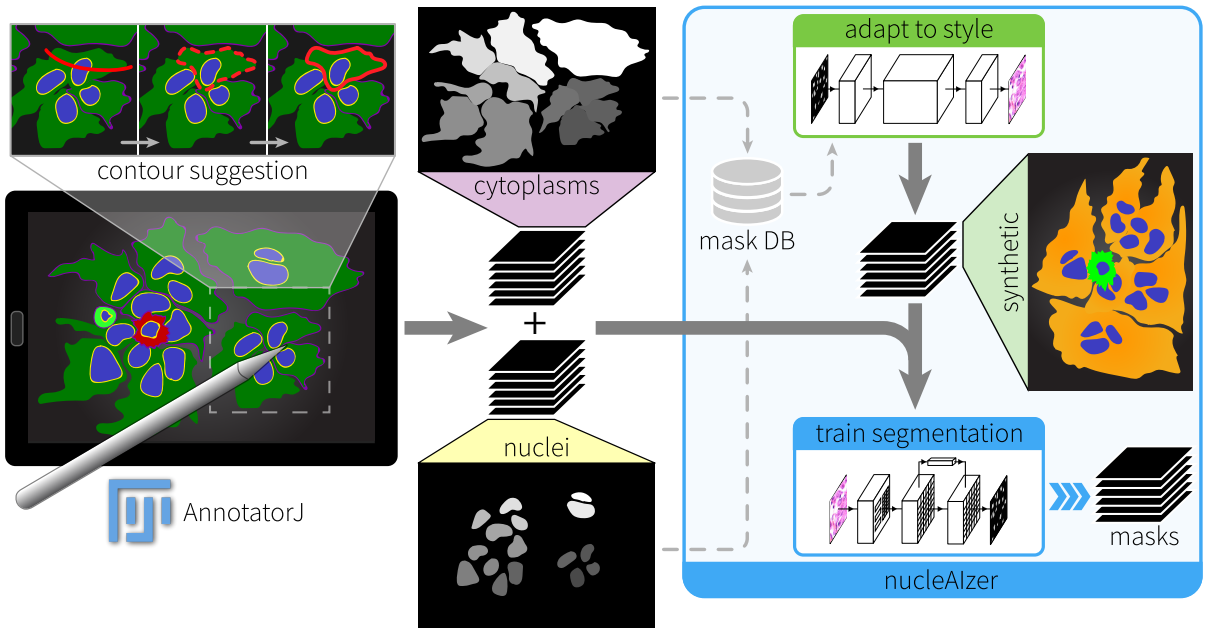


Figure 4.1. Connecting AnnotatorJ and nucleAIzer. Objects annotated in AnnotatorJ can be exported to proper training data format suitable to train different types of deep learning models, here multi-labelled masks are shown to fit to the nucleAIzer pipeline (see details in section 4.2). Contour assist mode may be used to suggest contours to speed up the annotation process (see details in subsection 3.2.2). The annotated object masks are stored in a mask database (denoted as 'mask DB' on the figure) that is used in the mask generation and image style transfer step of the nucleAIzer pipeline (green box on the figure; see in subsection 4.2.4). Resulting synthetic microscopy images adapted to the new experiment's style (appearance) are generated and forwarded to train a segmentation model together with the annotated masks exported from AnnotatorJ. Finally, the trained model yields segmentation masks as output.

After a short overview of methods found in the scientific literature, we focus on an approach that offers a new application of image style transfer [43] via generating such artificial

microscopy images that carry the new experiments' image domain and forward it to train a segmentation model (see in section 4.2); we call this approach *nucleAIzer* [16].

4.1. Deep learning segmentation methods

As briefly introduced in subsection 1.1.4, Mask R-CNN [27] is a segmentation CNN that can be efficiently applied on various image modalities (see e.g. the COCO dataset [76]). U-Net [26] may also be used to retrieve instance segmentation masks (as discussed in [26] [80]) using appropriate weights on object-separating border pixels during training. More recently, StarDist [39] and Cellpose [38] proposed novel networks also suitable for cell segmentation on instance level.

StarDist [39] was designed for the segmentation of *star-convex* polygons – making this method ideal for the usually elliptical nuclei but unfit for eccentric cytoplasm shapes observed in cell cultures. Since nuclei are round or elliptical, the authors claim their shape representation is more suitable for nucleus segmentation as opposed to most methods (like Mask R-CNN) depend on axis-aligned bounding boxes unable to properly take advantage of non-maximum suppression (NMS) in overlapping detections. StarDist predicts object probabilities and constructs polygons with radial distances for each, using a U-Net-based CNN, then yields instances via NMS. [39] suggests usage in crowded fluorescent images as an example of a challenging case. The method has a Fiji [62] plugin for prediction, and a 3D variant has also been published recently [81].

Cellpose [38] creates a vector flow representation of labelled instances (objects) and trains its network accordingly so that it predicts horizontal and vertical flows (i.e. gradients) of objects on test images along with a separate probability prediction whether a pixel belongs to objects, then these three maps are combined as a flow field. Its architecture is based on U-Net [26], and it considers image *styles* similar to our terminology (see in subsections 4.2.3-4) in training and prediction. Cellpose is designed to incorporate new user annotations potentially containing new experimental information in order to improve in time (with new releases) and be able to truly generalize well, these annotations – as previously discussed in chapter 3 – require expert knowledge to reliably contribute to a better performing model. It also has an online interface.

However, these solutions do not allow the incorporation of new unlabelled data modalities in training, thus require new annotations each time. On the contrary, the proposed instance

segmentation pipeline, *nucleAIzer* [16] is capable of adapting to new data via image style transfer without any new annotation. The method is explained in the following section (4.2).

4.2. nucleAIzer

We developed nucleAIzer [16] as a pipeline based on deep learning primarily intended for cell nucleus segmentation that has also been successfully applied in further cellular compartment segmentation tasks (see details in chapter 5) ever since. The method was motivated by the 2018 Data Science Bowl (DSB) [40] [50] nucleus segmentation competition detailed in subs. 4.3.1.

The main advantage of nucleAIzer compared to similar – let them be DL-based or classical – segmentation methods is its adaptation ability to such novel experimental scenarios that the pre-trained model was not prepared for, hence is expected to perform poorly on. Even though e.g. CellPose [38] plans to release new models trained on an extended image set of user-annotated custom experiment-derived images to improve its generalization ability, it still depends on new annotations (see issues in chapter 3) as most existing solutions do. nucleAIzer on the other hand synthesises artificial intensity images based on auto-generated masks and uses them to train its segmentation model preparing it to generalize well on the new image domain without new annotations. The synthetic image generation step is referred to as image style transfer learning [43]. Since such images can be generated in any desired number, it may also be used to emphasize rare types of images originally present in the training data.

The project is open source and freely available to download for local deployment on Windows and Unix-based systems at https://github.com/spreka/biomagdsb. Additionally, an online version is provided for the users' convenience for fast and easy testing on their own experimental images at www.nucleaizer.org. Finally, CellProfiler [37] integration is also possible via the plugin available at https://github.com/CellProfiler/CellProfiler-plugins/blob/master/nucleaizer.py (temporarily at https://github.com/CellProfiler/CellProfiler-plugins/blob/master/CellProfiler3/nucleaizer.py). In the near future, nucleAIzer will also be transformed into a napari [82] plugin.

4.2.1. 2D instance segmentation pipeline

A (heterogeneous) set of unlabelled microscopy images serves as input to the pipeline for which accurate instance segmentation masks are yielded. **Figure 4.2** shows the pipeline and displays

the synthetic image generation step in detail (**Fig. 4.2B**). In brief, test images are forwarded to a pre-segmentation model (Mask R-CNN-based), nucleus size and shape is estimated on these pre-segmented masks and used in subsequent steps. Input images are clustered based on visual similarity and a separate style model (pix2pix [43]) is trained on each cluster using the unlabelled original images and their corresponding pre-segmented masks – or in case of labelled training data, annotated masks might be used. Parallelly, new mask images are generated based on the nuclear morphology properties (size, shape, space distribution and density) measured on the pre-segmented masks. Then, for each cluster the trained image style transfer models are applied on the generated masks resulting in artificial microscopy images that mimic the given group's style each. A Mask R-CNN network is fine-tuned on the augmented training data and used to segment the input images rescaled according to the size estimation, and finally post-processed with U-Net and mathematical morphology operations (see in subsection 4.2.6).

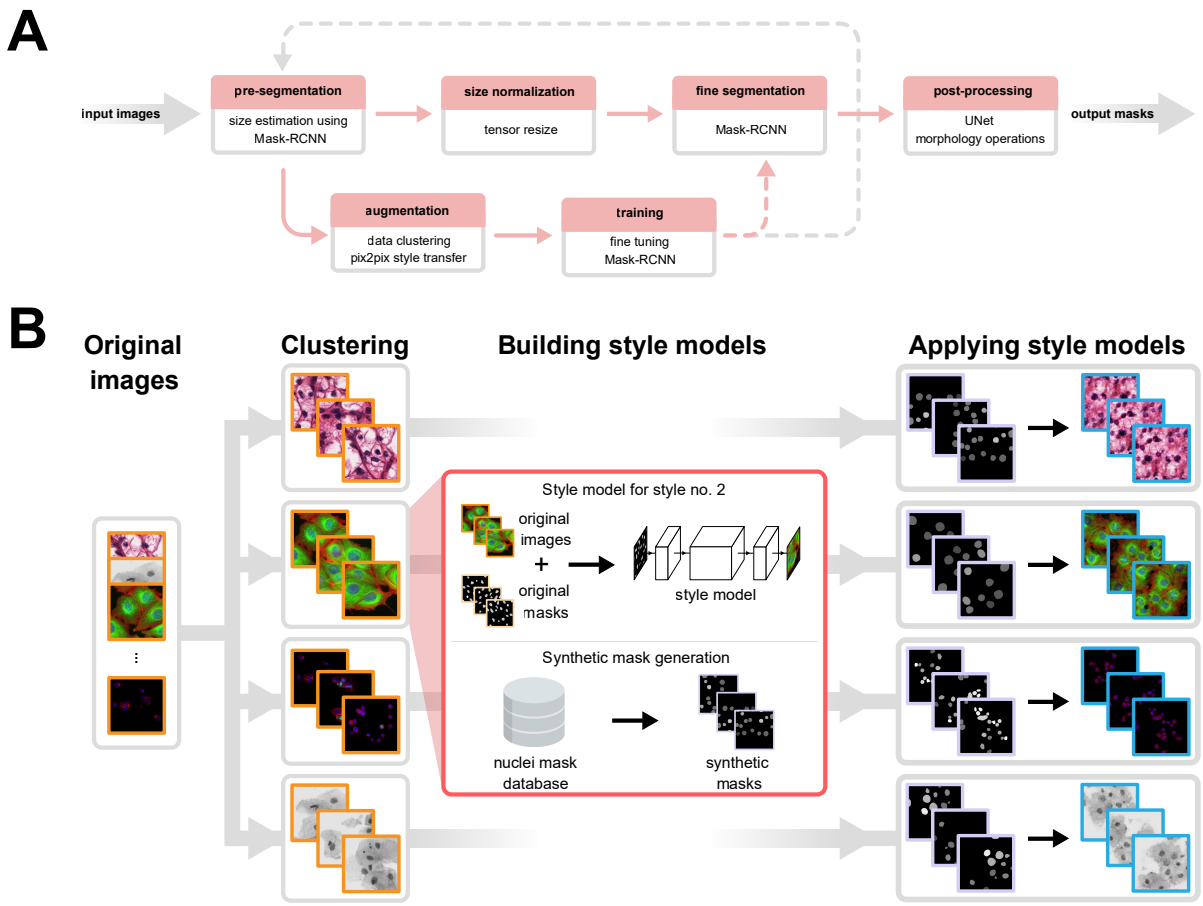


Figure 4.2. Overview of the nucleAIzer pipeline. A) Main components of the pipeline. A pre-trained Mask R-CNN segmentation model estimates nucleus sizes based on which images are rescaled to have uniform object sizes

and predicted by a fine-tuned Mask R-CNN model, then post-processed with U-Net and conventional morphology operations; as the top row shows. The bottom row explains the data augmentation and training steps: besides conventional augmentations (e.g. flips, rotations, crops etc.) artificially created microscopy images are appended to the training set (see on part B and subsection 4.2.4.) and a new Mask R-CNN network is trained. Training may be iteratively repeated as the grey dashed line suggests. B) Style transfer image generation. Firstly, images are clustered to groups based on visual similarity, then a style model is trained for each cluster so that our segmentation model is adapted to novel image domains (according to e.g. new experiments). Simultaneously, new labelled images (masks) are created to faithfully represent the nucleus shape, size and distribution of the given cluster. Finally, the learnt style models are applied on the masks to yield new synthetic training data. Figure is adapted from [16].

4.2.2. Pre-segmentation

As a first step, a Mask R-CNN-based network pre-trained on various nucleus images (referred to as *preseg* model) produces a pre-segmentation mask image used to estimate the approximate object size on each input image for subsequent training and inference as we found that models trained with uniform object size tend to perform better (see **Fig. 4.11**). The *preseg* model is robust to image modalities and object sizes for optimal performance, its purpose is to accurately segment *those* objects it does detect so that size and shape measurements may be precise; it is not expected to find all. Despite its primary goal, the *preseg* model was found to perform relatively well since it finds most objects on distinct types of microscopy images; see performance in subsections 4.3.1-2 and comparison to the fine-tuned segmentation model (dubbed *postComp*) on **Figures 4.11** and **S4**. *preseg* was trained using pre-trained weights on the COCO dataset [76], on a heterogeneous microscopy image set containing training data of the DSB [40] [50], public datasets [52] [68] [69] [83] [48] [84] [70] and experimental images from our own lab and our collaborators [85] [86] [87] (see also in [16]). Multiple cell lines, tissues of origin, imaged in different microscope types and magnifications, labelled fluorescently or brightfield-stained and label-free samples were represented in the training set.

4.2.3. Data augmentation and clustering

The training pipeline begins with data augmentation on two levels: 1) image style transfer learning (see in subsection 4.2.4) and 2) conventional image augmentations. Test images to which no ground truth annotation is available are clustered to separate groups based on pairwise similarity calculated from image-based features (intensity, texture) fetched using CellProfiler

[37] measurement modules. Similarities are learnt in a Weka-implementation of a shallow neural network [88] [89] as a label of 1 is assigned to images from the same experiment or imaging condition and 0 otherwise. This network returns a similarity matrix on test images that is subsequently clustered.

Our selected clustering algorithm was the traditional k-means, to which we pass a similarity matrix A computed from image features and the parameter k computed from the number of images N and the number of elements in a cluster $n_1, \dots, n_m$ through multiple runs of the algorithm with different n_i-s, then minimize to the intra-cluster distance. Thus, k is determined automatically to yield a higher number of clusters than visible image types present in the dataset, in order not to be error prone. Should a style be trained on such a cluster of images that arise from different types (of microscopy, sample origin, staining etc.), generated synthetic microscopy images would fail to resemble either of the original image types and corrupt the finally assembled training dataset.

For each cluster, an image style transfer model is trained, and artificial microscopy image/mask pairs are generated to boost the training set, see details in subsection 4.2.4.

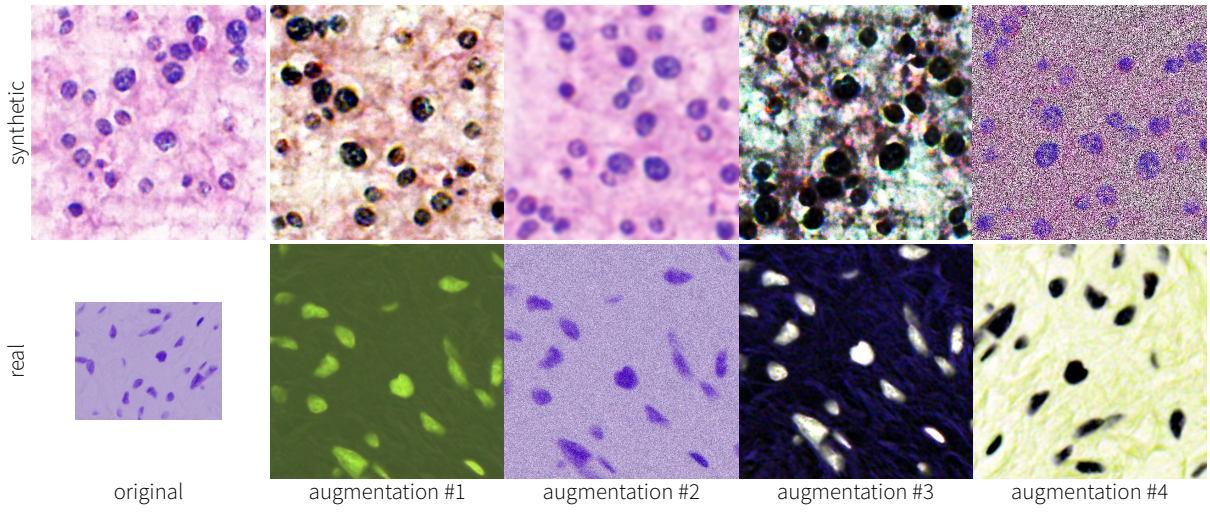


Figure 4.3. Example augmentations. Each row displays an original image and four arbitrarily selected augmentations of it. Top row shows an image style transfer learning-generated (synthetic) image, bottom row a real microscopy image from the DSB dataset [40]. See also Supplementary Figures S2-3.

Additionally, conventional augmentation techniques are also applied on original training data and synthetic images such as intensity stretching, histogram equalization, additive noise and

blur, colour channel swaps, inversion, translations, flips, rotations and crops, applied with random probabilities; see an example on **Figure 4.3**. We scale all training images to have uniform object size; see our evaluation of object size-dependent performance in subs. 4.3.4.

4.2.4. Synthetic microscopy images created with image style transfer

An image style transfer model is trained by clusters to learn the mapping from a set of labelled mask images to original (intensity) microscopy images (see on **Figures 4.2B**, **4.4** and **S2-3**). Image-to-image translation is realized via the pix2pix [43] implementation that consists of two generative adversarial networks (GANs) [44]: a generator whose task is to create new images resembling the training image as closely as possible, and a discriminator that attempts to identify the fake images. These two networks compete during training.

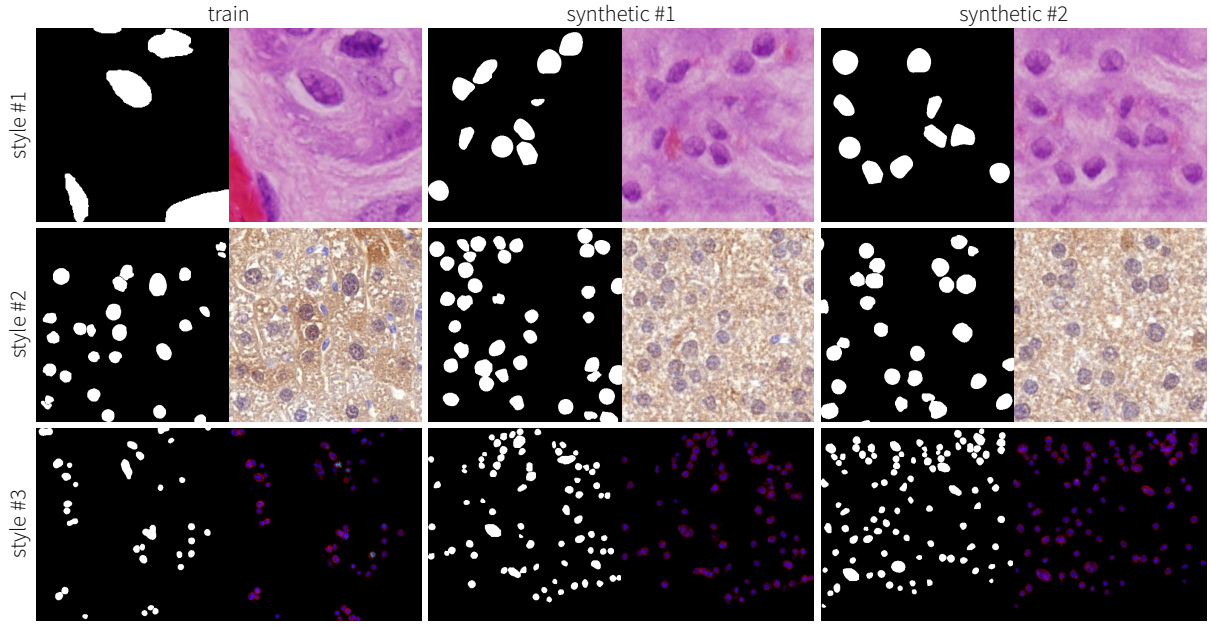


Figure 4.4. Example training image/mask pairs and generated artificial style images. Column 1 shows training mask/image pairs, 2-3 generated synthetic mask/image pairs. Rows correspond to three arbitrarily chosen styles. Style #1 is a histological tissue section imaged in brightfield mode, stained with Haematoxylin and eosin, from the DSB image set [40]; magnification and origin of tissue are unknown. Style #2 is also a histological tissue section imaged in brightfield microscopy, IHC-stained from the Peter Horvath group (BRC, Hungary). Style #3 is a cell line in culture labelled for nucleus and cytoplasm fluorescently (see source in Supplementary Table S2 of [16]). See also Supplementary Figures S2-3.

Figure 4.4 and **Supplementary Figures S2-3** present example train and test image pairs for styles (clusters) from the DSB dataset [40] and our own laboratory [85] [86] [87] (see also in

[16]). Masks are binary variations (label 1 for foreground and 0 for background) of the labelled masks either in our training dataset or the pre-segmented masks per se. We train each image style transfer model for a calculated number of epochs based on the number of training images available: the more images the fewer epochs.

To provide test input for the style models, new mask images are generated as follows. The pre-segmented masks – or annotated masks in the training set – are used to make measurements on such as object size, morphology and spatial distribution on the 2D image plane. According to the measured object features, appropriate (most similar) individual object (nucleus) masks are fetched from our previously collected object mask database, as well as new artificial shapes are generated using simcep [90] (a cell image simulator software) also based on these measurements; the two sources contribute to the simulated objects in equal splits (50-50%). Finally, these objects are placed on the new empty mask image to follow the localization of the given style. The trained image style transfer models are applied on the binary form of these masks for each cluster.

The output of the image style transfer network, synthetic microscopy images, are used to train a segmentation model with their corresponding multi-labelled mask images. We show in section 4.3 how such synthetic training data influences segmentation accuracy on various image sets, while the quality of our generated artificial microscopy images is demonstrated in subs. 4.3.3.

4.2.5. Instance segmentation model training and inference

Mask R-CNN [27] is used as the basis of our instance segmentation network in both the pre-segmentation step (see in subsection 4.2.2) and in the final segmentation using rescaled tensor sizes according to a uniform object size. The network is trained with the augmented training data (including image style transfer output) in three epoch groups with decreasing learning rate and targeting different layers of the architecture. Training images are rescaled such that they have a uniform object size we determined as 40 pixels in diameter (based on the DSB dataset [40], see quantitative evaluation on **Figure 4.11**), then cropped or padded to also have an equal image size of 512x512 pixels for optimal performance.

Once trained, the fine-tuned segmentation model is used to predict instance segmentation on input test microscopy images rescaled as discussed previously.

4.2.6. Post-processing

A post-processing step is appended to the nucleAIzer pipeline for ideal segmentation performance, even though in most applications such high accuracy is not needed e.g. when simply counting cells, thus we consider post-processing as an optional final step.

The input to post-processing consists of Mask R-CNN prediction in two steps: first on 2x then 4x scaled images, and ensembled U-Net-based [26] predictions used to refine contours, as displayed on **Figure 4.5**. Additionally, we use classical morphology operations such as erosion, dilation etc. and according to the optimized post-processing parameter list obtained with a genetic algorithm [91] we finalize the object segmentations; parameters p_1 to p_6 (see below) were optimized for the DSB dataset [40].

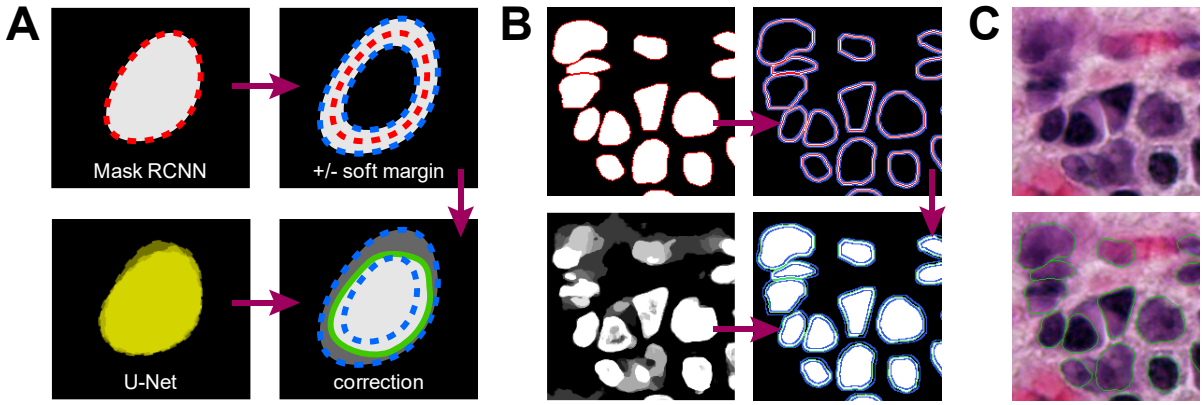


Figure 4.5. Post-processing steps. A) Schematic overview of the steps, B) real example as in A and C. Object segmentation from the fine-tuned Mask R-CNN (red) is extended and shrunk by x pixels to create a soft margin (blue) around the contour within which our trained U-Net models' ensembled prediction (yellow) is to yield the corrected contour (green). C) original image and the post-processed contour outlined in green. Figure is adapted from [16].

Firstly, objects are filtered such that those smaller than p_1 pixels and tiny ones detected inside other objects e.g. larger nucleoli inside the nucleus are excluded, and objects are merged when appropriate e.g. in case of one surrounded by the other by greater than p_2%. The Mask R-CNN prediction is dilated and eroded by p_3 and p_4 pixels respectively to create a soft margin around the contour. A threshold p_5 for U-Net probabilities is determined, and pixels with values above t contribute to correct the Mask R-CNN prediction within the soft margin, while objects with a total U-net probability below p_6 are removed.

4.3. Results

Intersection over union (IoU) was used as the primary evaluation metric in this study, according to the DSB 2018 [40] [50], formulated in subsection 3.3.2 (equation 3.1). Additionally, we also used *precision* (see equation 3.2), recall (equation 4.1) and *F1-score* (equation 4.2) to quantitatively evaluate our results regarding segmentation, while *accuracy* (equation 4.3) was used in our synthetic image quality test discussed in subsection 4.3.3; notation is as in subsection 3.3.2. IoU was calculated on ten thresholds from 0.5 to 0.95 then averaged for each image (as described in subsection 3.3.2) since it yields a fine balance between detection and pixel-wise precision both for small and large objects: small ones consisting of a few pixels are strictly penalized by the metric if the prediction does not properly overlap the ground truth as well as they are easily missed, while larger objects are considered more leniently pixel-wise.

$$recall(t) = \frac{TP(t)}{TP(t) + FN(t) +} \tag{4.1}$$

$$F1\ score(t) = 2 \cdot \frac{precision(t) \cdot recall(t)}{precision(t) + recall(t) +} \tag{4.2}$$

$$accuracy = \frac{\sum correctly\ classified\ instances}{\sum instances} \tag{4.3}$$

We discuss our segmentation accuracy evaluation on the DSB 2018 [40] [50] test in subsection 4.3.1, while on our custom fluorescent and histology image sets in 4.3.2.

4.3.1. Data Science Bowl 2018 competition

We originally developed nucleAIzer to compete in the DSB 2018 nucleus segmentation competition [40] [50] where 3891 teams participated in the first stage and 739 in the second. The competition set out to challenge image analysts and enthusiasts to create a method that is suitable to segment various, novel experiment-related microscopy images – most not included in the training set – representing different cell lines, tissues, stains, types of microscopy, magnification, image quality etc. The test set of the first stage (65 images) correlated with the official training set (670 images, later released in the Broad Bioimage Benchmark Collection, BBBC as the BBBC038 [40] dataset), whose ground truth annotations were released at the end of the first stage and could be used for training in the second stage. The latter test set consisted

of 3019 images mostly acquired of such samples that were labelled for non-nucleus cell compartments to prevent hand-labelled solutions. Even though it contained images labelled for the nucleus, most of these were not represented in the training set. Therefore, successful solutions needed to be prepared for generalization well enough on novel image domains. Our adaptation strategy discussed in subsections 4.2.3-4 above currently yields the best performance on this dataset according to the competition metric (IoU score as in equation 3.1). We finished in the top 1% in the first stage and top 4% in the second stage of the competition, then continued to improve the method (post-competition).

Additionally, as allowed in the DSB, we used external data sources for training from public nucleus datasets as well as our own laboratory and collaborators (see Supplementary Table S2 in [16]). Totally 1102 annotated nucleus image-mask pairs containing 80,692 nucleus masks were used and extended with artificially created pairs (263,701 nuclei on 2680 images).

We compared our approach to three classical methods: 1) an adaptive threshold-based object splitting method available in CellProfiler (CP) [37], 2) a gradient vector flow (GVF)-based solution [92] for object segmentation if the objects are bright regions on a darker background as gradient vectors point towards the brightest spots; this is ideal in case of fluorescent images while tends to fail on histology ones, and 3) ilastik [41] [42]: a pixel classification software with manual user annotation to classes. ilastik predictions are either probability maps or segmentations, we applied thresholding and object processing on the former. We also used unet4nuclei [40]: a U-Net-based DL method developed to efficiently segment nuclei on fluorescent cell line images, and the first two solutions of DSB stage 2 denoted as DSB1 [93] [40] and DSB2 [94] [40]. Both DSB1 and DSB2 are based on U-Nets, the former ensembles predictions of 32 DL networks and a boundary prediction of adjacent objects (training and prediction are time-consuming), while the latter also predicts locations relatively within objects.

To test the effect of synthetic images used in the training dataset we also compared our nucleAIzer pipeline trained with the following configurations regarding training data content. Whether and how image style transfer-generated images were included in training – based on ground truth or pre-segmented masks for *GTstyle* and *AUTOstyle*, respectively, and *NOstyle* was trained without style images used. *preseg* and *postComp* refer to our pre-trained and fine-tuned models, respectively, both optimized for the DSB. *preseg* is scale-independent and

robust, while *postComp* achieves the highest pixel-precision in general, as it was prepared for the DSB test2 set (*postComp* is the *AUTOstyle* model trained for DSB test2). **Table 4.1** presents mean IoU scores on the DSB test sets as well as on our custom image sets (*fluo* and *hist*, see in subsection 4.3.2).

method	**dataset**			
	fluo	hist	DSB test1	DSB test2
preseg	0.654	0.351	0.468	0.483
CP	0.599	0.255	0.333	0.528
unet4nuclei	0.653	0.052	0.138	-
GVF	0.519	0.039	0.258	0.168
NOstyle	0.676	0.381	0.542	0.621
AUTOstyle	0.626	0.421	**0.585**	**0.633**
GTstyle	0.695	0.430	0.551	-
postComp	**0.705**	**0.484**	0.561	**0.633**
ilastik	0.577	0.286	0.324	-
DSB1	-	-	-	0.631
DSB2	-	-	-	0.614

Table 4.1. IoU scores. Test sets denoted as DSB test1 and DSB test2 correspond to the stage 1 and stage 2 test sets of the DSB 2018 competition, both truly heterogeneous, while *fluo* is our custom fluorescent cell line image set [49] and *hist* is a heterogeneous histology image set (see in [16]) from our laboratory and collaborators. Highest score is highlighted in bold for each test set. Methods are detailed above. Image sources are detailed in Supplementary Table S2 of [16]. Table is adapted from [16].

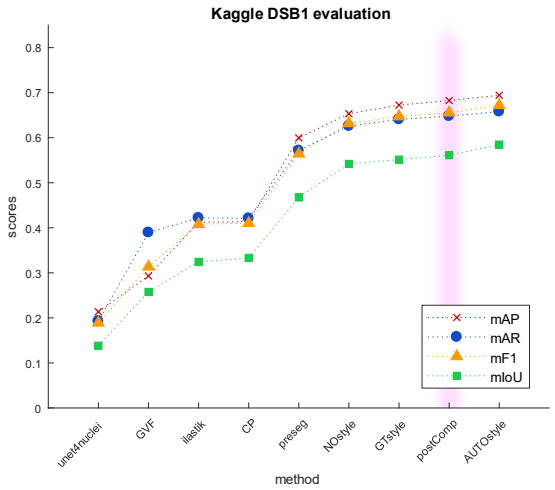


Figure 4.6. Quantitative evaluation on DSB test1 set compared to other methods and style variations. Metrics are defined in equations 3.1-2 and 4.1-2. See also Figure 4.8 and 5.2. Figure is adapted from [16].

Furthermore, we performed evaluation using three additional metrics: precision, recall and F1-score as shown on **Figures 4.6** and **4.8**. These results confirmed our method's superior accuracy as the applied metrics mostly correlated for all test sets.

Finally, we also performed a qualitative comparison of these methods to reveal weakness of each and show robustness of nucleAIzer to heterogeneous image domains; see segmentation examples on **Figure 4.7B**, while **Figure 4.7A** shows mean IoU scores on the test sets.

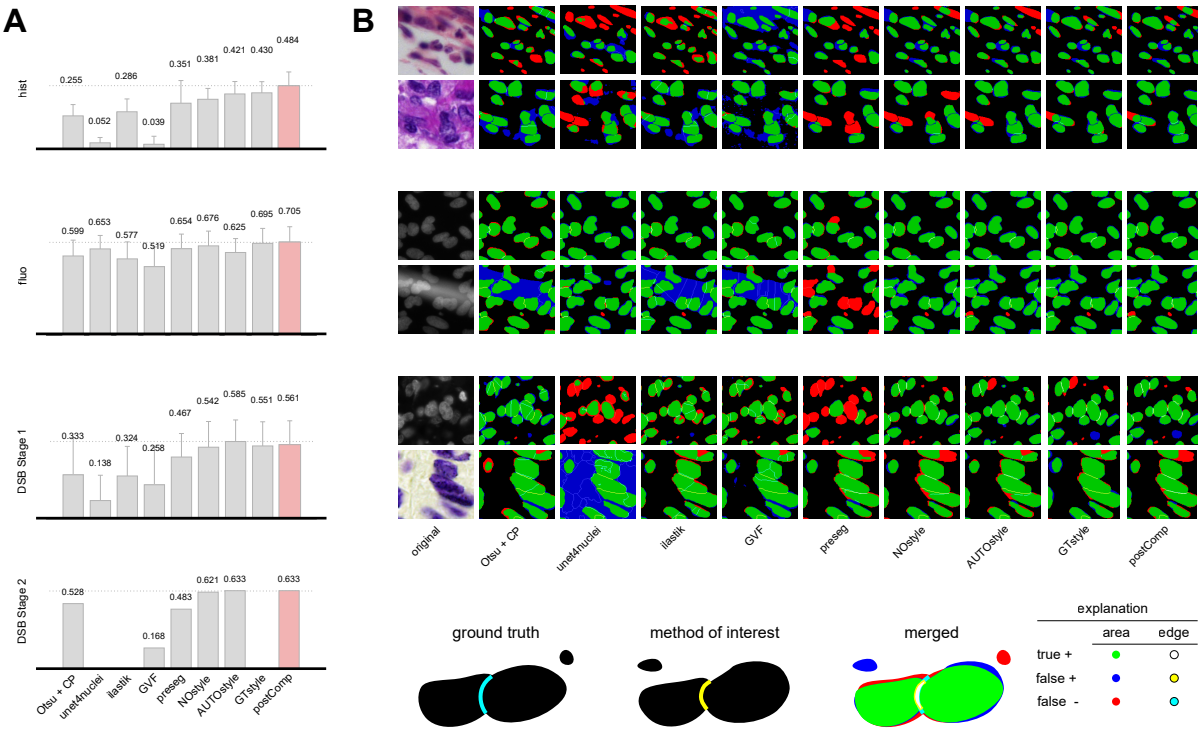


Figure 4.7. Quantitative and qualitative comparison of segmentation results. A) Mean IoU scores with error bars (standard deviation) on the four test image sets (see *fluo* and *hist* in subsection 4.3.2). The highest score is marked with a dashed line and pink colour. B) Example colour-coded segmentation results compared to ground truth annotations on difficult image regions cropped. Two examples are shown per test set, and rows correspond to A. See colour coding explained in the legend in the bottom row. We remark that ground truth annotations were not available for DSB stage 2. See also Table 4.1. Figure is adapted from [16].

Our evaluation confirmed that nucleAIzer achieves higher accuracy than the compared methods according to all metrics tested. We could also prove that segmentation performance increased when including image style transfer-generated artificial data in training: see columns *AUTOstyle* and *GTstyle* compared to *NOstyle*. When ground truth annotations are available

(*GTstyle*), this increase is even more prominent; unfortunately, we can rarely rely on such annotations in real experiments.

4.3.2. Custom fluorescent and histology image set performance

Our custom test image sets were denoted *fluo* and *hist* as follows. *fluo* is adapted from BBBC039 [49], 200 fluorescent images of U2OS cells in a chemical screen. *hist* consisted of 50 selected histopathology images from various sources mostly from our laboratory and the internet (see Supplemental Data in [16] and Supplementary Table S2 in [16]) to which no similar image domains were present in the training set so that we could efficiently test domain adaptation of nucleAIzer to these novel experiments. When testing on these image sets, they were completely excluded from training their respective models via automatic clustering to two disjoint parts: one was input to image style transfer image generation and the other held out solely for evaluation. These splits approximately halved the test sets, 100/200 *fluo*, 21/50 *hist* and 28/65 DSB test 1 test images were reserved for evaluation.

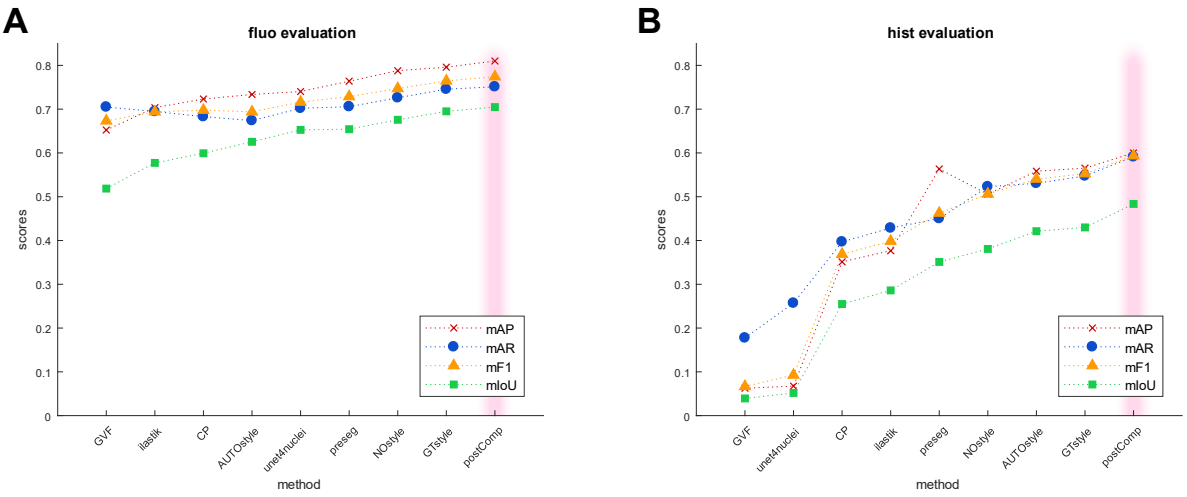


Figure 4.8. Quantitative evaluation on test sets *fluo* (A) and *hist* (B) compared to other methods and style variations. Metrics are defined in equations 3.1-2 and 4.1-2. See also Figure 4.6 and 5.2. Figure is adapted from [16].

Comparative analysis of segmentation performance was conducted as discussed in the previous subsection for the DSB test sets, see **Table 4.1** and **Figure 4.7** for mean IoU scores, the latter also for qualitative results, while evaluation using additional metrics is presented on **Figure 4.8** (see also **Figure 4.6**). Regarding the image style transfer performance increase reported on DSB

test1 (see in subsection 4.3.1), we could support our findings with independent tests on both image sets *fluo* and *hist* (as shown on **Figures 4.7-8**).

4.3.3. Fake or real?

We investigated how convincing our image style transfer-generated artificial microscopy images appeared to field experts with experience in cell biology and pathology by showing them mosaics of real and fake (synthetic) image tiles on a grid and asked them to spot the fakes. Each test grid contained 50-50% real and fake images, even though experts were merely told there were examples of both kind present on each grid (their ratio remained unknown to them), and their recognition accuracy was measured in a binary manner: 1 if they recognised the tile correctly and 0 otherwise. The tiles were cropped to a uniform image size and placed randomly on an 8-by-8 grid in two tests (two times 64 images, see an example crop on **Figure 4.9A** and full images on **Supplementary Figures S5-6**).

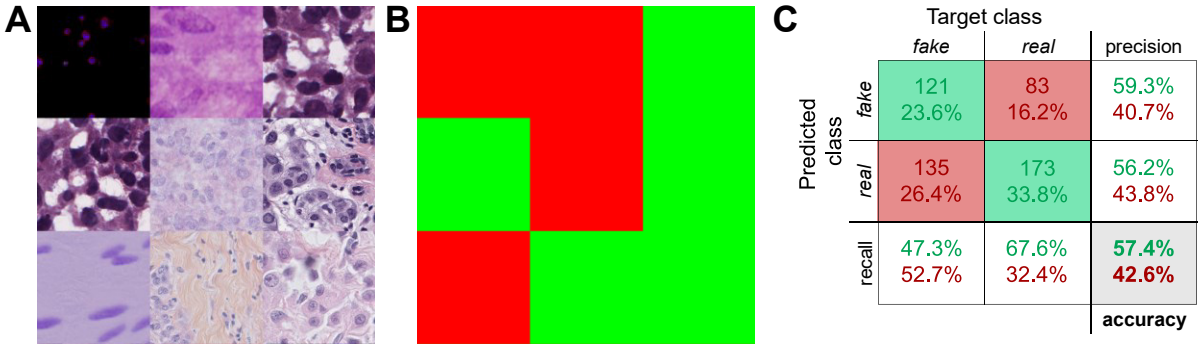


Figure 4.9. Synthetic image recognition test. A) sample tiled image crops, B) colour-coded representation: green corresponds to real and red to fake tiles, C) confusion matrix of experts' accuracy as follows. We counted predicted labels for both classes (real and fake) and compared them to true labels (target class) as light green (correctly classified) and red (incorrectly classified) shaded cells display, whereas precision and recall values calculated are shown in the border cells (green percentages correspond to correct classification for the given row or column, red to false). The bottom right cell presents mean accuracy. Precision, recall and accuracy values are as defined in equations 4.1-3. Figure is adapted from [16].

Our four experts reached an average of 57.4% accuracy (as defined in equation 4.3) on the two test grids, scoring between 42% and 73%; see summary on **Figure 4.9C** while individual expert results on **Figure 4.10**. Experts generally performed better on the first test (see on **Supplementary Figure S5**), especially experts 3 and 4. Our results confirm the high quality of our synthetic microscopy images rendering their appearance fit to their individual image

domains of real experimental images, and suggest their optimal applicability in cell segmentation.

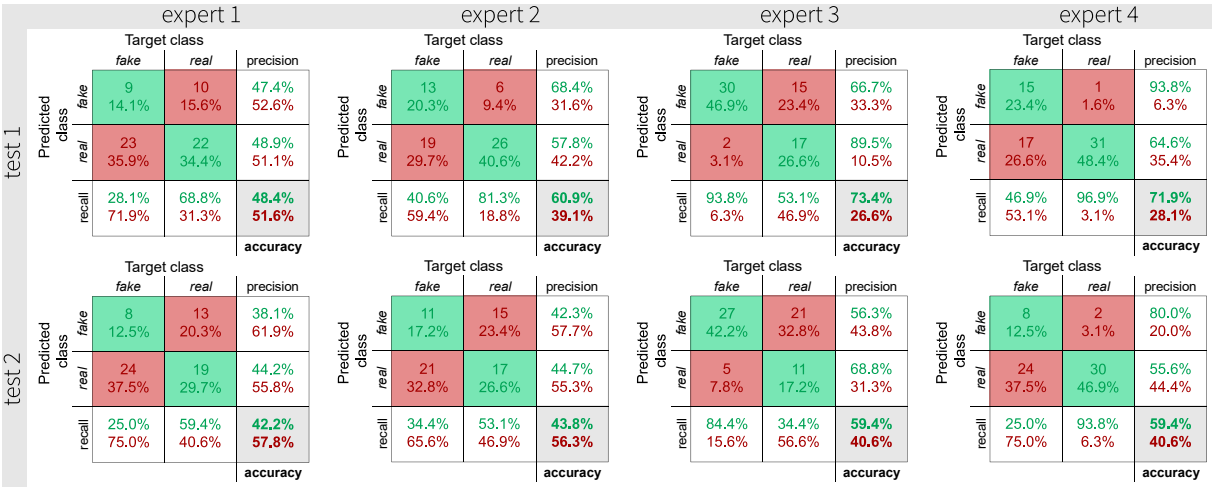


Figure 4.10. Expert accuracies in the synthetic image recognition test. Confusion matrices are as on Figure 4.9C. See also Figure 4.9 for summed results and Supplementary Figures S5-6 for the test images.

4.3.4. Object size and segmentation error analysis

Segmentation models were trained on such images that were rescaled to have a median 40-pixel diameter object size uniformly since it led to superior precision. We tested the robustness of our pre-trained models *preseg* and *postComp* in a wide eight-fold object size range (10-160 pixels) on a real test image set DSB test2 (**Figure 4.11**) and additionally on an artificially created image with simulated circles in the size range 1-125 pixels (**Supplementary Figure S4**). The first test displayed on **Figure 4.11A** confirmed the optimal object size to be around the expected 40 pixels; *postComp* performed the best in 1-2x of this size and remained relatively accurate up to 4x scaling, while scores dropped far more rapidly when downscaling images. Even though a similar trend was observed for *preseg*, it performed much better than *postComp* on smaller sizes.

We note that on real experimental images object sizes do not vary as drastically as we simulated in the previous tests, typically extending to ½-2x of the median as outliers. On **Figure 4.11B** we show that DSB test2 contained such example images that showed an extremely uneven size distribution around their median, suggesting that image rescaling depending on the median object size might not be optimal for all cases. We also note that such images are rare in real experiments. We simulated an extreme example of the aforementioned kind by placing white

circles with diameters from 1 to 125 pixels on a blank image and predicted it without rescaling using *preseg* and *postComp*; see **Supplementary Figure S4**. We found that both models detected the objects in a broad size range: *preseg* found the circles with diameter 7-125 (17.5%-312.5% of the expected 40) and *postComp* 7-100 (17.5%-250% of 40), respectively, with the former retrieving more precise contours on the larger circles than *postComp*.

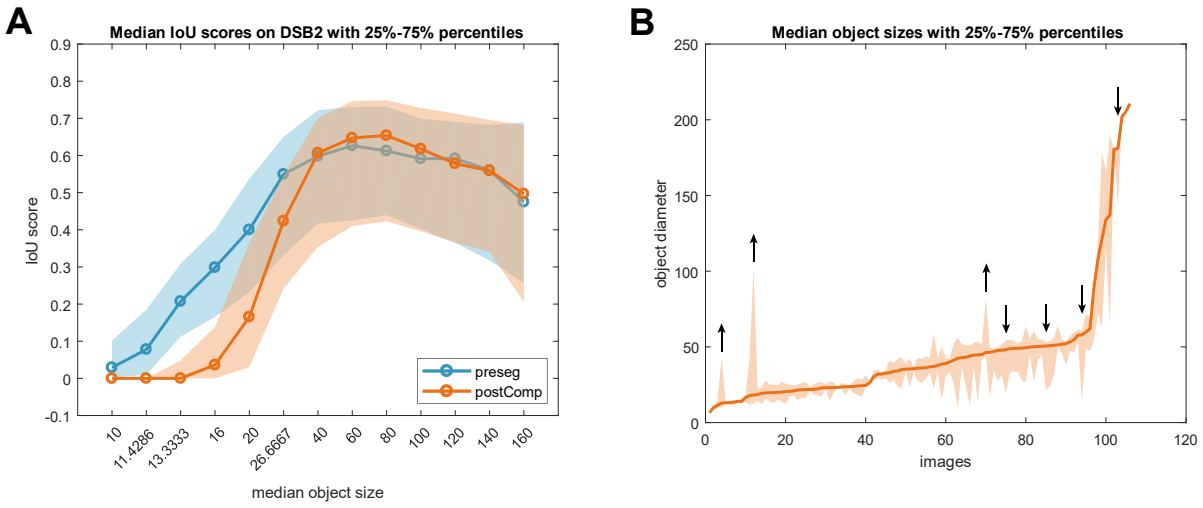


Figure 4.11. Model robustness to object sizes present on test images. A) We tested detected objects on the DSB test2 set in the size range ¼-4x compared to the uniform size of 40 pixels diameter we used for training and expected for prediction – image rescaling was applied accordingly. Solid lines with circle data markers show median IoU scores (over the test set), while shaded areas extend to 25-75% percentiles. B) Nucleus diameters measured on original images. Marking is as on A, while black arrows correspond to such images that have an uneven object size distribution, arrows point up when the majority of objects were larger than the median and down otherwise. See also Supplementary Figure S4. Figure is adapted from [16].

Secondly, we analysed the segmentation errors produced by our method and those we compared to, see **Figure 4.12**. Instance segmentation predictions were compared on an object-based manner and the following error types were counted on all images in a test set: 1) false positives (FP: a predicted non-nucleus object with no overlapping GT object), 2) false negatives (FN: a missed nucleus i.e. an object on GT with no corresponding prediction), 3) merges (multiple GT objects detected as one) and 4) splits (multiple adjacent objects detected over one real nucleus). For merges and splits object correspondence was determined as a minimum of 30% overlap in case of two adjacent objects compared to the best match while 15% for more.

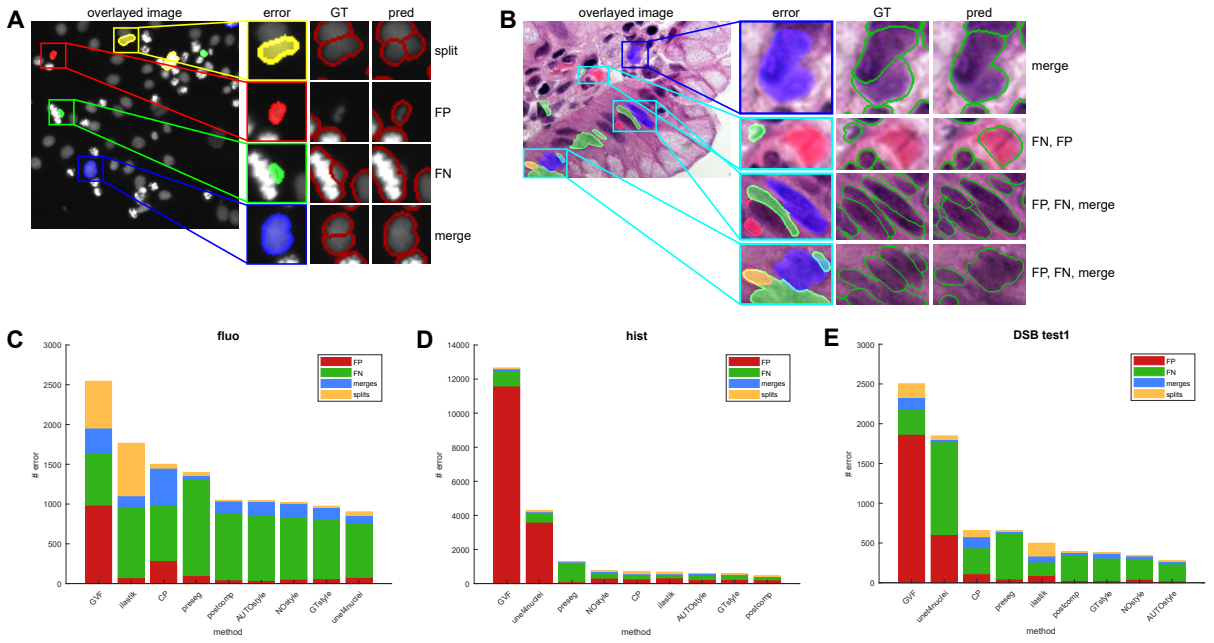


Figure 4.12. Comparative segmentation error analysis. A-B) Real example images with the typical error types outlined, colours denote the errors: yellow for split, red for FP, green for FN and blue for merge. Predictions (column pred) were produced with postComp, GT outlines are also shown for visual comparison in insets. B) displays a difficult histology image where our method as well as all compared methods tend to fail. C-E) Error type distribution on test sets fluo (C), hist (D) and DSB test1 (E), respectively. Methods are ordered by the sum of all errors. Figure is adapted from [16].

This evaluation was inspired by [49] where the authors compared DL networks (U-Net [26] and DeepCell [95]) to the baseline software solution widely used in bioimage analysis, CellProfiler [37], and concluded that DL methods help reduce the frequency of such segmentation errors while also retrieving more faithful contours (smoother and better fit to the objects). Furthermore, they found that small objects were the most likely to be missed by each method. Based on our test sets we can also confirm that DL methods (variations of nucleAIzer on all sets and unet4nuclei on *fluo*) indeed produce significantly fewer errors in total than the compared classical algorithms, as expected. The reported contour smoothness increase as well as more frequent miss of small objects were also observed.

The segmentation errors presented on **Figure 4.12A-B** are common in bioimage analysis, especially in classical methods such as CP (CellProfiler), GVF and ilastik who typically detect false objects more often and tend to split larger non-circular shapes or merge adjacent objects and background regions with similar texture. We note that unet4nuclei was fine-tuned and

published on test set *fluo* hence it performed the best on this set. However, it could not have been trained accurately enough on heterogeneous sets *hist* and DSB test1 since its U-Net basis retains uncertainty in complex image regions such as on histopathology images. Conclusively, nucleAIzer was the most robust on all sets in general and produced the fewest total errors, even though it had difficulties with complex cases such as tissue background regions resembling object texture or tiny nuclei as well as elongated shapes. Generally, false negative was the most common type of error for all methods.

4.3.5. Perspectives

nucleAIzer was appended with an online tool available at www.nucleaizer.org to enable biologists and pathologists worldwide to test our pre-trained models on their own experimental images for which no computer science knowledge is needed. Users can simply drag-and-drop their images, select one of our models and obtain segmentations returned in about a minute[8]; see **Supplementary Figure S7**.

Another method arose from the DSB 2018 competition [96] using test-time augmentation (TTA) to boost prediction performance of Mask R-CNN and U-Net with repeated predictions on augmented versions of test images (see **Supplementary Figure S8**), and achieved an even higher score on the DSB test2 set.

We benchmarked nucleAIzer against another DL network, CellPose in the Deep Visual Proteomics (DVP) project [18] (see details in section 5.2) and achieved superior performance on both nucleus and cytoplasm segmentation on various histology tissue samples. nucleAIzer was also successfully applied in a genome-wide screen project (results unpublished, see in section 5.1) and a SARS-COV-2 project [17] (see in section 5.3). Additionally, our collaborators both domestically and internationally use it to achieve state-of-the-art cellular compartment segmentation performance in ongoing projects (see chapter 5).

[8] Speed strongly depends on the number of requests in the queue, the uploaded number of images (limited to 10 per session), their image size and the number of objects on each.

5. Applications

High-content screening is a cell biology method to identify small molecules, peptides or RNAi highly parallelly, in a large sample, to change the phenotype of the cells, using automatic methods such as robot pipetting, microscopy and analysis software [97] [98]. A typical analysis pipeline is depicted on **Supplementary Figure S9**. After preparation of the sample to be analysed (treatment and labelling), images are acquired by a high-throughput microscope, then pre-processed (for quality control) and segmented to find cellular compartments e.g. nucleus and cytoplasm, descriptive features are extracted (intensity, texture, morphology) and finally, an automatic software classifies the cells (assigns phenotypes to them) and creates reports for downstream analysis. Remarkably, the majority of these processing steps (after image acquisition) may be substituted with a single appropriate DL network that can predict the phenotype of cells (and optionally also segment them, such as Mask R-CNN [27]). A pipeline is considered classical in this thesis if it does not include DL models for processing.

We successfully applied our methods in real-life projects with our collaborators internationally, and herein present a selection of them including but not limited to the following three detailed in the following sections.

5.1. Genome-wide screen project

In collaboration with the Kutay group at ETHZ (Eidgenössische Technische Hochschule, Zürich, Switzerland) we conducted a genome-wide 60S screen using siRNAs (small interfering RNAs) to study the genes affecting ribosome biogenesis pathways. 60S is the large subunit of the ribosome (the small subunit is referred as 40S), both their biogenesis begins in the nucleolus of the cell (small compartment in the nucleus) and is completed in the cytoplasm. Different siRNA knockdowns (or knockouts) were used to demonstrate the cellular localization of ribosomal proteins e.g. increased nucleolar- or decreased cytoplasmic signal, suggesting their potential role in ribosome biogenesis. Nuclei were labelled with DAPI (blue), RPL29 (Ribosomal Protein L29) was tagged with GFP (green, reporter gene).

Initially, when we started its preceding project about 40S several years ago, classical solutions were used to segment the cells (based on nucleus objects as seeds propagated to the boundaries of the cytoplasm [37]) and classical machine learning (ACC: Advanced Cell Classifier [99], an

automatic classification software tool) to find the distinctive phenotypes based on the GFP signal in nucleoli (inside the nucleus) and in the cytoplasm [100] (see **Fig. S9**).

Recently, we applied nucleAIzer [16] to segment the nuclei robustly and more efficiently than previously, and additionally trained a separate model for the nucleoli based on the GFP channel (see **Figure 5.1**). We aimed to increase phenotyping accuracy with these DL-retrieved contours. Results of this project are not yet published.

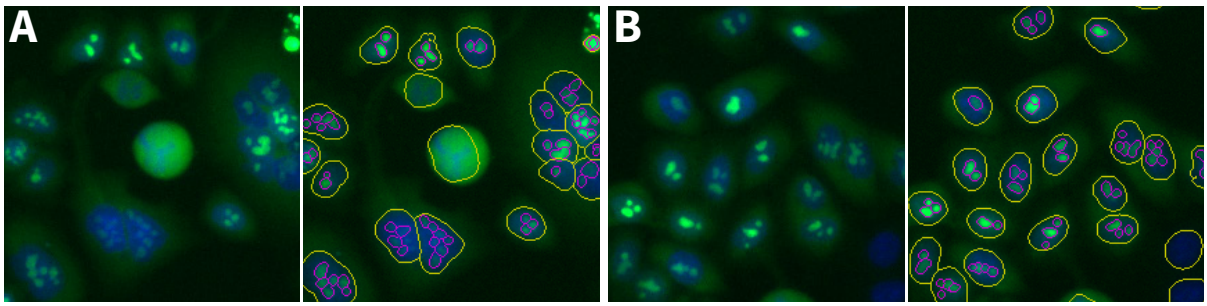


Figure 5.1. Example segmentation in the genome-wide screen project. Left: original image, right: outlines. Nuclei (yellow) were segmented on the DAPI- and nucleoli (magenta) on the GFP channel. A-B) Example regions. A) shows mitotic cells (strong green signal, round cells) in the middle. B) displays two cells with no GFP signal in the bottom right corner. Original image was courtesy of Kerstin Elisabeth Dörner (ETHZ).

5.2. Deep Visual Proteomics project

The Deep Visual Proteomics (DVP) project [18] is a collaboration of our laboratory with the group of Matthias Mann at the University of Copenhagen (Denmark) and Max Planck Institute (Germany), and Emma Lundberg at Karolinska Institute (Stockholm, Sweden), aiming to achieve single-cell resolution with automatic high-throughput analysis of plates (cell lines) and whole slides (tissue sections) using DL methods to find cells corresponding to phenotypes of interest, cut them with laser microdissection, collect and finally forward them to proteomic analysis such as mass spectrometry (MS) [101]. Currently, phases of cell cycle are used to prove the efficiency of the DVP, as the identified proteomic profile of the collected single cells from three distinct phases based on microscopy image analysis (using DL segmentation and classical machine learning-based classification) indeed show the proteins corresponding to the identified phases reported in scientific literature [102]. Furthermore, different types of histopathology samples including melanoma, ovarian cancer and salivary gland tissue slides are profiled with DVP.

We used nucleAIzer [16] to segment the nucleus and cytoplasm of single cells in each sample and benchmarked its performance to a baseline classical algorithm (CellProfiler, CP [37]), and two DL-based methods: 1) unet4nuclei [40] (see in subsection 4.3.1) and 2) Cellpose [38] (see in section 4.1). nucleAIzer was found to yield more accurate segmentations in all tested cases than the others promoting its applicability in DVP; see **Figures 5.2-3**. Multiple metrics (including mean IoU as used in chapter 4, mAP, IoU and F1-score at 0.7 IoU threshold; see definitions in equations 3.1-2 and 4.1-2) were used for benchmarking to emphasize the superior segmentation accuracy of nucleAIzer as the applied nucleus and cytoplasm segmentation method in the project.

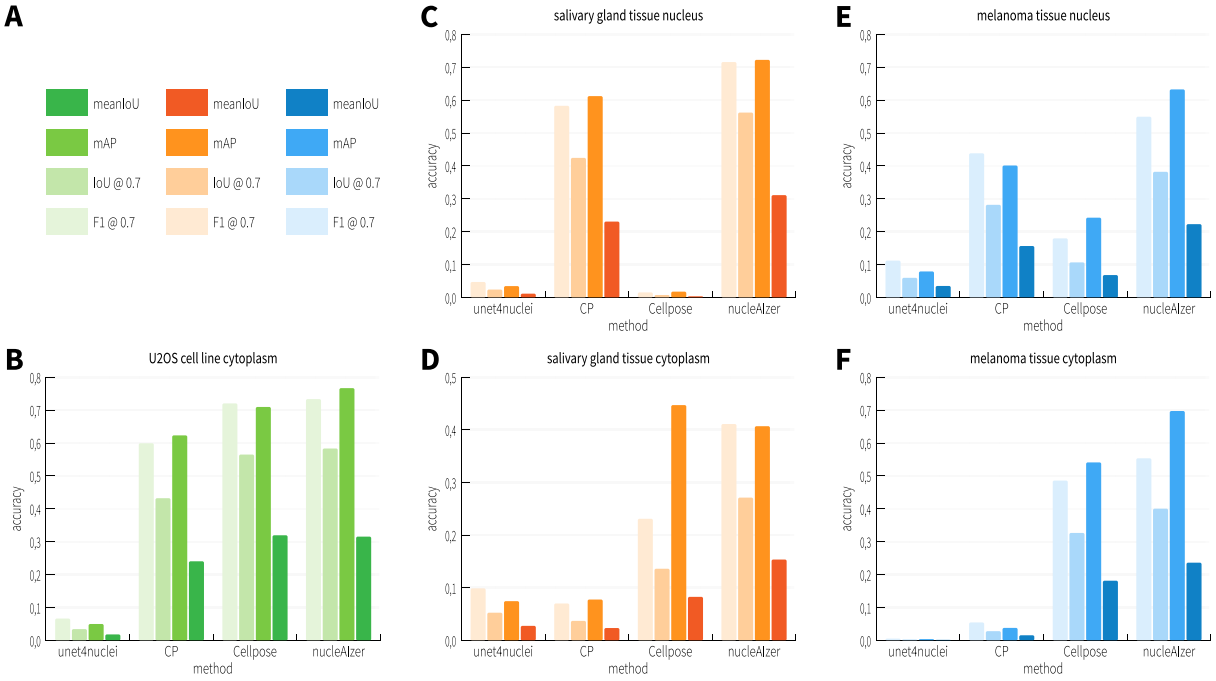


Figure 5.2. Comparative segmentation accuracy evaluation of nucleAIzer in DVP. Fluorescently labelled U2OS cell culture, as well as IHC-marked melanoma and salivary gland tissue images were segmented with nucleAIzer against CP (CellProfiler, classical algorithm), unet4nuclei and Cellpose (the last two are DL-based methods). Image sets were evaluated according to the following metrics: mean IoU, mAP, IoU at 0.7 threshold and F1-score also at 0.7 IoU threshold (see equations 3.1-2 and 4.1-2). nucleAIzer generally performs the best (in some cases by a large margin). A) Legend corresponding to the following panels, B) U2OS cytoplasm, C) salivary gland nucleus, D) salivary gland cytoplasm, E) melanoma nucleus, F) melanoma cytoplasm evaluation. Colours show the grouping of samples. See also Figures 4.6 and 4.8.

Pre-trained models released by Cellpose [38] were used to predict the images, nucleus and cytoplasm models as appropriate. unet4nuclei [40] was trained on the same heterogeneous

image set as described in chapter 4 (for *postComp* nucleAIzer model). Separate CellProfiler pipelines were created and fine-tuned manually by samples. Admirably, CellProfiler performed unexpectedly well on both tissue types when segmenting nuclei but had difficulties with cytoplasms. On the contrary, Cellpose returned cytoplasm segmentations with higher accuracy than nuclei for both tissues, respectively. nucleAIzer remained similarly accurate (relatively to the other methods) for both nucleus and cytoplasm on all samples.

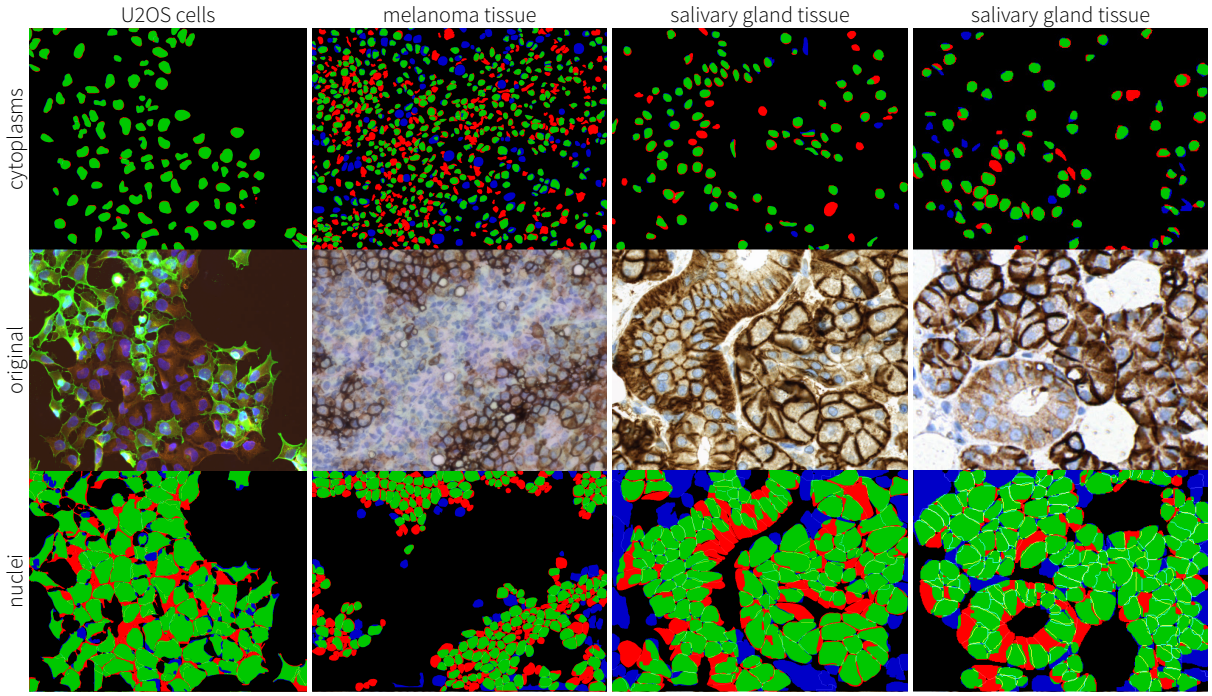


Figure 5.3. Example segmentation results of nucleAIzer. Fluorescent U2OS cell culture images, and melanoma as well as salivary gland tissue images were segmented with nucleAIzer and the resulting segmentations are displayed for the nucleus and cytoplasm, respectively, compared to ground truth annotations drawn by experts using AnnotatorJ [15]. Colour-coding is as on Figure 4.7. Original images are courtesy of Andreas Mund (University of Copenhagen, Denmark and Max Planck Institute, Germany). See also Figure 4.7.

5.3. Neuropilin-1 project

The year 2020 was unfortunately bound to trigger various scientific communities worldwide to rapidly start experimenting and developing solutions to the global pandemic of SARS-COV-2 (Severe acute respiratory syndrome coronavirus 2) or simply coronavirus responsible for COVID-19 (coronavirus disease 2019) [103]. Our collaboration with Yohei Yamauchi's laboratory at the University of Bristol (UK) was previously based on successful analysis of viral infection of influenza [104]. In the current large-scale project, we could identify a novel target

of SARS-COV-2 infection as the neuropilin-1 (NRP1) receptor on the cell surface (binding to the viral spike protein S1) [17] as an alternative to the previously published ACE2 (angiotensin-converting enzyme 2) receptor [105], suggesting that therapy may potentially target the blocking of NRP1 receptor binding (e.g. via an inhibitor) to reduce viral infection.

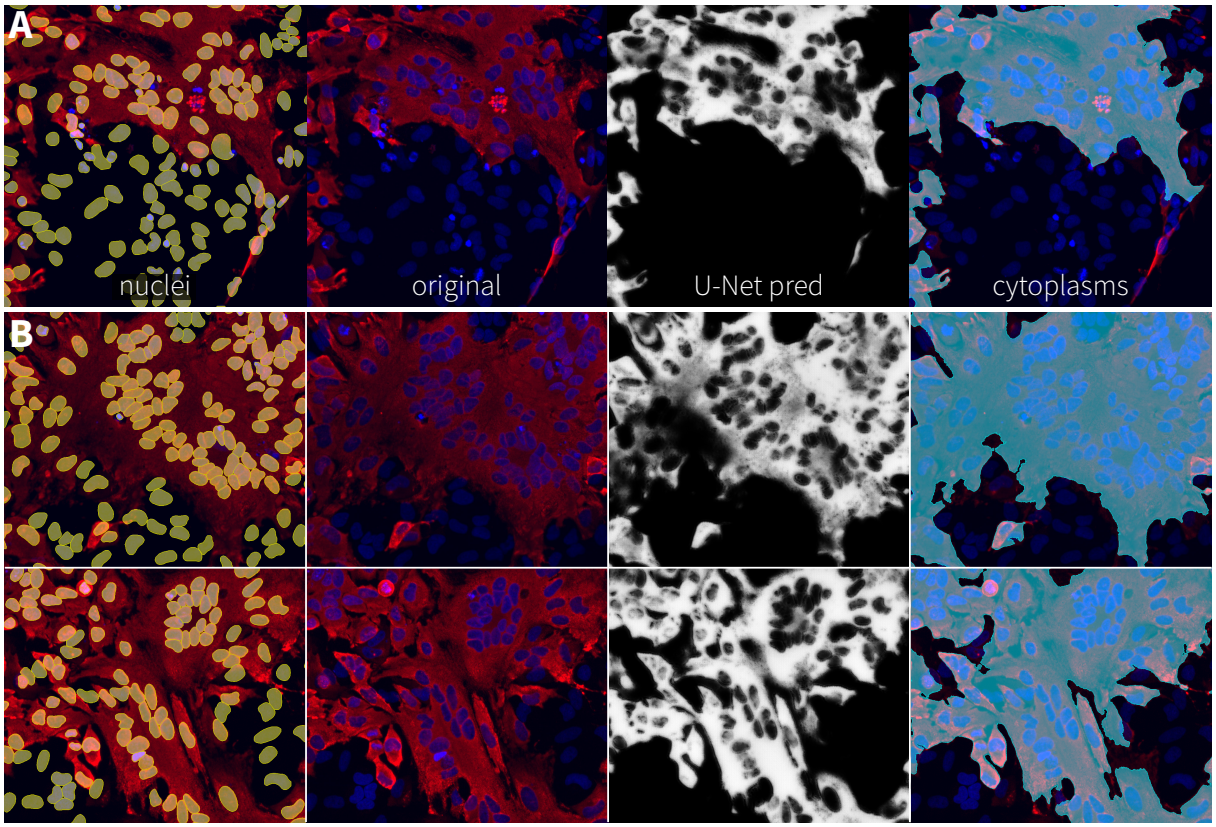


Figure 5.4. Segmentation results in the neuropilin-1 project. A) Zoomed-in crop showing processing steps. Nucleus segmentation (yellow overlay) was performed with nucleAIzer, cytoplasm segmentation (cyan overlay) with U-Net and a classical threshold-based algorithm. B) Further examples, notation (columns) is as in A.

The study was conducted on HeLa[9] and Vero E6[10] cells among others, both treated with the virus particles without or after appropriate knockout of the target protein NRP1. ACE2 expression was used to demonstrate the viral uptake in cell lines i.e. to show the cells can indeed be infected with the virus; infection was reduced in NRP1 knockouts. Samples were fluorescently labelled with mCherry (red) for the virus particle SARS-COV-2-N (nucleocapsid protein) specifically and imaged with a high-content confocal microscope. Images were first pre-processed – as mentioned at the beginning of chapter 5 and **Supplementary Figure S9B** –

[9] derived from the primary tumour of patient Henrietta Lacks in 1951. Cervical cancer epithelial cell line.
[10] African monkey kidney epithelial cell line.

and predicted with nucleAIzer to retrieve nucleus segmentations based on DAPI or Hoechst (blue) signal. Cytoplasm was labelled with GFP-NRP1 (green), thus only the wild-type (wt) was activated and expressed the signal. Both mCherry and GFP fluorescent signals were considerably weak even in case of proper emission. To be able to reliably quantify all cells in the sample, we prepended an additional processing step prior to cytoplasm segmentation: a U-Net trained on binary fluorescent cytoplasm masks was used to predict pixel probabilities based on which cytoplasms were found with a threshold-based algorithm. See **Figure 5.4** for demonstration.

Accurately segmented nucleus and cytoplasm masks were used for reliable feature extraction and classical supervised machine learning training to learn the phenotypes such as negative, single infected and multi-nucleated infected cells. The latter were found in large cell fusion areas regarded as *syncytia*, their area and number of included nuclei were important descriptors: NRP1 knockouts showed considerably reduced *syncytia* phenotype (both in frequency and area) than their wild-type variants.

5.4. Perspectives

Furthermore, nucleAIzer and AnnotatorJ are frequently applied by our collaborators worldwide, primarily in high-content screening projects and pathology laboratories. We briefly discuss a selection of such ongoing projects below.

We applied a segmentation strategy similar to as described in section 5.3 in our collaboration with the Martinek group (University of Szeged, Hungary) to quantify increased antibody cargo signal emission after appropriate treatment of cell lines. This project was a follow-up on a previous study [106] where cytoplasmic lipid droplet signal was quantified. HeLa cells were labelled with Hoechst (blue) for the nuclei and FITC (green) for the cell membrane, while Alexa647 (red) marked the antibody cargo bound to the target peptide carrier. The extension of individual cells i.e. cytoplasm segmentation was performed on a U-Net-enhanced version of the original image to compensate generally weak signal emission from the membrane further from the boundaries of the cell; see **Supplementary Figure S10**. Nuclei were precisely segmented with nucleAIzer [16] based on the Hoechst channel. Intensity measurements were

conducted in CellProfiler [37] to demonstrate cargo infiltration to the cytoplasm driven by the peptide carrier. Results of this study are yet to be published.

Together with dermatologist István Németh (University of Szeged, Hungary) we analysed patient-originated tissue samples stained with various melanoma markers and quantified the population of cells showing the phenotype melanoma among stromal cells. 4HNA, ACSL and MITF markers were used to label the cells, nucleAIzer [16] was used to segment single nuclei, and phenotypes were trained in ACC [99] using extracted intensity, texture and morphology features (CellProfiler [37], see **Supplementary Figure S9D-E**). Results are unpublished.

We are working on a large-scale project in collaboration with Hella Bolck (University Hospital Zürich, Switzerland) to classify ccRCC (clear cell renal cell carcinoma) cells to low-grade, high-grade and rhabdoid categories in H&E-stained whole-slide patient kidney tissue sections. As in our recent projects, nucleAIzer is used to segment nuclei and nucleoli, while phenotyping is based on feature extraction and classical machine learning (see also **Supplementary Figure S9D-E**). Training data annotations are partially created in AnnotatorJ [15]. Results of this study are not published yet.

6. Discussion

In this thesis, cell biology and pathology derived microscopy image analysis was addressed at a single-cell level to provide detailed insight to the mechanism of action, viability or disease progression of individual cells in a sample, that may potentially contribute to deeper understanding of specific treatment such as in drug screens or clinical trials, or diagnosis in personalised medicine. Our approaches are based on instance segmentation, that is to find the image region corresponding to an individual object (e.g. cell or cellular compartment such as nucleus or cytoplasm) which is to serve as basis of downstream analysis: image-based measurements can support (automatic software-guided) phenotyping decisions, cells of interest may be forwarded to molecular analysis including transcriptome (sequencing), proteome (e.g. mass spectrometry), lipidome etc. profiling.

Phenotypes can be retrieved with classical machine learning methods (such as SVM, kNN, clustering etc.) via expert-provided training examples (see e.g. ACC [99] – as applied in [100]; see also chapter 5). However, these strongly depend on the accuracy and descriptive power of coupled features (extracted from cells based on images) and reliability of the training set – depending on the expert to classify training instances correctly. The former constraint can be met via appropriate segmentation algorithms.

To obtain cell segmentations, either classical methods such as Otsu thresholding, watershed, object splitting, region growing [35] [36] etc. may be applied on microscopy images available in software broadly used in the bioimage analysis community including ImageJ/Fiji [61] [62] [63], Knime [107], Icy [108], CellProfiler [37] etc., or machine learning- (e.g. ilastik [41] [42]) based approaches offer convenient solutions. Recently, several deep learning (DL) methods have also been published for image- and more specifically, cell segmentation problems. Most of the mentioned software tools in the first category (classical) have progressed to keep up with the state-of-the-art, hence modern DL methods are being made available in latest core releases or as custom user-provided plugins (e.g. U-Net segmentation plugin in Fiji [62] [80], or our nucleAIzer plugin in CellProfiler [16] [37] – see later). The last group of methods (DL-based) often employ novel CNN (convolutional neural network) architectures to best fit to the given task and incorporate appropriate pre-processing of input data e.g. optical flow vectors in Cellpose [38] (published in 2020), or expect prior constraints to be met as in StarDist [39]

(2018) intended for nucleus-like (convex) shapes. The most commonly used DL methods many of the recent publications (e.g. unet4nuclei [40] (2019), Cellpose [38], StarDist [39], nucleAIzer [16] (2020)) build on are U-Net [26] (2015) and Mask R-CNN [27] (2017). Even though both can be trained to return precise segmentations on new image data, the bioimage analysis community tends to prefer such software tools that rather offer a pre-trained model or can easily be configured with a graphical user interface (GUI). One way of steps in this direction is releasing software as interactive python notebooks (e.g. Jupyter [109]) so that users can visually inspect the effect of configurations (such a solution is available in OpSeF [75] (2020), StarDist [39] or unet4nuclei [40]). Models pre-trained on general or experiment-specific data are also commonly released with publications (see Cellpose [38], OpSeF [75], StarDist [39], nucleAIzer [16], unet4nuclei [40]). Supplementarily, online interfaces are also created where users can quickly try DL segmentation (using pre-trained models) on their own experimental images (see e.g. nucleAIzer [16] or Cellpose [38]).

On the other hand, DL methods rely on a typically large (except for U-Net) training set whose accuracy depends on the experts collecting and labelling the data [20] [23]. Even though several datasets are available as free and open source (e.g. the Broad Bioimage Benchmark Collection [48] [40] [49], or annual DL challenge datasets like ImageNet [110], DSB2018 [40] [50] etc.) and usually cover a wide range of image modalities, new experiments might potentially fall out of their scope. Therefore, new annotations are often required to achieve desired precision for which purpose an appropriate annotation software can be used; some open-source tools are Diffgram [58], Cytomine [59] [60], LabKit [57] etc.

AnnotatorJ [15] is proposed as an annotation software using DL to assist contour definition, available as an ImageJ/Fiji [61] [62] [63] plugin. Beyond numerous convenient functions for easy data handling, saving, classification, editing, batch import/export, customization, shortcuts etc., an integrated pre-trained DL model (U-Net) can be loaded and used for assisted object annotation via contour suggestion. Additionally, custom user-provided models can also be used to annotate various object classes on images. The plugin allows export to various data formats suitable to directly train different DL models such as semantic masks for e.g. U-Net, instance labelled masks for e.g. Mask R-CNN or object coordinates according to the COCO dataset [76] format etc. We demonstrated the efficiency both in terms of annotation time and pixel precision on multiple image modalities including cell culture and histopathology images (from various

experiments, cell and tissue types, labelling, types of microscopy, image quality etc.) as well as conventional photos taken in traffic (Cityscapes dataset [47] collected in various cities in Germany, showing diverse weather and scenes). Consequently, we suggest AnnotatorJ shall be applied in cell biology or custom casual annotation projects to create training datasets for object detection, classification, semantic- or instance segmentation. The project is open source, available on GitHub and well documented.

Such a custom annotated dataset may serve as basis of application or development of novel segmentation methods and customized model training. An efficient and robust instance segmentation DL method, nucleAIzer [16] is proposed, primarily intended for nucleus segmentation – however, additional cellular compartments such as cytoplasms [18] or nucleoli were also successfully retrieved using distinct models trained with our approach; see projects in chapter 5. nucleAIzer utilizes artificially generated image-mask pairs for unknown domain adaptation such as novel experimental setups to which no similar training data is at disposal. These synthetic images are created with image style transfer learning [43] and enable precise segmentation in the domain adapted to with increased accuracy – as our comparative experiments confirmed with regards to both internal (our method with and without image style transfer training [16]) and external (against other DL methods e.g. unet4nuclei [40] [16] or Cellpose [38] [18]) tests on various types of microscopy images. Additionally, this approach requires no parameter tuning (like CellProfiler [37] or ilastik [41] [42] pipelines), estimates object size for appropriate image rescaling automatically (locally) – on the contrary, e.g. in Cellpose, size must be forwarded manually or a separate call must be made for size estimation – while online (for faster prediction) a single size parameter is expected – as in the online version of Cellpose too. We have successfully applied nucleAIzer in multiple real experimental projects [17] [18] (see more in chapter 5), thus we are confident it will positively contribute to the cyto-data processing community. It is also open source, applicable in a local environment, online or integrated in common analysis software such as CellProfiler.

6.1. Conclusions

A complete workflow of annotation, training and single-cell level segmentation with deep learning (DL) has been presented in this work. With the also DL-driven semi-automatic annotation tool AnnotatorJ, vast amounts of object annotation may be quickly created with ease

on images. Then, these annotated masks can be collected in a database serving as training data alongside artificial images representing novel experimental scenarios, for the instance segmentation method nucleAIzer. A single model may be trained for all potential imaging and sample conditions comprising of heterogeneous image domains, then applied conveniently either locally or online.

Acknowledgements

First of all, I would like to thank my supervisor **Péter Horváth** for supporting my scientific work and helping me in various tasks, I am grateful for his invaluable pieces of advice and for the final push I needed to start my PhD.

I would also like to express my gratitude to past and present members of the BIOMAG group for they welcomed me warmly and helped my scientific progress over the years. I especially appreciate the help of **Csaba Molnár** from whom I learnt scientific image processing, **Krisztián Koós** who always found time to lend advice in science and technical issues as well, **Tamás Balassa** who guided me in machine learning, I thank **Ábel Szkalisity** for the brainstorming discussions, and finally **András Kriston**, **Ferenc Kovács** and **Tivadar Danka** for sharing their experiences in various matters. I also thank all members of BIOMAG for the pleasant environment to work in.

Last but not least, I am grateful to my family for keeping my spirits high throughout the time of writing this thesis and my entire scientific career, it would not have been possible without the support of my mother **Erzsébet** and my brother **Gábor**.

My research was funded by the LENDULET-BIOMAG grant (2018-342) and the European Regional Development Funds (GINOP-2.3.2-15-2016-00001, GINOP-2.3.2-15-2016-00006, GINOP-2.3.2-15-2016-00026, GINOP-2.1.1-15-2016-000072, and GINOP-2.3.2-15-2016-00037), H2020-discovAIR (874656), and Chan Zuckerberg Initiative, Seed Networks for the HCA-DVP. I also acknowledge the NVIDIA Corporation for providing NVIDIA Titan Xp GPUs.

References

[1] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems 25,* vol. 25, pp. 1097-1105, 2012.

[2] Y. Sun, X. Wang and X. Tang, "Deep Learning Face Representation from Predicting 10,000 Classes," *IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1891-1898, 2014.

[3] Y. Taigman, M. Yang, M. Ranzato and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," *IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1701-1708, 2014.

[4] F. Schroff, D. Kalenichenko and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 815-823, 2015.

[5] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV,* pp. 779-788, 2016.

[6] V. Badrinarayanan, A. Kendall and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 39, no. 12, pp. 2481-2495, 2017.

[7] S. Grigorescu, B. Trasnea, T. Cocias and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics,* vol. 37, no. 3, p. 362– 386, 2020.

[8] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Signal Processing Magazine,* vol. 29, no. 6, pp. 82-97, 2012.

[9] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature,* vol. 542, no. 7639, p. 115–118, 2017.

[10] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization," *Psychological review,* vol. 65, no. 6, 1958.

[11] S. Linnainmaa, "The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors," *Master's Thesis, University of Helsinki,* 1970.

[12] P. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science," *Thesis (Ph. D.). Appl. Math. Harvard University,* 1974.

[13] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *Nature,* vol. 323, no. 6088, p. 533–536, 1986.

[14] A. G. Ivakhnenko and V. G. Lapa, "Cybernetic Predicting Devices," *CCM Information Corporation,* 1965.

[15] R. Hollandi, Á. Diósdi, G. Hollandi, N. Moshkov and P. Horváth, "AnnotatorJ: an ImageJ plugin to ease hand-annotation of cellular compartments," *Molecular Biology of the Cell,* vol. 31, no. 20, pp. 2179-2186, 2020.

[16] R. Hollandi, A. Szkalisity, T. Toth, E. Tasnadi, C. Molnar, B. Mathe, I. Grexa, J. Molnar, A. Balind, M. Gorbe, M. Kovacs, E. Migh, A. Goodman, T. Balassa, K. Koos, W. Wang, J. C. Caicedo, N. Bara, F. Kovacs, L. Paavolainen, T. Danka, A. Kriston, A. E. Carpenter, K. Smith and P. Horvath, "nucleAIzer: a parameter-free deep learning framework for nucleus segmentation using image style transfer," *Cell Systems,* vol. 10, no. 5, pp. 453-458.e6, 2020.

[17] J. L. Daly, B. Simonetti, C. Antón-Plágaro, M. K. Williamson, D. K. Shoemark, L. Simón-Gracia, K. Klein, M. Bauer, R. Hollandi, U. F. Greber, P. Horvath, R. B. Sessions, A. Helenius, J. A. Hiscox, T. Teesalu, D. A. Matthews, A. D. Davidson, P. J. Cullen and Y. Yamauchi, "Neuropilin-1 is a host factor for SARS-CoV-2 infection," *Science,* vol. 370, no. 6518, pp. 861-865, 2020.

[18] A. Mund, F. Coscia, R. Hollandi, F. Kovacs, A. Kriston, A.-D. Brunner, M. Bzorek, S. Naimy, L. M. R. Gjerdrum, B. Dyring-Andersen, J. M. Bulkescher, C. Lukas, C. Gnann, E. Lundberg, P. Horvath and M. Mann, "AI-driven Deep Visual Proteomics defines cell identity and heterogeneity," *bioRxiv,* 2021.

[19] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature,* vol. 521, no. 7553, p. 436–444, 2015.

[20] A. Gupta, P. J. Harrison, H. Wieslander, N. Pielawski, K. Kartasalo, G. Partel, L. Solorzano, A. Suveer, A. H. Klemm, O. Spjuth, I. Sintorn and C. Wählby, "Deep Learning in Image Cytometry: A Review," *Cytometry Part A,* vol. 95, no. 4, pp. 366-380, 2019.

[21] R. Raina, A. Madhavan and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," *Proceedings of the 26th Annual International Conference on Machine Learning,* p. 873–880, 2009.

[22] Y. E. Wang, G.-Y. Wei and D. Brooks, "Benchmarking TPU, GPU, and CPU Platforms for Deep Learning," *arXiv,* 2019.

[23] E. Moen, D. Bannon, T. Kudo, W. Graf, M. Covert and D. V. Valen, "Deep learning for cellular image analysis," *Nature Methods,* vol. 16, no. 12, p. 1233–1246, 2019.

[24] Y. LeCun, "Handwritten digit recognition with a back-propagation network," *Proc. Advances in Neural Information Processing Systems,* p. 396–404, 1990.

[25] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[26] O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *Lecture Notes in Computer Science,* pp. 234-241, 2015.

[27] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," *IEEE International Conference on Computer Vision (ICCV),* pp. 2980-2988, 2017.

[28] K. Sirinukunwattana, J. P. Pluim, H. Chen, X. Qi, P.-A. Heng, Y. B. Guo, L. Y. Wang, B. J. Matuszewski, E. Bruni, U. Sanchez, A. Böhm, O. Ronneberger, B. B. Cheikh, D. Racoceanu, P. Kainz, M. Pfeiffer, M. Urschler, D. R. Snead and N. M. Rajpoot, "Gland segmentation in colon histology images: The glas challenge contest," *Medical Image Analysis,* vol. 35, pp. 489-502, 2017.

[29] Y. Song, E.-L. Tan, X. Jiang, J.-Z. Cheng, D. Ni, S. Chen, B. Lei and T. Wang, "Accurate Cervical Cell Segmentation from Overlapping Clumps in Pap Smear Images," *IEEE Transactions on Medical Imaging,* vol. 36, no. 1, pp. 288-300, 2017.

[30] R. Vaillant, C. Monrocq and Y. LeCun, "Original approach for the localisation of objects in images," *IEE Proceedings: Vision, Image and Signal Processing,* vol. 141, no. 4, p. 245–250, 1994.

[31] S. Lawrence, C. Giles, A. C. Tsoi and A. Back, "Face recognition: a convolutional neural-network approach," *IEEE Transactions on Neural Networks,* vol. 8, no. 1, pp. 98-113, 1997.

[32] O. Dürr and B. Sick, "Single-Cell Phenotype Classification Using Deep Convolutional Neural Networks," *Journal of Biomolecular Screening,* vol. 21, no. 9, pp. 998-1003, 2016.

[33] O. Z. Kraus, B. T. Grys, J. Ba, Y. Chong, B. J. Frey, C. Boone and B. J. Andrews, "Automated analysis of high-content microscopy data with deep learning," *Molecular Systems Biology,* vol. 13, no. 4, 2017.

[34] P. Eulenberg, N. Köhler, T. Blasi, A. Filby, A. E. Carpenter, P. Rees, F. J. Theis and F. A. Wolf, "Reconstructing cell cycle and disease progression using deep learning," *Nature Communications,* vol. 8, no. 1, p. 463, 2017.

[35] R. M. Haralick and L. G. Shapiro, "Image Segmentation Techniques," *Proceedings Volume 0548, Applications of Artificial Intelligence II,* 1985.

[36] R. Adams and L. Bischof, "Seeded region growing," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 16, no. 6, pp. 641-647, 1994.

[37] A. E. Carpenter, T. R. Jones, M. R. Lamprecht, C. Clarke, I. H. Kang, O. Friman, D. A. Guertin, J. H. Chang, R. A. Lindquist, J. Moffat, P. Golland and D. M. Sabatini, "CellProfiler: image analysis software for identifying and quantifying cell phenotypes," *Genome Biology,* vol. 7, no. 10, 2006.

[38] C. Stringer, T. Wang, M. Michaelos and M. Pachitariu, "Cellpose: a generalist algorithm for cellular segmentation," *Nature Methods,* 2020.

[39] U. Schmidt, M. Weigert, C. Broaddus and G. Myers, "Cell Detection with Star-convex Polygons," *Medical Image Computing and Computer Assisted Intervention - {MICCAI} 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part {II},* pp. 265-273, 2018.

[40] J. C. Caicedo, A. Goodman, K. W. Karhohs, B. A. Cimini, J. Ackerman, M. Haghighi, C. Heng, T. Becker, M. Doan, C. McQuin, M. Rohban, S. Singh and A. E. Carpenter, "Nucleus segmentation across imaging experiments: the 2018 Data Science Bowl," *Nature Methods,* vol. 16, no. 12, p. 1247–1253, 2019.

[41] C. Sommer, C. N. Straehle, U. Köthe and F. A. Hamprecht, "ilastik: Interactive Learning and Segmentation Toolkit," *Eighth IEEE International Symposium on Biomedical Imaging (ISBI). Proceedings,* pp. 230-233, 2011.

[42] S. Berg, D. Kutra, T. Kroeger, C. N. Straehle, B. X. Kausler, C. Haubold, M. Schiegg, J. Ales, T. Beier, M. Rudy, K. Eren, J. I. Cervantes, B. Xu, F. Beuttenmueller, A. Wolny, C. Zhang, U. Koethe, F. A. Hamprecht and A. Kreshuk, "ilastik: interactive machine learning for (bio)image analysis," *Nature Methods,* vol. 16, no. 12, p. 1226–1232, 2019.

[43] P. Isola, J.-Y. Zhu, T. Zhou and A. A. Efros, “Image-to-Image Translation with Conditional Adversarial Networks,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 5967-5976, 2017.

[44] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, “Generative Adversarial Networks,” *Proceedings of the International Conference on Neural Information Processing Systems,* p. 2672–2680, 2014.

[45] S. Ren, K. He, R. Girshick and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 39, no. 6, pp. 1137-1149, 2017.

[46] T. Litman, “Personalized medicine—concepts, technologies, and applications in inflammatory skin diseases,” *APMIS,* vol. 127, no. 5, pp. 386-424, 2019.

[47] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth and B. Schiele, “The Cityscapes Dataset for Semantic Urban Scene Understanding,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 3213-3223, 2016.

[48] V. Ljosa, K. L. Sokolnicki and A. E. Carpenter, “Annotated high-throughput microscopy image sets for validation,” *Nature Methods,* vol. 9, no. 7, p. 637, 2012.

[49] J. C. Caicedo, J. Roth, A. Goodman, T. Becker, K. W. Karhohs, M. Broisin, C. Molnar, C. McQuin, S. Singh, F. J. Theis and A. E. Carpenter, “Evaluation of Deep Learning Strategies for Nucleus Segmentation in Fluorescence Images,” *Cytometry Part A,* vol. 95, no. 9, pp. 952-965, 2019.

[50] “2018 Data Science Bowl,” 2018. [Online]. Available: https://www.kaggle.com/c/data-science-bowl-2018.

[51] V. Ulman, M. Maška, K. E. G. Magnusson, O. Ronneberger, C. Haubold, N. Harder, P. Matula, P. Matula, D. Svoboda, M. Radojevic, I. Smal, K. Rohr, J. Jaldén, H. M. Blau, O. Dzyubachyk, B. Lelieveldt, P. Xiao, Y. Li, S.-Y. Cho, A. C. Dufour, J.-C. Olivo-Marin, C. C. Reyes-Aldasoro, J. A. Solis-Lemus, R. Bensch, T. Brox, J. Stegmaier, R. Mikut, S. Wolf, F. A. Hamprecht, T. Esteves, P. Quelhas, Ö. Demirel, L. Malmström, F. Jug, P. Tomancak, E. Meijering, A. Muñoz-Barrutia, M. Kozubek and C. Ortiz-de-Solorzano, “An objective comparison of cell-tracking algorithms,” *Nature Methods,* vol. 14, no. 12, p. 1141–1152, 2017.

[52] N. Kumar, R. Verma, D. Anand, Y. Zhou, O. F. Onder, E. Tsougenis, H. Chen, P.-A. Heng, J. Li, Z. Hu, Y. Wang, N. A. Koohbanani, M. Jahanifar, N. Z. Tajeddin, A. Gooya, N. Rajpoot, X. Ren, S. Zhou, Q. Wang, D. Shen, C.-K. Yang, C.-H. Weng, W.-H. Yu, C.-Y. Yeh, S. Yang, S. Xu, P. H. Yeung, P. Sun, A. Mahbod, G. Schaefer, I. Ellinger, R. Ecker, O. Smedby, C. Wang, B. Chidester, T.-V. Ton, M.-T. Tran, J. Ma, M. N. Do, S. Graham, Q. D. Vu, J. T. Kwak, A. Gunda, R. Chunduri, C. Hu, X. Zhou, D. Lotfi, R. Safdari, A. Kascenas, A. O’Neil, D. Eschweiler, J. Stegmaier, Y. Cui, B. Yin, K. Chen, X. Tian, P. Gruening, E. Barth, E. Arbel, I. Remer, A. Ben-Dor, E. Sirazitdinova, M. Kohl, S. Braunewell, Y. Li, X. Xie, L. Shen, J. Ma, K. D. Baksi, M. A. Khan, J. Choo, A. Colomer, V. Naranjo, L. Pei, K. M. Iftekharuddin, K. Roy, D. Bhattacharjee, A. Pedraza, M. G. Bueno, S. Devanathan, S. Radhakrishnan, P. Koduganty, Z. Wu, G. Cai, X. Liu, Y. Wang and A. Sethi, “A Multi-Organ Nucleus Segmentation Challenge,” *IEEE Transactions on Medical Imaging,* vol. 39, no. 5, pp. 1380-1391, 2020.

[53] R. Verma, N. Kumar, A. Patil, N. Kurian, S. Rane and A. Sethi, “Multi-organ Nuclei Segmentation and Classification Challenge 2020,” 2020.

[54] “Histopathologic Cancer Detection,” 2019. [Online]. Available: https://www.kaggle.com/c/histopathologic-cancer-detection/overview.

[55] A. Gupta and R. Gupta, “ISBI 2019 C-NMC Challenge: Classification in Cancer Cell Imaging,” in *Lecture Notes in Bioengineering*, Singapore, Springer, 2020.

[56] I. Arganda-Carreras, S. C. Turaga, D. R. Berger, D. Cireşan, A. Giusti, L. M. Gambardella, J. Schmidhuber, D. Laptev, S. Dwivedi, J. M. Buhmann, T. Liu, M. Seyedhosseini, T. Tasdizen, L. Kamentsky, R. Burget, V. Uher, X. Tan, C. Sun, T. Pham, E. Bas, M. Uzunbas, A. Cardona, J. Schindelin and S. Seung, “Crowdsourcing the creation of image segmentation algorithms for connectomics,” *Frontiers in Neuroanatomy,* vol. 9, p. 142, 2015.

[57] M. Arzt, “LabKit,” 2017. [Online]. Available: https://imagej.net/Labkit.

[58] A. Sarkis and P. Estrada, “Diffgram,” [Online]. Available: https://diffgram.com/.

[59] R. Marée, L. Rollus, B. Stévens, R. Hoyoux, G. Louppe, R. Vandaele, J.-M. Begon, P. Kainz, P. Geurts and L. Wehenkel, “Collaborative analysis of multi-gigapixel imaging data using Cytomine,” *Bioinformatics,* vol. 32, no. 9, p. 1395–1401, 2016.

[60] U. Rubens, R. Hoyoux, L. Vanosmael, M. Ouras, M. Tasset, C. Hamilton, R. Longuespée and R. Marée, “Cytomine: Toward an Open and Collaborative Software Platform for Digital Pathology Bridged to Molecular Investigations,” *Proteomics Clinical Applications,* vol. 13, no. 1, 2019.

[61] C. A. Schneider, W. S. Rasband and K. W. Eliceiri, “NIH Image to ImageJ: 25 years of image analysis,” *Nature Methods,* vol. 9, no. 7, p. 671–675, 2012.

[62] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, J.-Y. Tinevez, D. J. White, V. Hartenstein, K. Eliceiri, P. Tomancak and A. Cardona, “Fiji: an open-source platform for biological-image analysis,” *Nature Methods,* vol. 9, no. 7, p. 676–682, 2012.

[63] M. D. Abramoff, P. J. Magalhaes and S. J. Ram, “Image processing with ImageJ,” *Biophotonics international,* vol. 11, no. 7, pp. 36-42, 2004.

[64] I. Arganda-Carreras, V. Kaynig, C. Rueden, K. W. Eliceiri, J. Schindelin, A. Cardona and H. S. Seung, “Trainable Weka Segmentation: a machine learning tool for microscopy pixel classification,” *Bioinformatics,* vol. 33, no. 15, p. 2424–2426, 2017.

[65] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, M. Kudlur, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, M. Schuster, R. Monga, S. Moore, D. Murray, C. Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015. [Online]. Available: https://www.tensorflow.org/.

[66] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv,* 2018.

[67] H. Irshad, L. Montaser-Kouhsari, G. Waltz, O. Bucur, J. A. Nowak, F. Dong, N. W. Knoblauch and A. H. Beck, “Crowdsourcing image annotation for nucleus detection and segmentation in computational pathology: evaluating experts, automated methods, and the crowd,” *Pacific Symposium on Biocomputing,* pp. 294-305, 2015.

[68] T. C. G. A. R. Network, R. McLendon and e. al., “Comprehensive genomic characterization defines human glioblastoma genes and core pathways,” *Nature,* vol. 455, no. 7216, p. 1061–1068, 2008.

[69] N. Kumar, R. Verma, S. Sharma, S. Bhargava, A. Vahadane and A. Sethi, “A Dataset and a Technique for Generalized Nuclear Segmentation for Computational Pathology,” *IEEE transactions on medical imaging,* vol. 36, no. 7, p. 1550–1560, 2017.

[70] L. P. Coelho, A. Shariff and R. F. Murphy, “Nuclear segmentation in microscope cell images: a hand-segmented dataset and comparison of algorithms,” *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro,* vol. 5193098, p. 518–521, 2009.

[71] F. P. Miller, A. Vandome and J. McBrewster, “Apache Maven,” 2010.

[72] E. D. D. Team, “Deeplearning4j: Open-source distributed deep learning for the JVM,” 2016. [Online]. Available: http://deeplearning4j.org.

[73] F. Chollet, “Keras,” 2015. [Online]. Available: https://keras.io.

[74] M. Kass, A. Witkin and D. Terzopoulos, “Snakes: Active contour models,” *International Journal of Computer Vision,* vol. 1, no. 4, p. 321–331, 1988.

[75] T. M. Rasse, R. Hollandi and P. Horvath, “OpSeF: Open Source Python Framework for Collaborative Instance Segmentation of Bioimages,” *Frontiers in Bioengineering and Biotechnology,* vol. 8, p. 1171, 2020.

[76] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. L. Zitnick, “Microsoft COCO: Common Objects in Context,” *Lecture Notes in Computer Science,* pp. 740-755, 2014.

[77] J. Guérin, S. Thiery, E. Nyiri and O. Gibaru, “Unsupervised robotic sorting: Towards autonomous decision making robots,” *International Journal of Artificial Intelligence and Applications (IJAIA),* vol. 9, no. 2, 2018.

[78] “Image Annotation Services - Lionbridge.AI,” [Online]. Available: https://lionbridge.ai/services/image.

[79] “Hive,” [Online]. Available: https://thehive.ai/.

[80] T. Falk, D. Mai, R. Bensch, Ö. Çiçek, A. Abdulkadir, Y. Marrakchi, A. Böhm, J. Deubner, Z. Jäckel, K. Seiwald, A. Dovzhenko, O. Tietz, C. D. Bosco, S. Walsh, D. Saltukoglu, T. L. Tay, M. Prinz, K. Palme, M. Simons, I. Diester, T. Brox and O. Ronneberger, “U-Net: deep learning for cell counting, detection, and morphometry,” *Nature Methods,* vol. 16, no. 1, p. 67–70, 2019.

[81] M. Weigert, U. Schmidt, R. Haase, K. Sugawara and G. Myers, “Star-convex Polyhedra for 3D Object Detection and Segmentation in Microscopy,” *The IEEE Winter Conference on Applications of Computer Vision (WACV),* 2020.

[82] n. contributors, “napari: a multi-dimensional image viewer for python,” 2019. [Online]. Available: https://napari.org/.

[83] N. P. Jack, W. Thomas, L. Marick and R. Fabien, “Nuclei Segmentation in Histopathology Images Using Deep Neural Networks,” *Zenodo. Dataset,* 2018.

[84] P. D. Caie, R. E. Walls, A. Ingleston-Orme, S. Daya, T. Houslay, R. Eagle, M. E. Roberts and N. O. Carragher, "High-Content Phenotypic Profiling of Drug Response Signatures across Distinct Cancer Cells," *Molecular Cancer Therapeutics,* vol. 9, no. 6, pp. 1913-1926, 2010.

[85] K. Smith, Y. Li, F. Piccinini, G. Csucs, C. Balazs, A. Bevilacqua and P. Horvath, "CIDRE: an illumination-correction method for optical microscopy," *Nature Methods,* vol. 12, no. 5, p. 404–406, 2015.

[86] V. Frismantas, M. P. Dobay, A. Rinaldi, J. Tchinda, S. H. Dunn, J. Kunz, P. Richter-Pechanska, B. Marovca, O. Pail, S. Jenni, E. Diaz-Flores, B. H. Chang, T. J. Brown, R. H. Collins, S. Uhrig, G. P. Balasubramanian, O. R. Bandapalli, S. Higi, S. Eugster, P. Voegeli, M. Delorenzi, G. Cario, M. L. Loh, M. Schrappe, M. Stanulla, A. E. Kulozik, M. U. Muckenthaler, V. Saha, J. A. Irving, R. Meisel, T. Radimerski, A. Von Stackelberg, C. Eckert, J. W. Tyner, P. Horvath, B. C. Bornhauser and J.-P. Bourquin, "Ex vivo drug response profiling detects recurrent sensitivity patterns in drug-resistant acute lymphoblastic leukemia," *Blood,* vol. 129, no. 11, p. e26–e37, 2017.

[87] C. Brasko, K. Smith, C. Molnar, N. Farago, L. Hegedus, A. Balind, T. Balassa, A. Szkalisity, F. Sukosd, K. Kocsis, B. Balint, L. Paavolainen, M. Z. Enyedi, I. Nagy, L. G. Puskas, L. Haracska, G. Tamas and P. Horvath, "Intelligent image-based in situ single-cell isolation," *Nature Communications,* vol. 9, no. 1, 2018.

[88] E. Frank, M. A. Hall and I. H. Witten, "The WEKA Workbench," in *Data Mining: Practical Machine Learning Tools and Techniques*, 2016.

[89] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations,* vol. 11, no. 1, 2009.

[90] A. Lehmussola, P. Ruusuvuori, J. Selinummi, H. Huttunen and O. Yli-Harja, "Computational Framework for Simulating Fluorescence Microscope Images With Cell Populations," *IEEE Transactions on Medical Imaging,* vol. 26, no. 7, pp. 1010-1016, 2007.

[91] J. H. Holland, Adaptation in Natural and Artificial Systems, Michigan: University ofMichigan Press, 1975.

[92] G. Li, T. Liu, J. Nie, L. Guo, J. Chen, J. Zhu, W. Xia, A. Mara, S. Holley and S. T. C. Wong, "Segmentation of touching cell nuclei using gradient flow tracking," *Journal of microscopy,* vol. 231, p. 47–58, 2008.

[93] A. Buslaev, S. Seferbekov, V. Durnov and Anton, "DSB2018 [ods.ai] topcoders," 2018. [Online]. Available: https://github.com/selimsef/dsb2018_topcoders/.

[94] M. Jiang, "2018 Data Science Bowl 2nd Place Solution," 2018. [Online]. Available: https://github.com/jacobkie/2018DSB.

[95] D. A. V. Valen, T. Kudo, K. M. Lane, D. N. Macklin, N. T. Quach, M. M. DeFelice, I. Maayan, Y. Tanouchi, E. A. Ashley and M. W. Covert, "Deep Learning Automates the Quantitative Analysis of Individual Cells in Live-Cell Imaging Experiments," *PLoS Computational Biology,* vol. 12, no. 11, 2016.

[96] N. Moshkov, B. Mathe, A. Kertesz-Farkas, R. Hollandi and P. Horvath, "Test-time augmentation for deep learning-based cell segmentation on microscopy images," *Scientific Reports,* vol. 10, no. 1, 2020.

[97] S. A. Haney, High Content Screening: Science, Techniques and Applications, New York: Wiley-Interscience, 2008.

[98] J. R. H. D. L. Taylor and K. A. Giuliano, High Content Screening: A Powerful Approach to Systems Cell Biology and Drug Discovery, Totowa, NJ: Humana Press, 2010.

[99] F. Piccinini, T. Balassa, A. Szkalisity, C. Molnar, L. Paavolainen, K. Kujala, K. Buzas, M. Sarazova, V. Pietiainen, U. Kutay, K. Smith and P. Horvath, "Advanced Cell Classifier: user-friendly machine-learning-based software for discovering phenotypes in high-content imaging data," *Cell Systems,* vol. 4, no. 6, pp. 651-655, 2017.

[100] L. Badertscher, T. Wild, C. Montellese, L. T. Alexander, L. Bammert, M. Sarazova, M. Stebler, G. Csucs, T. U. Mayer, N. Zamboni, I. Zemp, P. Horvath and U. Kutay, "Genome-wide RNAi Screening Identifies Protein Modules Required for 40S Subunit Synthesis in Human Cells," *Cell Reports,* vol. 13, no. 12, pp. 2879-2891, 2015.

[101] A.-D. Brunner, M. Thielert, C. Vasilopoulou, C. Ammar, F. Coscia, A. Mund, O. B. Horning, N. Bache, A. Apalategui, M. Lubeck, O. Raether, M. A. Park, S. Richter, D. S. Fischer, F. J. Theis, F. Meier and M. Mann, "Ultra-high sensitivity mass spectrometry quantifies single-cell proteome changes upon perturbation," *bioRxiv,* 2020.

[102] D. Mahdessian, A. J. Cesnik, C. Gnann, F. Danielsson, L. Stenström, M. Arif, C. Zhang, R. Shutten, A. Bäckström, P. Thul, N. H. Cho, O. Carja, M. Uhlén, A. Mardinoglu, C. Stadler, C. Lindskog, B. Ayoglu, M. D. Leonetti, F. Pontén, D. Sullivan and E. Lundberg, "Spatiotemporal dissection of the cell cycle with single-cell proteogenomics," *bioRxiv,* 2020.

[103] N. Zhu, D. Zhang, W. Wang, X. Li, B. Yang, J. Song, X. Zhao, B. Huang, W. Shi, R. Lu, P. Niu, F. Zhan, X. Ma, D. Wang, W. Xu, G. Wu, G. F. Gao and W. Tan, "A Novel Coronavirus from Patients with Pneumonia in China, 2019," *New England Journal of Medicine,* vol. 382, no. 8, pp. 727-733, 2020.

[104] I. Banerjee, Y. Yamauchi, A. Helenius and P. Horvath, "High-Content Analysis of Sequential Events during the Early Phase of Influenza A Virus Infection," *PLOS ONE,* vol. 8, no. 7, 2013.

[105] M. Hoffmann, H. Kleine-Weber, S. Schroeder, N. Krüger, T. Herrler, S. Erichsen, T. S. Schiergens, G. Herrler, N.-H. Wu, A. Nitsche, M. A. Müller, C. Drosten and S. Pöhlmann, "SARS-CoV-2 Cell Entry Depends on ACE2 and TMPRSS2 and Is Blocked by a Clinically Proven Protease Inhibitor," *Cell,* vol. 181, no. 2, p. 271–280.e8, 2020.

[106] N. Imre, A. Hetényi, E. Szabó, B. Bodnár, A. Szkalisity, I. Gróf, A. Bocsik, M. A. Deli, P. Horvath, Á. Czibula, É. Monostori and T. A. Martinek, "Routing Nanomolar Protein Cargoes to Lipid Raft-Mediated/Caveolar Endocytosis through a Ganglioside GM1-Specific Recognition Tag," *Advanced Science,* vol. 7, no. 4, 2020.

[107] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel and B. Wiswedel, "KNIME: The Konstanz Information Miner," in *Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007)*, Springer, 2007, pp. 319-326.

[108] F. d. Chaumont, S. Dallongeville, N. Chenouard, N. Hervé, S. Pop, T. Provoost, V. Meas-Yedid, P. Pankajakshan, T. Lecomte, Y. L. Montagner, T. Lagache, A. Dufour and J.-C. Olivo-Marin, "Icy: an open bioimage informatics platform for extended reproducible research," *Nature Methods,* vol. 9, no. 7, p. 690–696, 2012.

[109] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing and J. D. Team, "Jupyter Notebooks – a

publishing format for reproducible computational workflows," *Positioning and Power in Academic Publishing: Players, Agents and Agendas,* pp. 87-90, 2016.

[110] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision,* vol. 115, no. 3, p. 211–252, 2015.

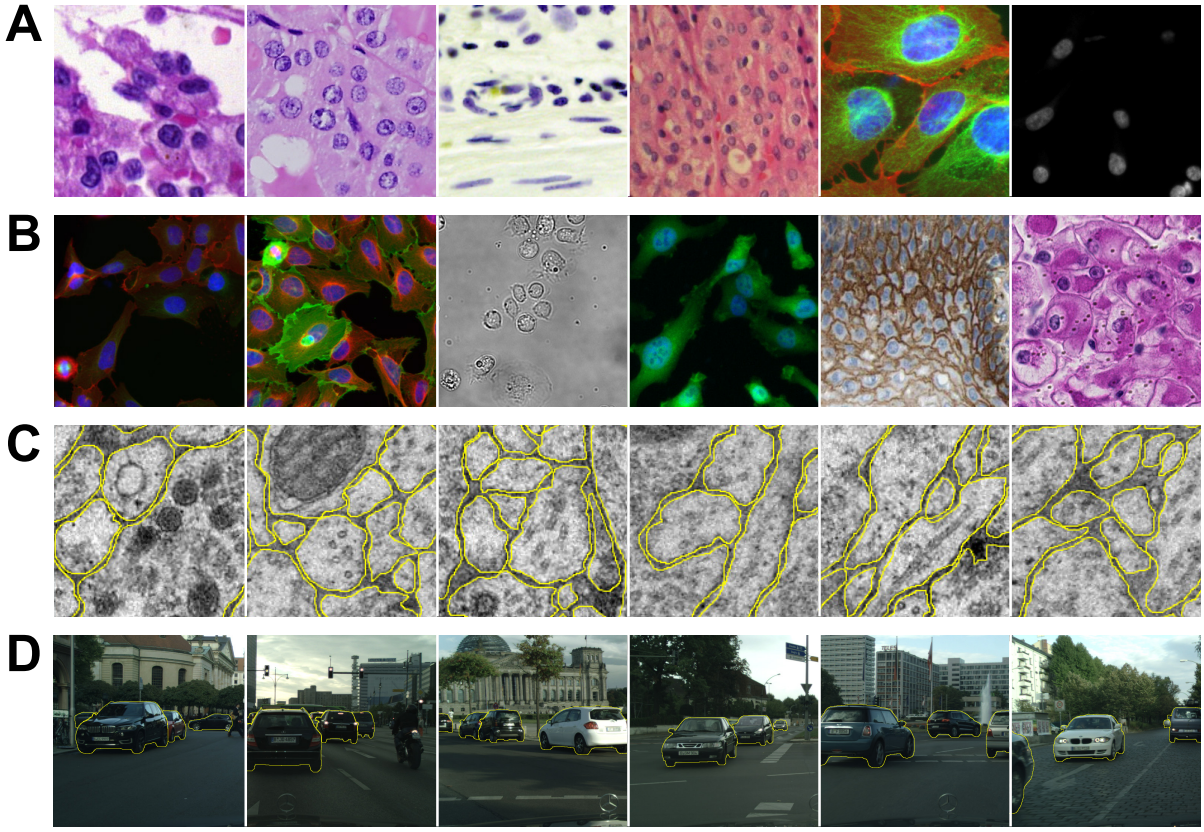
[111] R. Hollandi, E. Tasnádi, T. Balassa, E. Migh, I. B. Németh, K. Koós and P. Horváth, "Mesterséges intelligencia alapú digitális képelemzés lehetőségei a patológiában," in *Patológiai és molekuláris onkodiagnosztikai módszerek: Kézikönyv patológusoknak, kutatóknak, analitikusoknak, asszisztenseknek és a társszakmák képviselőinek*, Budapest, Hungary, Medicina, 2020, pp. 452-469.

Supplementary figures and tables

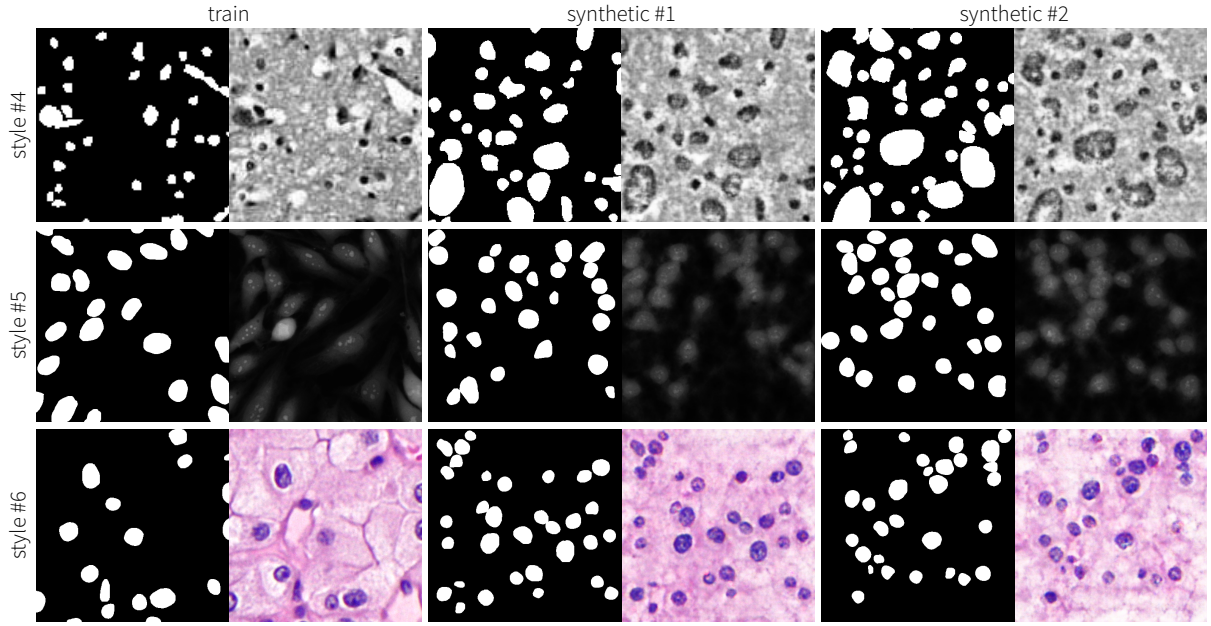
feature		tool							
		LabelImg	Lionbridge. AI	Hive	VGG Image Annotator	Diffgram	CytoMine	SlideRunner	AnnotatorJ
Open source		✓	x	x	✓	✓	✓	✓	✓
Cross-platform		✓	service	service	✓	✓	Ubuntu	✓	✓
Implementation		Python	N/A	N/A	web	Python, web	Docker, web	Python	Java
Annotation	Bounding box	✓	✓	✓	✓	✓	✓	✓	✓
	Freehand ROI	x	x	N/A	x	x	✓	x	✓
	Polygonal region	x	✓	✓	✓	✓	✓	✓	✓ (ImageJ)
	Semantic	x	✓	✓	x	x	✓	x	✓
	Single click*	x	x	x	x	x	✓ (magic wand)	✓	✓ (drag)
	Edit selection	x (drag)	N/A	N/A	✓	✓	✓	N/A	✓
Class option		✓	✓	✓	✓	✓	✓	✓	✓
DL	Support	x	AI assist	N/A	x	✓	✓	x	✓
	Model import	x	N/A	N/A	x	Tensorflow	N/A	x	Keras, DL4J

**: bounding box drawing with a single click and drag is not considered a single click annotation.*

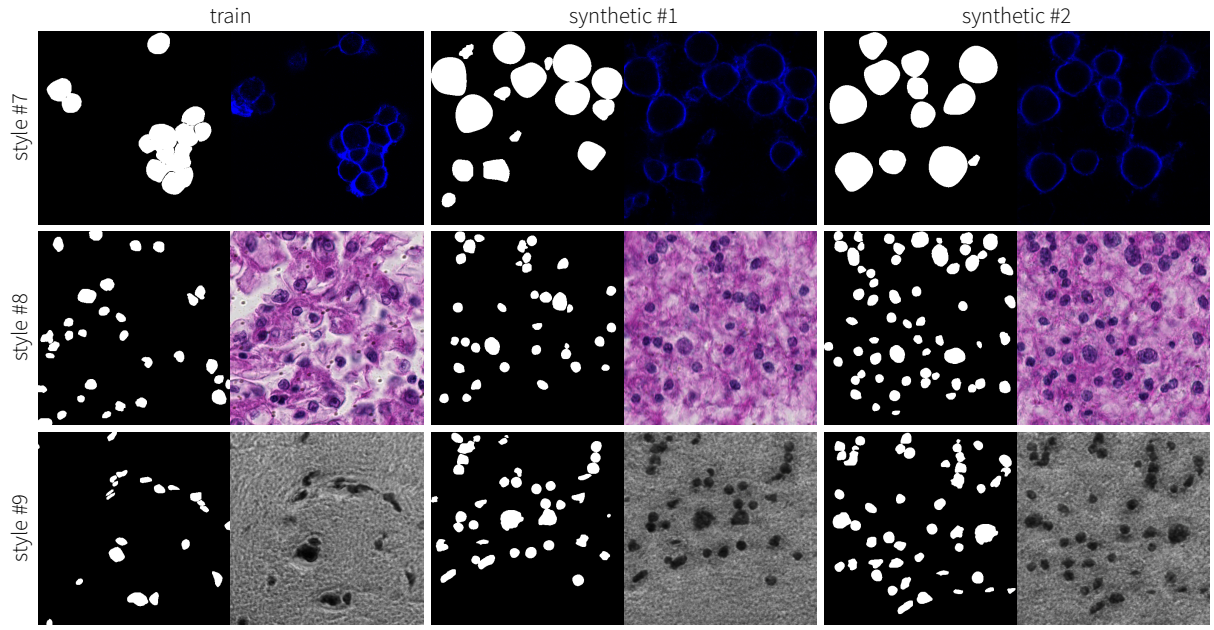
Supplementary Table 1. Annotation software tools compared to AnnotatorJ. Table is adapted from [15].



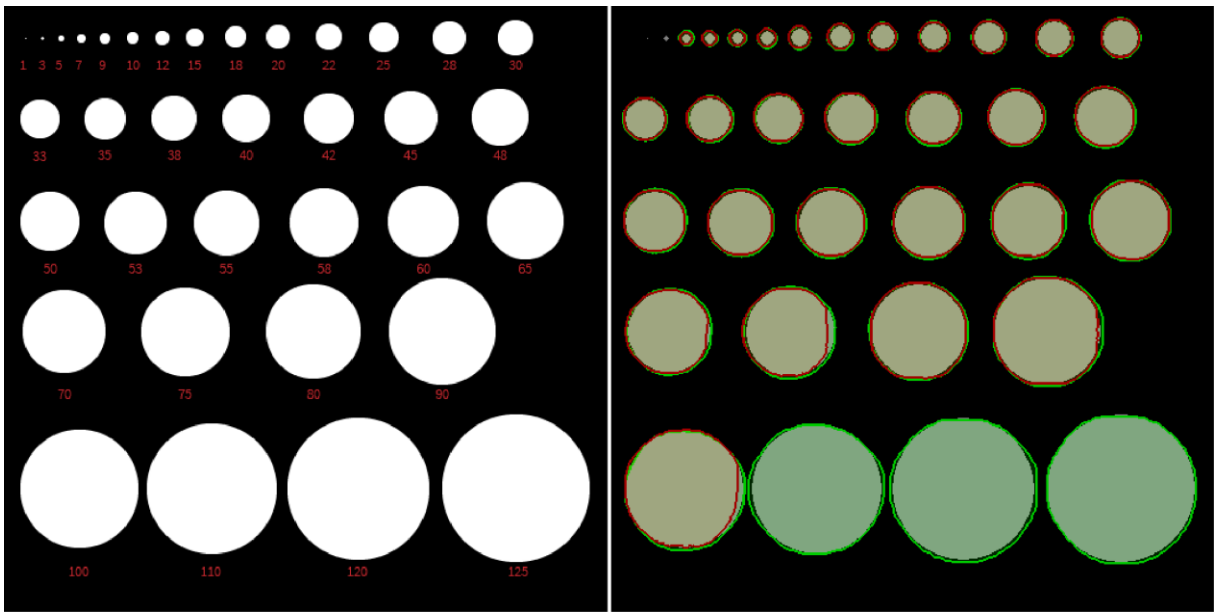
Supplementary Figure S1. Example images of test sets used in the evaluation of AnnotatorJ in chapter 3. Panels A-D correspond to test sets *i-iv* as in Figure 3.3: A) nucleus images containing histology and cell culture images acquired in brightfield or fluorescent microscopy at different magnifications. B) cytoplasm images as in A. C) EM (electron microscopy) records of neuronal structures in a label-free tissue section [56]. Objects are outlined in yellow for better visibility. D) traffic video frames from the Cityscapes dataset [47], the target object class (car) is highlighted as in C. Figure is based on [15].



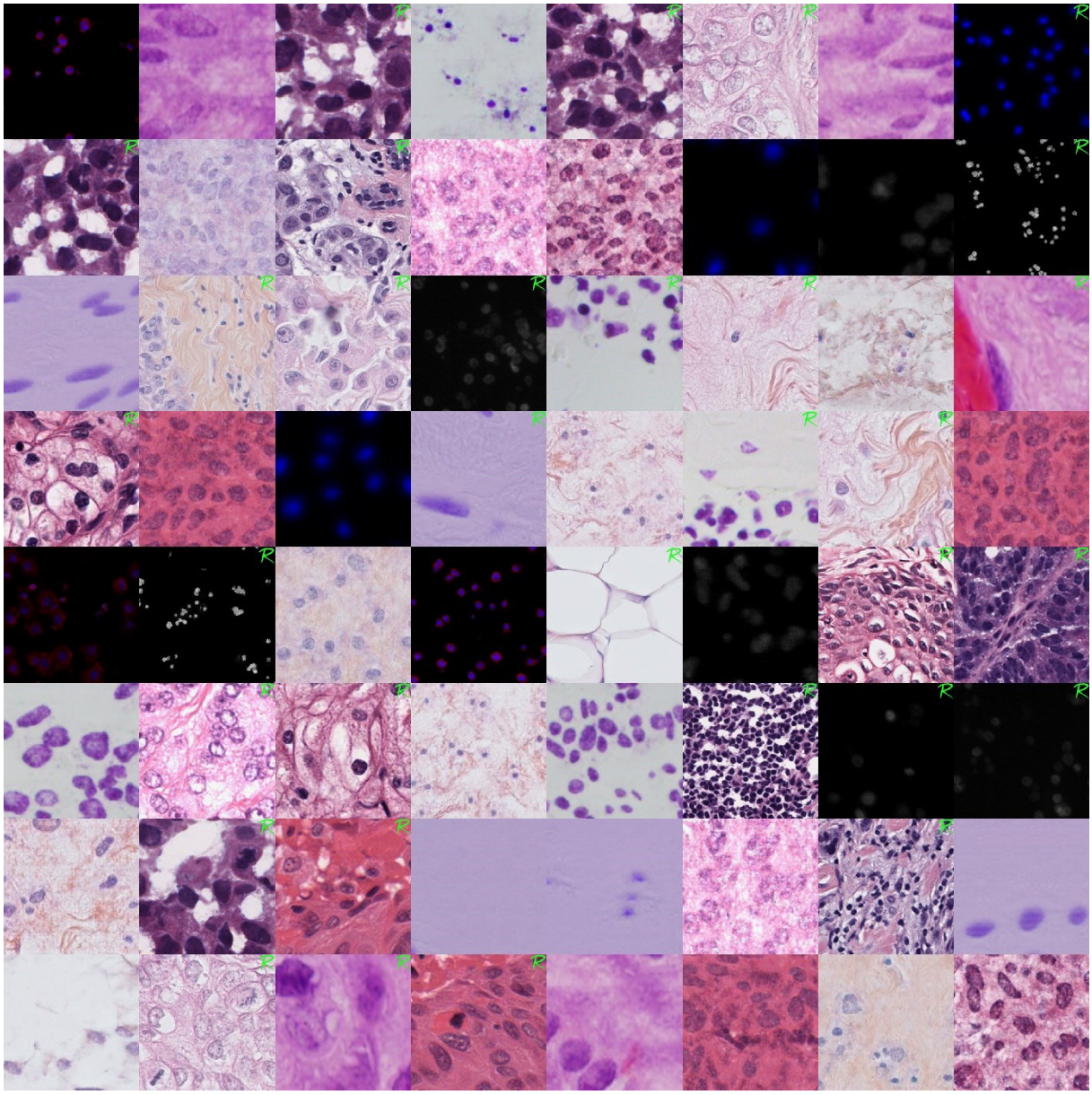
Supplementary Figure S2. Example training image/mask pairs and generated artificial style images. As on Figure 4.4, column 1 shows training mask/image pairs, 2-3 generated synthetic mask/image pairs. Rows correspond to three arbitrarily chosen styles. Style #4 is an unlabelled histological tissue section imaged in brightfield mode, from the DSB image set [40]; magnification and origin of tissue are unknown. Style #2 is a cell culture sample imaged in fluorescent microscopy, labelled for the nucleoli and cytoplasm, from the DSB stage 2 test set [40] [50]; magnification, staining and origin of sample are unknown. Style #3 is a histological tissue section imaged in brightfield mode, see source in Supplementary Table S2 of [16]. This figure is an extension of Figure 4.4. See also Supplementary Figure S3.



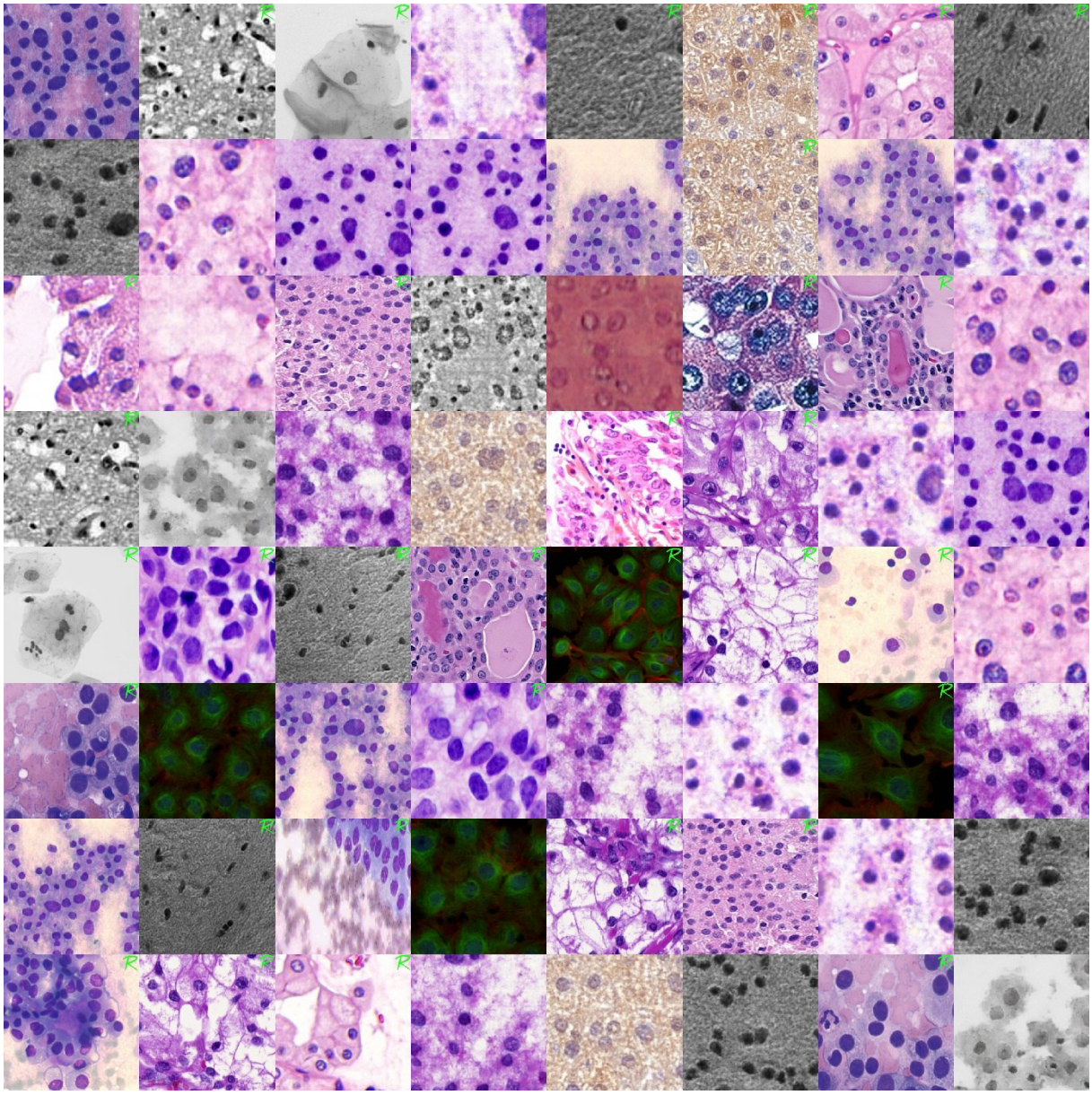
Supplementary Figure S3. Example training image/mask pairs and generated artificial style images. As on Figure 4.4 and Supplementary Figure S2, column 1 shows training mask/image pairs, 2-3 generated synthetic mask/image pairs. Rows correspond to three arbitrarily chosen styles. Style #7 is a cell culture sample imaged in fluorescent microscopy, labelled for the cell membrane, from our collaborator Imre Gombos (BRC, Szeged, Hungary); magnification, staining and origin of sample are unknown. Style #8 is a histological tissue section imaged in brightfield mode, patient-retrieved kidney sample H&E-stained, from our collaborator Hella Bolck (University Hospital Zürich, Switzerland); magnification is unknown. Style #9 is an unlabelled histological mouse brain tissue section imaged in brightfield mode at 40x magnification, from [85]. This figure is an extension of Figure 4.4. See also Supplementary Figure S2.



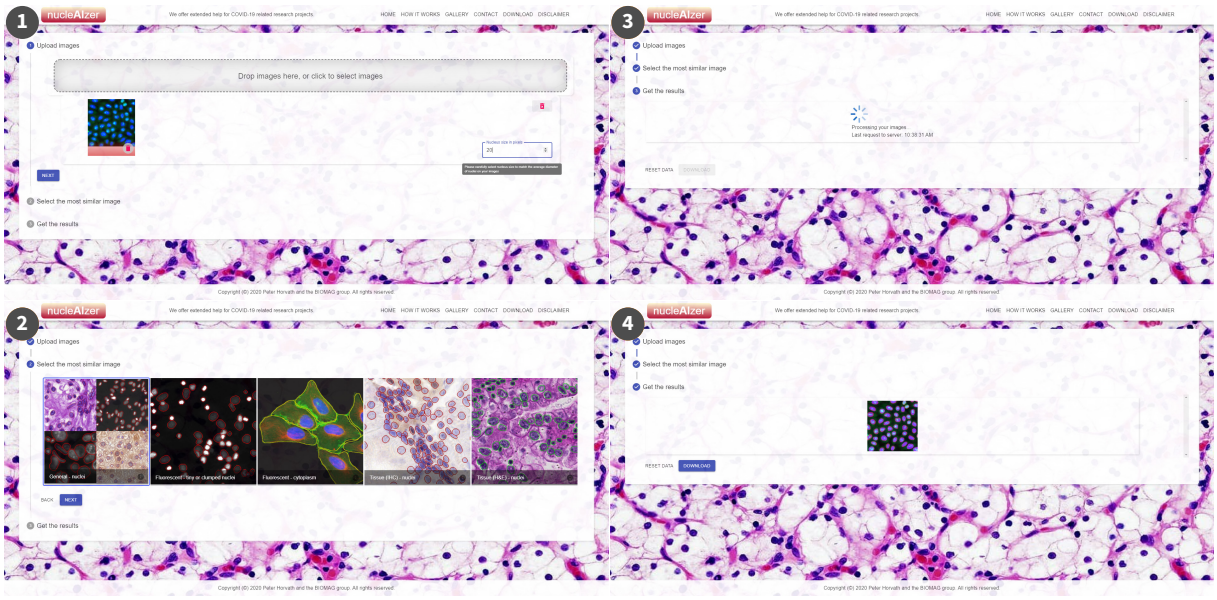
Supplementary Figure S4. Size robustness of nucleAIzer pre-trained models. White circles in the size range 1-125 pixels diameter were placed on a blank (black) image (left) and predicted with our pre-trained models (right): *preseg* predictions are overlayed in green, *postComp* in red. Despite the latter expects objects of approximately its trained size (40 pixels diameter) it detected them in the range 7-100, while the former is scale-independent and expected to find most objects in a wider range, it found circles with 7-125 pixels diameter even more accurately than the *postComp* model. See also Figure 4.11.



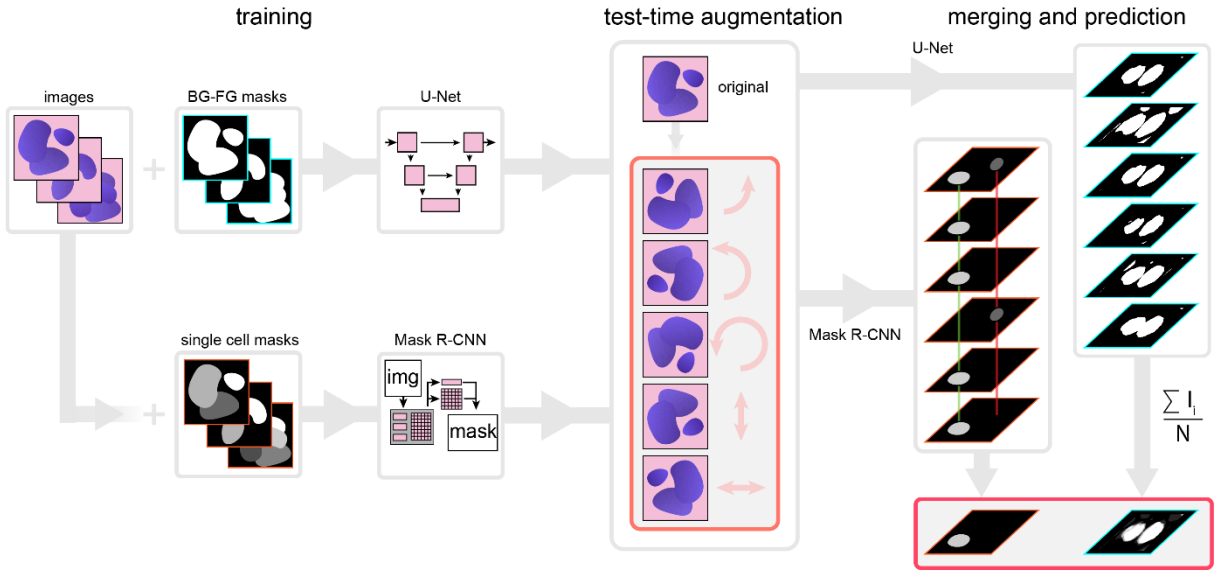
Supplementary Figure S5. Synthetic image recognition test image #1. Tiles are 50% real and fake, respectively, real ones are marked with a green R in the top right corner. Image sources can be found in Supplementary Table S2 of [16]. See also Supplementary Figure S6.



Supplementary Figure S6. Synthetic image recognition test image #2. Tiles are 50% real and fake, respectively, real ones are marked with a green R in the top right corner. Image sources can be found in Supplementary Table S2 of [16]. See also Supplementary Figure S5.

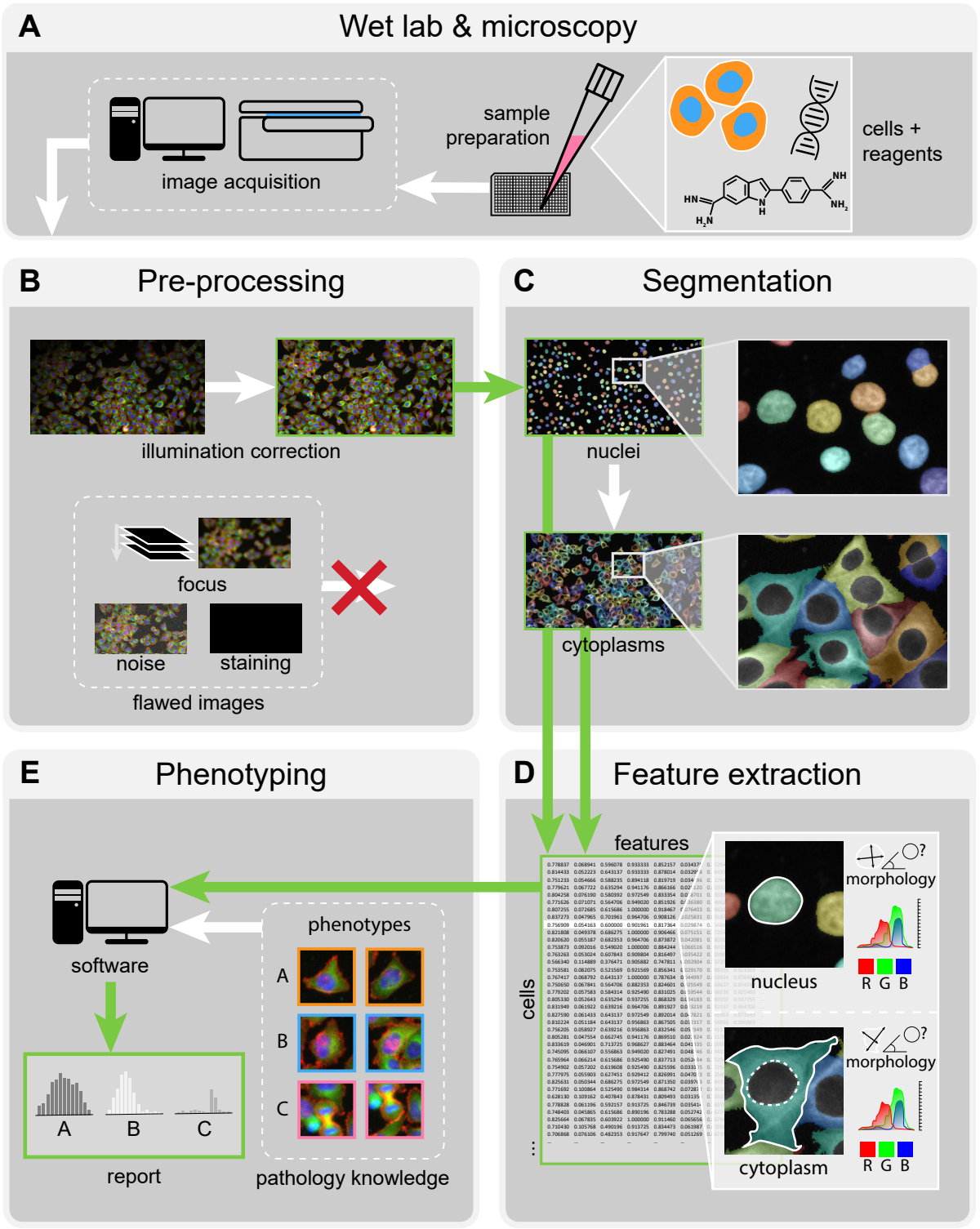


Supplementary Figure S7. nucleAIzer online tool. Numbers in grey circles in the upper left corner show the order of steps. 1) Image upload and estimated object size setting, 2) an appropriate model most similar to the current test image(s) is selected, 3) users wait for processing on the server side to complete and 4) segmentation results are displayed (as overlay) and can be downloaded. Users can upload their own experimental images via drag and drop, while downloadable results contain a 16-bit multi-labelled .tiff mask of all detected objects, individual object mask images and contours overlayed on an RGB image, for each input test image. The object size to be estimated refers to a median diameter in pixels (default is 20). A 256x256 8-bit 3-channel RGB image of a fluorescent cell culture is shown as an example.

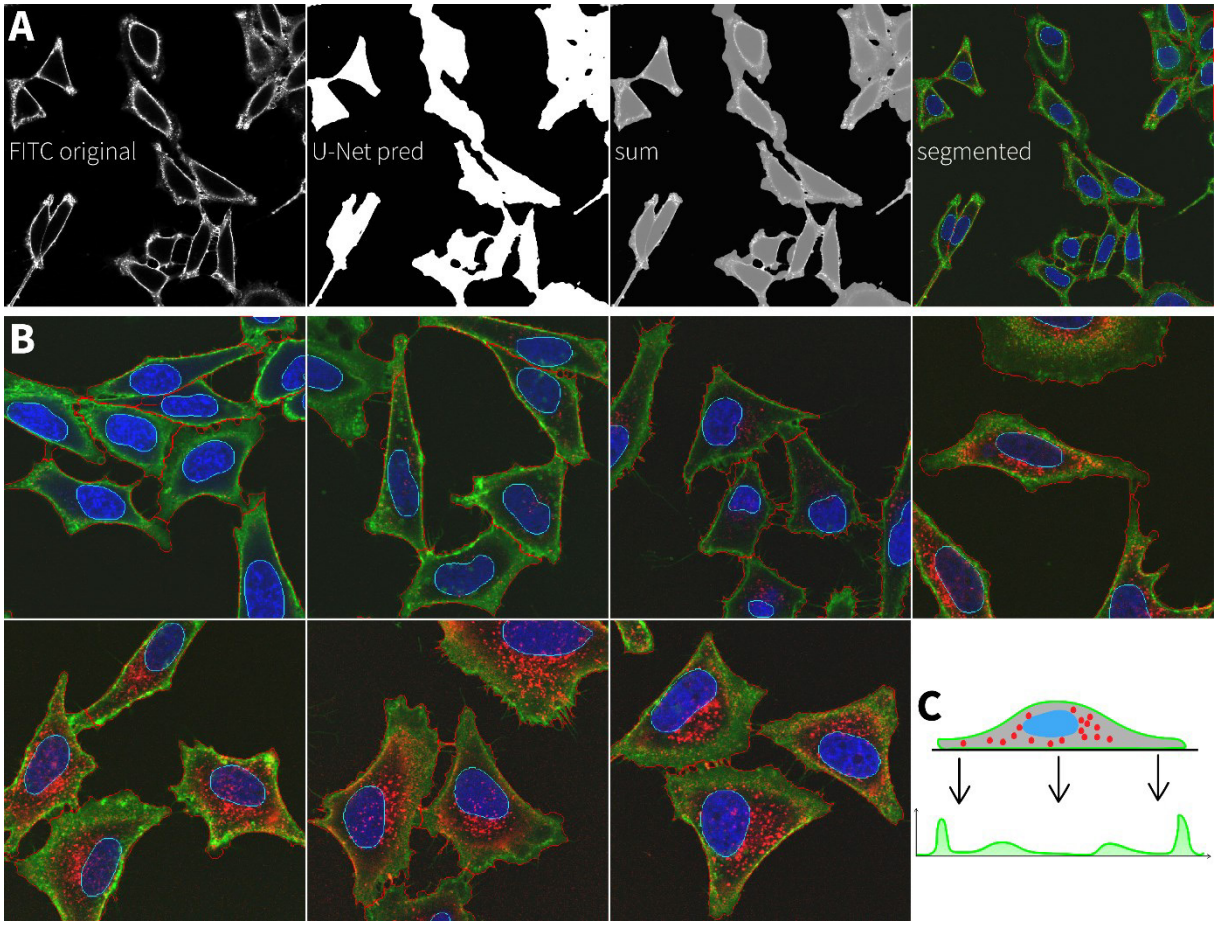


Supplementary Figure S8. TTA (test-time augmentation) pipeline. Training images have foreground-background masks in case of U-Net, while labelled instances for Mask R-CNN. Test images are augmented prior to prediction by flipping and rotating them, then each augmented version is predicted by the given DL model (U-Net or Mask

R-CNN) and finally, predictions are combined to yield a single predicted segmentation for each test image. Figure is adapted from [96].



Supplementary Figure S9. High-content screening pipeline. A) Wet lab and microscopy. The biologist prepares the sample, adds reagents, compounds and labels to the cells, and pipettes them to separate wells of multi-well plates by treatments. Highly automated robotic pipetting is available in most high-content screening laboratories. After proper incubation and treatment, the plates are acquired in a high-content microscope automatically. B) The acquired images are pre-processed as a quality control step to ensure the reliability of further analysis steps. Those burdened with severe focus, noise or staining issues (such as large areas occluded by aggregated stain or very out-of-focus) are excluded from the image set, while illumination pattern can be corrected via e.g. CIDRE [85]. C) Segmentation of appropriate cellular compartments is usually based on the nucleus identification as a first step. Cell lines are typically labelled for the nucleus with a separate dye to enable easy and accurate cell detection. Further compartments e.g. cytoplasm, membrane, mitochondria, lipid droplets, encapsulated virus particles etc. may be segmented using the nuclei as seed objects (in classical algorithms such as propagation). DL methods can also be used to segment all these compartments much more efficiently and accurately. D) Features are extracted from the image regions corresponding to the identified cellular compartments each; these usually include intensity-, morphology- and texture features that describe the appearance of cells distinctively enough for automatic phenotyping (in E). E) Software is used to automatically classify (determine the phenotype of) cells using the feature vectors obtained in D. Expert knowledge is usually required to provide examples of each phenotype (except for unsupervised classification via e.g. *k*-means clustering). Classification software generally create statistics and graphs about the dataset for further data analysis. Such a software is e.g. ACC (Advanced Cell Classifier) [99]. We note that an appropriate DL method might be used to cover steps B-E in a single prediction (segmentation and classification together, as in e.g. Mask R-CNN [27]). Figure is partially adapted from [111].



Supplementary Figure S10. Nucleus and cytoplasm segmentation applied in the peptide carrier-antibody cargo analysis project. A) Processing steps for cytoplasm segmentation. Cytoplasms (red contours) were detected with a threshold-based algorithm on a U-Net-enhanced version of the FITC (green) channel images marking the membrane. The weighted sum of the U-Net prediction (pixel probability map) and the FITC channel image served as input to cytoplasm segmentation. B) Example image crops with segmented nuclei (cyan contours) and cytoplasm outlines on images from increasing dosage of treatment – indicated by the increased number and intensity of antibody cargos (red spots). nucleAIzer was used to segment nuclei accurately and robustly on the Hoechst (blue) channel. Cargo intensity (red) was quantified in the cytoplasm. C) Schematic representation of the FITC membrane intensity profile observed on images. Original images were courtesy of Norbert Imre (University of Szeged, Hungary).

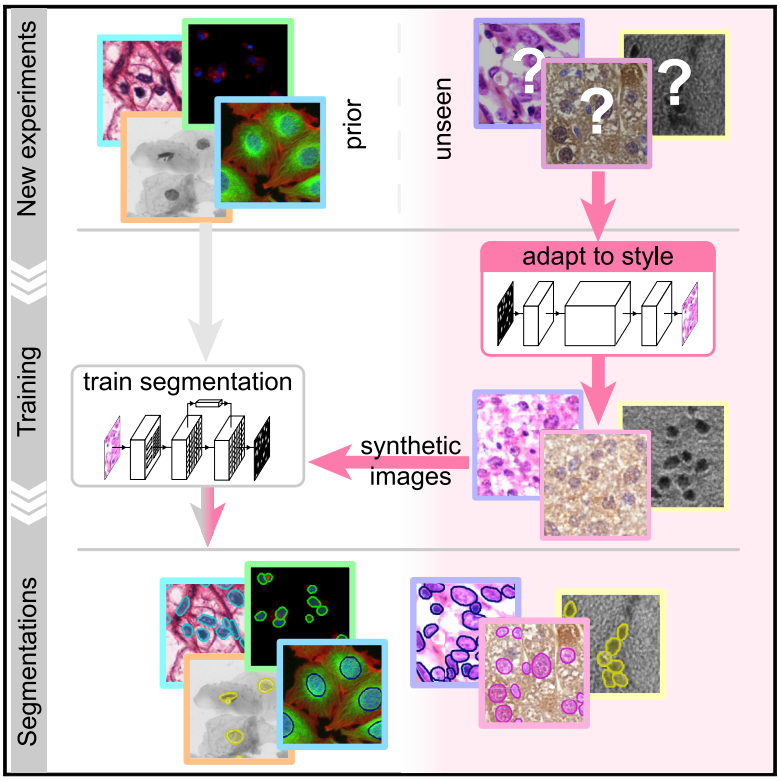
I.

Methods in Brief

Cell Systems

nucleAIzer: A Parameter-free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer

Graphical Abstract



Highlights

- Robust method automatically adapting to various unseen experimental scenarios
- Deep learning solution for accurate nucleus segmentation without user interaction
- Accelerates, improves quality, and reduces complexity of bioimage analysis tasks
- Easy-to-use online tool provided for the method

Authors

Reka Hollandi, Abel Szkalisity, Timea Toth, ..., Anne Elizabeth Carpenter, Kevin Smith, Peter Horvath

Correspondence

horvath.peter@brc.hu

In Brief

Microscopy image analysis of single cells can be challenging but also eased and improved. We developed a deep learning method to segment cell nuclei. Our strategy is adapting to unexpected circumstances automatically by synthesizing artificial microscopy images in such a domain as training samples.

Hollandi et al., 2020, Cell Systems *10*, 1–6
May 20, 2020
https://doi.org/10.1016/j.cels.2020.04.003

CellPress

Please cite this article in press as: Hollandi et al., nucleAIzer: A Parameter-free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer, Cell Systems (2020), https://doi.org/10.1016/j.cels.2020.04.003

Cell Systems

CellPress
OPEN ACCESS

Methods in Brief

nucleAIzer: A Parameter-free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer

Reka Hollandi,[1] Abel Szkalisity,[1] Timea Toth,[1,2] Ervin Tasnadi,[1,3] Csaba Molnar,[1,3] Botond Mathe,[1] Istvan Grexa,[1,4] Jozsef Molnar,[1] Arpad Balind,[1] Mate Gorbe,[1] Maria Kovacs,[1] Ede Migh,[1] Allen Goodman,[6] Tamas Balassa,[1,5] Krisztian Koos,[1] Wenyu Wang,[7] Juan Carlos Caicedo,[6] Norbert Bara,[1,8] Ferenc Kovacs,[1,8] Lassi Paavolainen,[7] Tivadar Danka,[1] Andras Kriston,[1,8] Anne Elizabeth Carpenter,[6] Kevin Smith,[9,10] and Peter Horvath[1,7,8,11,*]

[1]Synthetic and Systems Biology Unit, Hungarian Academy of Sciences, Biological Research Center (BRC), Temesvári körút 62, Szeged 6726, Hungary

[2]Doctoral School of Biology, University of Szeged, Közép fasor 52, Szeged 6726, Hungary

[3]Doctoral School of Computer Science, University of Szeged, Árpád tér 2, Szeged 6720, Hungary

[4]Doctoral School of Interdisciplinary Medicine, University of Szeged, Koranyi fasor 10, Szeged 6720, Hungary

[5]Doctoral School of Informatics, Eötvös Loránd University, Pázmány Péter sétány 1/C, Room 2.317, Budapest 1117, Hungary

[6]Imaging Platform, Broad Institute of Harvard and MIT, 415 Main Street, Cambridge, MA 02142, USA

[7]Institute for Molecular Medicine Finland (FIMM), University of Helsinki, Tukholmankatu 8, Helsinki 00014, Finland

[8]Single-Cell Technologies Ltd, Szeged 6726, Hungary

[9]KTH Royal Institute of Technology, School of Computer Science and Communication, Lindstedtsvägen 3, Stockholm 10044, Sweden

[10]Science for Life Laboratory, Solna, Sweden

[11]Lead Contact

*Correspondence: horvath.peter@brc.hu

https://doi.org/10.1016/j.cels.2020.04.003

SUMMARY

Single-cell segmentation is typically a crucial task of image-based cellular analysis. We present nucleAIzer, a deep-learning approach aiming toward a truly general method for localizing 2D cell nuclei across a diverse range of assays and light microscopy modalities. We outperform the 739 methods submitted to the 2018 Data Science Bowl on images representing a variety of realistic conditions, some of which were not represented in the training data. The key to our approach is that during training nucleAIzer automatically adapts its nucleus-style model to unseen and unlabeled data using image style transfer to automatically generate augmented training samples. This allows the model to recognize nuclei in new and different experiments efficiently without requiring expert annotations, making deep learning for nucleus segmentation fairly simple and labor free for most biological light microscopy experiments. It can also be used online, integrated into CellProfiler and freely downloaded at www.nucleaizer.org.

A record of this paper's transparent peer review process is included in the Supplemental Information.

INTRODUCTION

Identifying nuclei is the starting point for many microscopy-based cellular analyses, which are widespread in biomedical research. Accurate localization of the nucleus is the basis of a variety of quantitative measurements of important cell functions but is also a first step for identifying individual cell borders, which enables a multitude of further analyses. Until recently, the dominant approaches for this task have been based on classic image processing algorithms (e.g., thresholding and seeded watershed; Carpenter et al., 2006), guided by shape and spatial priors (Molnar et al., 2016). These methods require expert knowledge to properly adjust the parameters, which typically must be retuned when experimental conditions change.

Recently, deep learning has revolutionized an assortment of tasks in image analysis, from image classification (Krizhevsky et al., 2017) to face recognition (Taigman et al., 2014) and scene segmentation (Badrinarayanan et al., 2017). It is also responsible for breakthroughs in diagnosing retinal images (De Fauw et al., 2018), classifying skin lesions with superhuman performance (Esteva et al., 2017), and correcting artifacts in fluorescence images (Weigert et al., 2017). Initial work (reviewed in Moen et al., 2019) indicates that deep learning is effective for nucleus segmentation (Falk et al., 2019; Van Valen et al., 2016; Cui et al., 2018); however, these methods often fail to properly separate touching nuclei well and most importantly lack robustness to unseen domains.

The 2018 Data Science Bowl (DSB) organized by Kaggle, Booz Allen Hamilton, and the Broad Institute challenged participants to push the state of the art in nucleus segmentation. The goal of the challenge was to develop fully automated and robust methods effective in a variety of conditions, including differing

Please cite this article in press as: Hollandi et al., nucleAIzer: A Parameter-free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer, Cell Systems (2020), https://doi.org/10.1016/j.cels.2020.04.003

cell lines, treatments, and types of light microscopy. The challenge attracted thousands of data scientists from around the world. Approaches using deep learning dominated the competition, achieving scores that shattered what was previously possible: the best performing traditional methods we submitted ranked no higher than 1,000 out of 3,891 submissions in stage 1 (data not shown); even classical methods hand-tuned to five subsets of the testing data were beaten by 85 out of 739 submissions in stage 2 testing (Caicedo et al., 2019b). The top deep-learning-based methods relied on only a handful of different architectures, namely Mask R-CNN, U-Net, and feature-pyramid networks; the factors that participants commonly believed had most influence over their method's ranking were the amount of data, the pre-processing, and methods used to augment the data.

We present here a superior approach we named nucleAIzer, which, unlike the previous best submissions, applies image style transfer (Isola et al., 2017): an image-to-image translation using a pixel-wise mapping from one image to the other that ensures the generated synthetic output image resembles the original as closely as possible. It aims to overcome one of the greatest challenges of deep learning, the extent of the annotated training set. In particular, we address the unsupervised domain adaptation problem in which the target (test) samples are drawn from a different distribution than the labeled training samples, but we have access to some unlabeled samples from the target distribution. We augment the training samples by creating realistic-looking artificial sample images with the texture, coloration, and pattern elements from source images not included in the training set using image style transfer (Figure 1). Combining this with a segmentation network based on Mask R-CNN (He et al., 2017), an instance segmentation and classification network, along with boundary correction using U-Net (Ronneberger et al., 2015), a semantic segmentation network for biomedical images, (Figure S4) and mathematical morphology, our method outperforms all other methods reported on the final DSB leaderboard (post-competition) (Our method achieved the top-score after the competition ended. An early version of our approach placed 27th out of 739 submissions in round 2 of the competition). We also demonstrate that our method outperforms similar baselines on public fluorescent and histology datasets. Our trained model does not require parameter tuning or specialized knowledge to use and can be applied on a wide variety of conditions and imaging modalities.

Our software is open source and freely available (Data S1 at https://github.com/spreka/biomagdsb). Pre-trained networks for DSB, fluorescent, and histology data can be applied to new images via CellProfiler (Data S2 and at https://github.com/CellProfiler/CellProfiler-plugins/blob/master/nucleaizer.py) or through an online interface at www.nucleaizer.org.

Our approach (Figures 1A and S1; STAR Methods) begins by automatically rescaling the images such that nucleus size is approximately uniform, as the performance of the network is improved if the nucleus size is fixed during training and inference (see STAR Methods; Figures S3 and S6). To do this, we estimate the typical nucleus size in the provided images with a Mask R-CNN-based network pretrained on a large set of diverse images with nucleus segmentations and fine-tuned using the provided training data and label masks. The output of this network is an initial segmentation we use to estimate the typical nucleus size. Alternatively, if the typical nucleus size is known a priori, it can be provided manually and the images rescaled accordingly.

Next, to adapt our model to handle a wide variety of cell types, staining methods, and imaging modalities, even those for which no segmentation annotations are available, we augment the training set with an artificially generated set of representative image-label pairs. This is accomplished using image style transfer. Training and inference both begin by automatically clustering training images into similar styles based on their appearance, using *k*-means (see STAR Methods; Figure 1B). For each cluster of similar image types, a style transfer network (Isola et al., 2017) is trained to generate synthetic images of the desired style with nuclei at specified locations. During training, nucleus annotations are used to train the style transfer network; during inference on out-of-domain target images, we use nucleus masks output from the initial segmentation network. After a style transfer network is trained for each image style, we generate a set of artificial nucleus masks representative of the shape, size, and spatial distribution of nuclei belonging to that style. For this, we used ~100,000 manually labeled single nucleus masks from the DSB set. A subset of these nuclei is selected that represent the shape distribution of the original morphologies, and they are placed such that they follow the spatial distribution of the image style (see STAR Methods). With trained style networks and representative nucleus masks in hand, we generate synthetic images in the desired style nearly indistinguishable from real microscopy images (see STAR Methods) with nuclei in locations defined by the artificial masks. The synthetic image-mask pairs make up the augmented dataset; samples are shown in Figures 1B and S7A. The augmented data are added to the training data for the segmentation network and further extended with conventional augmentations (rotation, cropping, intensity stretching, etc., see STAR Methods). For this experiment, we generated 20 synthetic image/mask pairs for each of the 134 style clusters we identified in the final round data.

Finally, the ultimate Mask R-CNN segmentation model is trained on the combined augmented and rescaled training data. All images are adjusted such that the estimated nuclei size is uniform. To refine the segmentations for high pixel-level accuracy, the edges of each detected nucleus are corrected using a U-Net-based model trained on the same data, followed by some mathematical morphology-based post-processing (see STAR Methods). This step may be skipped if such accuracy is unnecessary for the application, for example, if simply counting nuclei.

RESULTS

We evaluated our approach on four different datasets: DSB stage 1, DSB stage 2, our own set of fluorescence microscopy images, and our own set of histology images from various sources (*DSB1, DSB2, fluo*, and *hist*, respectively, details in Table S2). We compare our approach against submissions from other teams on *DSB1* and *DSB2* (nearly 3,000 in stage 1 and 739 in stage 2). As benchmarks, we include the results reported in the first and second positions of the leaderboard, which was frozen

Please cite this article in press as: Hollandi et al., nucleAIzer: A Parameter-free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer, Cell Systems (2020), https://doi.org/10.1016/j.cels.2020.04.003

Cell Systems
Methods in Brief

CellPress
OPEN ACCESS

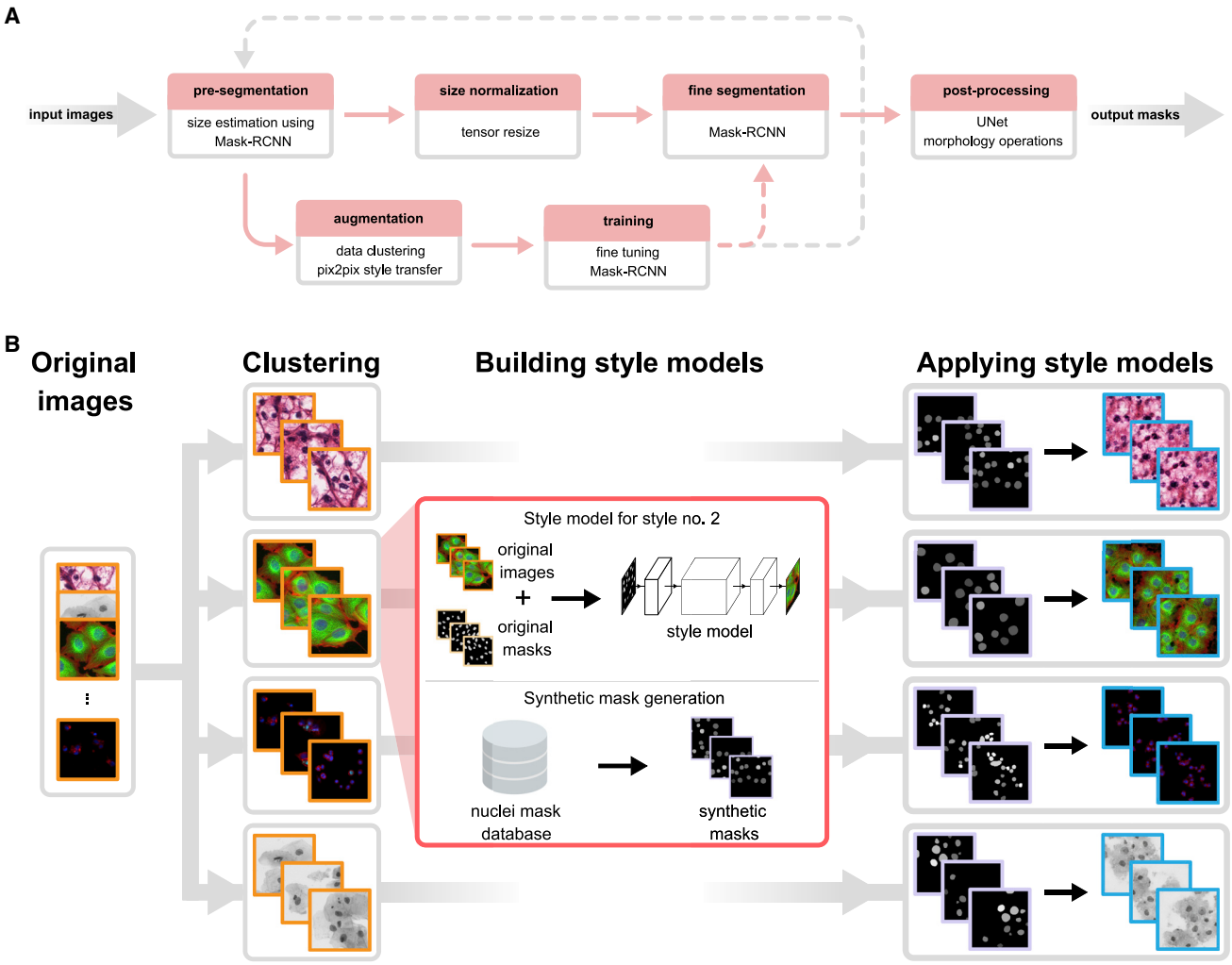


Figure 1. Overview of Our Approach

(A) Upper row of boxes presents the nucleus segmentation and pre-processing; an initial Mask R-CNN network estimates typical nucleus sizes, then images are rescaled such that mean nucleus size is uniform and a Mask R-CNN network trained on images with uniform nucleus size predicts segmentations. A contour refinement step using a U-Net-based network with a morphology operation is applied to obtain the final segmentation result. The data augmentation pipeline is depicted in the bottom row, the training set is augmented with an artificially generated set of image/label pairs in the target domain(s), and a pre-trained Mask R-CNN method is fine-tuned using the augmented images. Augmentation and training steps may be iteratively repeated as the gray dashed line suggests. Upper row depicts the inference pipeline; bottom row, training. Solid lines indicate data flow; dashed lines indicate transfer of a trained model.

(B) Image style-transfer-based data augmentation. To adapt our model to handle out-of-domain image types for which we have no segmentation labels, we synthesize new training data by first clustering images into similar groups, then learn a style transfer model. The style transfer model is provided with simulated nucleus masks, which mimic the number, shape, and size of the unseen nuclei, and then synthetic training image/label pairs are generated using the masks and the style transfer models. These data are added to the standard training data provided to Mask R-CNN, and the network learns to segment nuclei in the new domain. See also Figure S1.

at the close of the competition (https://www.kaggle.com/c/data-science-bowl-2018/leaderboard), a recent deep learning method, unet4nuclei (Caicedo et al., 2019a), which is based on a U-Net (Ronneberger et al., 2015) structure, a widely used Otsu threshold and seeded watershed method with object splitting (Carpenter et al., 2006), the pixel-based classification software ilastik (Sommer et al., 2011), and a more sophisticated but still classical gradient vector flow (GVF) based method, where an active contour is driven to edges using gradient vectors pointing to bright regions (Li et al., 2008) (Figure 2; Table S1; Data S2). Notably, the DSB stage 2 evaluation is performed on an unknown subset of the provided test images, many of which are outside the domain of the training images, truly challenging the ability of the model to generalize. We provide additional benchmarks and variations of our approach for comparison—including how our proposed style transfer learning step improves performance—in STAR Methods and Figure S2. Training a model on the same data with and without style transfer augmentation showed increased accuracy with style.

Our method scores higher (DSB-score, 0.633) than the top ranked deep learning approach (0.631, the highest of 739 teams) on the DSB stage 2 test set and has a simpler

Please cite this article in press as: Hollandi et al., nucleAIzer: A Parameter-free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer, Cell Systems (2020), https://doi.org/10.1016/j.cels.2020.04.003

CellPress
OPEN ACCESS

Cell Systems
Methods in Brief

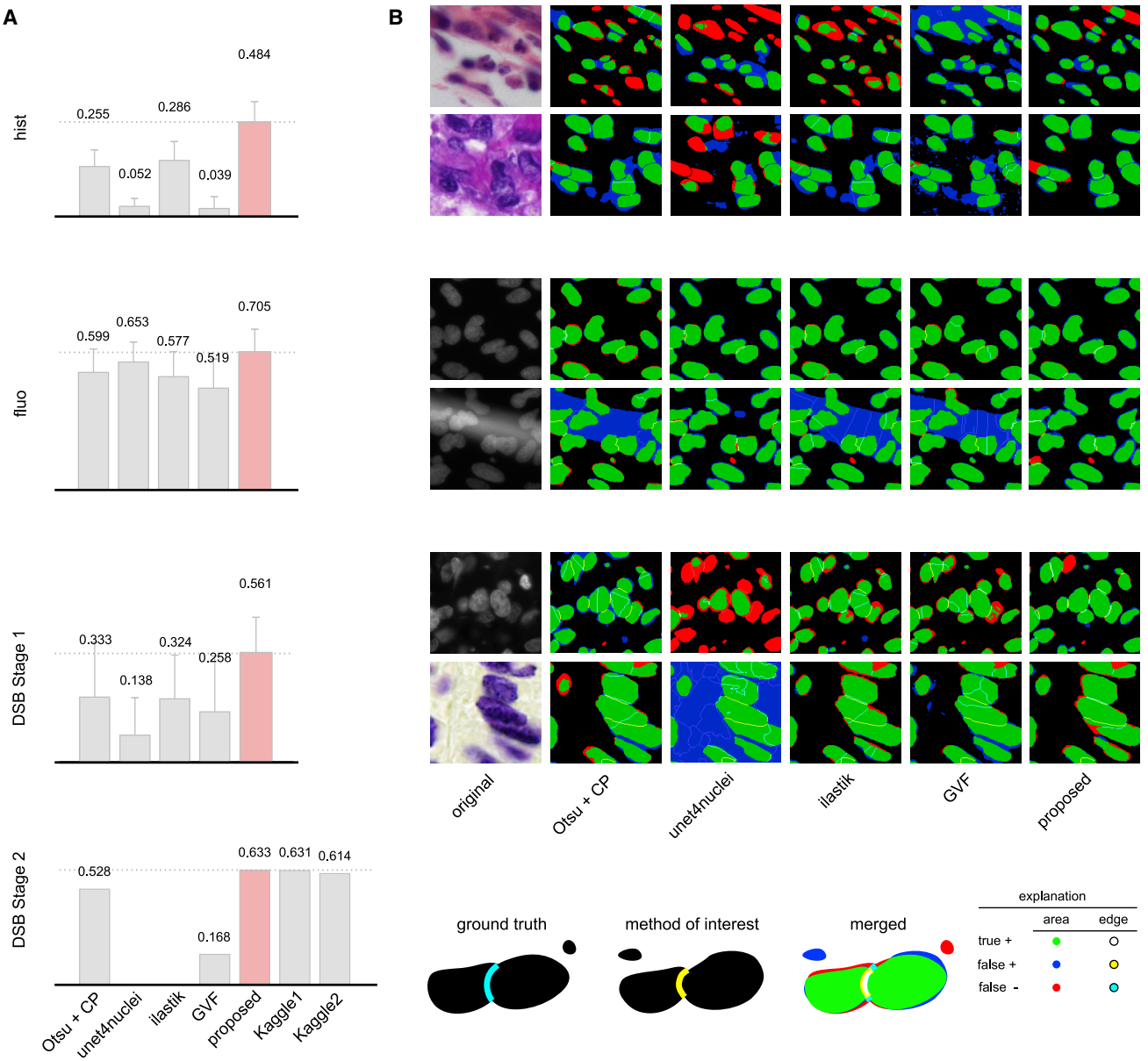


Figure 2. Results

(A) DSB-scores with error bars (standard deviation) for four image sets: hist, fluo, DSB stage 1, and DSB stage 2 (see details in STAR Methods). DSB-score is a modified mean average precision of segmented nuclei (see STAR Methods). Highest scores are marked with dashed lines and red color.

(B) Segmentation results for various methods on sample image crops with difficult cases (two example images of each); rows match those of (A) (note: ground truth is not public for DSB stage 2). A crop of the original image is provided in the first column, followed by segmentation results predicted by various methods. The color coding of the results is explained in the legend at the bottom. See also Figures S2, S5, and S8; Table S1.

architecture with fewer parameters. Our method outperforms all other tested methods, too, including a classical baseline (0.528) (Caicedo et al., 2019b; Carpenter et al., 2006) (Figure 2A). In addition, our proposed method outperformed all prior published results on *hist*, a diverse set of histology images and on *fluo*, a fluorescent image set (BBBC039; Caicedo et al., 2019a) (see Data S1 and S2 for details). A detailed comparison of our results against six other methods evaluated with additional metrics is provided in Table S1; Figures S5 and S8 (see details in STAR Methods).

DISCUSSION

We proposed a deep-learning-based nucleus segmentation approach designed for robustness to new experimental settings, using image style transfer to augment our training data with valuable out-of-domain samples. Our segmentation network learned from these artificially generated image/mask pairs, which mimic the patterns of new data types. This approach helped the network adapt to a diverse set of test data outside the domain of the training data, outperforming every other deep learning

and classical method tested. Our generalized models successfully segment images across several domains, achieving performance close to or matching that achieved by models derived from and applied to a specific domain. The idea of augmenting difficult-to-obtain data using style transfer has enormous potential not only for nucleus detection but also more broadly in applications requiring some form of image understanding.

Key Changes Prompted by Reviewer Comments

The manuscript was extended with the section Segmentation Error Analysis describing both advantages and limitations of our approach compared with other methods, while practical runtime and resource details were also given in section Methods Used for Comparison for training and inference so that the reader might have a better overview of applicability. Specific algorithmic considerations were clarified more extensively, e.g., clustering and image style transfer or post-processing. For context, the complete transparent peer review record is included within the Supplemental Information.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- KEY RESOURCES TABLE
- RESOURCE AVAILABILITY
  - Lead Contact
  - Materials Availability
  - Data and Code Availability
- EXPERIMENTAL MODEL AND SUBJECT DETAILS
  - Kaggle Competition
  - Data
  - Computational environment
  - Related work
- METHOD DETAILS
  - Overview of the Pipeline
  - Training and Style Transfer Data Augmentation
  - Clustering for Style Transfer Learning
  - Inference
- QUANTIFICATION AND STATISTICAL ANALYSIS
  - Evaluation Metrics
  - Methods Used for Comparison
  - Detailed Results

SUPPLEMENTAL INFORMATION

Supplemental Information can be found online at https://doi.org/10.1016/j.cels.2020.04.003.

ACKNOWLEDGMENTS

We acknowledge support from the LENDULET-BIOMAG grant (2018-342) and from the European Regional Development Funds (GINOP-2.3.2-15-2016-00006, GINOP-2.3.2-15-2016-00026, and GINOP-2.3.2-15-2016-00037), the Academy of Finland grant 310552 to L.P. Funding was provided by the National Institute of General Medical Sciences of the National Institutes of Health under MIRA R35 GM122547 to A.E.C. Funding was provided by the Swedish Research Council (Vetenskapsrädet) under research project grant 2017-040609_3 to K.S. We gratefully acknowledge the NVIDIA Corporation for the donation of NVIDIA TITAN Xp GPUs for our research.

AUTHOR CONTRIBUTIONS

R.H., A.S., E.T., C.M., F.K., T.D., A.K., K.S., and P.H. designed the method. R.H., A.S., T.T., E.T., C.M., B.M., I.G., J.M., A.B., M.G., M.K., E.M., T.B., K.K., W.W., J.C.C., N.B., F.K., L.P., T.D., A.K., and P.H. performed annotation, testing, and benchmarking. C.M., K.K., E.T., A.K., R.H., and P.H. designed the online tool. R.H., A.G., K.K., L.P., A.E.C., K.S., and P.H. wrote the manuscript. A.G. and A.E.C. enabled CellProfiler connection and co-organized the 2018 Data Science Bowl.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Received: August 5, 2019
Revised: January 22, 2020
Accepted: April 13, 2020
Published: May 7, 2020

REFERENCES

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). SegNet: a deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. *39*, 2481–2495.

Caicedo, J.C., Goodman, A., Karhohs, K.W., Cimini, B.A., Ackerman, J., Haghighi, M., Heng, C., Becker, T., Doan, M., McQuin, C., et al. (2019a). Nucleus segmentation across imaging experiments: the 2018 Data Science Bowl. Nat. Methods. *16*, 1247–1253.

Caicedo, J.C., Roth, J., Goodman, A., Becker, T., Karhohs, K.W., Broisin, M., Molnar, C., McQuin, C., Singh, S., Theis, F.J., and Carpenter, A.E. (2019b). Evaluation of deep learning strategies for nucleus sgmentation in fluorescence images. Cytometry A *95*, 952–965.

Carpenter, A.E., Jones, T.R., Lamprecht, M.R., Clarke, C., Kang, I.H., Friman, O., Guertin, D.A., Chang, J.H., Lindquist, R.A., Moffat, J., et al. (2006). CellProfiler: image analysis software for identifying and quantifying cell phenotypes. Genome Biol *7*, R100.

Cui, Y., Zhang, G., Liu, Z., Xiong, Z., and Hu, J. (2018). A deep learning algorithm for one-step contour aware nuclei segmentation of histopathological images. arXiv http://arxiv.org/abs/1803.02786.

De Fauw, J., Ledsam, J.R., Romera-Paredes, B., Nikolov, S., Tomasev, N., Blackwell, S., Askham, H., Glorot, X., O'Donoghue, B., Visentin, D., et al. (2018). Clinically applicable deep learning for diagnosis and referral in retinal disease. Nat. Med. *24*, 1342–1350.

Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M., and Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Nature *542*, 115–118.

Falk, T., Mai, D., Bensch, R., Çiçek, Ö., Abdulkadir, A., Marrakchi, Y., Böhm, A., Deubner, J., Jäckel, Z., Seiwald, K., et al. (2019). U-net: deep learning for cell counting, detection, and morphometry. Nat. Methods *16*, 67–70.

Frank, E., Hall, M.A., and Witten, I.H. (2016). https://waikato.github.io/weka-wiki/citing_weka/.

He, K., Gkioxari, G., Dollar, P., and Girshick, R. (2017). Mask R-CNN. IEEE International Conference on Computer Vision (ICCV) *2017*, 2980–2988.

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M.M.A., Yang, Y., and Zhou, Y. (2017). Deep learning scaling is predictable, empirically. arXiv https://arxiv.org/abs/1712.00409.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A.A. (2017). Image-to-image translation with conditional adversarial networks IEEE Conference on Computer Vision and Pattern Recognition, *2017* (CVPR)), pp. 5967–5976.

Krizhevsky, A., Sutskever, I., and Hinton, G.E. (2017). ImageNet classification with deep convolutional neural networks. Commun. ACM *60*, 84–90.

Lehmussola, A., Ruusuvuori, P., Selinummi, J., Huttunen, H., and Yli-Harja, O. (2007). Computational framework for simulating fluorescence microscope images with cell populations. IEEE Trans. Med. Imaging *26*, 1010–1016.

Li, G., Liu, T., Nie, J., Guo, L., Chen, J., Zhu, J., Xia, W., Mara, A., Holley, S., and Wong, S.T. (2008). Segmentation of touching cell nuclei using gradient flow tracking. J. Microsc. *231*, 47–58.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C.L. (2014). Microsoft COCO: common objects in context. Lect. Notes Comput. Sci. 740–755.

Moen, E., Bannon, D., Kudo, T., Graf, W., Covert, M., and Van Valen, D. (2019). Deep learning for cellular image analysis. Nat. Methods *16*, 1233–1246.

Molnar, C., Jermyn, I.H., Kato, Z., Rahkama, V., Östling, P., Mikkonen, P., Pietiäinen, V., and Horvath, P. (2016). Accurate morphology preserving segmentation of overlapping cells based on active contours. Sci. Rep. *6*, 32412.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: convolutional networks for biomedical image segmentation. Lect. Notes Comput. Sci. 234–241.

Van Valen, D.A., Kudo, T., Lane, K.M., Macklin, D.N., Quach, N.T., DeFelice, M.M., Maayan, I., Tanouchi, Y., Ashley, E.A., and Covert, M.W. (2016). Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments. PLoS Comput. Biol. *12*, e1005177.

Weigert, M., Schmidt, U., Boothe, T., Müller, A., Dibrov, A., Jain, A., Wilhelm, B., et al. (2017). Content-aware image restoration: pushing the limits of fluorescence microscopy. Nat Methods *15*, 1090–1097.

Weng, L. (2017). Object detection for dummies part 3: R-CNN family. https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html.

Sommer, C., Straehle, C., Kothe, U., and Hamprecht, F.A. (2011). Ilastik: interactive learning and segmentation toolkit IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 2011, 230–233.

Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). DeepFace: closing the gap to human-level performance in face verification IEEE Conference on Computer Vision and Pattern Recognition, 2014, 1701–1708.

Please cite this article in press as: Hollandi et al., nucleAIzer: A Parameter-free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer, Cell Systems (2020), https://doi.org/10.1016/j.cels.2020.04.003

Cell Systems
Methods in Brief

CellPress
OPEN ACCESS

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Software and Algorithms		
Code repository	This manuscript	https://github.com/spreka/biomagdsb
NucleAIzer online tool	This manuscript	www.nucleaizer.org
CellProfiler plugin	This manuscript	https://github.com/CellProfiler/CellProfiler-plugins/blob/master/nucleaizer.py

RESOURCE AVAILABILITY

Lead Contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the Lead Contact, Peter Horvath (horvath.peter@brc.hu).

Materials Availability

This study did not generate new unique reagents.

Data and Code Availability

The authors declare that the data supporting the findings of this study are available within the paper and its Supplemental Information files.

The authors also declare that the software supporting the findings of this study are available within the paper, its Supplemental Information files, under www.nucleaizer.org, and https://github.com/spreka/biomagdsb.

EXPERIMENTAL MODEL AND SUBJECT DETAILS

Kaggle Competition

We designed our pipeline to recognize nuclei as accurately as possible in a wide variety of images acquired with different microscopes, under varying imaging conditions with different stains for nuclei of various cell types. This was the challenge set forth in the 2018 Data Science Bowl (DSB) by Kaggle, Booz Allen Hamilton and the Broad Institute. The competition included a preparatory stage 1, to which teams could submit their solutions during a four-month period and a 4-day long stage 2 final scoring period.

Existing nucleus segmentation methods do not generalize well, they perform well only on the limited experimental conditions they are designed or tuned for. The Data Science Bowl was highly successful in the sense that many robust solutions were developed that pushed the state-of-the-art in terms of segmentation performance and insensitivity to image type and quality. Solutions such as ours are now being developed into toolkits for biologists that will accelerate science by improving automation in identifying nuclei.

We participated in the competition in both stages, reaching the top 1% in stage1 and top 4% in stage 2. The presented results are based on further improvements post-competition.

Data

The official dataset for the challenge is composed of a training set and two tests sets, one for each stage. The number of images in each set is 670 (training), 65 (stage 1 test), and 3019 (stage 2 test), stage 1 test masks were released in the second stage. The final evaluation of the teams' performance was measured on a subset of the stage 2 test set (the identity of the subset remained hidden to the competitors). Many of the competitors used additional data besides the provided training data, as this was permitted as long as participants shared their sources on the official competition website (https://www.kaggle.com/c/data-science-bowl-2018). Our annotated training data included 12 additional data sources besides the DSB data, including some data sources annotated by experts in our institution. This extended the total number of training image/mask pairs from 735 to 1,102, and the number of annotated nuclei from 33,814 to 80,692 (not including the synthetic data). A summary of the data we used is provided in Table S2.

Using style transfer, we augmented our training data with synthetic image/mask pairs generated in the style of k=134 clusters of images from the DSB Stage 2 set, as described in Sections Clustering to Synthesizing new image/mask pairs. This added 2,680 synthetic image/mask pairs to the training data (approximately 263,701 annotated nuclei).

We tested various versions of our method along with several competing methods on four test datasets: *DSB test1*, *DSB test2*, *fluo*, and *hist*. DSB test1 and DSB test2 are heterogeneous test sets from the Kaggle challenge (stage 1 and stage 2). The *fluo* dataset is fluorescence images of U2OS cells in a chemical screen taken from the Broad Bioimage Benchmark Collection

(BBBC039) (Caicedo et al., 2019a). The *hist* dataset is a mixture of histology images collected from the internet and prostate H&E stained slides collected in-house.

A fraction of the histological images manually annotated in our lab were used as test set *hist* (see Supplemental Data). BBBC039 (Caicedo et al., 2019a) images were used to train a fluorescent segmentation model, we refer as *fluo*. The *hist* and *fluo* test sets are disjoint from the respective training data.

We carefully prepared our test sets for evaluation by automatic clustering as follows. Each test set was split into disjoint parts; one was completely held out of all training procedures and solely used for evaluation, while the remaining part served as out-of-domain unannotated data, was clustered by *k*-means and forwarded to style transfer and subsequent training steps.

We collected histopathology images of test set *hist* intentionally from such experiments that lacked similar instances in our entire training set to test how well our approach would perform on various out-of-domain experiments. Hence, only style transfer learning could be used to input these missing domains' information to our segmentation network.

All input images were initially converted to 8-bit 3-channel RGB images in.png format as well as images produced by our pipeline (except masks).

Computational environment

Software

Our pipeline is implemented using a shell script to allow continuous execution of the entire pipeline. Python 3 scripts execute the training and inference of Mask R-CNN, U-Net, and pix2pix which rely on the TensorFlow, Keras, and PyTorch environments. The clustering, post-processing, and initial steps of style transfer are implemented in Matlab. Our software is available for download at: https://github.com/spreka/biomagdsb where a detailed documentation can also be found discussing the required versions of frameworks and details about the architecture parameters.

The entire pipeline can be run both under Linux and Windows. In a typical use case, it is not necessary to retrain any of the models. Calling the *postComp* method without post processing provides excellent results. For specific experiments with no ground truth annotations, performing the style transfer learning part of our pipeline generates new synthetic training data in the missing domain on which training a new model results in fine segmentation. Alternatively, an online version of our method is available at www.nucleaizer.org.

Hardware

Our methods were trained and tested on a variety of Nvidia graphics cards, including GTX 1070, 1080Ti, and Titan Xp.

Related work

Mask R-CNN

He et al. (2017) published Mask R-CNN as an extension of Faster R-CNN to allow simultaneous instance detection and segmentation. The network architecture is similar to that of Faster R-CNN: feature extraction uses ResNet (50 or 101 layers) or alternatively Feature Pyramid Network (FPN), while head is as in Faster R-CNN extended with a fully convolutional mask prediction branch. A detailed discussion of extended R-CNN versions can be found in Weng, 2017.

We decided to incorporate Mask R-CNN in our pipeline due to its robustness, scalability and instance-awareness. It is currently one of the leading computational architectures in instance segmentation of arbitrary object classes, and its applications dominated the methods submitted to the DSB 2018 competition alongside solutions based on U-Net.

U-Net

U-Net (Ronneberger et al., 2015) was specifically created for bioimage segmentation with an encoder-decoder architecture and skip connections between layers of the encoding branch and decoding branch to provide the decoder with access to spatial information to reason about upsampling the segmentation.

We applied U-Net in our post-processing pipeline as it can efficiently be used to detect subtle differences such as those around the edges of objects. The network structure is straightforward and computationally feasible.

Post-processing the segmented nuclei per se is needed due to the inevitable uncertainty in marginal cases, like relatively small objects most likely corresponding to false detections. We found probability maps predicted by U-Net helpful in such scenarios.

METHOD DETAILS

Overview of the Pipeline

As a first step, pre-segmentation of the input images is performed using a pre-trained deep convolutional model (which we refer as *preseg*) to estimate nuclei sizes as well as to create a mask input for image style transfer learning. Simultaneously, we cluster similar images of the input data into groups, and learn styles on these clusters (see Figure 1B and sections Clustering for Style Transfer Learning and Learning Image Style Transfer Models for details). As a next step, we extend the training data with artificially created style transferred images for fine-tuning a Mask R-CNN (He et al., 2017) pre-trained on our nucleus segmentation dataset. For inference on unseen data, we use the refined Mask R-CNN network incorporating knowledge about estimated cell sizes. The resulting contours are refined with U-Net (Ronneberger et al., 2015) and a morphology step.

The proposed method consists of procedures for training and inference, as shown in Figure S1. Inference merely requires unannotated images as its input – provided the pre-trained models are available. Training the network produces a learned segmentation

Please cite this article in press as: Hollandi et al., nucleAIzer: A Parameter-free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer, Cell Systems (2020), https://doi.org/10.1016/j.cels.2020.04.003

model, and requires a set of annotated training data and a pre-trained segmentation network (pre-segmentation network), as well as any available unannotated images that can be used for data augmentation. The pretrained segmentation network is crucial to both the training and inference procedures, so we discuss it first and then continue with training and prediction steps.

Training and Style Transfer Data Augmentation

Pre-segmentation

The architecture for the segmentation networks is based on the Mask R-CNN architecture. The pretrained segmentation network (pre-segmentation network) is used to make rough estimates about the nucleus size and shape while being robust to changes in imaging modality or magnification. The network is initialized with pretrained weights from the MS-COCO dataset, which contains images and segmentation masks for 91 object types including people, trucks, sheep, dogs, etc. For details about the original COCO competition see http://cocodataset.org or the corresponding publication (Lin et al., 2014). The network was trained using a diverse set of annotated images containing various imaging modalities, cell lines, magnifications, etc. For more information see Section Data. The annotations consisted of segmentation masks for the nuclei. Augmentation was used during training including geometric transformations, intensity stretching, cropping, noise, and blur (see Data S1 documentation for details).

The resulting network, which we refer to as *preseg*, already performed reasonably well on unannotated images in the test set (Figure S2), although this was not its purpose. The preseg network is used to: estimate properties of nuclei in new unannotated images (size, shape, and area) in clustering, and to generate rough segmentations on unannotated images for the style transfer data augmentation step (see the following two sections for details).

Clustering for Style Transfer Learning

Images without annotations are automatically clustered to define multiple groups with similar properties: textures, imaging modalities, cell lines, sample type (tissue or culture), etc. These groups are used as data sources to learn style transfer models to generate additional synthetic data that mimics the properties of each cluster of unannotated images.

To perform the clustering, we use a pairwise similarity metric between feature vectors describing each unannotated image. Features were extracted using CellProfiler (Carpenter et al., 2006) modules including intensity and texture and a similarity metric was computed by a shallow fully connected neural network (Frank et al., 2016). This similarity network was trained on the DSB train1 data set, where images taken with the same condition are given a label of 1 and images from different conditions are given a label of 0. The output of this network on the unannotated data yielded a similarity matrix which we clustered with k-means. The number of clusters, k=134 for DSB stage 2 test set, was chosen automatically based on the number of images to over-segment the groups to avoid accidental mixing of the true underlying groups. Ideally, each obtained cluster of unannotated images represents a ''style'' or distribution of data which can be augmented with style transfer (e.g. digital slides of H&E stained breast cancer histology samples at 63x magnification, or fluorescent images of Human MCF7 cell nuclei at 40x).

Learning Image Style Transfer Models

We use the pix2pix (Isola et al., 2017) framework for image style transfer (https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix). The architecture consists of two adversarial networks, a *generator* tasked with synthesizing realistic looking images, and a *discriminator* tasked with identifying real images from synthesized images. This model learns to map one image domain to another through an adversarial loss that encourages the generator to learn to fool the discriminator. The input to the generator in our case is a binary mask containing 1's at the locations of the desired nuclei, and 0's elsewhere. The input of the discriminator is an image/mask pair (either a real pair, or a synthetically generated pair). The generator learns to transform the binary mask into the desired style of the real images from the cluster, and the discriminator encourages this by trying to identify real image/mask pairs from fakes. We use the rough segmentations provided by the *preseg* network as masks for the unannotated images in the style cluster during learning. We train a pix2pix style transfer network to synthesize realistic images from masks for each of the style clusters.

Synthesizing New Image/Mask Pairs

Using our set of 134 trained style transfer networks, we synthesized 20 new image/mask pairs for each of the styles in the unannotated data. A crucial step for this task was to generate novel binary masks to provide as input to the style transfer network, which uses the mask to generate a realistic image of the cells with nuclei in the locations defined in the mask. We generated the masks algorithmically as a combination of 1) fetching real nuclei masks from a database, and 2) synthesizing nuclei using software (simcep; Lehmussola et al., 2007). Approximately 50% of the nucleus masks were created using each approach. In this manner, we generated 20 masks for each of the 134 style clusters, and then used the style transfer network to generate the corresponding images.

We assembled our nucleus mask database from images of the official DSB training set and further external datasets (see Table S2) - some of which we corrected for slight contour errors - and added each nucleus mask to the database. We fetched such nuclei masks that follow the features of the desired style and placed them on the synthetic mask images in accordance with the localization properties of the given style.

Training the Mask R-CNN Segmentation Network

The synthetic image/mask pairs generated by the style transfer network were added to the annotated training data to update the Mask R-CNN segmentation network. We used the implementation of Matterport (https://github.com/matterport/Mask_RCNN) and wrote handler scripts in Python to create the appropriate data structures and call functions. Training was performed in 3 steps with decreasing learning rate and targeted different layers of the Mask R-CNN network, as described in the documentation of the aforementioned Matterport repository.

The loss function was as defined in (He et al., 2017): it comprises of classification, localization and segmentation mask losses: $L=L_{cls}+L_{box}+L_{mask}$ by ROIs, and defines mask loss as follows. Given the *k*-th region does belong to ground truth class *k* it takes the average binary cross-entropy loss which is formulated as

$$L_{mask} = -\frac{1}{m^2}\sum_{1\le i,j\le m}\left[y_{ij}\cdot\log\hat{y}_{ij}^{k} + (1-y_{ij})\log\left(1-\hat{y}_{ij}^{k}\right)\right] \quad \text{(Equation 1)}$$

where y_{ij} is the true label of a cell (*i,j*) from a ROI of *mxm* size on the ground truth mask of class *k* and $\hat{y}_{ij}^{k}$ is the predicted class label of the same cell. The formula only includes masks for ground truth class *k* that are associated with the *k*-th class.

Image Augmentation and Resizing

The performance of deep learning networks is known to scale with the size of the dataset (Hestness et al., 2017). Therefore, we use a number of approaches to augment the training data. The first, as we described above, is to add new synthetic image/mask pairs generated in the style of unseen examples to the existing annotated training data. Each minibatch contained 10-50% synthetic images. We also used standard data augmentation techniques including random cropping, colour channel swapping, intensity modification by histogram stretching or equalization and inversion, rotation to an arbitrary degree and translation as geometric transformations and finally, to better resemble low-quality images, blur and additive noise were used as well. These operations were applied to all the input training data – style transfer results too – with a random probability.

MASK R-CNN is reasonably robust to changes in scale, but superior performance is obtained if the nucleus size is approximately 40 pixels in diameter for the data and parameters we used. Figure S3 shows the results of the robustness of our method with a fixed parameter against different nuclei sizes. Quantitative evaluation is shown in Figure S6.

Another preprocessing step was to resize the images by a scaling factor to obtain a training dataset homogeneous both in cell and image size. The scaling factors were computed from the size estimation of the *preseg* nucleus masks such that the resulting mean cell size is set to 40 pixels diameter. Images were then either cropped or padded so that the resulting image was 512 x 512 pixels.

Inference

Mask R-CNN Prediction

The Mask R-CNN model trained as described above is used to predict segmentation masks when new images are provided as input. The images are resized before they are input to the network as described in the previous section.

Post-processing and U-Net Correction

We found that the segmentations could be further improved by postprocessing and refining nucleus contours using U-Net (Ronneberger et al., 2015). This encouraged better boundary reasoning between adjacent nuclei, and finer segmentations with the background. First, outlier objects were removed or merged as follows: 1) Smaller objects that were entirely within another object were eliminated. 2) objects that were surrounded by another object more than p_1% were merged, and 3) objects smaller than p_2 pixels area were removed. Next, U-Net based correction was performed (Figure S4): 1) an optimal threshold p_3 for U-Net probability values was determined, 2) a soft margin around the Mask R-CNN contour was defined for each object, with an extension of p_4 pixels inwards and p_5 outwards. The contour was extended/shrunk based on the U-Net predictions. 3) objects that had in total less than p_6 mean U-Net probability were removed. Parameters $p_1..p_6$ were optimized on the training set with a genetic algorithm to the DSB-score function (see formulation in section Evaluation Metrics). Best values were: (0.17, 44, 0.9375, 1, 1, 0.8).

QUANTIFICATION AND STATISTICAL ANALYSIS

Evaluation Metrics

The evaluation metric used for the DSB competition is based on the mean average precision, as defined on the competition website, at different intersection-over-union (IoU) thresholds. A successful nucleus detection was determined by an IoU test (also known as the Jaccard index):

$$IoU(x,y) = \frac{|x\cap y|}{|x\cup y|} = \frac{|x\cap y|}{|x|+|y|-|x\cap y|} \quad \text{(Equation 2)}$$

which measures the overlap between prediction pixels *x* and the annotation pixels *y* over the intersection of the two areas. Using a threshold ranging from 0.5 to 0.95 with steps of 0.05, true positive (*TP*) detections, false positive (*FP*) detections and false negative (*FN*) detections were identified. For a threshold of 0.5, a predicted object is considered a "hit" if the IoU is greater than 0.5. For each threshold *t*, a modified version of precision was calculated

$$DSB\ score(t) = \frac{TP(t)}{TP(t)+FP(t)+FN(t)+\varepsilon} \quad \text{(Equation 3)}$$

for all thresholds in (0.5, 0.95). These scores were averaged for all thresholds, and then the mean of the average scores is reported over the images in the test dataset. In addition to the DSB-score, we evaluated our results with three additional metrics based on the IoU detection test: mean average precision- (mAP), recall and F1-score. We used the same *t*, TP, FP and FN values as above. We also added a small $\varepsilon = 10^{-40}$ value to the denominators.

Please cite this article in press as: Hollandi et al., nucleAIzer: A Parameter-free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer, Cell Systems (2020), https://doi.org/10.1016/j.cels.2020.04.003

$$precision(t) = \frac{TP(t)}{TP(t) + FP(t) + \varepsilon} \quad \text{(Equation 4)}$$

$$recall(t) = \frac{TP(t)}{TP(t) + FN(t) + \varepsilon} \quad \text{(Equation 5)}$$

$$F1\ score(t) = 2 \cdot \frac{precision(t) \cdot recall(t)}{precision(t) + recall(t) + \varepsilon} \quad \text{(Equation 6)}$$

The same strategy was used to calculate mean values for these measures as was for the DSB-score, taking the average over various thresholds *t*, and the mean among the test images. In the following sections, we refer to these measures as mAP (mean average precision), mAR (mean average recall), and mF1 (mean average F1-score).

We also introduce classification accuracy regarding our style-transfer generated image quality evaluation as follows:

$$accuracy = \frac{\sum correctly\ classified\ instances}{\sum instances} \quad \text{(Equation 7)}$$

Methods Used for Comparison

Our tests included several variations of our method along with six competing methods and several variations of our approaching using different style augmentation: *NOstyle* did not contain style augmented images, *AUTOstyle* used nuclei masks generated by the preseg network, and *GTstyle* used hand annotated ground truth to generate nuclei masks. CellProfiler (*CP*) (Carpenter et al., 2006) is a widely-used bioimage analysis software incorporating several methods to segment, measure and analyze cellular compartments. We created multiple pipelines for the different image types of the test sets – except for our fluorescent set which comprised of a single experiment. *Preseg* refers to our general scale-independent pre-segmentation model while *postComp* is our final refined post-competition submission (an *AUTOstyle* model customized for DSB test2).

We compared against several other approaches including *ilastik* (Sommer et al., 2011), which provides a pixel classification setup where users can manually annotate regions of the input images to desired classes and obtain predictions as either probability maps or segmented images. Segmentations were obtained by applying a threshold to probabilities from *ilastik* (with additional object splitting). *Unet4nuclei* (Caicedo et al., 2019a) is an implementation of the popular U-Net deep learning approach to segmentation. *GVF* (Li et al., 2008), or gradient vector flow, is an active contour-based segmentation method suitable if objects are bright regions on a dark background. Pipelines of these compared methods are provided in Data S2. *DSB1* and *DSB2* are the first and second place entries on the final Kaggle leaderboard. The approach from *DSB1* (https://www.kaggle.com/c/data-science-bowl-2018/discussion/54741) uses a very deep U-Net architecture along with prediction of touching borders. *DSB2* also uses a U-Net approach, and forces the network to predict relative locations within each nucleus (https://github.com/jacobkie/2018DSB).

Comparing the complexity as well as the computation time and resources needed to train *DSB1*, we are confident to claim that our method is considerably simpler and much faster. *DSB1* combines a total of 32 trained deep neural networks to achieve their reported score on *DSB test2* set, the training of which can take days even when performed on a high computation-capable GPU (Nvidia GTX 1080Ti). In contrast, in our method only a Mask R-CNN and U-Net models are trained for prediction, taking approximately 10 hours for training on the same GPU. The computation time for image style transfer strictly depends on the number of different styles present in the target data as one style model is trained for each, individually taking about 15 minutes. *DSB2* uses a simpler architecture.

We also investigated computation time regarding inference with our method. Even though inference time is affected by multiple circumstances including image size, number of objects on the image and VRAM of the GPU used, an approximate one image per 2 seconds can be achieved given the following. An image of 520x696 pixels size having about 120 objects of ~20 pixels median diameter size, rescaled to 2x its original size to have ~40 pixels diameter sized objects, i.e. 1040×1396 pixels resized image, on an Nvidia GTX 1080Ti GPU having 11 GB VRAM can be predicted in 2 seconds.

Detailed Results

Style Transfer Increases Performance

We tested the methods outlined in Section Methods Used for Comparison on four test datasets: *DSB test1*, *DSB test2*, *fluo*, and *hist*, described in Section Data. The resulting DSB-scores are presented in Table S1. When running these tests, the test data was never included in the data to train the model, e.g. when testing on *DSB test1*, the *DSB test1* data was held out from the training set. Similarly, when testing on hist, *biomag2* and *biomag6* subsets were held out.

The test image sets were used as style transfer learning input as determined by our automatic clustering method: a portion of the set was left out when the clustering algorithm could not find a sufficient number of images for a cluster. Therefore, we report our results on such fractions of the test sets that none of the deep learning networks have seen prior to inference as follows. 100/200 *fluo*, 21/50 *hist*, 28/65 *DSB test1* images were used for evaluation. None of the final *DSB test2* evaluation image set was used for training.

Please cite this article in press as: Hollandi et al., nucleAIzer: A Parameter-free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer, Cell Systems (2020), https://doi.org/10.1016/j.cels.2020.04.003

The results demonstrate that our style transfer approach improves performance in test sets containing data from heterogeneous sources: *hist*, *DSB test1* and *DSB test2*. We also see excellent performance on single domain fluorescence data, *fluo*. Comparing the results of our method with (*AUTOstyle* [postComp is the *AUTOstyle* for DSB stage 2 test] and *GTstyle*) and without style transfer augmentation (*NOstyle*), we see a clear trend towards increased performance with style transfer augmentation. If we have access to ground truth nucleus masks (*GTstyle*) our performance improves, though in many realistic scenarios such masks will not be available. Figure S2 shows the output of the various methods we tested on challenging examples (note that *DSB1* and *DSB2* are not reported because we did not have access to their code). In Figure S5, we present mAP, mAR, mF1 and mIoU metrics for the various methods on each dataset. As expected, there is a strong correlation between the metrics.

Objects of Various Sizes can Be Detected Accurately

In addition to the qualitative demonstration on Figure S3, we provide a quantitative analysis of the range of object sizes correctly detected by two of our compared methods: *preseg* and *postComp*. Note that while *postComp* was trained on fixed sized (40 pixels diameter) nuclei images and is expected to perform best on objects of approximately the fixed size, *preseg* is more flexible as we intentionally included images presenting a wide range of object sizes in its training to prepare it for an initial robustness. Therefore we expect *preseg* to detect objects robustly in a wider size range. We tested both models on DSB stage 2 test set and scaled the images to 0.25-4.0 times relative to our generally expected median 40 pixels diameter objects. Our results confirm our expectations of *preseg* (our scale-independent model) which performs significantly better than *postComp* (scaled model) on shrunk images as presented on Figure S6 below. We found that the accuracy of both models is decreased far less rapidly when enlarging the images.

We also note that the object sizes can vary on individual images (Figure S6B) suggesting the scaling procedure by median object sizes cannot necessarily be optimal for all images; we mark some of the extremes with black arrows.

Synthetic Images Are often Mistaken for Real

We tested how well our style transfer-generated synthetic images compared to real microscopy images by showing a representative selection of both to field experts (pathologists and biologists) and asked them to tell the synthetic images apart from the real ones. The only prior information forwarded to the participating experts was there are fake images in the collection. Their decision accuracy was measured in a binary fashion: whether the expert could identify a truly synthetic image (1) or not (0). We show an example test image montage below (Figure S7) with the average detection of experts and the labels (real or fake). We collected 64 cropped images each for our two test image mosaics comprising of 50% real and fake tiles, respectively.

We report an approximate 57% accuracy (ranging from 42% to 73%) of fake image recognition averaging both our experts and the test cases. Based on the performance of the experts we can conclude the visual quality of the style transfer-generated images is on par with real microscopy images suggesting the advance our approach may bring to cellular compartment segmentation.

Segmentation Error Analysis

We visually compare segmentation errors and improvements on Figures 2 and S2. To better understand the distribution of such common errors in any of the analyzed segmentation methods we compared how well they perform in terms of avoiding the main error types: 1) missing a nucleus, 2) falsely detecting an object as nucleus, 3) splitting a nucleus and 4) merging adjacent nuclei unnecessarily. An example image presented on Figure S8A shows them visually. All existing methods fail to overcome these issues in at least some instances, as they significantly depend on the experimental and imaging conditions used to produce the images. Our method aims to help reduce these issues.

We measured such types of errors as follows. 1) a missed nucleus is a false negative (FN) i.e. present on ground truth (GT) with no corresponding object on the prediction. 2) A falsely detected nucleus is a false positive (FP): a predicted object with no corresponding GT. 3) A split nucleus is identified as two or more predicted objects that overlap with a significant region of the best corresponding GT object, respectively; we considered an overlap of at least 30% as significant in this case if two objects contributed to the overlap, and 15% if more. Splits were only considered if the given GT object did not have a single matching predicted object. 4) A merged nucleus is a single predicted object that has a significant overlap with multiple GT objects each. We calculated merges similarly to splits but swapped the role of GT and predicted objects.

We conducted our evaluation on the same subsets of each test set discussed in the previous sections. Quantitative analysis of segmentation errors support our results: our method (and its modified versions) generally outperform the compared methods. Comparative results are displayed on Figures S8C–S8E. Remarkably, *unet4nuclei* produced in total fewer errors than our methods on test set *fluo* but it has been trained and published on this image set.

Segmentation errors naturally occur in automatic methods. Classical methods (*CP*, *ilastik*, *GVF*) tend to predict a higher frequency of false positive objects, typically on complex background regions similar to e.g. Figure S4C. They are also more prone to merging touching nuclei or background regions around them to the objects (see Figure S2B rows 1–2) and to split larger, irregularly shaped objects. *Unet4nuclei* could not have been trained accurately enough for heterogeneous sets (*hist*, *DSB test1*) due to the inevitable uncertainty of U-Net in complex histological regions while it excelled on the single-domain set *fluo*.

Our method typically failed to split (i.e. merged) very small or elongated adjacent nuclei with weak textural difference from the dividing background region. Similarly, it unnecessarily split nuclei in cases where texture or edge information may suggest multiple nuclei-like structures inside a single nucleus.

Cell Systems, Volume *10*

Supplemental Information

nucleAIzer: A Parameter-free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer

Reka Hollandi, Abel Szkalisity, Timea Toth, Ervin Tasnadi, Csaba Molnar, Botond Mathe, Istvan Grexa, Jozsef Molnar, Arpad Balind, Mate Gorbe, Maria Kovacs, Ede Migh, Allen Goodman, Tamas Balassa, Krisztian Koos, Wenyu Wang, Juan Carlos Caicedo, Norbert Bara, Ferenc Kovacs, Lassi Paavolainen, Tivadar Danka, Andras Kriston, Anne Elizabeth Carpenter, Kevin Smith, and Peter Horvath

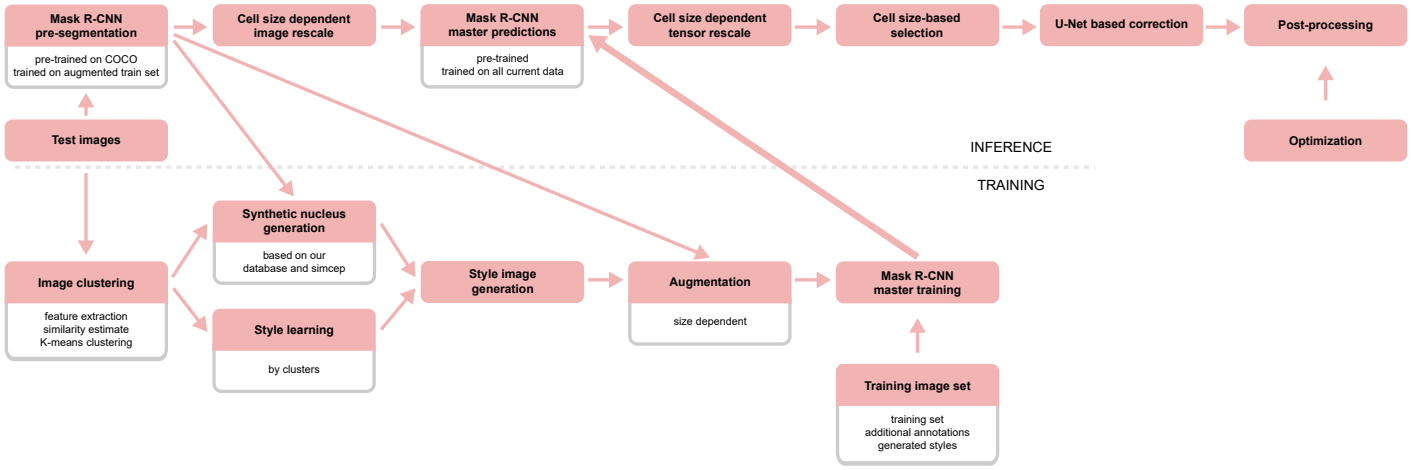


Figure S1. Related to Figure 1. A detailed overview of the proposed pipeline.

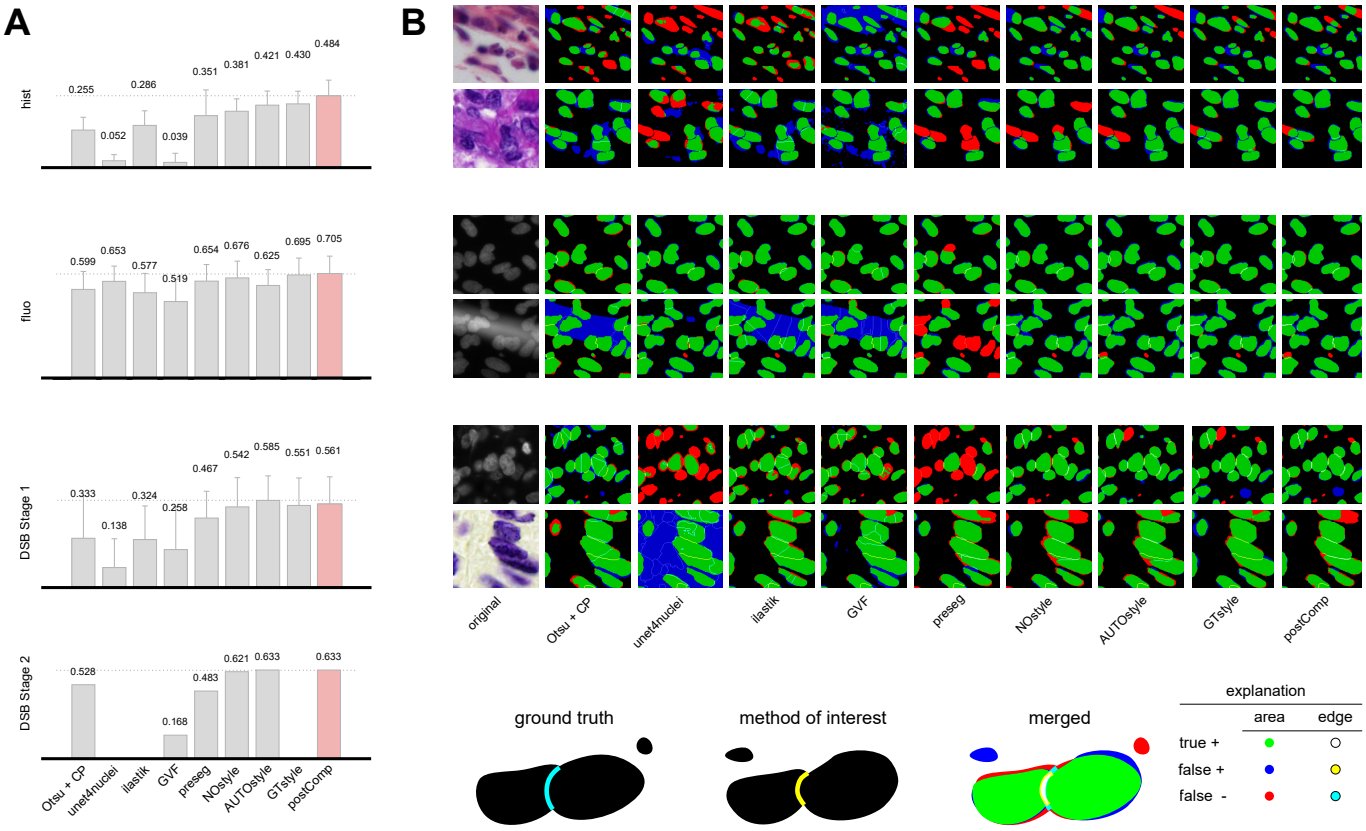


Figure S2. Related to Figure 2. Results from Figure 2 in more detail.
(A) DSB scores with error bars for the image sets tested.
(B) Segmentation results for various methods. preseg, NOstyle, AUTOstyle, GTstyle and postComp refer to variations of our approach. See sections Data and Methods used for comparison for details.

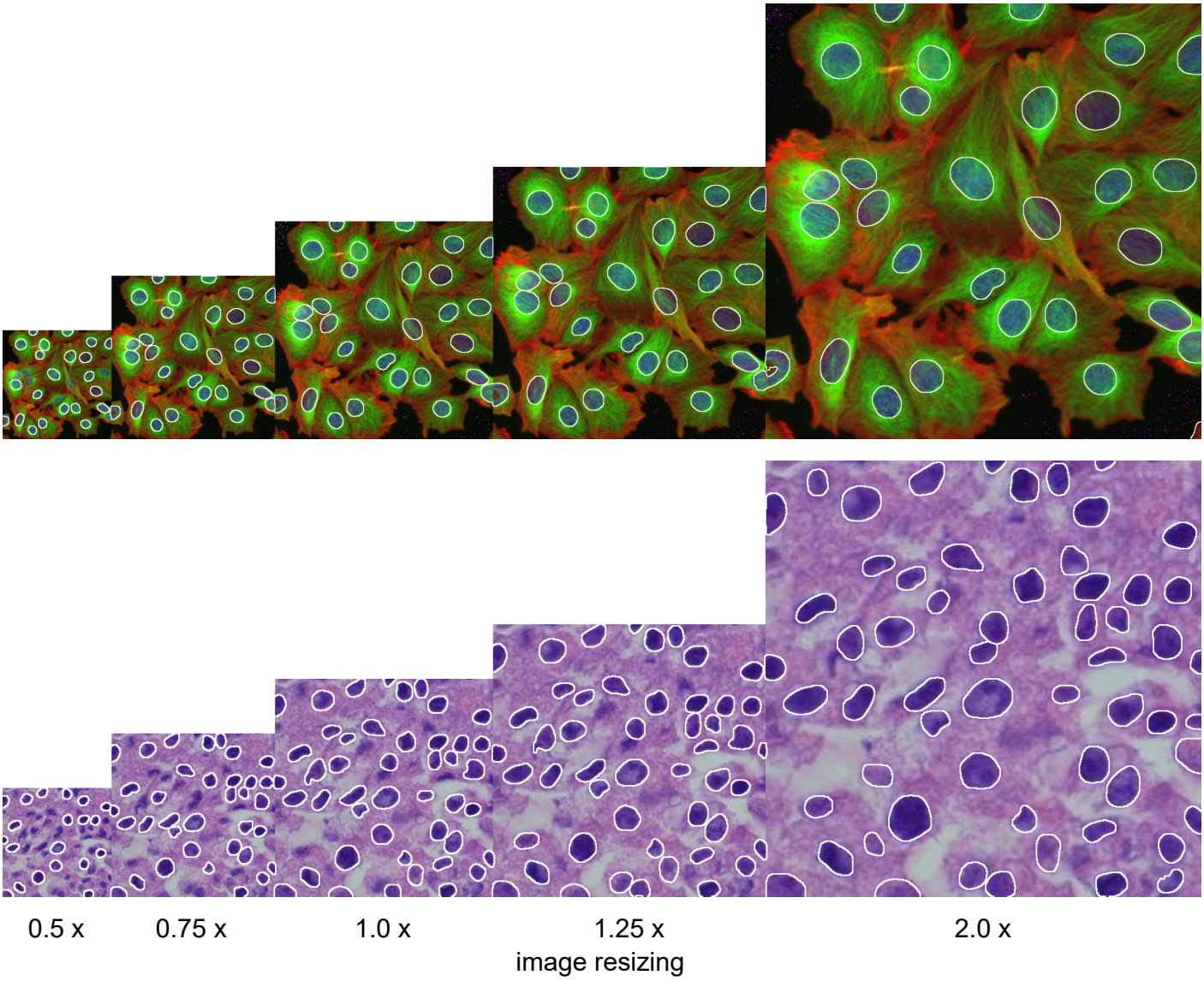


Figure S3. Related to STAR Methods. Mask R-CNN nucleus segmentations are reasonably robust to changes in scale, but we found that the best performance can be achieved if nucleus size is fixed to 40 pixels diameter (at 1.0 scale factor).

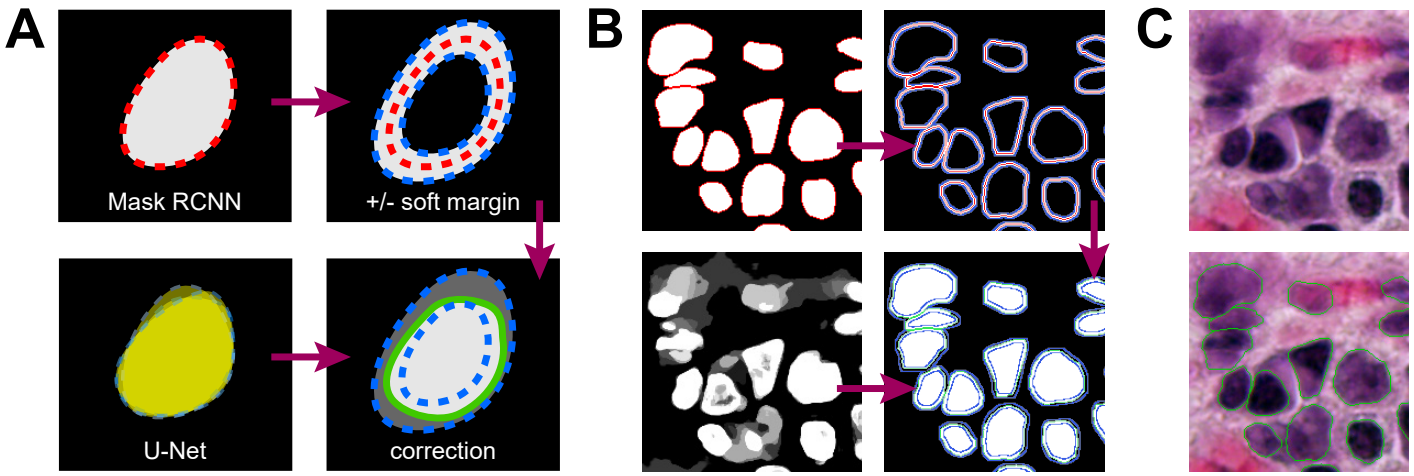


Figure S4. Related to Figure 1 and STAR Methods. Post-processing contour correction with U-Net. Mask R-CNN segmentation contour was refined within a given maximum margin using U-Net segmentation. See Section Post-processing and U-Net correction for details.
(A) Correction steps.
(B) Real example as in (A).
(C) Original image and result.

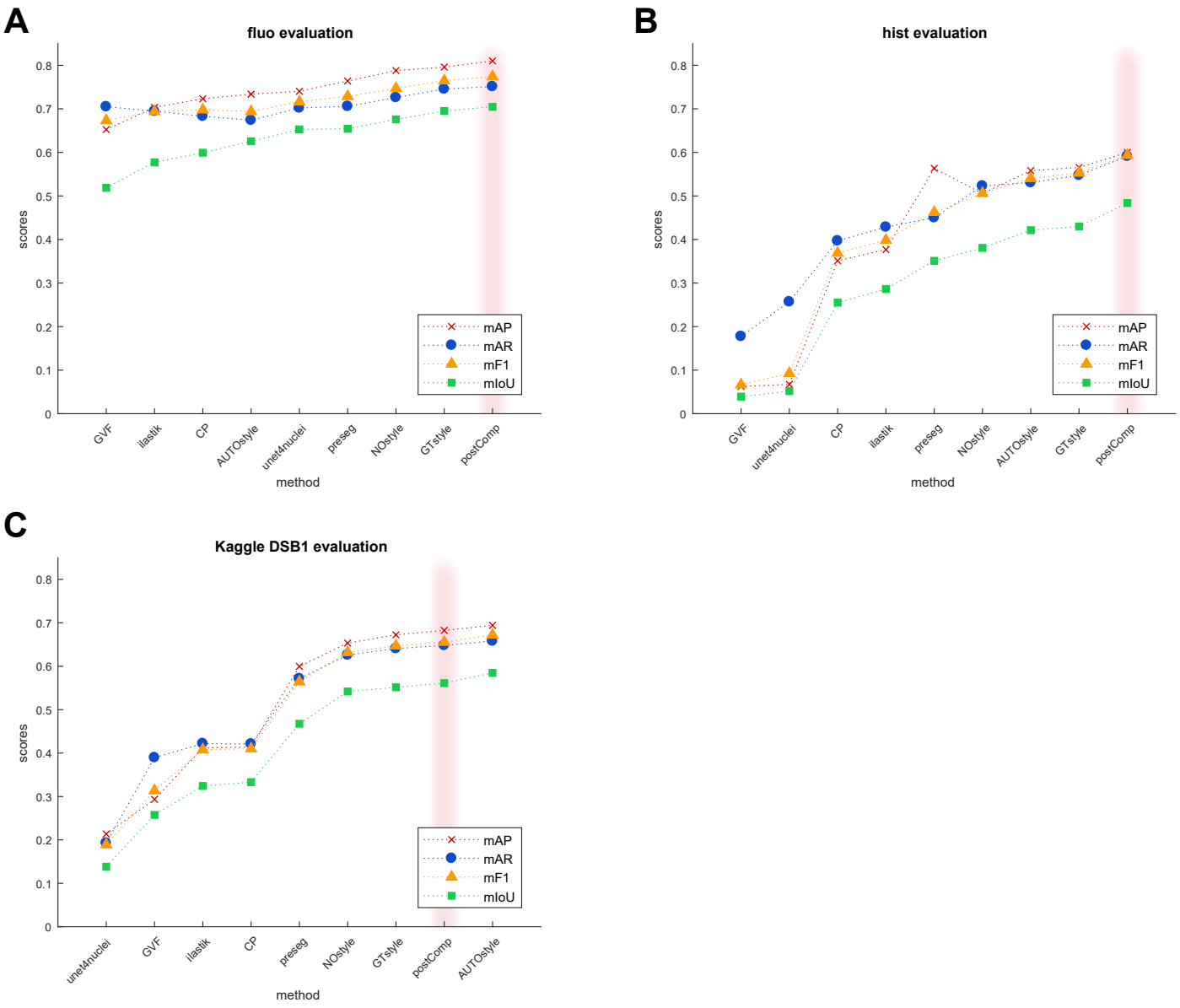


Figure S5. Related to Figure 2. Evaluation of the methods on fluo, hist, and DSB test1 using different metrics: mAP: mean average precision, mAR: mean average recall, mF1: mean average F1-score, mIoU: mean intersection over union. See definitions in Section Evaluation metrics.
(A) test set fluo
(B) test set hist
(C) test set DSB test1.
postComp is highlighted in light pink shade. Methods are sorted by mIoU. Results on DSB test2 are not reported because ground truths are not available.

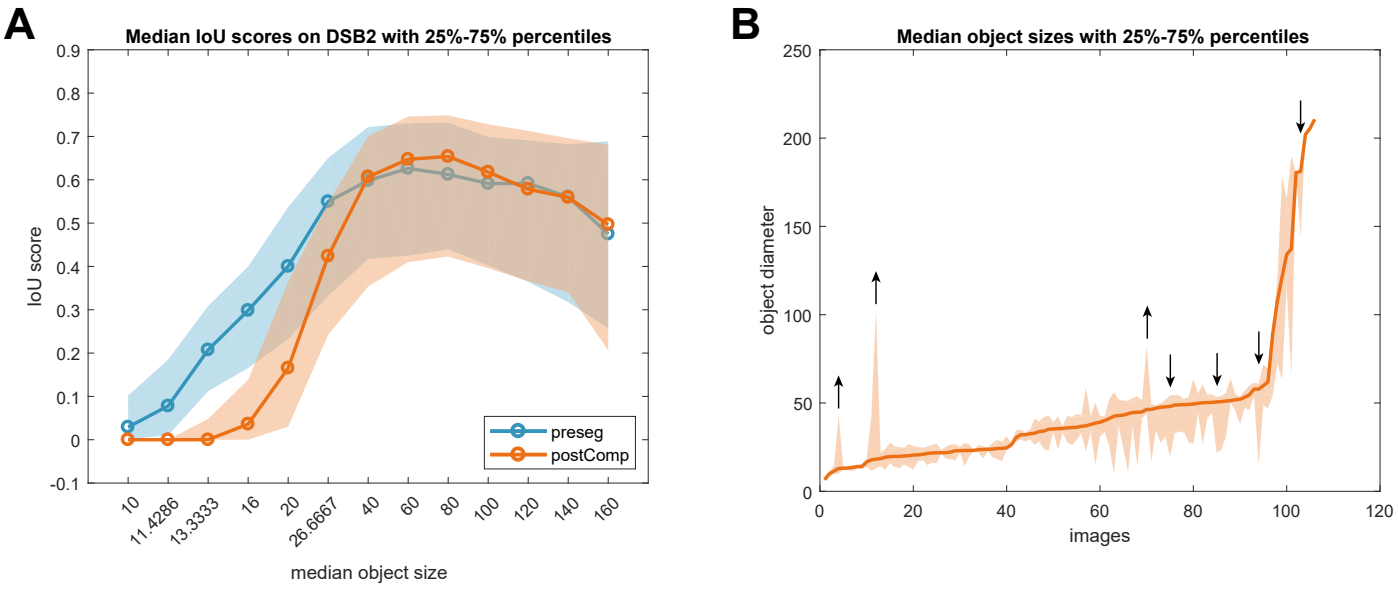


Figure S6. Related to STAR Methods. Size range analysis of our models.
(A) IoU scores on DSB stage 2 test set in the size range 0.25-4.0x obtained by our preseg and postComp models (without post-processing). Solid lines with circles show medians of the test set for each scale, while shades extend to the 25% and 75% percentiles, respectively.
(B) Object diameters in pixels on the original images, median is marked with a solid line, shades are as on a), black arrows indicate some extreme images with an unevenly wide object size range whose direction shows whether the majority of objects are smaller (down) or larger (up) than the median.

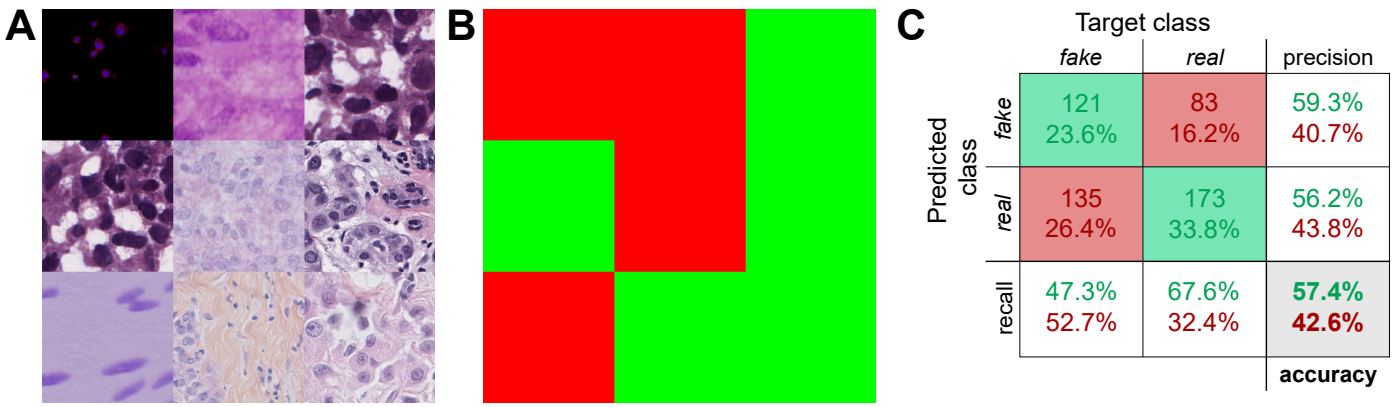


Figure S7. Related to STAR Methods. Style transfer-generated image spotting test results. We asked experts with relevant biological background if they could find synthetically generated images among real ones.
(A) Test image montage (crop).
(B) Labels: green tiles correspond to real, red to fake images in (A).
(C) Accuracy of experts in a confusion matrix: predicted labels are counted against true labels (target class) for both classes (real and fake) as marked in light green (correctly classified) and red (incorrectly classified) shaded cells, while precision as well as recall values are calculated in the border cells (green percentages refer the correct classification for the given row or column, red the false). Mean accuracy is in the bottom right cell. Precision, recall and accuracy values are as defined in STAR Methods Section Evaluation metrics (4)-(5) and (7).

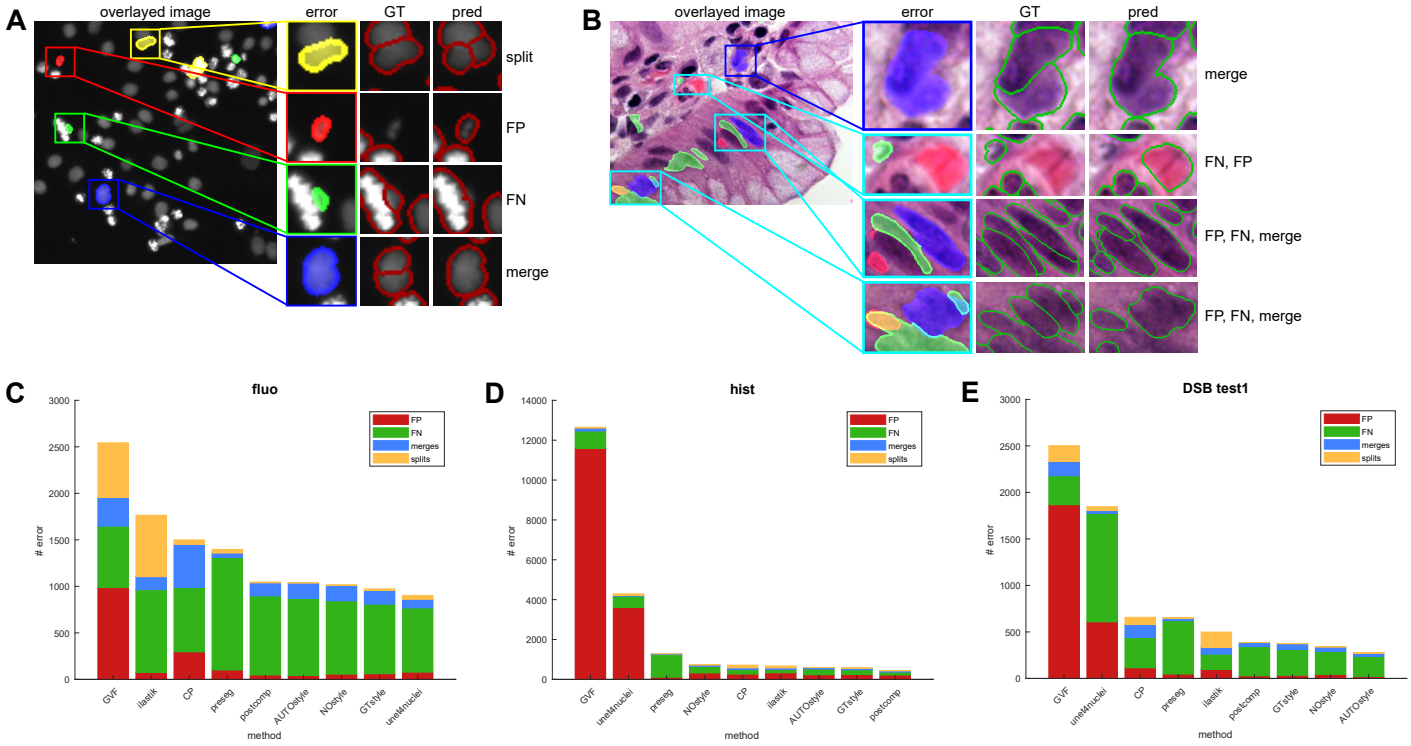


Figure S8. Related to STAR Methods. Segmentation errors.
(A) Error types overlayed on a real image in a colour-coded manner; red for FP (false), green for FN (missed), blue for merge and yellow for split. Insets show comparison to GT. Prediction of postComp.
(B) A real example as in (A) where our method typically fails similarly to all compared methods.
(C-E) Distribution of error types on the test sets: (C) fluo, (D) hist, (E) DSB test1. Colours are as in (A), methods are ordered by sum of all errors.

Table S1: DSB scores

method	dataset			
	fluo	hist	DSB test1	DSB test2
preseg	0.654	0.351	0.468	0.483
CP	0.599	0.255	0.333	0.528
unet4nuclei	0.653	0.052	0.138	-
GVF	0.519	0.039	0.258	0.168
NOstyle	0.676	0.381	0.542	0.621
AUTOstyle	0.626	0.421	**0.585**	**0.633**
GTstyle	0.695	0.430	0.551	-
postComp	**0.705**	**0.484**	0.561	**0.633**
Ilastik	0.577	0.286	0.324	-
DSB1	-	-	-	0.631
DSB2	-	-	-	0.614

Table S1. DSB scores. Related to Figure 2. We tested the methods described in Section Methods used for comparison on four test datasets: DSB test1, DSB test2, fluo, and hist. Fluo was the BBBC039 (Caicedo et al., 2018) image set of 200 DAPI-labelled fluorescent images while hist was a randomly selected set of 50 in-house labeled histological images. DBS test1 and test2 are the official test sets of the DSB 2018 competition. The DSB-score is the mean average precision at different intersection-over-union (IoU) thresholds (see Section Evaluation metrics). GTStyle is not reported for DSB test2 because annotations are not available, as a result the postComp model is the same as AUTOstyle. The best performing method's score is highlighted in bold for each test set.

Table S2: Training dataset statistics

property		subset													
		DSB train	DSB test1	Weebly	TNBC	BBBC021	Internet	biomag 02	biomag 04	biomag 05	biomag 06	jw	ISBI 2009	AS	00733
image type	fluo normal	x	x	-	-	x	x	-	-	-	-	x	x	x	-
	fluo clumped	x	x	-	-	x	x	-	x	x	x	x	-	x	x
	tissue simple	x	x	-	-	-	x	x	-	-	-	-	-	-	-
	tissue complex	x	x	x	x	-	x	-	-	-	x	-	-	-	-
image size	min	256x256	256x256	1000x1000	512x512	1280x1024	213x65	812x828	288x286	256x256	256x340	956x1242	1030x1349	512x512	512x512
	max	1040x1388	520x696	-	-	-	2848x4272	1024x1344	326x327	1024x1360	2048x2048	1040x1388	-	-	-
number of images		670	65	30	50	5	97	11	4	5	32	20	48	11	54
number of nuclei	mean/image	44.27	4152	565.23	81.12	124.40	99.18	94.27	53.25	197.60	67.03	42.30	38.10	110.27	136.15
	total	29662	63.88	16957	4056	622	9620	1037	213	988	2145	846	1829	1213	7352
models trained on		all	all but this	all	all	all	all but hist	all but hist	all	all	all but hist	all	all	all	all
models tested on		-	DSB test1	-	-	-	hist	hist	-	-	hist	-	-	-	-
data source		DSB	DSB	1	2	3, 4	5, 6	7	8	9	10	11	11, 12	11	11

Table S2. Related to STAR Methods. Statistics about the training data. Data sources in the last row are as follows.
1: https://nucleisegmentationbenchmark.weebly.com/dataset.html
2: Naylor Peter Jack, Walter Thomas, Laé Marick, & Reyal Fabien. (2018). Nuclei Segmentation in Histopathology Images Using Deep Neural Networks (Version 1.0) [Data set]. Zenodo. http://doi.org/10.5281/zenodo.1174353
3: http://mct.aacrjournals.org/content/9/6/1913
4: https://www.nature.com/articles/nmeth.2083
5: https://www.google.hu/search?q=histology+microscope+nucleus&-source=lnms&tbm=isch&sa=X&ved=0ahUKEwiIxb6v_J3aAhXFFJoKHYdzAxUQ_AUICigB&biw=1366&bih=654
6: https://www.google.hu/search?q=dapi+stained+nuclei&client=-firefox-b-ab&dcr=0&source=lnms&tbm=isch&sa=X&ved=0ahUKEwik4bK8_Z3aAhXLDZoKHZXTDZkQ_AUICigB&biw=1366&bih=654
7: https://www.nature.com/articles/nmeth.3323
8: https://www.ncbi.nlm.nih.gov/pubmed/28122742
9: https://www.nature.com/articles/s41467-017-02628-4
10: Peter Horvath Laboratory
11: shared by other participating teams of the DSB on the competition website
12: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2901896/

II.

MBoC | BRIEF REPORT

AnnotatorJ: an ImageJ plugin to ease hand annotation of cellular compartments

Réka Hollandi[a], Ákos Diósdi[a,b], Gábor Hollandi[a], Nikita Moshkov[a,c,d], and Péter Horváth[a,e,*]
[a]Synthetic and Systems Biology Unit, Biological Research Center, 6726 Szeged, Hungary; [b]Doctoral School of Biology, University of Szeged, 6726 Szeged, Hungary; [c]Doctoral School of Interdisciplinary Medicine, University of Szeged, Koranyi fasor 10, 6720 Szeged, Hungary; [d]National Research University Higher School of Economics, Faculty of Computer Science, 101000 Moscow, Russia; [e]Institute for Molecular Medicine Finland, University of Helsinki, 00014 Helsinki, Finland

ABSTRACT **AnnotatorJ combines single-cell identification with deep learning (DL) and manual annotation. Cellular analysis quality depends on accurate and reliable detection and segmentation of cells so that the subsequent steps of analyses, for example, expression measurements, may be carried out precisely and without bias. DL has recently become a popular way of segmenting cells, performing unimaginably better than conventional methods. However, such DL applications may be trained on a large amount of annotated data to be able to match the highest expectations. High-quality annotations are unfortunately expensive as they require field experts to create them, and often cannot be shared outside the lab due to medical regulations. We propose AnnotatorJ, an ImageJ plugin for the semiautomatic annotation of cells (or generally, objects of interest) on (not only) microscopy images in 2D that helps find the true contour of individual objects by applying U-Net–based presegmentation. The manual labor of hand annotating cells can be significantly accelerated by using our tool. Thus, it enables users to create such datasets that could potentially increase the accuracy of state-of-the-art solutions, DL or otherwise, when used as training data.**

Monitoring Editor
Jennifer Lippincott-Schwartz
Howard Hughes Medical Institute

Received: Feb 27, 2020
Revised: Jun 24, 2020
Accepted: Jul 17, 2020

INTRODUCTION

Single-cell analysis pipelines begin with an accurate detection of the cells. Even though microscopy analysis software tools aim to become more and more robust to various experimental setups and imaging conditions, most lack efficiency in complex scenarios such as label-free samples or unforeseen imaging conditions (e.g., higher signal-to-noise ratio, novel microscopy, or staining techniques), which opens up a new expectation of such software tools: adaptation ability (Hollandi *et al.*, 2020). Another crucial requirement is to maintain ease of usage and limit the number of parameters the users need to fine-tune to match their exact data domain.

Recently, deep learning (DL) methods have proven themselves worthy of consideration in microscopy image analysis tools as they have also been successfully applied in a wider range of applications including but not limited to face detection (Sun *et al.*, 2014; Taigman *et al.*, 2014; Schroff *et al.*, 2015), self-driving cars (Redmon *et al.*, 2016; Badrinarayanan *et al.*, 2017, Grigorescu *et al.*, 2019), and speech recognition (Hinton *et al.*, 2012). Caicedo *et al.* (Caicedo *et al.*, 2019) and others (Hollandi *et al.*, 2020; Moshkov *et al.*, 2020) proved that single-cell detection and segmentation accuracy can be significantly improved utilizing DL networks. The most popular and widely used deep convolutional neural networks (DCNNs) include Mask R-CNN (He *et al.*, 2017): an object detection and instance segmentation network; YOLO (Redmon *et al.*, 2016; Redmon and Farhadi, 2018): a fast object detector; and U-Net (Ronneberger *et al.*, 2015): a fully convolutional network specifically intended for bioimage analysis purposes and mostly used for pixel classification. StarDist (Schmidt *et al.*, 2018) is an instance segmentation DCNN optimal for convex or elliptical shapes (such as nuclei).

As robustly and accurately as they may perform, these networks rely on sufficient data, both in amount and quality, which tends to be the bottleneck of their applicability in certain cases such as single-cell detection. While in more industrial applications (see Grigorescu *et al.*, 2019 for an overview of autonomous driving) a large amount of training data can be collected relatively easily (see the cityscapes dataset [Cordts *et al.*, 2016; available at https://www.cityscapes-dataset.com/] of traffic video frames using a car and camera to

This article was published online ahead of print in MBoC in Press (http://www.molbiolcell.org/cgi/doi/10.1091/mbc.E20-02-0156). on July 22, 2020

*Address correspondence to: Péter Horváth (horvath.peter@brc.hu).

Abbreviations used: DL, deep learning; DL4J, Deeplearning4j; IoU, intersection over union; ROI, region of interest.

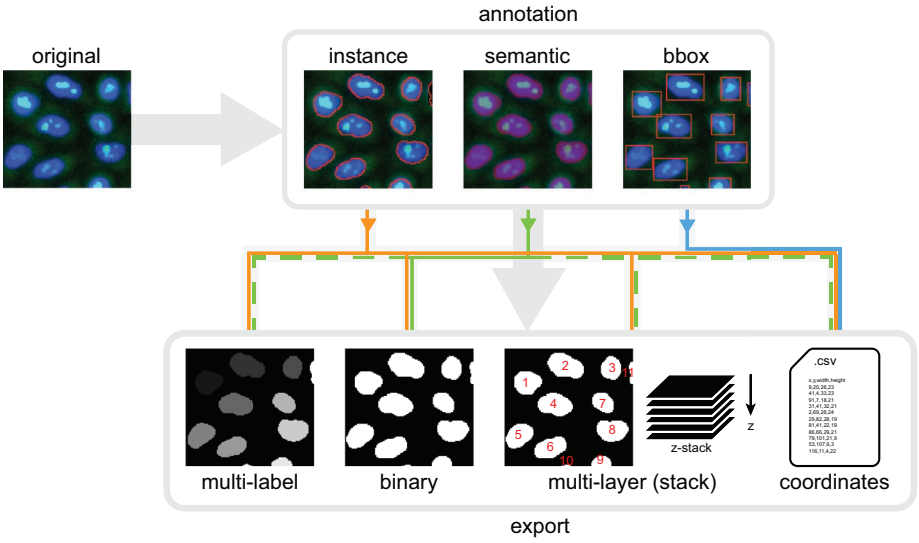


FIGURE 1: Annotation types. The top row displays our supported types of annotation: instance, semantic, and bounding box (noted as "bbox" in the figure) based on the same objects of interest, in this case nuclei, shown in red. Instances mark the object contours, semantic overlay shows the regions (area) covered, while bounding boxes are the smallest enclosing rectangles around the object borders. Export options are shown in the bottom row: multilabel, binary, multilayer images, and coordinates in a text file. Lines mark the supported export options for each annotation type by colors: orange for instance, green for semantic, and blue for bounding box. Dashed lines indicate additional export options for semantics that should be used carefully.

record and potentially nonexpert individuals to label the objects), clinical data is considerably more difficult, due to ethical constraints, and expensive to gather as expert annotation is required. Datasets available in the public domain such as BBBC (Ljosa *et al.*, 2012) at https://data.broadinstitute.org/bbbc/, TNBC (Naylor *et al.*, 2017, 2019) or TCGA (Cancer Genome Atlas Research Network, 2008; Kumar *et al.*, 2017), and detection challenges including ISBI (Coelho *et al.*, 2009), Kaggle (https://www.kaggle.com/, e.g., Data Science Bowl 2018; see at https://www.kaggle.com/c/data-science-bowl-2018), ImageNet (Russakovsky *et al.*, 2015), etc., contribute to the development of genuinely useful DL methods; however, most of them lack heterogeneity of the covered domains and are limited in data size. Even combining them one could not possibly prepare their network/method to generalize well (enough) on unseen domains that vastly differ from the pool they covered. On the contrary, such an adaptation ability can be achieved if the target domain is represented in the training data, as proposed in Hollandi *et al.*, 2020, where synthetic training examples are generated automatically in the target domain via image style transfer.

Eventually, similar DL approaches' performance can only be increased over a certain level if we provide more training examples. The proposed software tool was created for this purpose: the expert can more quickly and easily create a new annotated dataset in their desired domain and feed the examples to DL methods with ease. The user-friendly functions included in the plugin help organize data and support annotation, for example, multiple annotation types, editing, classes, etc. Additionally, a batch exporter is provided offering different export formats matching typical DL models'; supported annotation and export types are visualized in Figure 1; open-source code is available at https://github.com/spreka/annotatorj under GNU GPLv3 license.

We implemented the tool as an ImageJ (Abramoff *et al.*, 2004, Schneider *et al.*, 2012) plugin because ImageJ is frequently used by bioimage analysts, providing a familiar environment for users. While other software also provide a means to support annotation, for example, by machine learning–based pixel classification (see a detailed comparison in *Materials and Methods*), AnnotatorJ is a lightweight, free, open-source, cross-platform alternative. It can be easily installed via its ImageJ update site at https://sites.imagej.net/Spreka/ or run as a standalone ImageJ instance containing the plugin.

In AnnotatorJ we initialize annotations with DL presegmentation using U-Net to suggest contours from as little as a quickly drawn line over the object (see Supplemental Material and Figure 2). U-Net predicts pixels belonging to the target class with the highest probability within a small bounding box (a rectangle) around the initially drawn contour; then a fine approximation of the true object boundary is calculated from connected pixels; this is referred to as the suggested contour. The user then manually refines the contour to create a pixel-perfect annotation of the object.

RESULTS AND DISCUSSION

Performance evaluation

We quantitatively evaluated annotation performance and speed in AnnotatorJ (see Figures 3 and 4) with the help of three annotators who had experience in cellular compartment annotation. Both annotation accuracy and time were measured on the same two test sets: a nucleus and a cytoplasm image set (see also Supplemental Figure S1 and Supplemental Material). Both test sets contained images of various experimental conditions, including fluorescently labeled and brightfield-stained samples, tissue section, and cell culture images. We compared the effectiveness of our plugin using *Contour assist* mode to only allowing the use of *Automatic adding*. Even though the latter is also a functionality of AnnotatorJ, it ensured that the measured annotation times correspond to a single object each. Without this option the user must press the key "t" after every contour drawn to add it to the region of interest (ROI) list, which can be unintendedly missed, increasing its time as the same contour must be drawn again.

For the annotation time test presented in Figure 3 we measured the time passed between adding new objects to the annotated object set in ROI Manager for each object, then averaged the times for each image and each annotator, respectively. Time was measured in the Java implementation of the plugin in milliseconds. Figures 3 and 4 show SEM error bars for each mean measurement (see Supplemental Material for details).

In the case of annotating cell nuclei, results confirm that hand-annotation tasks can be significantly accelerated using our tool. Each of the three annotators were faster by using *Contour assist*; two of them nearly double their speed.

To ensure efficient usage of our plugin in annotation assistance, we also evaluated the accuracies achieved in each test case by calculating mean intersection over union (IoU) scores of the annotations as segmentations compared with ground truth masks previously created by different expert annotators. We used the mean IoU score defined in the Data Science Bowl 2018 competition (https://www.kaggle.com/c/data-science-bowl-2018/overview/evaluation) and in Hollandi *et al.*, 2020:

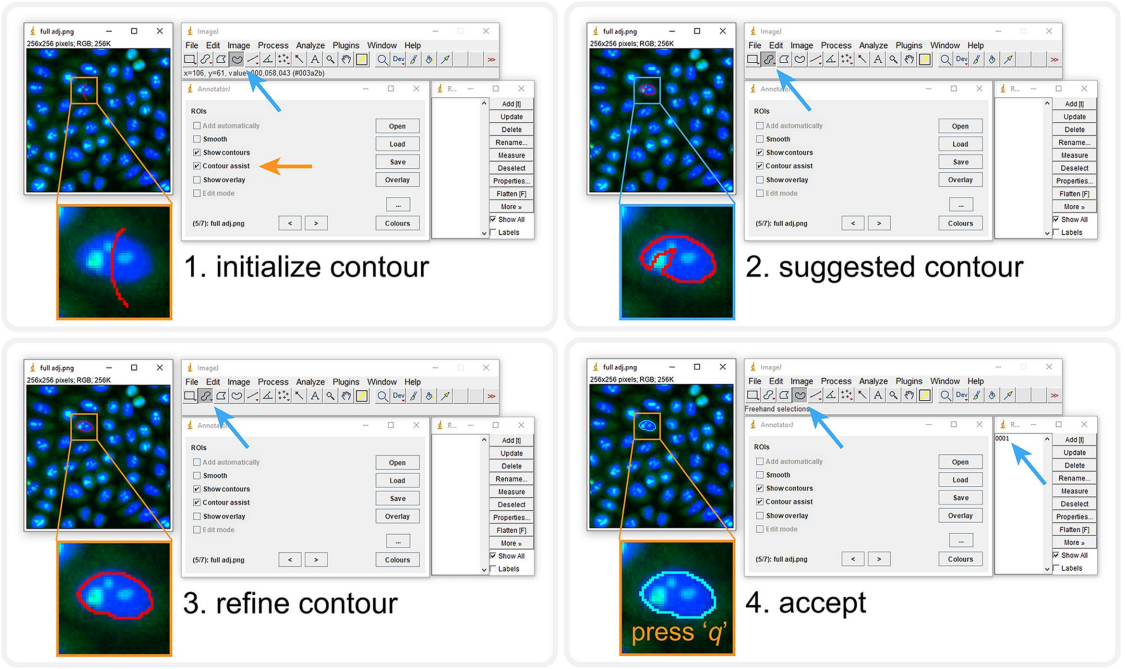


FIGURE 2: Contour assist mode of AnnotatorJ. The blocks show the order of steps; the given tool needed is automatically selected. User interactions are marked with orange arrows, and automatic steps with blue. 1) Initialize the contour with a lazily drawn line; 2) the suggested contour appears (a window is shown until processing completes), brush selection tool is selected automatically; 3) refine the contour as needed; 4) accept it by pressing the key "q" or reject with "Ctrl" + "delete." Accepting adds the ROI to ROI Manager with a numbered label. See also Supplemental Material for a demo video (figure2.mov).

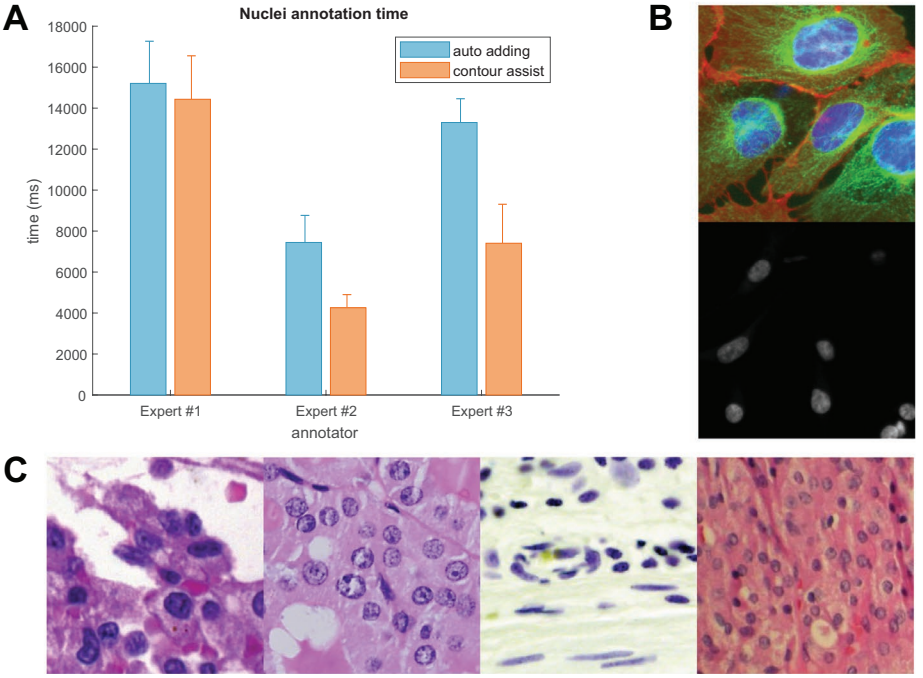


FIGURE 3: Annotation times on nucleus images. AnnotatorJ was tested on sample microscopy images (both fluorescent and brightfield, as well as cell culture and tissue section images); annotation time was measured on a per-object (nucleus) level. Bars represent the mean annotation times on the test image set; error bars show SEM. Orange corresponds to *Contour assist* mode and blue to only allowing the *Automatic adding* option. (A) Nucleus test set annotation times. (B) Example cell culture test images. (C) Example histopathology images. Images shown in B and C are 256 × 256 crops of original images. Some images are courtesy of Kerstin Elisabeth Dörner, Andreas Mund, Viktor Honti, and Hella Bolck.

$$\text{IoU score}(t) = \frac{\text{TP}(t)}{\text{TP}(t) + \text{FP}(t) + \text{FN}(t) + \varepsilon} \quad (1)$$

IoU determines the overlapping pixels of the segmented mask with the ground truth mask (intersection) compared with their union. The IoU score is calculated at 10 different thresholds from 0.5 to 0.95 with 0.05 steps; at each threshold true positive (TP), false positive (FP), and false negative (FN) objects are counted. An object is considered TP if its IoU is greater than the given threshold *t*. IoU scores calculated at all 10 thresholds were finally averaged to yield a single IoU score for a given image in the test set.

An arbitrarily small $\varepsilon = 10^{-40}$ value was added to the denominators for numerical stability. Equation 1 is a modified version of mean average precision (mAP) typically used to describe the accuracy of instance segmentation approaches. Precision is formulated as

$$\text{precision}(t) = \frac{\text{TP}(t)}{\text{TP}(t) + \text{FP}(t) + \varepsilon} \quad (2)$$

Nucleus and cytoplasm image segmentation accuracies were averaged over the test sets, respectively. We compared our

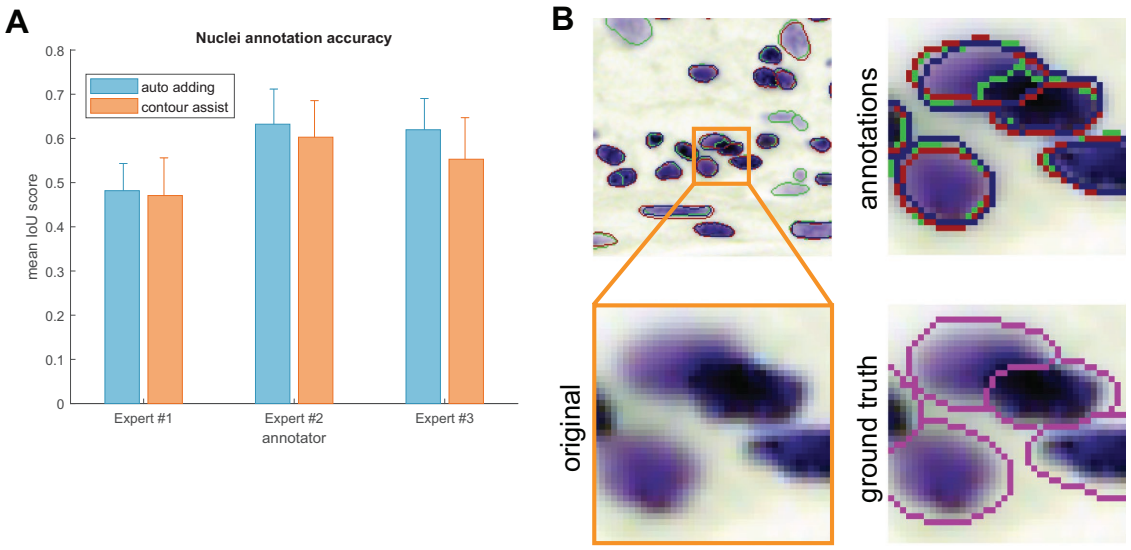


FIGURE 4: Annotation accuracies. Annotations created in the same test as the times measured in Figure 3 were evaluated using mean IoU scores for the nucleus test set. Error bars show SEM; colors are as in Figure 3. (A) Nucleus test set accuracies. (B) Example contours drawn by our expert annotators. Inset highlighted in orange is zoomed in showing the original image; annotations are marked in red, green, and blue corresponding to experts #1–3, respectively, and ground truth contours (in magenta) are overlayed on the original image for comparison.

annotators using and not using *Contour assist* mode (Figure 4). The results show greater interexpert than intraexpert differences, allowing us to conclude that the annotations created in AnnototarJ are nearly as accurate as freehand annotations.

Export evaluation

As the training data annotation process for deep learning applications requires the annotated objects to be exported in a manner that DL models can load them, which typically covers the four types of export options offered in AnnotatorJExporter, it is also important to investigate the efficiency of export. We measured export times similarly to annotation times. For the baseline results, each object defined by their ROI was copied to a new empty image, then filled and saved to create a segmentation mask image file. Exportation from AnnotatorJExporter was significantly faster and only required a few clicks: it took four orders of magnitude less time to export the annotations (~60 ms). Export times reported correspond to a randomly selected expert so that computer hardware specifications remain the same.

Comparison to other tools and software packages

The desire to collect annotated datasets has arisen with the growing popularity and availability of application-specific DL methods. Object classification on natural images (photos) and face recognition are frequently used examples of such applications in computer vision. We discuss some of the available software tools created for image annotation tasks and compare their feature scope in the following table (Table 1; see also Supplemental Table S1) and in the Supplemental Material.

We collected our list of methods to compare following Morikawa, 2019 and "The best image annotation platforms", 2018. While there certainly is a considerable amount of annotation tools for object detection purposes, most of them are not open source. We included Lionbridge.AI (https://lionbridge.ai/services/image-annotation/) and Hive (https://thehive.ai/), two service-based solutions, because of their wide functionality and artificial intelligence support. Both of them work in a project-management way and outsource the annotation task to enable fast and accurate results. Their main application spectra cover more general object detection tasks like classification of traffic video frames. LabelImg (https://github.com/tzutalin/labelImg), on the other hand, as well as the following tools, is open source but offers a narrower range of annotation options and lacks machine learning support making it a lightweight but free alternative. VGG Image Annotator (Dutta and Zisserman, 2019) comes on a web-based platform, therefore making it very easy for the user to become familiarized with the software. It enables multiple types of annotation with class definition. Diffgram (https://diffgram.com/) is available both online and as a locally installable version (Python) and adds DL support which speeds up the annotation process significantly; that is, provided the intended object classes are already trained and the DL predictions only need minor edit. A similar, also web-based approach is provided by supervise.ly (https://supervise.ly/; see the Supplemental Material), which is free for research purposes. Even though web-hosted services offer a convenient solution for training new models (if supported), handling sensitive clinical data may be problematic. Hence, locally installable software is more desirable in biological and medical applications. A software closer to the bioimage analyst community is CytoMine (Marée *et al.*, 2016; Rubens *et al.*, 2019), a more general image processing tool with a lot of annotation options that also provides DL support and has a web interface. SlideRunner (Aubreville *et al.*, 2018) was created for large tissue section (slide) annotation specifically, but similar to others it does not integrate machine learning methods to help annotation and rather focuses on the classification task.

AnnotatorJ, on the other hand, as an ImageJ (Fiji) plugin should provide a familiar environment for bioimage annotators to work in. It offers all the functionality available in similar tools (such as different annotation options: bounding box, polygon, freehand drawing, semantic segmentation, and editing them) while it also incorporates support for a popular DL model, U-Net. Furthermore, any user-trained Keras model can be loaded into the plugin with ease because of the DL4J framework, extending its use cases to general object annotation tasks (see Supplemental Figure S2 and Supplemental Material). Due to its open-source implementation, the users can modify or extend the plugin to even better fit their needs.

Feature		Tool							
		LabelImg	Lionbridge.AI	Hive	VGG Image Annotator	Diffgram	CytoMine	SlideRunner	AnnotatorJ
Open source		✓	×	×	✓	✓	✓	✓	✓
Cross-platform		✓	service	service	✓	✓	Ubuntu	✓	✓
Implementation		Python	N/A	N/A	web	Python, web	Docker, web	Python	Java
Annotation	Bounding box	✓	✓	✓	✓	✓	✓	✓	✓
	Freehand ROI	×	×	N/A	×	×	✓	×	✓
	Polygonal region	×	✓	✓	✓	✓	✓	✓	✓ (ImageJ)
	Semantic	×	✓	✓	×	×	✓	×	✓
	Single click[a]	×	×	×	×	×	✓ (magic wand)	✓	✓ (drag)
	Edit selection	x (drag)	N/A	N/A	✓	✓	✓	N/A	✓
Class option		✓	✓	✓	✓	✓	✓	✓	✓
DL	Support	×	AI assist	N/A	×	✓	✓	×	✓
	Model import	×	N/A	N/A	×	Tensorflow	N/A	×	Keras, DL4J

[a]Bounding box drawing with a single click and drag is not considered a single click annotation.

TABLE 1: Comparison of annotation software tools.

Additionally, as an ImageJ plugin it requires no software installation, can be downloaded inside ImageJ/Fiji (via its update site, https://sites.imagej.net/Spreka/), or run as a standalone ImageJ instance with this plugin.

We also briefly discuss ilastik (Sommer *et al.*, 2011; Berg *et al.*, 2019) and Suite2p (Pachitariu *et al.*, 2017) in the Supplemental Material because they are not primarily intended for annotation purposes. Two ImageJ plugins that offer manual annotation and machine learning–generated outputs, Trainable Weka Segmentation (Arganda-Carreras *et al.*, 2017) and LabKit (Arzt, 2017), are also detailed in the Supplemental Material.

We presented an ImageJ plugin, AnnotatorJ, for convenient and fast annotation and labeling of objects on digital images. Multiple export options are also offered in the plugin.

We tested the efficiency of our plugin with three experts on two test sets comprising nucleus and cytoplasm images. We found that our plugin accelerates the hand-annotation process on average and offers up to four orders of magnitude faster export. By integrating the DL4J Java framework for U-Net contour suggestion in *Contour assist* mode any class of object can be annotated easily: the users can load their own custom models for the target class.

MATERIALS AND METHODS

Motivation

We propose AnnotatorJ, an ImageJ (Abramoff *et al.*, 2004, Schneider *et al.*, 2012) plugin for the annotation and export of cellular compartments that can be used to boost DL models' performance. The plugin is mainly intended for bioimage annotation but could possibly be used to annotate any type of object on images (see Supplemental Figure S2 for a general example). During development we kept in mind that the intended user should be able to get comfortable with the software very quickly and quicken the otherwise truly time-consuming and exhausting process of manually annotating single cells or their compartments (such as individual nucleoli, lipid droplets, nucleus, or cytoplasm).

The performance of DL segmentation methods is significantly influenced by both the training data size and its quality. Should we feed automatically segmented objects to the network, errors present in the original data will be propagated through the network during training and bias the performance, hence such training data should always be avoided. Hand-annotated and curated data, however, will minimize the initial error boosting the expected performance increase on the target domain to which the annotated data belongs. NucleAIzer (Hollandi *et al.*, 2020) showed an increase in nucleus segmentation accuracy when a DL model was trained on synthetic images generated from ground truth annotations instead of presegmented masks.

Features

AnnotatorJ helps organize the input and output files by automatically creating folders and matching file names to the selected type and class of annotation. Currently, the supported annotation types are 1) instance, 2) semantic, and 3) bounding box (see Figure 1). Each of these are typical inputs of DL networks; instance annotation provides individual objects separated by their boundaries (useful in the case of, e.g., clumped cells of cancerous regions) and can be used to provide training data for instance segmentation networks such as Mask R-CNN (He *et al.*, 2017). Semantic annotation means foreground–background separation of the image without distinguishing individual objects (foreground); a typical architecture using such segmentations is U-Net (Ronneberger *et al.*, 2015). And finally, bounding box annotation is done by identifying the object's bounding

rectangle, and is generally used in object detection networks (like YOLO [Redmon *et al.*, 2016] or R-CNN [Girshick *et al.*, 2014]).

Semantic annotation is done by painting areas on the image overlay. All necessary tools to operate a given function of the plugin are selected automatically. Contour or overlay colors can be selected from the plugin window. For a detailed description and user guide please see the documentation of the tool (available at https://github.com/spreka/annotatorj repository).

Annotations can be saved to default or user-defined "classes" corresponding to biological phenotypes (e.g., normal or cancerous) or object classes—used as in DL terminology (such as person, chair, bicycle, etc.), and later exported in a batch by class. Phenotypic differentiation of objects can be supported by loading a previously annotated class's objects for comparison as overlay to the image and toggling their appearance by a checkbox.

We use the default ImageJ ROI Manager to handle instance annotations as individual objects. Annotated objects can be added to the ROI list automatically (without the bound keystroke "t" as defined by ROI Manager) when the user releases the mouse button used to draw the contour by checking its option in the main window of the plugin. This ensures that no annotation drawn is missing from the ROI list.

Contour editing is also possible in our plugin using "Edit mode" (by selecting its checkbox) in which the user can select any already annotated object on the image by clicking on it, then proceed to edit the contour and either apply modifications with the shortcut "Ctrl" + "q," discard them with "escape," or delete the contour with "Ctrl" + "delete." The given object selected for edit is highlighted in inverse contour color.

Object-based classification is also possible in "Class mode" (via its checkbox): similarly to "Edit mode," ROIs can be assigned to a class by clicking on them on the image which will also update the selected ROI's contour to the current class's color. New classes can be added and removed, and their class color can be changed. A default class can be set for all unassigned objects on the image. Upon export (using either the quick export button "[^]" in the main window or the exporter plugin) masks are saved by classes.

In the options (button "…" in the main window) the user can select to use either U-Net or a classical region-growing method to initialize the contour around the object marked. Currently only instance annotation can be assisted.

Contour suggestion using U-Net

Our annotation helper feature "*Contour assist*" (see Figure 2) allows the user to work on initialized object boundaries by roughly marking an object's location on the image which is converted to a well-defined object contour via weighted thresholding after a U-Net (Ronneberger *et al.*, 2015) model trained on nucleus or other compartment data predicts the region covered by the object. We refer to this as the *suggested contour* and expect the user to refine the boundaries to match the object border precisely. The suggested contour can be further optimized by applying *active contour* (AC; Kass *et al.*, 1988) to it. We aim to avoid fully automatic annotation (as previously argued) by only enabling one object suggestion at a time and requiring manual interaction to either refine, accept, or reject the suggested contour. These operations are bound to keyboard shortcuts for convenience (see Figure 2). When using the *Contour assist* function automatic adding of objects is not available to encourage the user to manually validate and correct the suggested contour as needed.

In Figure 2 we demonstrate *Contour assist* using a U-Net model trained on versatile microscopy images of nuclei in Keras and on a fluorescent microscopy image of a cell culture where the target objects, nuclei, are labeled with DAPI (in blue). This model is provided at https://github.com/spreka/annotatorj/releases/tag/v0.0.2-model in the open-source code repository of the plugin.

Contour suggestions can be efficiently used for proper initialization of object annotation, saving valuable time for the expert annotator by suggesting a nearly perfect object contour that only needs refinement (as shown in Figure 2). Using a U-Net model accurate enough for the target object class, the expert can focus on those image regions where the model is rather uncertain (e.g., around the edges of an object or the separating line between adjacent objects) and fine-tune the contour accurately while sparing considerable effort on more obvious regions (like an isolated object on simple background) by accepting the suggested contour after marginal correction.

The framework of the chosen U-Net implementation, DL4J (available at http://deeplearning4j.org/ or https://github.com/eclipse/deeplearning4j), supports Keras model import, hence custom, application-specific models can be loaded in the plugin easily by either training them in DL4J (Java) or Python (Keras) and saving the trained weights and model configuration in .h5 and .json files. This vastly extends the possible fields of application for the plugin to general object detection or segmentation tasks (see Supplemental Material and Supplemental Figures S2 and S3).

Exporter

The annotation tool is supplemented by an exporter, AnnotatorJ-Exporter plugin, also available in the package. It was optimized for the batch export of annotations created by our annotation tool. For consistency, one class of objects can be exported at a time. We offer four export options: 1) multilabeled, 2) multilayered, 3) semantic images, and 4) coordinates (see Figure 1). Instance annotations are typically expected to be exported as multilabeled (instance-aware) or multilayered (stack) grayscale images, the latter of which is useful for handling overlapping objects such as cytoplasms in cell culture images. Semantic images are binary foreground–background images of the target objects while coordinates (top-left corner [*x*,*y*] of the bounding rectangle appended by its width and height in pixels) can be useful training data for object detection applications including astrocyte localization (Suleymanova *et al.*, 2018) or in a broader aspect, face detection (Taigman *et al.*, 2014). All export options are supported for semantic annotation; however, we note that in instance-aware options (multilabeled or multilayered mask and coordinates) only such objects are distinguished whose contours do not touch on the annotation image.

OpSeF compatibility

OpSeF (Open Segmentation Framework; Rasse *et al.*, 2020) is an interactive python notebook-based framework (available at https://github.com/trasse/OpSeF-IV) that allows users to easily try different DL segmentation methods in customizable pipelines. We extended AnnotatorJ to support the data structure and format used in OpSeF to allow seamless integration in these pipelines, so users can manually modify, create, or classify objects found by OpSeF in AnnotatorJ, then export the results in a compatible format for further use in the former software. A user guide is provided in the documentation of https://github.com/trasse/OpSeF-IV.

ImageJ

ImageJ (or Fiji: Fiji is just ImageJ; Schindelin *et al.*, 2012) is an open-source, cross-platform image analysis software tool in Java that has

been successfully applied in numerous bioimage analysis tasks (segmentation [Legland *et al.*, 2016; Arganda-Carreras *et al.*, 2017], particle analysis [Abramoff *et al.* 2004], etc.) and is supported by a broad range of community, comprising of biomage analyst end users and developers as well. It provides a convenient framework for new developers to create their custom plugins and share them with the community. Many typical image analysis pipelines have already been implemented as a plugin, for example, U-Net segmentation plugin (Falk *et al.*, 2019) or StarDist segmentation plugin (Schmidt *et al.*, 2018).

U-Net implementation

We used the DL4J (http://deeplearning4j.org/) implementation of U-Net in Java. DL4J enables building and training custom DL networks, preparing input data for efficient handling and supports both GPU and CPU computation throughout its ND4J library.

The architecture of U-Net was first developed by Ronneberger *et al.* (Ronneberger *et al.*, 2015) and was designed to learn medical image segmentation on a small training set when a limited amount of labeled data is available, which is often the case in biological contexts. To handle touching objects as often is the case in nuclei segmentation, it uses a weighted cross entropy loss function to enhance the object-separating background pixels.

Region growing

A classical image processing algorithm, region growing (Haralick and Shapiro, 1985; Adams and Bischof, 1994) starts from initial seed points or objects and expands the regions towards the object boundaries based on the intensity changes on the image and constraints on distance or shape. We used our own implementation of this algorithm.

ACKNOWLEDGMENTS

R.H., N.M., A.D., and P.H. acknowledge support from the LENDULET-BIOMAG grant (Grant no. 2018-342), from the European Regional Development Funds (GINOP-2.3.2-15-2016-00006, GINOP-2.3.2-15-2016-00026, and GINOP-2.3.2-15-2016-00037), from the H2020-discovAIR (874656), and from Chan Zuckerberg Initiative, Seed Networks for the HCA-DVP. We thank Krisztián Koós for the technical help, and Máté Görbe and Tamás Monostori for testing the plugin.

REFERENCES

The best image annotation platforms for computer vision (2018, October 30). + an honest review of each, https://hackernoon.com/the-best-image-annotation-platforms-for-computer-vision-an-honest-review-of-each-dac7f565fea.

Abramoff MD, Magalhaes PJ, Ram SJ (2004). Image processing with ImageJ. Biophoton Int, 11, 36–42.

Adams R, Bischof L (1994). Seeded region growing, IEEE Trans Pattern Anal Mach Intell 16, 641–647.

Arganda-Carreras I, Kaynig V, Rueden C, Eliceiri KW, Schindelin J, Cardona A, Sebastian Seung H (2017). Trainable Weka segmentation: a machine learning tool for microscopy pixel classification. Bioinformatics 33, 2424–2426.

Arganda-Carreras I, Turaga SC, Berger DR, Cireşan D, Giusti A, Gambardella LM, Schmidhuber J, Laptev D, Dwivedi S, Buhmann JM, *et al.* (2015). Crowdsourcing the creation of image segmentation algorithms for connectomics. Front Neuroanat 9, 142.

Arzt M (2017). https://imagej.net/Labkit.

Aubreville M, Bertram C, Klopfleisch R, Maier A (2018). SlideRunner. In: Bildverarbeitung für die Medizin 2018, Heidelberg, Berlin: Informatik aktuell, Springer Vieweg, pp. 309–314. https://doi.org/10.1007/978-3-662-56537-7_81.

Badrinarayanan V, Kendall A, Cipolla R (2017). SegNet: a deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans Pattern Anal Mach Intell 39, 2481–2495.

Berg S, Kutra D, Kroeger T, StraehleCN, KauslerBX, HauboldC, SchieggM, AlesJ, BeierT, RudyM, *et al.* (2019). ilastik: interactive machine learning for (bio)image analysis. Nat Methods 16, 1226–1232.

Caicedo JC, Roth J, Goodman A, Becker T, Karhohs KW, Broisin M, Molnar C,BeckerT, KarhohsKW, BroisinM, *et al.* (2019). Evaluation of deep learning strategies for nucleus segmentation in fluorescence images. Cytometry A 95, 952–965.

Cancer Genome Atlas Research Network (2008). Comprehensive genomic characterization defines human glioblastoma genes and core pathways. Nature 455, 1061–1068.

Coelho LP, Shariff A, Murphy RF (2009). Nuclear segmentation in microscope cell images: a hand-segmented dataset and comparison of algorithms. 2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro. Available at https://doi.org/10.1109/isbi.2009.5193098.

Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, Franke U, Roth S, Schiele B, *et al.* (2016). The cityscapes dataset for semantic urban scene understanding. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Available at https://doi.org/10.1109/cvpr.2016.350.

Dutta A, Zisserman A (2019). The VIA annotation software for images, audio and video. In Proceedings of the 27th ACM International Conference on Multimedia - MM '19. Available at https://doi.org/10.1145/3343031.3350535.

Eclipse Deeplearning4j Development Team. Deeplearning4j: open-source distributed deep learning for the JVM, Apache Software Foundation License 2.0. http://deeplearning4j.org

Falk T, Mai D, Bensch R, Çiçek Ö, Abdulkadir A, Marrakchi Y, Böhm A, Deubner J, Jäckel Z, Seiwald K, *et al.* (2019). U-Net: deep learning for cell counting, detection, and morphometry. Nat Methods 16, 67–70.

Frank E, Hall MA, Witten IH (2016). "The WEKA Workbench", online appendix for Data Mining: Practical Machine Learning Tools and Techniques, 4th ed., Morgan Kaufmann.

Girshick R, Donahue J, Darrell T, Malik J (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In 2014 IEEE Conference on Computer Vision and Pattern Recognition. Available at https://doi.org/10.1109/cvpr.2014.81.

Grigorescu S, Trasnea B, Cocias T, Macesanu G (2019). A survey of deep learning techniques for autonomous driving. J Field Rob 37, 362–386.

Haralick RM, Shapiro LG (1985). Image segmentation techniques. Applications of Artificial Intelligence II. Available at https://doi.org/10.1117/12.948400.

He K, Gkioxari G, Dollar P, Girshick R (2017). Mask R-CNN. In 2017 IEEE International Conference on Computer Vision (ICCV). Available at https://doi.org/10.1109/iccv.2017.322.

Hinton G, Deng L, Yu D, Dahl G, Mohamed A-R, Jaitly N, Senior A, Vanhoucke V, Nguyen P, Sainath TN, *et al.* (2012). Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process Mag 29, 82–97.

Hollandi R, Szkalisity A, Toth T, Tasnadi E, Molnar C, Mathe B, Grexa I, Molnar J, Balind A, Gorbe M, *et al.* (2020). nucleAIzer: a parameter-free deep learning framework for nucleus segmentation using image style transfer. Cell Syst 10, 453–458.e6.

Kass M, Witkin A, Terzopoulos D (1988). Snakes: active contour models. Int J Comput Vis 1, 321–331.

Kumar N, Verma R, Sharma S, Bhargava S, Vahadane A, Sethi A (2017). A dataset and a technique for generalized nuclear segmentation for computational pathology. IEEE Trans Med Imaging 36, 1550–1560.

Legland D, Arganda-Carreras I, Andrey P (2016). MorphoLibJ: integrated library and plugins for mathematical morphology with ImageJ. Bioinformatics 32, 3532–3534.

Ljosa V, Sokolnicki KL, Carpenter AE (2012). Annotated high-throughput microscopy image sets for validation. Nat Methods 9, 637.

Marée R, Rollus L, Stévens B, Hoyoux R, Louppe G, Vandaele R, Begon J-M, Kainz P, Geurts P, Wehenkel L, *et al.* (2016). Collaborative analysis of multi-gigapixel imaging data using Cytomine. Bioinformatics 32, 1395–1401.

Morikawa R (July 18, 2019). 24 best image annotation tools for computer vision. Available at https://lionbridge.ai/articles/image-annotation-tools-for-computer-vision/.

Moshkov N, Mathe B, Kertesz-Farkas A, Hollandi R, Horvath P (2020). Test-time augmentation for deep learning-based cell segmentation on microscopy images. Sci Rep 10, 5068.

Naylor P, Lae M, Reyal F, Walter T (2017). Nuclei segmentation in histopathology images using deep neural networks. In 2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017). Available at https://doi.org/10.1109/isbi.2017.7950669.

Naylor P, Lae M, Reyal F, Walter T (2019). Segmentation of nuclei in histopathology images by deep regression of the distance map. IEEE Trans Med Imaging 38, 448–459.

Pachitariu M, Stringer C, Dipoppa M, Schröder S, Rossi LF, Dalgleish H, Carandini M, (2017). Suite2p: beyond 10,000 neurons with standard two-photon microscopy. BioRxiv. https://doi.org/10.1101/061507.

Rasse TM, Hollandi R, Horvath P (2020). OpSeF IV: open source Python framework for segmentation of biomedical images. BioRxiv. https://doi.org/10.1101/2020.04.29.068023.

Redmon J, Divvala S, Girshick R, Farhadi A (2016). You only look once: unified, real-time object detection. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Available at https://doi.org/10.1109/cvpr.2016.91.

Redmon J, Farhadi A (2018). YOLOv3: an incremental improvement. arXiv https://arxiv.org/abs/1804.02767.

Ronneberger O, Fischer P, Brox T (2015). U-Net: convolutional networks for biomedical image segmentation. In Lecture Notes in Computer Science, Springer, Vol. 9351, 234–241.

Rubens U, Hoyoux R, Vanosmael L, Ouras M, Tasset M, Hamilton C, Longuespée R, Marée R (2019). Cytomine: toward an open and collaborative software platform for digital pathology bridged to molecular investigations. Prot Clin Appl 13, 1800057.

Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, *et al.* (2015). ImageNet large scale visual recognition challenge. Int J Comput Vis, 115, 211–252.

Schindelin J, Arganda-Carreras I, Frise E, Kaynig V, Longair M, Pietzsch T, Preibisch S, Rueden C, Saalfeld S, Schmid B, *et al.* (2012). Fiji: an open-source platform for biological-image analysis. Nat Methods 9, 676–682.

Schmidt U, Weigert M, Broaddus C, Myers G (2018). Cell detection with star-convex polygons. In Lecture Notes in Computer Science, Springer, 265–273. Crossref. Web.

Schneider CA, Rasband WS, Eliceiri KW (2012). NIH image to ImageJ: 25 years of image analysis. Nat Methods 9, 671–675.

Schroff F, Kalenichenko D, Philbin J (2015). FaceNet: a unified embedding for face recognition and clustering. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 815–823. doi: 10.1109/CVPR.2015.7298682.

Sommer C, Straehle C, Kothe U, Hamprecht FA (2011). Ilastik: interactive learning and segmentation toolkit. In IEEE International Symposium on Biomedical Imaging: From Nano to Macro 2011, 230–233.

Suleymanova I, Balassa T, Tripathi S, Molnar C, Saarma M, Sidorova Y, Horvath P (2018). A deep convolutional neural network approach for astrocyte detection. Sci Rep 8, 12878.

Sun Y, Wang X, Tang X, (2014). Deep learning face representation from predicting 10,000 classes. In 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1891—1898. doi: 10.1109/CVPR.2014.244.

Taigman Y, Yang M, Ranzato M'A, Wolf L (2014). DeepFace: closing the gap to human-level performance in face verification. In 2014 IEEE Conference on Computer Vision and Pattern Recognition. Available at https://doi.org/10.1109/cvpr.2014.220.

Supplemental Materials

Molecular Biology of the Cell

Hollandi *et al*.

Supplementary Material

Test data

We tested the efficiency in terms of both annotation time and accuracy on two test image sets: nucleus and cytoplasm. A sample of the test sets is displayed on **Figures 3B-C** and **S1B-C**; images are collected from publicly available datasets and our collaborators (see sources in (Hollandi et al. 2020)). We collected images from various modalities including different microscopy imaging (brightfield, fluorescent), sample origin (cell lines, histological tissue sections), staining technique (label-free, immunohistochemical or fluorescent markers for specific cellular compartments) that show the target object (nucleus or cytoplasm) in a heterogeneous environment so that we can test our plugin in general cases.

Evaluation

We used SEM (standard error of the mean) in our evaluation of annotation time and accuracy which is formulated as

$$SEM(x) = \frac{\sigma(x)}{\sqrt{N}} \#(3)$$

where the sample is denoted as x, $\sigma(x)$ marks the standard deviation of x and N is the number of elements in x.

We also tested annotation on a cytoplasm test image set containing more complex regions e.g. overlapping objects (see **Figure S1**). Our cytoplasm model aided annotation, however, due to its semantic nature it could not always separate touching objects properly hence the increased annotation time in certain cases.

Supplementary Figure S1. Annotation times on cytoplasm images. AnnotatorJ was tested on sample microscopy images (both fluorescent and brightfield, as well as cell culture and tissue section images), annotation time was measured on a per-object (cytoplasm) level. Bars represent the mean annotation times on the test image set, error bars show SEM (standard error of the mean). Orange corresponds to Contour assist mode and blue to only allowing the Automatic adding option. A) Cytoplasm annotation times, B) example histopathology test images, C) example cell culture images. Images shown on B-C) are 256x256 crops of original images. Some images are courtesy of Kerstin Elisabeth Dörner, Andreas Mund, Viktor Honti and Hella Bolck.

As any user-defined U-Net model can be easily imported in AnnotatorJ via DL4J, we tested general applicability on the Cityscapes dataset (Cordts et al., 2016): we trained a custom model on the car object class. Expert annotators were not needed for testing on these everyday objects. **Figure S2** shows annotation times and

example images. Acceleration in annotation time is also significant (similarly to the nucleus test set); contour suggestion helps the most in complex-shaped regions such as outside mirrors and bumpers.

Supplementary Figure S2. Annotation times on car images. AnnotatorJ was tested on sample images of the Cityscapes dataset (Cordts et al., 2016), annotation time was measured on a per-object (car) level. Bars represent the mean annotation times on the test image set, error bars show SEM (standard error of the mean). Orange corresponds to Contour assist mode and blue to only allowing the Automatic adding option. A) Car annotation times, B) example images as 1024x1024 crops of original images outlined by a randomly selected annotator.

Additionally, we tested AnnotatorJ using a model trained on electron microscopy (EM) data from the ISBI 2012 challenge (Arganda-Carreras et al., 2015) (fetched from https://github.com/zhixuhao/unet, which repository was also used to train our U-Net models in Keras) to help annotate neuronal cells (see **Fig. S3**). The complex shapes these structures had also emphasized the efficiency of our contour suggestion.

Supplementary Figure S3. Annotation times on electron microscopy (EM) images. AnnotatorJ was tested on sample images of the ISBI 2012 challenge (Arganda-Carreras et al., 2015), annotation time was measured on a per-object (cell) level. Bars represent the mean annotation times on the test image set, error bars show SEM (standard error of the mean). Orange corresponds to Contour assist mode and blue to only allowing the Automatic adding option. A) EM annotation times, B) example images as 256x256 crops of original images outlined by a randomly selected expert.

Additional methods

Active contours (AC)

Active contours (Kass et al., 1988) is a generally well applicable image processing technique to fit an initial contour to the boundaries of an object by evolving a so-called *snake* contour driven by 2 energy functions as follows.

$$E = \int_0^1 E_{int}\left[v(s)\right] ds + \int_0^1 E_{ext}\left[v(s)\right] ds \#(4)$$

$$E_{int}\left[v(s)\right] = \frac{1}{2}\left[\alpha(s)\left|\frac{\partial v(s)}{\partial s}\right|^2 + \beta(s)\left|\frac{\partial^2 v(s)}{\partial s^2}\right|^2\right] \#(5)$$

$$E_{ext}\left[v(s)\right] = -\gamma \cdot \left|I\big(x(s), y(s)\big) * \nabla^2 G_\sigma\right|^2 \#(6)$$

(4) is the combination of the internal (5) and external (6) energy functions, and is minimized during the iterative evolution process. The internal energy is responsible for extending and smoothing the contour while the external energy restrictively drives it towards the edges of the image which are enhanced by the LoG (Laplacian of Gaussian) filter-convolved image (data term). α, β and γ are the regularization parameters.

Gradient vector flow (GVF)

GVF was first published in (Xu and Prince 1997) as a method to define a vector field based on the image gradient, that is the derivatives of the image, as enhanced by edges. It is attracted to bright regions (ideally edges) on the image.

Additional annotation tools

Since open source or local software is of primary interest in medical research (see in Introduction and *Comparison to other tools and software packages*), we extend our previous list of compared tools here.

We briefly discussed supervise.ly (https://supervise.ly/) above, a service-based web-hosted solution for data labelling with DL model support. Training of various DL models including Mask R-CNN, U-Net, YOLO and others is possible. It supports annotation as bounding box, semantic- and polygonal drawing, editing points in polygons and class labelling. However, its free community edition intended for research purposes provides limited functionality.

FastAnnotationTool (https://github.com/christopher5106/FastAnnotationTool) is a lightweight, open source bounding box annotation tool with editing option in C++.

Make Sense (https://www.makesense.ai/) is another open source labelling tool that also adds DL support for annotation suggestion (using pre-trained models) and labelling. Point, polygon and bounding box annotation is possible, and classes can be assigned. Implemented in TypeScript, it offers both a web-based and a local (https://github.com/SkalskiP/make-sense) version.

ilastik (Sommer et al., 2011; Berg et al., 2019) is a pixel-based classification software that can be trained via brush strokes on the image. It can export semantic segmentation masks and their probability maps as well. However, these are returned by machine learning algorithms; users can annotate regions (pixel-based) with a brush or objects (instances) by single clicks to train them.

Suite2p (Pachitariu et al., 2017) was designed for a specific task: neuronal two-photon microscopy image analysis. It has a pipeline that can be modified in each step so that it fits the user's data, automatic detection and classification of ROIs (cells) can be manually curated.

LabKit (Arzt 2017) is an ImageJ plugin for machine learning-supported labelling that similarly to the Trainable Weka Segmentation (ImageJ) plugin (Arganda-Carreras et al., 2017) and ilastik (Sommer et al., 2011; Berg et al., 2019) uses user-defined regions marked by brush strokes to train a machine learning algorithm and automatically segment image pixels to the defined classes. Both plugins offer several machine learning algorithms implemented in the Weka framework (Frank et al., 2016). Trainable Weka Segmentation can save semantic segmentation masks or their probability maps (like ilastik) while LabKit can also save manually drawn objects as multi-labelled .tiff files (like AnnotatorJExporter).

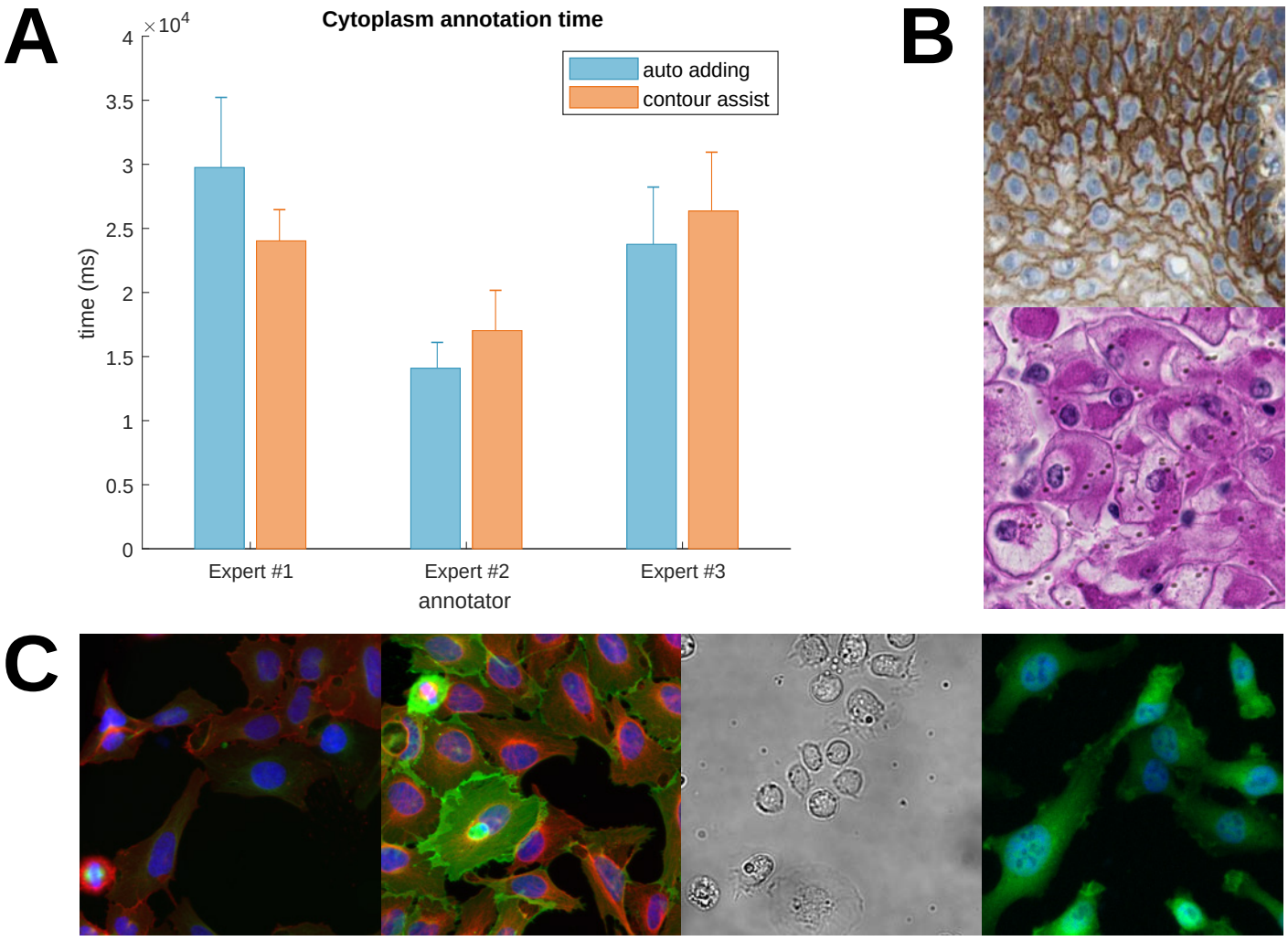
We collected the sources of annotation tools in Table S1.

Table S1. Sources of the compared software tools.

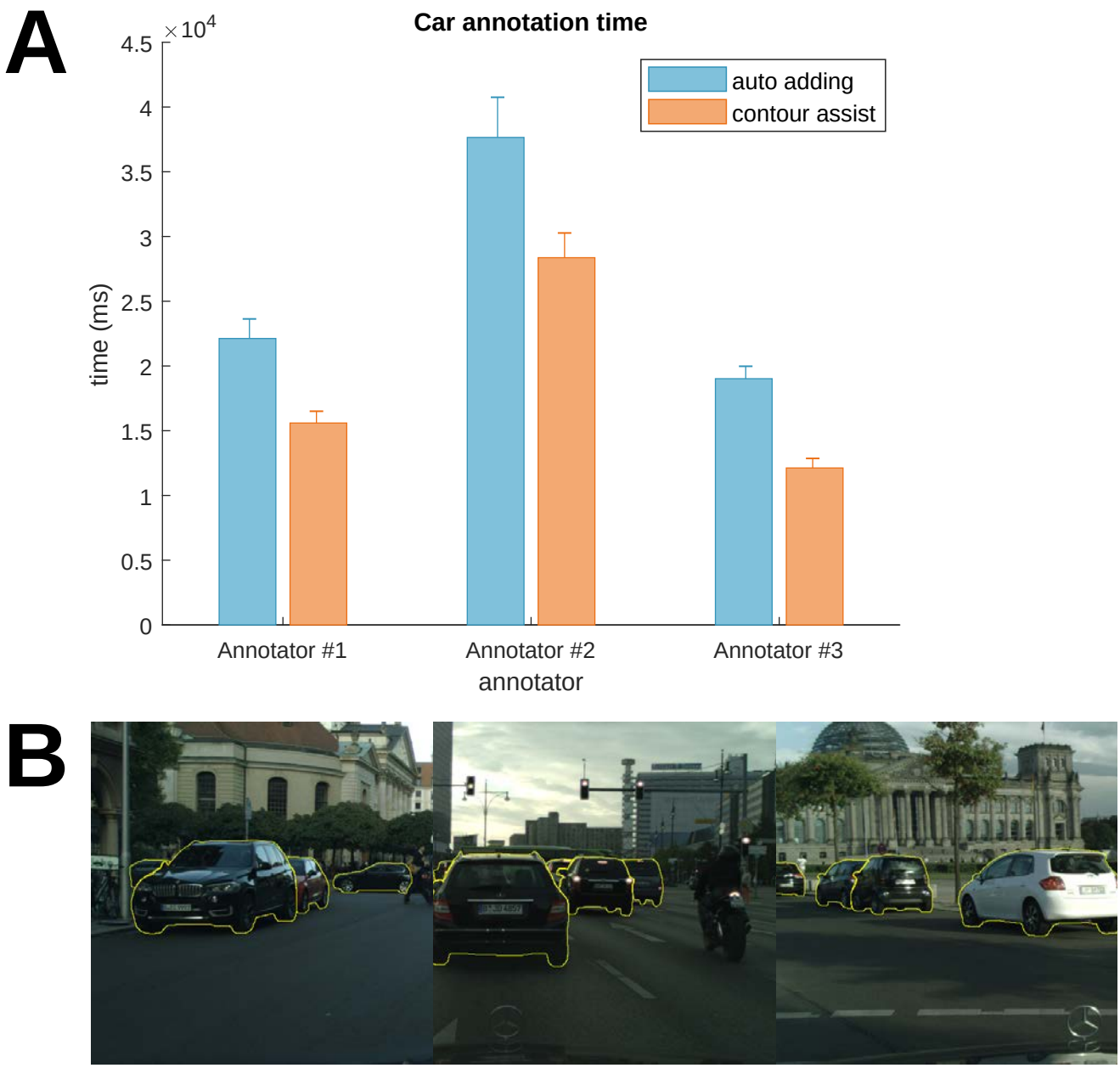
tool	source
LabelImg	https://github.com/tzutalin/labelImg
Lionbridge.AI	https://lionbridge.ai/services/image-annotation/
Hive	https://thehive.ai/hive-data
VGG Image Annotator	http://www.robots.ox.ac.uk/~vgg/software/via/via-1.0.6.html
Diffgram	https://diffgram.com/ https://github.com/diffgram/diffgram/tree/master/sdk
CytoMine	http://www.cytomine.org/ https://github.com/cytomine/Cytomine-bootstrap
SlideRunner	https://github.com/maubreville/SlideRunner
supervise.ly	https://supervise.ly/
FastAnnotationTool	https://github.com/christopher5106/FastAnnotationTool
Make Sense	https://www.makesense.ai/ https://github.com/SkalskiP/make-sense
ilastik	https://www.ilastik.org/ https://github.com/ilastik/ilastik
Suite2p	https://github.com/MouseLand/suite2p https://github.com/cortex-lab/Suite2P
LabKit	https://imagej.net/Labkit http://sites.imagej.net/Labkit/ https://github.com/maarzt/imglib2-labkit
Trainable Weka Segmentation	https://imagej.net/Trainable_Weka_Segmentation

Supplementary video

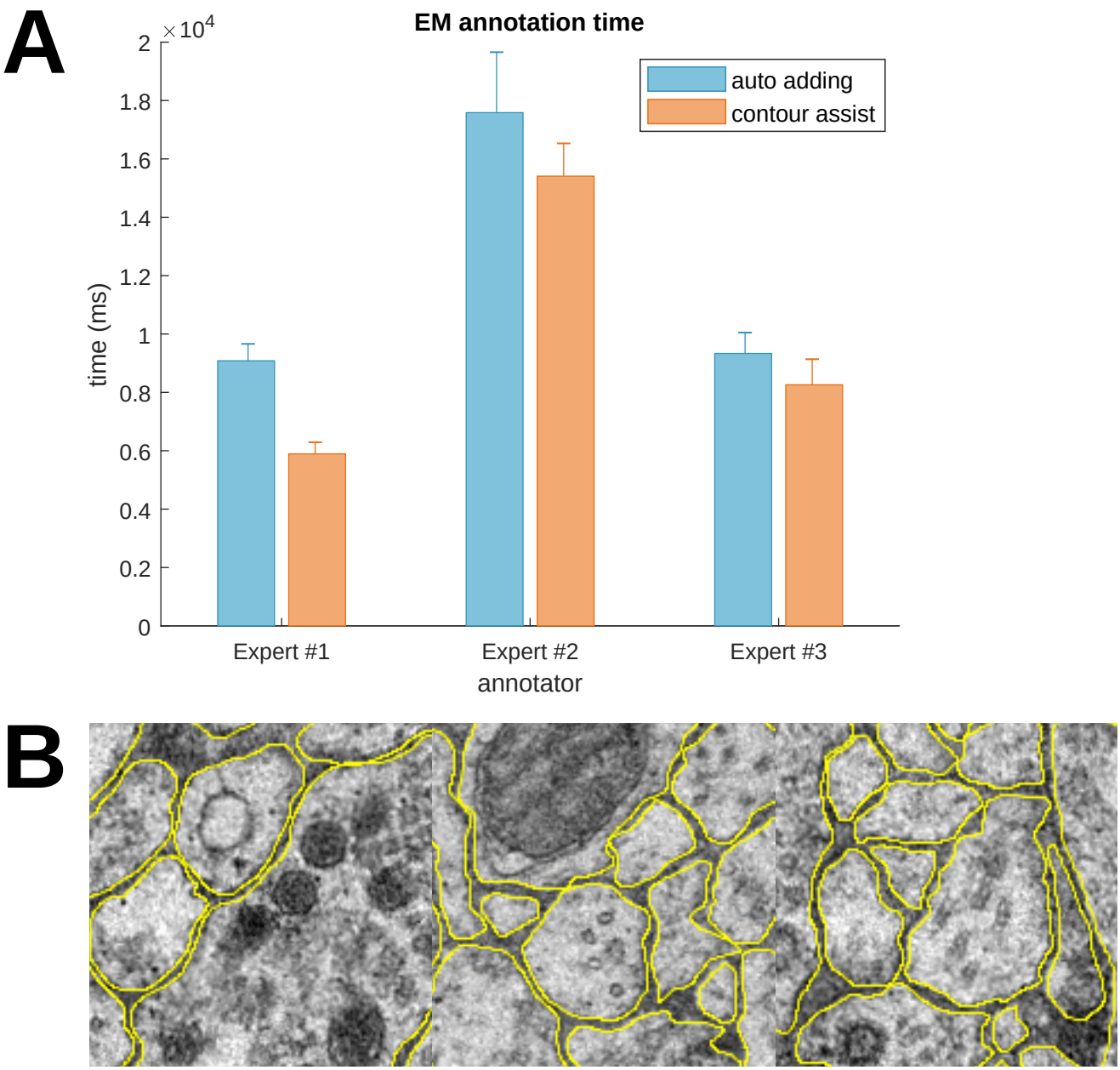
A short demonstration of *Contour assist* is provided as the video *figure2.mov*; available in higher resolution at https://drive.google.com/file/d/1dk3FrX-KIhTpaNYSKv-zVsoAoVUGtEYp/view?usp=sharing.



Supplementary Figure S1. Annotation times on cytoplasm images. AnnotatorJ was tested on sample microscopy images (both fluorescent and brightfield, as well as cell culture and tissue section images), annotation time was measured on a per-object (cytoplasm) level. Bars represent the mean annotation times on the test image set, error bars show SEM (standard error of the mean). Orange corresponds to Contour assist mode and blue to only allowing the Automatic adding option. A) Cytoplasm annotation times, B) example histopathology test images, C) example cell culture images. Images shown on B-C) are 256x256 crops of original images. Some images are courtesy of Kerstin Elisabeth Dörner, Andreas Mund, Viktor Honti and Hella Bolck.



Supplementary Figure S2. Annotation times on car images. AnnotatorJ was tested on sample images of the Cityscapes dataset (Cordts et al., 2016), annotation time was measured on a per-object (car) level. Bars represent the mean annotation times on the test image set, error bars show SEM (standard error of the mean). Orange corresponds to Contour assist mode and blue to only allowing the Automatic adding option. A) Car annotation times, B) example images as 1024x1024 crops of original images outlined by a randomly selected annotator.



Supplementary Figure S3. Annotation times on electron microscopy (EM) images. AnnotatorJ was tested on sample images of the ISBI 2012 challenge (Arganda-Carreras et al., 2015), annotation time was measured on a per-object (cell) level. Bars represent the mean annotation times on the test image set, error bars show SEM (standard error of the mean). Orange corresponds to Contour assist mode and blue to only allowing the Automatic adding option. A) EM annotation times, B) example images as 256x256 crops of original images outlined by a randomly selected expert.

III.

frontiers in Bioengineering and Biotechnology

TECHNOLOGY AND CODE
published: 06 October 2020
doi: 10.3389/fbioe.2020.558880

OpSeF: Open Source Python Framework for Collaborative Instance Segmentation of Bioimages

Tobias M. Rasse[1], Réka Hollandi[2] and Peter Horvath[2,3]*

[1] Scientific Service Group Microscopy, Max Planck Institute for Heart and Lung Research, Bad Nauheim, Germany, [2] Synthetic and Systems Biology Unit, Biological Research Center (BRC), Szeged, Hungary, [3] Institute for Molecular Medicine Finland (FIMM), University of Helsinki, Helsinki, Finland

Various pre-trained deep learning models for the segmentation of bioimages have been made available as developer-to-end-user solutions. They are optimized for ease of use and usually require neither knowledge of machine learning nor coding skills. However, individually testing these tools is tedious and success is uncertain. Here, we present the Open Segmentation Framework (OpSeF), a Python framework for deep learning-based instance segmentation. OpSeF aims at facilitating the collaboration of biomedical users with experienced image analysts. It builds on the analysts' knowledge in Python, machine learning, and workflow design to solve complex analysis tasks at any scale in a reproducible, well-documented way. OpSeF defines standard inputs and outputs, thereby facilitating modular workflow design and interoperability with other software. Users play an important role in problem definition, quality control, and manual refinement of results. OpSeF semi-automates preprocessing, convolutional neural network (CNN)-based segmentation in 2D or 3D, and postprocessing. It facilitates benchmarking of multiple models in parallel. OpSeF streamlines the optimization of parameters for pre- and postprocessing such, that an available model may frequently be used without retraining. Even if sufficiently good results are not achievable with this approach, intermediate results can inform the analysts in the selection of the most promising CNN-architecture in which the biomedical user might invest the effort of manually labeling training data. We provide Jupyter notebooks that document sample workflows based on various image collections. Analysts may find these notebooks useful to illustrate common segmentation challenges, as they prepare the advanced user for gradually taking over some of their tasks and completing their projects independently. The notebooks may also be used to explore the analysis options available within OpSeF in an interactive way and to document and share final workflows. Currently, three mechanistically distinct CNN-based segmentation methods, the U-Net implementation used in Cellprofiler 3.0, StarDist, and Cellpose have been integrated within OpSeF. The addition of new networks requires little; the addition of new models requires no coding skills. Thus, OpSeF might soon become both an interactive model repository, in which pre-trained models might be shared, evaluated, and reused with ease.

Keywords: deep learning, biomedical image analysis, segmentation, convolutional neural network, U-net, cellpose, StarDist, python

OPEN ACCESS

Edited by:
Mehdi Pirooznia, National Heart, Lung, and Blood Institute (NHLBI), United States

Reviewed by:
Alexandr Kalinin, Shenzhen Research Institute of Big Data, China
Hady Ahmady Phoulady, California State University, Sacramento, United States

****Correspondence:***
Tobias M. Rasse
tobias.rasse@mpi-bn.mpg.de

Specialty section:
This article was submitted to Computational Genomics, a section of the journal Frontiers in Bioengineering and Biotechnology

Received: *04 May 2020*
Accepted: *15 September 2020*
Published: *06 October 2020*

Citation:
Rasse TM, Hollandi R and Horvath P (2020) OpSeF: Open Source Python Framework for Collaborative Instance Segmentation of Bioimages. Front. Bioeng. Biotechnol. 8:558880. doi: 10.3389/fbioe.2020.558880

INTRODUCTION

Phenomics, the assessment of the set of physical and biochemical properties that completely characterize an organism, has long been recognized as one of the most significant challenges in modern biology (Houle et al., 2010). Microscopy is a crucial technology to study phenotypic characteristics. Advances in high-throughput microscopy (Lang et al., 2006; Neumann et al., 2010; Chessel and Carazo Salas, 2019), slide scanner technology (Webster and Dunstan, 2014; Wang et al., 2019), light-sheet microscopy (Swoger et al., 2014; Ueda et al., 2020), semi-automated (Bykov et al., 2019; Schorb et al., 2019) and volume electron microscopy (Titze and Genoud, 2016; Vidavsky et al., 2016), as well as correlative light- and electron microscopy (Hoffman et al., 2020) have revolutionized the imaging of organisms, tissues, organoids, cells, and subcellular structures. Due to the massive amount of data produced by these approaches, the traditional biomedical image analysis tool of "visual inspection" is no longer feasible, and classical, non-machine learning-based image analysis is often not robust enough to extract phenotypic characteristics reliably in a non-supervised manner.

Thus, the advances mentioned above were enabled by breakthroughs in the application of machine learning methods to biological images. Traditional machine learning techniques, based on random-forest classifiers and support vector machines, were made accessible to biologists with little to no knowledge in machine learning, using stand-alone tools such as ilastik (Haubold et al., 2016; Berg et al., 2019; Kreshuk and Zhang, 2019) or QuPath (Bankhead et al., 2017). Alternatively, they were integrated into several image analysis platforms such as Cellprofiler (Lamprecht et al., 2007), Cellprofiler Analyst (Jones et al., 2009), Icy (de Chaumont et al., 2012), ImageJ (Schneider et al., 2012; Arganda-Carreras et al., 2017) or KNIME (Sieb et al., 2007).

More recently, deep learning methods, initially developed for computer vision challenges, such as face recognition or autonomous cars, have been applied to biomedical image analysis (Cireşan et al., 2012, 2013). The U-Net is the most commonly used deep convolutional neural network specifically designed for semantic segmentation of biomedical images (Ronneberger et al., 2015). In the following years, neural networks were broadly applied to biomedical images (Zhang et al., 2015; Akram et al., 2016; Albarqouni et al., 2016; Milletari et al., 2016; Moeskops et al., 2016; Van Valen et al., 2016; Çiçek et al., 2016; Rajchl et al., 2017). Segmentation challenges like the 2018 Data Science Bowl (DSB) further promoted the adaptation of computer vision algorithms like Mask R-CNN (He et al., 2017) to biological analysis challenges (Caicedo et al., 2019). The DSB included various classes of nuclei. Schmidt et al. use the same dataset to demonstrate that star-convex polygons are better suited to represent densely packed cells (Schmidt et al., 2018) than axis-aligned bounding boxes used in Mask R-CNN (Hollandi et al., 2020b). Training of deep learning models typically involves tedious annotation to create ground truth labels. Approaches that address this limitation include optimizing the annotation workflow by starting with reasonably good predictions (Hollandi et al., 2020a), applying specific preprocessing steps such that an existing model can be used (Whitehead, 2020), and the use of generalist algorithms trained on highly variable images (Stringer et al., 2020). Following the latter approach, Stringer et al. trained a neural network to predict vector flows generated by the reversible transformation of a highly diverse image collection. Their model includes a function to auto-estimate the scale. It works well for specialized and generalized data (Stringer et al., 2020).

Recently, various such pre-trained deep learning segmentation models have been published that are intended for non-machine learning experts in the field of biomedical image processing (Schmidt et al., 2018; Hollandi et al., 2020b; Stringer et al., 2020). Testing such models on new data sets can be time-consuming and might not always give good results. Pre-trained models might fail because the test images do not resemble the data network was trained on sufficiently well. Alternatively, the underlying network architecture and specification, or the way data is internally represented and processed might not be suited for the presented task. Biomedical users with no background in computer science are often unable to distinguish these possibilities. They might erroneously conclude that their problem is in principle not suited for deep learning-based segmentation. Thus, they might hesitate to create annotations to re-train the most appropriate architecture. Here, we present the **Op**en **Se**gmentation **F**ramework OpSeF, a Python framework for deep-learning-based instance segmentation of cells and nuclei. OpSeF has primarily been developed for staff image analysts with solid knowledge in image analysis, thorough understating of the principles of machine learning, and basic skills in Python. It wraps scikit-image, a collection of Python algorithms for image processing (van der Walt et al., 2014), the U-Net implementation used in Cellprofiler 3.0 (McQuin et al., 2018), StarDist (Schmidt et al., 2018; Weigert et al., 2019, 2020), and Cellpose (Stringer et al., 2020) in a single framework. OpSeF defines the standard in- and outputs, facilitates modular workflow design, and interoperability with other software (Weigert et al., 2020). Moreover, it streamlines and semi-automates preprocessing, CNN-based segmentation, postprocessing as well as evaluation of results. Jupyter notebooks (Kluyver et al., 2016) serve as a minimal graphical user interface. Most computations are performed head-less and can be executed on local workstations as well as on GPU clusters. Segmentation results can be easily imported and refined in ImageJ using AnnotatorJ (Hollandi et al., 2020a).

MATERIALS AND METHODS

Data Description

Cobblestones

Images of cobblestones were taken with a Samsung Galaxy S6 Active Smartphone.

Leaves

Noise was added to the demo data from "YAPiC - Yet Another Pixel Classifier" available at https://github.com/

yapic/yapic/tree/master/docs/example_data using the *Add Noise* function in ImageJ.

Small Fluorescent Nuclei

Images of Hek293 human embryonic kidney stained with a nuclear dye from the image set BBBC038v1 (Caicedo et al., 2019) available from the Broad Bioimage Benchmark Collection (BBBC) were used. Metadata is not available for this image set to confirm staining conditions. Images were rescaled from 360 × 360 pixels to 512 × 512 pixels.

3D Colon Tissue

We used the low signal-to-noise variant of the image set BBBC027 (Svoboda et al., 2011) from the BBBC showing 3D colon tissue images.

Epithelial Cells

Images of cervical cells from the image set BBBC038v1 (Caicedo et al., 2019) available from the BBBC display cells stained with a dye that labels membranes weakly and nuclei strongly. The staining pattern is reminiscent of images of methylene blue-stained cells. However, metadata is not available for this image set to confirm staining conditions.

Skeletal Muscle

A methylene blue-stained skeletal muscle section was recorded on a Nikon Eclipse Ni-E microscope equipped with a Märzhäuser SlideExpress2 system for automated handling of slides. The pixel size is 0.37 × 0.37 μm. Thirteen large patches of 2048 × 2048 pixels size were manually extracted from the original 44712 × 55444 pixels large image. Color images were converted to grayscale.

Kidney

HE stained kidney paraffin sections were recorded on a Nikon Eclipse Ni-E microscope equipped with a Märzhäuser SlideExpress2 system for automated handling of slides. The pixel size is 180 × 180 nm. The original, stitched, 34816 × 51200 pixels large image was split into two large patches (18432 × 6144 and 22528 × 5120 pixel). Next, the Eosin staining was extracted using the *Color Deconvolution* ImageJ plugin. This plugin implements the method described by Ruifrok and Johnston (2001).

Arabidopsis Flowers

H2B:mRuby2 was used for the visualization of somatic nuclei of *Arabidopsis thaliana* flower. The flower was scanned from eight views differing by 45° increments in a Zeiss Z1 light-sheet microscope (Valuchova et al., 2020). We used a single view to mimic a challenging 3D segmentation problem. Image files are available in the Image Data Resource (Williams et al., 2017) under the accession code: idr0077.

Mouse Blastocysts

The DAPI signal from densely packed E3.5 mouse blastocysts nuclei was recorded on a Leica SP8 confocal microscope using a 40× 1.30 NA oil objective (Blin et al., 2019). Image files are available in the Image Data Resource (Williams et al., 2017) under the accession code: idr0062.

Neural Monolayer

The DAPI signal of a neural monolayer was recorded on a Leica SpE confocal microscope using a 63× 1.30 NA oil objective (Blin et al., 2019). Image files are available in the Image Data Resource (Williams et al., 2017) under the accession code: idr0062.

Algorithm

Ideally, OpSeF is used as part of collaborative image analysis projects, to which both the user and the image analyst contribute their unique expertise (**Figure 1A**). All analyst tasks are optimized for deployment on Linux workstations or GPU clusters, all user tasks may be performed on any laptop in ImageJ. If challenges arise, the image analyst (**Figure 1A**) might consult other OpSeF users or the developer of tools used within OpSeF. The analyst will – to the benefit of future users – become more skilled using CNN-based segmentation in analysis workflows. The user, who knows the sample best, plays an important role in validating results and discovering artifacts (**Figure 1A**). Exemplary workflows and new models might be shared to the benefit of other OpSeF users (**Figure 1B**).

OpSeF's analysis pipeline consists of four principal sets of functions to *import and reshape* the data, to *preprocess* it, to *segment* objects, and to *analyze and classify* results (**Figure 1C**). Currently, OpSeF can process individual tiff files and the proprietary Leica '.lif' container file format. During *import and reshape*, the following options are available for tiff-input: *tile* in 2D and 3D, *scale*, and make *sub-stacks*. For lif-files, only the make *sub-stacks* option is supported. Preprocessing is mainly based on scikit-image (van der Walt et al., 2014). It consists of a linear workflow in which 2D images are filtered, the background is removed, and stacks are projected. Next, the following optional preprocessing operations might be performed: histogram adjustment (Zuiderveld, 1994), edge enhancement, and inversion of images. Available segmentation options include the pre-trained U-Net used in Cellprofiler 3.0 (McQuin et al., 2018), the so-called "2D_paper_dsb2018" StarDist model (Schmidt et al., 2018) and Cellpose (Stringer et al., 2020). The 2D StarDist model (Schmidt et al., 2018) was trained on a subset of fluorescent images from the 2018 Data Science Bowl (DSB) (Caicedo et al., 2019). Although good performance on non-fluorescent images cannot be taken for granted, the StarDist versatile model, which was trained on the same data, generalizes well and can be used to segment cells in diaminobenzidene and hematoxylin stained tissue sections (Whitehead, 2020). We thus used the 2D_paper_dsb2018 StarDist model for all 2D examples. Available options for preprocessing in 3D are limited (**Figure 1B**, lower panel). Segmentation in 3D is computationally more demanding. Thus, we recommend a two-stage strategy for Cellpose 3D. Preprocessing parameters are first explored on representative planes in 2D. Next, further optimization in 3D is performed. Either way, preprocessing and selection of the ideal model for segmentation are one functional unit. **Figure 1D** illustrates this concept with a processing pipeline, in which three different models are applied to four different preprocessing pipelines each. The resulting images are classified

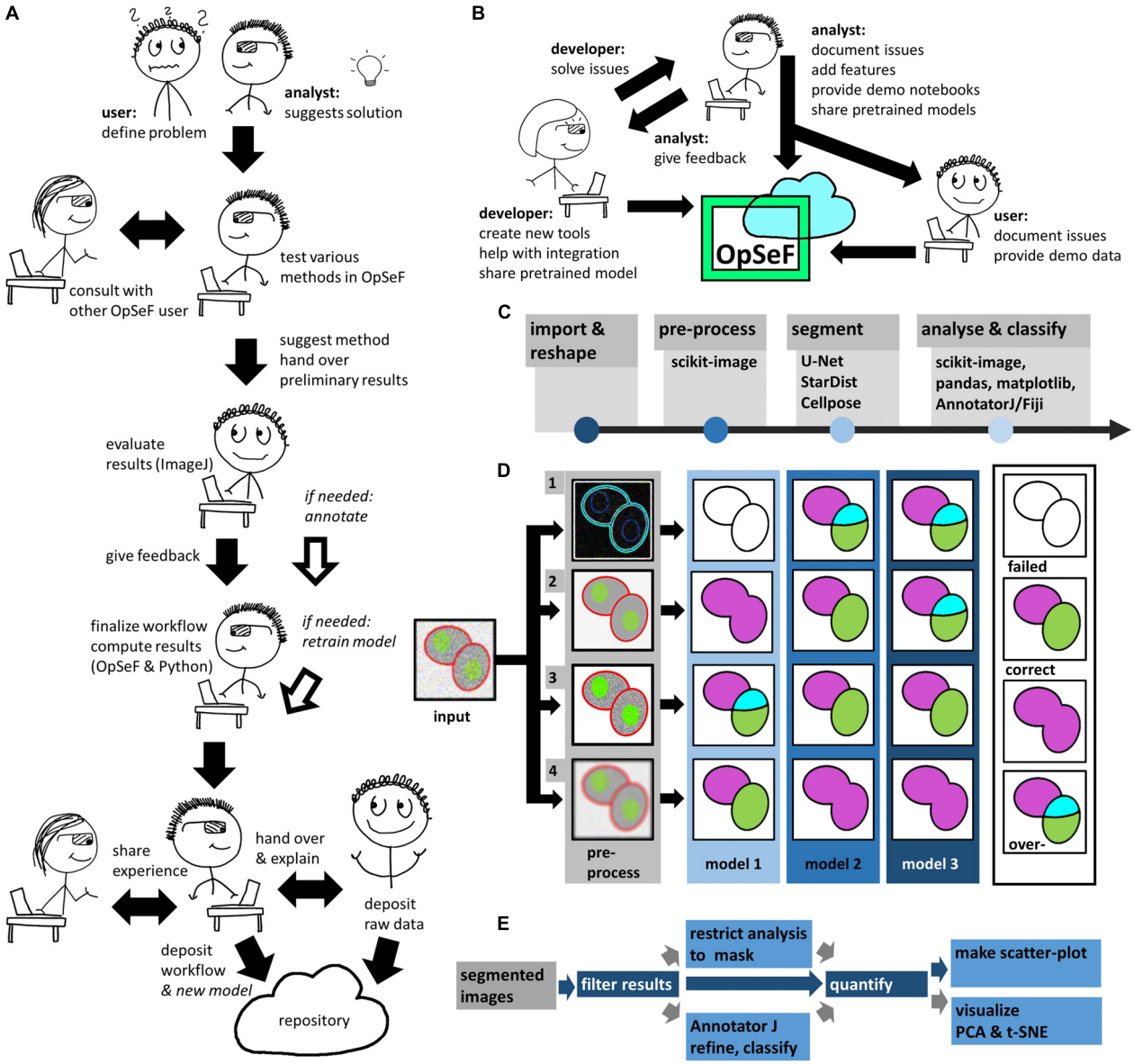


FIGURE 1 | Image analysis using OpSeF **(A)** Illustration on how to use OpSeF for collaborative image analysis **(B)** Illustration on how developers, image analysts and users might contribute toward the further development of OpSeF. **(C)** OpSeF analysis pipeline consists of four groups of functions used to *import and reshape* the data, to *preprocess* it, to *segment objects*, and to *analyze and classify* results. **(D)** Optimization procedure. Left panel: Illustration of a processing pipeline, in which three different models are applied to data generated by four different preprocessing pipelines each. Right panel: Resulting images are classified into results that are correct; suffer from under- or over-segmentation or fail to detect objects. **(E)** Illustration of postprocessing pipeline. Segmented objects might be filtered by their region properties or a mask, results might be exported to AnnotatorJ and re-imported for further analysis. Blue arrows define the default processing pipeline, gray arrows feature available options. Dark blue boxes are core components, light blue boxes are optional processing steps.

into results that are mostly correct, suffer from under- or over-segmentation, or largely fail to detect objects. In the given example, the combination of preprocessing pipeline three and model two gives overall the best result. We recommend an iterative optimization which starts with a large number of models, and relatively few, but conceptually different preprocessing pipelines. For most datasets, some models outperform others. In this case, we recommend fine-tuning the most promising preprocessing pipelines in combination with the most promising model. OpSeF uses matplotlib (Virtanen et al., 2020) to visualize results in Jupyter notebooks and to export exemplary results that may be used as figures in publications. All data is managed in pandas (Virtanen et al., 2020) and might be exported as csv file. Scikit-image (van der Walt et al., 2014), and scikit-learn (Pedregosa et al., 2011; **Figures 1E**, **2A–C**) are used for pre- and postprocessing of segmentation results, which might e.g., be filtered based on their size, shape or other object properties (**Figure 2B**). Segmentation objects may further be

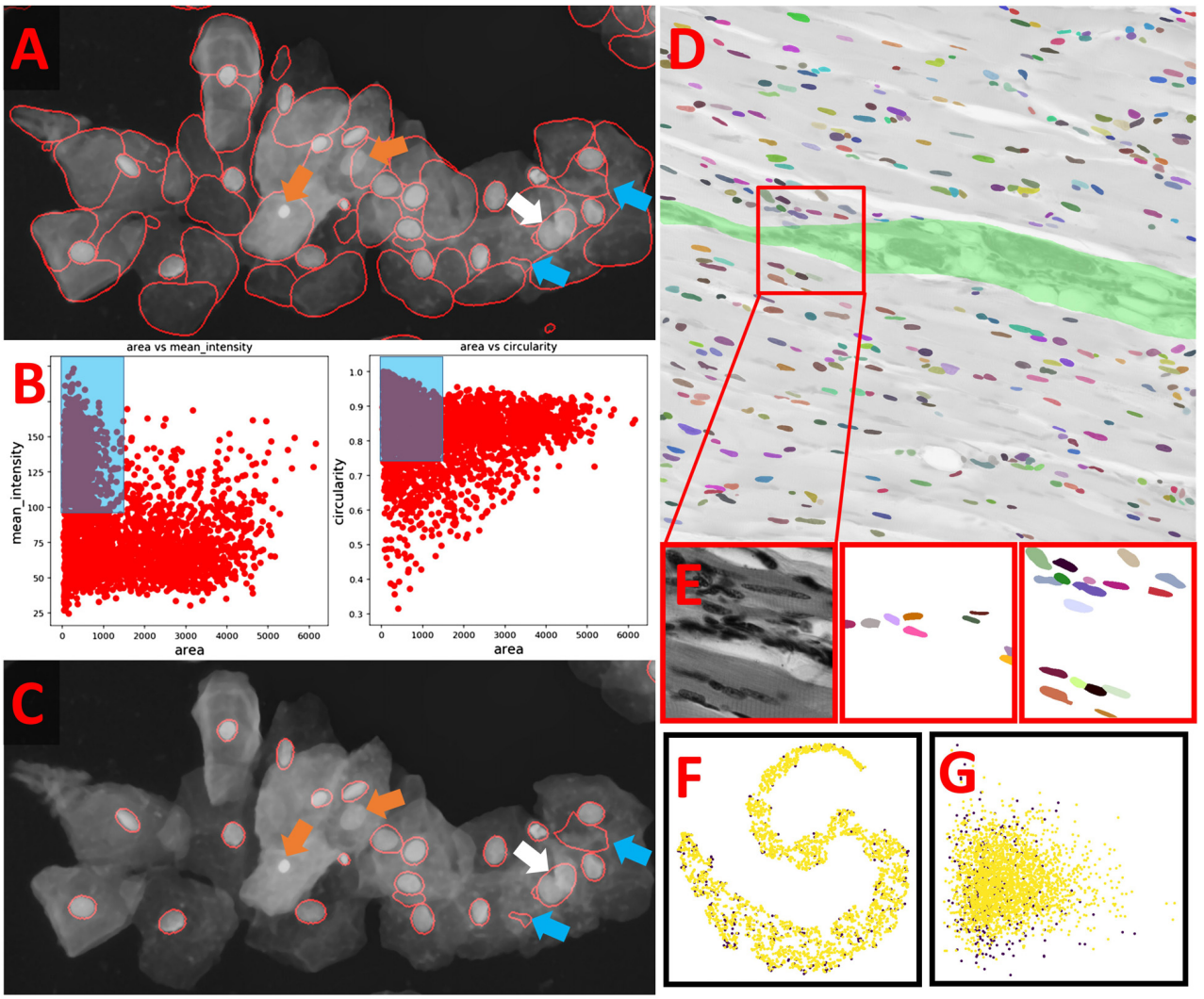


FIGURE 2 | Example of how postprocessing can be used to refine results **(A)** StarDist segmentation of the multi-labeled cells dataset detected nuclei reliably but caused many false positive detections. These resemble the typical shape of cells but are larger than true nuclei. Orange arrows point at nuclei that were missed, the white arrow at two nuclei that were not split, the blue arrows at false positive detections that could not be removed by filtering. **(B)** Scatter plot of segmentation results shown in panel **(A)**. Left panel: Mean intensity plotted against object area. Right panel: Circularity plotted against object area. Blue Box illustrates the parameter used to filter results. **(C)** Filtered Results. Orange arrows point at nuclei that were missed, the white arrow at two nuclei that were not split, the blue arrows at false positive detections that could not be removed by filtering. **(D,E)** Example for the use of a user-provided mask to classify segmented objects. The segmentation results (false-colored nuclei) are superimposed onto the original image subjected to [median 3 × 3] preprocessing. All nuclei located in the green area are assigned to Class 1, all others to Class 2. The red box indicates the region shown enlarged in panel **(E)**. From left to right in panel **(E)**: original image, nuclei assigned to class 1, nuclei assigned to class 2. **(F)** T-distributed Stochastic Neighbor Embedding (t-SNE) analysis of nuclei assigned to class 1 (purple) or class 2 (yellow). **(G)** Principal component analysis (PCA) of nuclei assigned to class 1 (purple) or class 2 (yellow).

refined by a user-provided (**Figures 2D,E**) or an autogenerated mask. Results might be exported to AnnotatorJ (Hollandi et al., 2020a) for editing or classification in ImageJ. AnnotatorJ is an ImageJ plugin that helps hand-labeling data with deep learning-supported semi-automatic annotation and further convenient functions to create and edit object contours easily. It has been extended with a classification mode and import/export fitting the data structure used in OpSeF. After refinement, results can be re-imported and further analyzed in OpSeF. Analysis options include scatter plots of region properties (**Figure 2B**), T-distributed Stochastic Neighbor Embedding (t-SNE) analysis (**Figure 2F**), and principal component analysis (PCA) (**Figure 2G**).

RESULTS

We provide demonstration notebooks to illustrate how OpSeF might be used to elucidate efficiently whether a given segmentation task is solvable with state of the art deep

convolutional neural networks (CNNs). In the first step, preprocessing parameters are optimized. Next, we test whether the chosen model performs well without re-training. Finally, we assess how well it generalizes on heterogeneous datasets.

Preprocessing can be used to make the input image more closely resemble the visual appearance of the data on which the models for the CNNs bundled with OpSeF were trained on, e.g., by filtering and resizing. Additionally, preprocessing steps can be used to normalize data and reduce heterogeneity. Generally, there is not a single, universally best preprocessing pipeline. Instead, well-performing combinations of preprocessing pipelines and matching CNN-models can be identified. Even the definition of a "good" result depends on the biological question posed and may vary from project to project. For cell tracking, very reproducible cell identification will be of utmost importance; for other applications, the accuracy of the outline might be more crucial. To harness the full power of CNN-based segmentation models and to build trust in their more widespread use, it is essential to understand under which conditions they are prone to fail.

We use various demo datasets to challenge the CNN-based segmentation pipelines. Jupyter notebooks document how OpSeF was used to obtain reliable results. These notebooks are provided as a starting point for the iterative optimization of user projects and as a tool for interactive user training.

The first two datasets – cobblestones and leaves – are generic, non-microscopic image collections, designed to illustrate common analysis challenges. Further datasets exemplify the segmentation of a monolayer of fluorescent cells, fluorescent tissues, cells in which various compartments have been stained with the same dye, as well as histological sections stained with one or two dyes. The latter dataset exemplifies additionally how OpSeF can be used to process large 2D images.

Nuclei and cells used to train CNN-based segmentation are most commonly round or ellipsoid shaped. Objects in the cobblestone dataset are approximately square-shaped. Thus, the notebook may be used as an example to explore the segmentation of non-round cells (e.g., many plant cells, neurons). Heterogeneous intensities within objects and in the border region, as well as a five-fold variation of object size, challenge segmentation pipelines further. In the first round of optimization, minor smoothing [median filter with 3 × 3 kernel (median 3 × 3)] and background subtraction were applied. Next, the effect of additional histogram equalization, edge enhancement, and image inversion was tested. The resulting four preprocessed images were segmented with all models [Cellpose nuclei, Cellpose Cyto, StarDist, and U-Net]. The Cellpose scale-factor range [0.2, 0.4, 0.6] was explored. Among the 32 resulting segmentation pipelines, the combination of image inversion and the Cellpose Cyto 0.4 model produced the best results in both training images (**Figures 3A–C**) without further optimization. The segmentation generalized well to the entire dataset. Only in one image, three objects were missed, and one object was over-segmented. Borders around these stones are very hard to distinguish for a human observer, and even further training might not resolve the presented segmentation tasks (**Figures 3D–F**). Cellpose has been trained on a large variety of images and had been reported to perform well on objects of similar shape [compare **Figure 4**, Images 21, 22, 27 in Stringer et al. (2020)]. Thus, it is no surprise that Cellpose outperformed the StarDist 2D model (Schmidt et al., 2018), which had been trained only on fluorescent images.

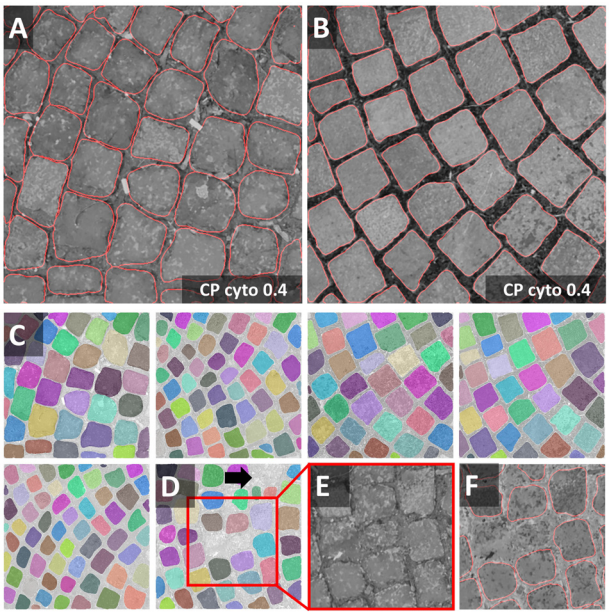


FIGURE 3 | Cobblestones notebook: Segmentation of non-roundish cells **(A,B)** Segmentations (red line) of the Cellpose Cyto 0.4 model are superimposed onto the original image subjected to [median 3 × 3] preprocessing. The inverted image (not shown) was used as input to the segmentation. Outlines are well defined, no objects were missed, none over-segmented. These settings fit accurately to the entire dataset (train and test) shown in panels **(C,D)**. Only in one image, three objects were missed and one was over-segmented. Borders around these stones are hard to discern. Individual objects are false color-coded in panels **(C,D)**. The red squares in panel **(D)** highlight one of the two problematic regions shown as a close-up in panels **(E,F)**.

Segmentation of the leaves dataset seems trivial and could easily be solved by any threshold-based approach. Nevertheless, it challenges CNN-based segmentation due to the presence of concave and convex shapes. Moreover, objects contain dark lines, vary 20-fold in area, and are presented on a heterogeneous background. Preprocessing was performed as described for the cobblestone dataset. The most promising result was obtained with the Cellpose Cyto 0.5 model in combination with [median 3 × 3 & image inversion] preprocessing (**Figures 4A,B**) and the StarDist model with [median 3 × 3 & histogram equalization] preprocessing (**Figure 4C**). Outlines were well defined, few objects were missed (blue arrow in **Figure 4A**), few over-segmented (green and orange arrow in **Figures 4B,C**). The Cellpose Cyto 0.7 model gave similar results.

Maple leaves (orange arrows in **Figures 4B,C**) were most frequently over-segmented. Their shape resembles a cluster of touching cells. Thus, the observed over-segmentation might be caused by the attempt of the CNN to reconcile their shapes with structures it has been trained on. Oak leaves were the second most frequently over-segmented leaf type. These leaves contain dark leaf veins that might be interpreted as

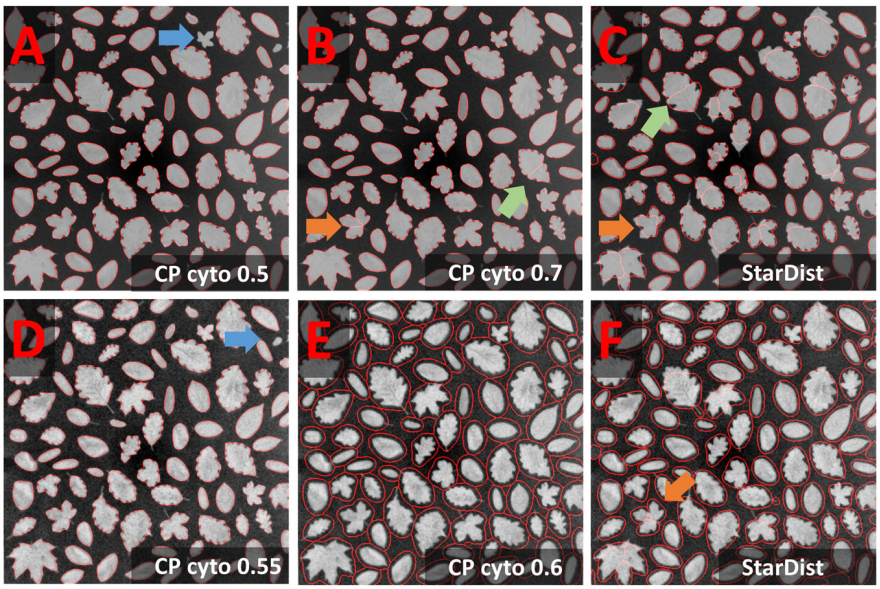


FIGURE 4 | Leaves notebook: Segmentation of rare concave cells **(A–C)** Segmentation results (red line) of the Cellpose Cyto 0.5 and 0.7 and StarDist model are superimposed onto the original image subjected to [median 3 × 3] preprocessing. The inverted image (not shown) was used as input to the segmentation. Outlines are well defined, few objects were missed (**A:** blue arrow), some over-segmented (**B,C:** green and orange arrow). Green arrow points at an oak leave with prominent leaf veins, orange arrow at maple leaves with less prominent leave veins. **(D–F)** Results of further optimization. Further smoothing **(E)** reduced the rate of false negatives (blue arrow) and over-segmentation in the Cellpose Cyto model. However, object outlines were less precise **(E)**.

cell borders. However, erroneous segmentation mostly does not follow these veins (green arrow in **Figure 4B**). Next, the effect of stronger smoothing [mean 7 × 7] was explored. The Cellpose nuclei model (**Figure 4E**) reduced the rate of false-negative detections (**Figure 4D** blue arrow) and over-segmentation (**Figure 4F** orange arrow) at the expense of loss in precision of object outlines. Parameter combinations tested in **Figures 4D,E** generalize well in the entire dataset.

Next, we used OpSeF to segment nuclei in a monolayer of cells. Most nuclei are well separated. We focused our analysis on the few touching nuclei. Both the Cellpose nuclei model and the Cellpose Cyto model performed well across a broad range of scale-factors. Interestingly, strong smoothing made the Cellpose nuclei but not the Cellpose Cyto model more prone to over-segmentation (**Figure 5A**). The StarDist model performed well, while the U-Net failed surprisingly, given the seemingly simple task. Pixel intensities have a small dynamic range, and nuclei are dim and rather large. To elucidate whether any of these issues led to this poor performance, we binned the input 2 × 2 (U-Net+BIN panel in **Figure 5A**) and adjusted brightness and contrast. Adjusting brightness and contrast alone had no beneficial effect (data not shown). The U-Net performed much better on the binned input. Subsequently, we batch-processed the entire dataset. StarDist was more prone to over-segmentation (green arrow in **Figure 5B**), but detected smaller objects more faithfully (orange arrow in **Figure 5B**). This might indicate that the size of test objects was larger than the size of train objects. StarDist was more likely to include atypical objects, e.g., nuclei during cell division that display a strong texture (blue arrow in **Figure 5B**). Substantial variation in brightness was well tolerated by both models (white arrow in **Figure 5B**). Both models complement each other well.

We also tested a more complex dataset: 3D colon tissue from the Broad Bioimage Benchmark Collection. This synthetic dataset is ideally suited to assess segmenting clustered nuclei in tissues. We chose the low signal-to-noise variant, which allowed us to test denoising strategies. Sum, maximum, and median projection of three Z-planes was tested in combination with the preprocessing variants previously described for the monolayer of cells dataset. Twelve different preprocessing pipelines were combined with all models [Cellpose nuclei, Cellpose Cyto, StarDist, and U-Net]. The Cellpose scale-factor range [0.15, 0.25, 0.4, 0.6] was explored. Many segmentation decisions in the 3D colon tissue dataset are hard to perform even for human experts. Within this limitation, [median projection & histogram equalization] preprocessing produced reasonable results without any further optimization in combination with either Cellpose nuclei 0.4 or the StarDist model (**Figures 5C,D**). Only a few cell clusters were not segmented (**Figures 5C,D** white arrow). Both models performed equally well on the entire data set.

We subsequently tried to segment a single layer of irregular-shaped epithelial cells, in which the nucleus and cell membranes had been stained with the same dye. In the first run, minor [median 3 × 3] or strong [mean 7 × 7] smoothing was applied. Next, the effect of additional histogram equalization, edge enhancement, and image inversion was tested. The resulting eight preprocessed images were segmented with all models [Cellpose nuclei, Cellpose Cyto, StarDist, and U-Net]. The Cellpose scale-factor range [0.6, 0.8, 1.0, 1.4, 1.8] was explored. The size of nuclei varied more than five-fold. We thus focused

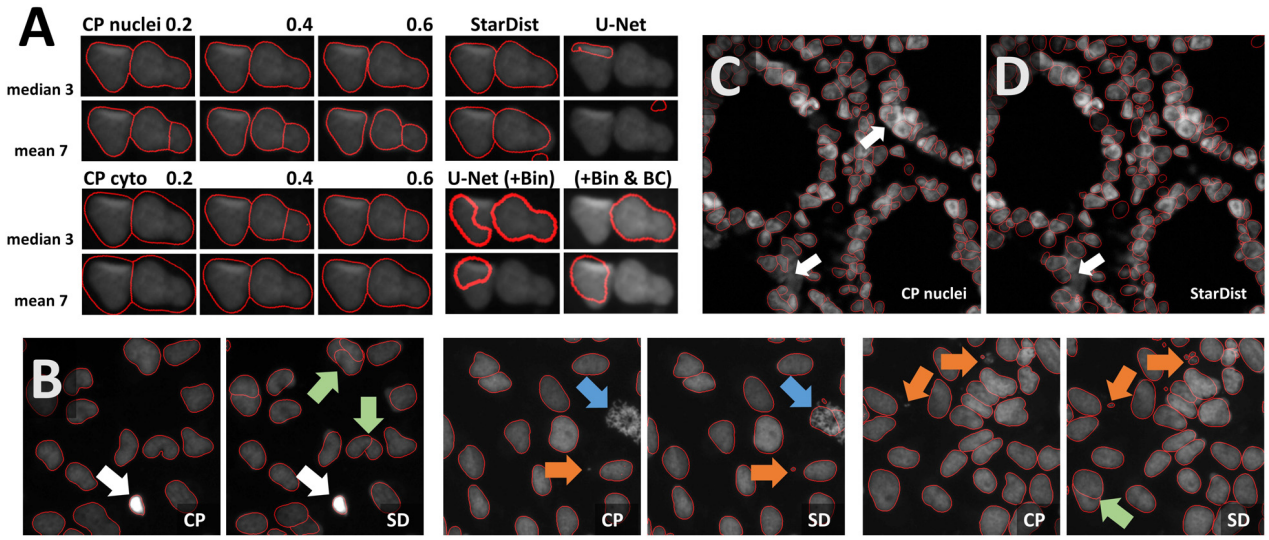


FIGURE 5 | Segmentation of dense fluorescent nuclei **(A)** Segmentation of the sparse nuclei data set. Example of two touching cells. The segmentation result (red line) superimposed onto the original image subjected to [median 3 × 3] preprocessing. Columns define the model used for segmentation, the rows the filter used for preprocessing. Cellpose nuclei model and the Cellpose Cyto model performed well across a large range of scale-factors. The U-Net failed initially. Improved results after binning are shown in the lower right panel. **(B)** Comparison of Cellpose nuclei (CP) and StarDist (SD) segmentation results (red line). StarDist is more prone to over-segmentation (green arrow), but detects smaller objects more reliably (orange arrow) and tends to include objects with strong texture (blue arrow). Strong variation in brightness was tolerated well by both models (white arrow). **(C,D)** 2D Segmentation of colon tissue from the Broad BioImage Benchmark Collection with Cellpose nuclei (CP) or StarDist (SD). Both models gave reasonable results. Only a few dense clusters could not be segmented (white arrow).

our analysis on a cluster of particularly large nuclei and a cluster of small nuclei. The Cellpose nuclei 1.4 and StarDist model detected both small and large nuclei similarly well (**Figure 6A**). StarDist segmentation results included many cell-shaped false positive detections. Given the model was trained on different data, retraining would be the best way to improve performance. Alternatively, false-positive detections, which were much larger than true nuclei, could be filtered out during postprocessing. While the U-Net did not perform well on the same input [median 3 × 3] (**Figure 6A**), it returned better results (**Figure 6A**) upon [mean 7 × 7 & histogram equalization] preprocessing. As weak smoothing was beneficial for the Cellpose and StarDist pipelines and stronger smoothing for the U-Net pipelines, we explored the effect of intermediate smoothing [median 5 × 5] for Cellpose and StarDist and even stronger smoothing [mean 9 × 9] for the U-Net pipelines. A slight improvement was observed. Thus, we used [median 5 × 5] preprocessing in combination with Cellpose nuclei 1.5 or StarDist model to process the entire dataset. Cellpose frequently failed to detect bright, round nuclei (**Figure 6B**, arrows) and StarDist (**Figure 6C**) had many false detections. Thus, re-training or postprocessing is required.

In the DSB, most algorithms performed better on images classified as small or large fluorescent, compared to images classified as "purple tissue" or "pink and purple" tissue. We used methylene blue-stained skeletal muscle sections as a sample dataset for tissue stained with a single dye and Hematoxylin and eosin (HE) stained kidney paraffin sections as an example for multi-dye stained tissue. Analysis of tissue sections might be compromised by heterogenous image quality cause e.g., by artifacts created at the junctions of tiles. To account for these artifacts all workflows used the fused image as input to the analysis pipeline.

While most nuclei in the skeletal muscle dataset are well separated, some form dense clusters, others are out of focus (**Figure 7A**). The size of nuclei varies ten-fold; their shape ranges from elongated to round. The same preprocessing and model as described for the epithelial cells dataset were used; the Cellpose scale-factor range [0.2, 0.4, 0.6] was explored. [Median 3 × 3 & invert image] preprocessing combined with the Cellpose nuclei 0.6 model produced satisfactory results without further optimization (**Figure 7B**). Outlines were well defined, some objects were missed, few over-segmented. Neither StarDist nor the U-Net performed similarly well. We could not overcome this limitation by adaptation of preprocessing or binning. The performance of other – most likely more appropriate – StarDist model (2D_versatile_fluo, 2D_versatile_he) was not tested. Processing of the entire dataset identified inadequate segmentation of dense clusters (**Figure 7C**, white arrow) and occasional over-segmentation of large, elongated nuclei (**Figure 7C**, orange arrow) as the main limitations. Nuclei that are out-of-focus were frequently missed (**Figure 7C**, blue arrow). Limiting the analysis to in-focus nuclei is feasible.

Cell density is very heterogeneous in the kidney dataset. The Eosin signal from a HE stained kidney paraffin section (**Figures 8A,B**) was obtained by color deconvolution. Nuclei are densely packed within glomeruli and rather sparse in the proximal and distal tubules. Two stitched images were split using OpSeF's "to tiles" function. Initial optimization was performed on a batch of 16 image tiles, the entire dataset contains 864 tiles.

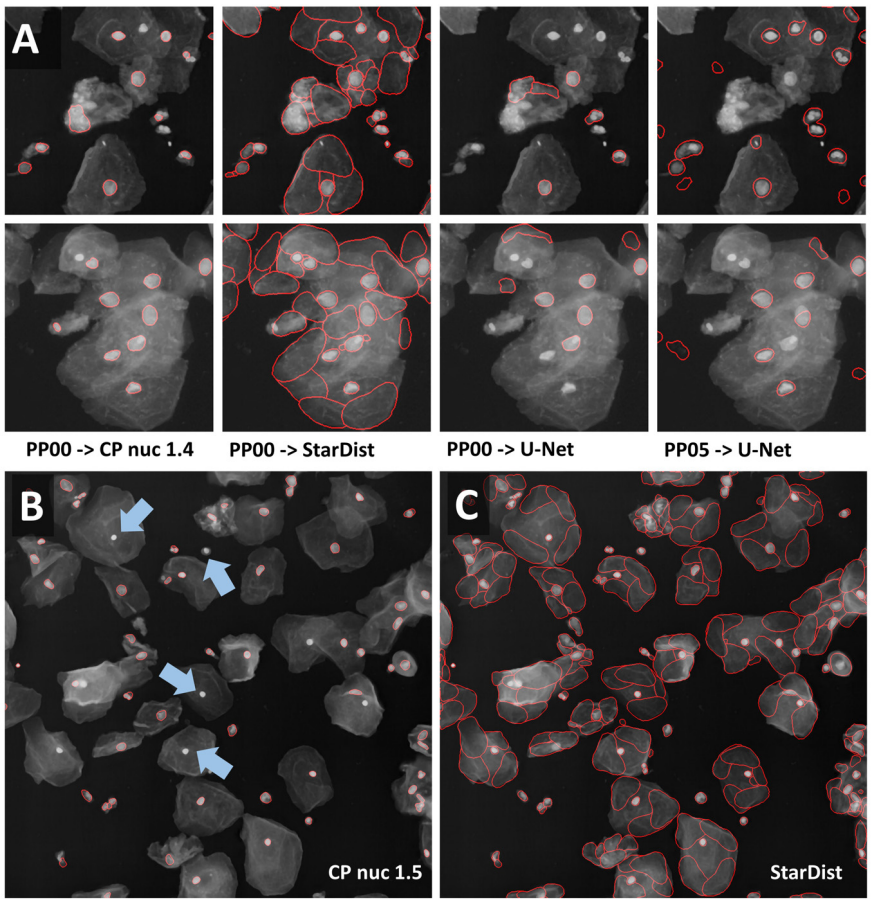


FIGURE 6 | Segmentation of multi-labeled cells **(A)** Images of large epithelial cells from the 2018 Data Science Bowl collection were used to test the segmentation of a single layer of cells. These cells vary in size. The Cellpose nuclei 1.4 model and [median 3 × 3] preprocessing gave a reasonable segmentation for large and small nuclei. StarDist segmentation based on the same input detected nuclei more reliably. However, many false-positive detections were present. Interestingly, the shape of false detections resembles the typical shape of cells well. The U-Net did not perform well with the same preprocessing, but with [mean 7 × 7 & histogram equalization] preprocessing. **(B,C)** [Median 5 × 5] preprocessing in combination with the Cellpose 1.5 nuclei or the StarDist model was applied to the entire data set. **(B)** The Cellpose model missed reproducibly round, very bright nuclei (blue arrow). **(C)** StarDist predicted many false-positive cells.

The same preprocessing and model were used as described for the skeletal muscle dataset, the Cellpose scale-factor range [0.6, 0.8, 1.0, 1.4, 1.8] was explored. [Median 3 × 3 & histogram equalization] preprocessing in combination with the Cellpose nuclei 0.6 model produced fine results (**Figure 8C**). [Mean 7 × 7 & histogram equalization] preprocessing in combination with StarDist performed similarly well (**Figure 8C**). The latter pipeline resulted in more false-positive detections (**Figure 8C**, purple arrows). The U-Net performed worse, and more nuclei were missed (**Figure 8C**, blue arrow). All models failed for dense cell clusters (**Figures 8C,D**, white arrow).

Next, we sought to expand the capability of OpSeF to volume segmentation. To this aim, we trained a StarDist 3D model using the annotation of *Arabidopsis thaliana* lateral root nuclei dataset provided by Wolny et al. (2020). Images were obtained on a Luxendo MuVi SPIM light-sheet microscope (Wolny et al., 2020). We first tested the model with a similar, publically available dataset. Valuchova et al. (2020) studied differentiation within *Arabidopsis* flowers. To this aim, the authors obtained eight views of H2B:mRuby2 labeled somatic nuclei on a Zeiss Z1 light-sheet microscope. We used a single view to mimic a challenging 3D segmentation (**Figures 9A–C**). Changes in image quality along the optical axis, in particular, deteriorating image quality deeper in the tissue (**Figure 9C**) are a major challenge for any segmentation algorithm. While the segmentation quality of the interactive H-Watershed ImageJ plugin (Vincent and Soille, 1991; Najman and Schmitt, 1996; Lotufo and Falcao, 2000; Schindelin et al., 2015), a state of the art traditional image processing method, is still acceptable in planes with good contrast (**Figure 9B**, xy Slice), results deeper in the tissue are inferior to CNN-based segmentation (data not shown). The H-Watershed plugin consequently fails to segment precisely in 3D (**Figure 9C**, zy-slices). The StarDist 3D model, which was trained on a similar dataset, performs slightly better than the Cellpose nuclei model. To evaluate the performance of these models further, we used the DISCEPTS dataset (Blin et al., 2019). DICEPTS stands for "DifferentiatingStemCells & Embryos are a PainToSegment" and contains various datasets of densely packed

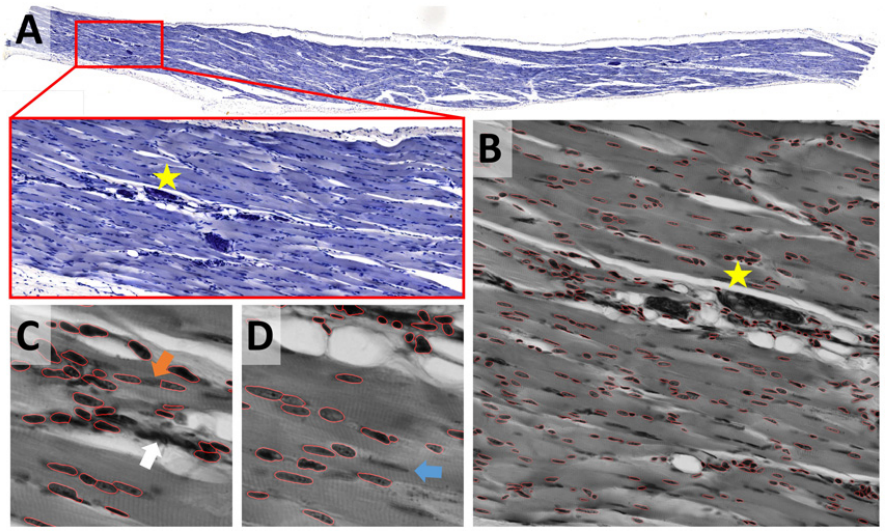


FIGURE 7 | Skeletal muscle notebook: Segmenting irregular nuclei in tissue **(A)** A methylene blue-stained skeletal muscle section was used to test the segmentation of tissue that has been stained with one dye. **(B)** Segmentation was tested on 2048 × 2048 pixels large image patches. The star shown in panels **(A,B)** is located at the same position within the image displayed at different zoom factors. The segmentation result (red line) of the Cellpose nuclei 0.6 model is super-imposed onto the original image subjected to [median 3 × 3] preprocessing. **(C,D)** Close-up on regions that were difficult to segment. Segmentation of dense clusters (white arrow) often failed, and occasional over-segmentation of large, elongated nuclei (orange arrow) was observed. Nuclei that are out-of-focus (blue arrow) were frequently missed (blue arrow).

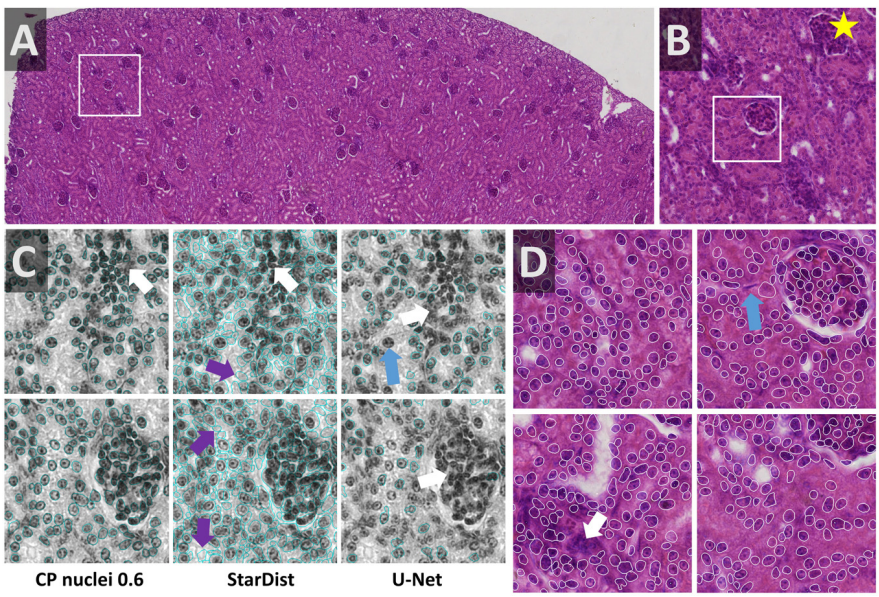


FIGURE 8 | Kidney notebook: Segmenting cell cluster in tissues **(A,B)** Part of a HE stained kidney paraffin section used to test segmentation of tissue stained with two dyes. The white box in panel **(A)** highlights the region shown enlarged in panel **(B)**. Star in panels **(A,B,D)** marks the same glomerulus. **(C)** Eosin signal was extracted by color deconvolution. The segmentation result (blue line) is superimposed onto the original image subjected to [median 3 × 3] preprocessing. Cellpose nuclei 0.6 model with scale-factor 0.6 in combination with [median 3 × 3 & histogram equalization] preprocessing and the StarDist model with [mean 7 × 7 & histogram equalization] performed similarly well **(C,D)**. StarDist resulted in more false-positive detections (purple arrows). The U-Net performed worse, more nuclei were missed (blue arrow). All models failed in very dense areas (white arrow).

nuclei that are heterogeneous in shape, size, or texture. Blin et al. elegantly solved the segmentation challenge by labeling the nuclear envelope. We thought to assess the performance of models contained in OpSeF on the more challenging DAPI signal. While the StarDist model trained on *Arabidopsis thaliana* lateral root nuclei shows satisfactory performance on E3.5 mouse

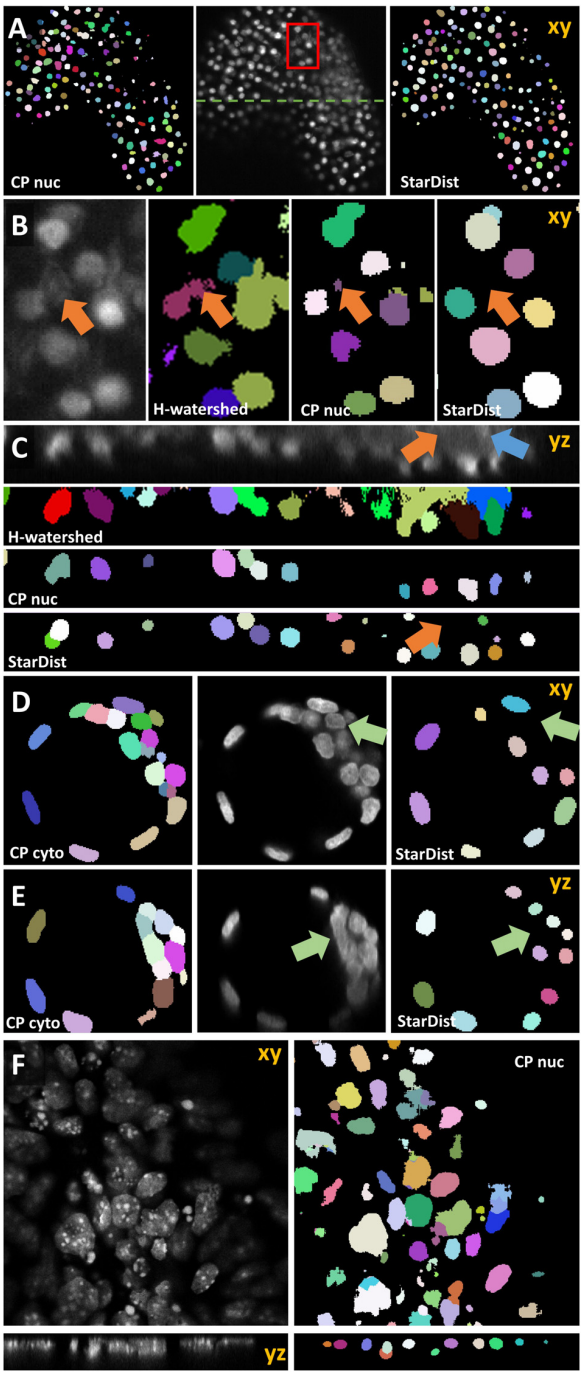


FIGURE 9 | Volume segmentation. In all images, segmented nuclei are assigned a unique random color. **(A)** Side-by-side comparison of segmented somatic nuclei of *Arabidopsis thaliana* flowers obtained with the Cellpose 3D nuclei model (left) and a StarDist 3D model that was trained on a similar dataset (right). The middle panel shows the central xy-plane of the image volume obtained by light-sheet microscopy. The red box marks the area that is shown enlarged in panel **(B)**. The green line indicates the location of the yz-plane shown in panel **(C)**. In panels **(B,C)** the results obtained manually using the interactive ImageJ H-watershed plugin are additionally shown. Orange arrows point at non-nuclear artifact signal that is ignored by StarDist 3D model trained on a dataset containing similar artifacts. Blue arrows point at *(Continued)*

FIGURE 9 | Continued
nuclear signal in a low-contrast area. **(D–F)** Evaluation of 3D model based on the DISCEPTS dataset. Central xy- or yz-planes of original images (gray) are shown side-by-side to the segmentation results of the model as indicated. All data were obtained on confocal microscopes. **(D,E)** Segmentation of E3.5 mouse blastocysts. The StarDist 3D model trained on well-spaced nuclei missed nuclei in dense-cluster (green arrow) that were reliably identified by the Cellpose 3D cyto model. **(F)** Segmentation results of Cellpose nuclei model (right panel) on the neural "monolayer" dataset (left). These cells contain – in contrast to *Arabidopsis thaliana* lateral root nuclei – strong texture in their nuclei, are densely spaced and vary in size.

blastocysts, notably, the size of nuclei is underestimated, and cells in dense clusters are sometimes missed. Fine-tuning of the non-maximum suppression and the detection threshold might suffice to obtain more precise segmentation results.

Interestingly, the more versatile Cellpose cyto, rather than the Cellpose nuclei model, is ideally suited for segmenting the E3.5 mouse blastocysts nuclei (**Figures 9D,E**). Next, we used the neural "monolayer" dataset (**Figure 9F**). In this dataset flat cells form tight 3D clusters. It proved to be challenging to segment (Blin et al., 2019). Our pre-trained StarDist model failed to give satisfactory segmentation results (data not shown). The presented cells contain – in contrast to *Arabidopsis thaliana* lateral root nuclei – strong texture in their nuclei. The more versatile Cellpose nuclei model displayed promising initial results with little fine-tuning (**Figure 9F**) that might be further improved by re-training the model on the appropriate ground truth.

DISCUSSION

Intended Use and Future Developments

Examining the relationship between biochemical changes and morphological alterations in diseased tissues is crucial to understand and treat complex diseases. Traditionally, microscopic images are inspected visually. This approach limits the possibilities for the characterization of phenotypes to more obvious changes that occur later in disease progression. The manual investigation of subtle alterations at the single-cell level, which often requires quantitative assays, is hampered by the data volume. A whole slide tissue image might contain over one million cells. Despite the improvement in machine learning technology, completely unsupervised analysis pipelines have not been widely accepted. Thus, one of the major challenges for the coming years will be the development of efficient strategies to keep the human-expert in the loop. Many biomedical users still perceive deep learning models as black boxes. The mathematical foundation of how CNNs make decisions is improving. OpSeF facilitates understanding the strength of pre-trained models and network architecture on the descriptive, operational level. Thereby, awareness of intrinsic limitations such as the inability of StarDist to segment non-star-convex shapes well, or issues relating to the limited field-of-view of neural networks can be reached. It further allows us to quickly assess how robust models are against artifacts such as shadows present in light-sheet microscopy or how well they are in

predicting cell shapes accurately that are neither round nor ellipsoid shaped (e.g., neurons, amoebas). Collectively, increased awareness of limitations and better interpretability of results will be pivotal to increase the acceptance of machine learning methods. It will improve the quality control of results and allow efficient integration of expert knowledge in analysis pipelines (Holzinger et al., 2019a,b).

As illustrated in the provided Jupyter notebooks, the U-Net often performed worst. Why is that the case? As previously reported the learning capacity of a single, regular U-Net is limited (Caicedo et al., 2019). Alternatively, the similarity of train and test images might be insufficient. Either way, the provision of a set of U-Nets trained on diverse data, might be a promising approach to address this limitation. [ods.ai] topcoders, the winning team of the 2018 Data Science Bowl (Caicedo et al., 2019), combined simple U-Nets with dedicated pre- and postprocessing pipelines. In doing so, they outperformed teams using more complex models like Mask R-CNN (Caicedo et al., 2019). OpSeF allows for the straightforward integration of a large number of pre-trained CNNs. We plan to include the possibility of saving pixel probabilities in future releases of OpSeF. This option will grant users more flexibility in designing custom postprocessing pipelines that ensemble results from a set of useful intermediate predictions.

OpSeF allows semi-automated exploration of a large number of possible combinations of preprocessing pipelines and segmentation models. Even if satisfactory results are not achievable with pre-trained models, OpSeF results may be used as a guide for which CNN architecture, re-training on manually created labels might be promising. The generation of training data is greatly facilitated by a seamless integration in ImageJ using the AnnotatorJ plugin. We hope that many OpSeF users will contribute their training data to open repositories and will make new models available for integration in OpSeF. Thus, OpSeF might soon become, an interactive model repository, in which an appropriate model might be identified with reasonable effort. Community provided Jupyter notebooks might be used to teach students in courses how to optimize CNN based analysis pipelines. This could educate them and make them less dependent on turn-key solutions that often trade performance for simplicity and offer little insight into the reasons why the CNN-based segmentation works or fails. The better users understand the model they use, the more they will trust them and, the better they will be able to quality control them. We hope that OpSeF will be widely accepted as a framework through which novel models might be made available to other image analysts in an efficient way.

Integrating Various Segmentation Strategies and Quality Control of Results

Multiple strategies for instance segmentation have been pursued. The U-Net belongs to the "pixel to object" class of methods: each pixel is first assigned to a semantic class (e.g., cell or background), then pixels are grouped into objects (Ronneberger et al., 2015). Mask R-CNNs belong to the "object to pixel" class of methods (He et al., 2017): the initial prediction of bounding boxes for each object is followed by a semantic segmentation. Following an intermediate approach, Schmidt et al. first predict star-convex polygons that approximate the shape of cells and use non-maximum suppression to prune redundant predictions (Schmidt et al., 2018; Weigert et al., 2019, 2020). Stringer et al. use stimulated diffusion originating from the center of a cell to convert segmentation masks into flow fields. The neural network is then trained to predict flow fields, which can be converted back into segmentation masks (Stringer et al., 2020). Each of these methods has specific strengths and weaknesses. The use of flow fields as auxiliary representation proved to be a great advantage for predicting cell shapes that are not roundish. At the same time, Cellpose is the most computationally demanding model used. In our hands, Cellpose tended to result in more obviously erroneously missed objects, in particular, if objects displayed a distinct appearance compared to their neighbors (blue arrows in **Figures 5B**, **6B**, **7D**). StarDist is much less computationally demanding, and star-convex polygons are well suited to approximate elliptical cell shapes. The pre-trained StarDist model implemented in OpSeF might be less precise in predicting novel shapes it has not been trained on, e.g., maple leaves (**Figure 4C**). This limitation can be overcome by retraining, given the object is star-convex, which includes certain concave shapes such as maple leaves. However, some cell-types (e.g., neurons, amoeba) are typically non star-convex, and StarDist – due to "limitation by design" – cannot be expected to segment these objects precisely. Segmentation errors by the StarDist model were generally plausible. It tended to predict cell-like shapes, even if they are not present (**Figure 6B**). Although the tendency of StarDist to fail gracefully might be advantageous in most instances, this feature requires particularly careful quality control to detect and fix errors. The "pixel-to-object" class of methods is less suited for segmentation of dense cell clusters. The misclassification of just a few pixels might lead to the fusion of neighboring cells.

OpSeF integrates three mechanistically distinct methods for CNN-based segmentation in a single framework. This allows comparing these methods easily. Nonetheless, we decided against integrating an automated evaluation, e.g., by determining F1 score, false positive and false negative rates, and accuracy. Firstly, for most projects no ground-truth is available. Secondly, we want to encourage the user to visually inspect segmentation results. Reviewing 100 different segmentation results opened in ImageJ as stack takes only a few minutes and gives valuable insight into when and how segmentations fail. This knowledge is easily missed when just looking at the output scores of commonly used metrics but might have a significant impact on the biological conclusion. Even segmentation results from a model with 95% precision and 95% recall for the overall cell population might be not suited to determine the abundance of a rare cell type if these cells are systematically missed, detected less accurately in the mutant situation, or preferentially localized to areas in the tissue that are not segmented well. Although it is difficult to capture such issues with standard metrics, they are readily observed by a human expert. Learning more about the circumstances in which

certain types of CNN-based segmentation fail helps to decide when human experts are essential for quality control of results. Moreover, it is pivotal for the design of postprocessing pipelines. These might select among multiple segmentation hypotheses – on an object by object basis – the one which gives the most consistent results for reconstructing complex cells-shapes in large 3D volumes or for cell-tracking.

Optimizing Results and Computational Cost

Image analysis pipelines are generally a compromise between ease-of-use and performance as well as between computational cost and accuracy. Until now, rather simple, standard U-Nets are most frequently used models in the major image analysis tools. In contrast, the winning model of the 2018 Data Science Bowl by the [ods.ai] topcoders team used sophisticated data postprocessing to combine the output of 32 different neural networks (Caicedo et al., 2019). The high computational cost currently limits the widespread use of this or similar approaches. OpSeF is an ideal platform to find the computationally most efficient solution to a segmentation task. The [ods.ai] topcoders algorithm was designed to segment five different classes of nuclei: "small" and "large fluorescent," "grayscale," "purple tissue" and "pink and purple tissue" (Caicedo et al., 2019). Stringer et al. used an even broader collection of images that included cells of unusual appearance and natural images of regular cell-like shapes such as shells, garlic, pearls, and stones (Stringer et al., 2020).

The availability of such versatile models is precious, in particular, for users, who are unable to train custom models or lack resources to search for the most efficient pre-trained model. For most biological applications, however, no one-fits-all solution is required. Instead, potentially appropriate models might be pre-selected, optimized, and tested using OpSeF. Ideally, an image analyst and a biomedical researcher will jointly fine-tune the analysis pipeline and quality control results. This way, resulting analysis workflows will have the best chances of being both robust and accurate, and an ideal balance between manual effort, computational cost, and accuracy might be reached.

Comparison of the models available within OpSeF revealed that the same task of segmenting 100 images using StarDist took 1.5-fold, Cellpose with fixed scale-factor 3.5-fold, and Cellpose with flexible scale-factor 5-fold longer compared to segmentation with the U-Net.

The systematic search of the optimal parameter and ideal performance might be dispensable if only a few images are to be processed that can be easily manually curated, but highly valuable if massive datasets produced by slide-scanner, light-sheet microscopes or volume EM techniques are to be processed.

Deployment Strategies

We decided against providing OpSeF as an interactive cloud solution. A local solution uses existing resources best, avoids limitations generated by the down- and upload of large datasets, and addresses concerns regarding the security of clinical datasets. Although the provision of plugins is perceived as crucial to speed up the adoption of new methods, image analysts increasingly use the Jupyter notebooks that allow them to document workflows step-by-step. This is a significant advantage compared to interactive solutions, in which parameters used for analysis are not automatically logged. Biologists might hesitate to use Jupyter notebooks for analysis due to an initial steep learning curve. Once technical challenges such as the establishment of the conda environment are overcome, notebooks allow them to integrate data analysis and documentation with ease. Notebooks might be deposited in repositories along with the raw data. This builds more trust in published results by improving transparency and reproducibility.

DATA AVAILABILITY STATEMENT

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found below:
Test data for the Jupyter notebooks is available at:
https://owncloud.gwdg.de/index.php/s/nSUqVXkkfUDPG5b.
Test data for AnnotatorJ is available at:
https://owncloud.gwdg.de/index.php/s/dUMM6JRXsuhTncS.
The source code is available at:
https://github.com/trasse/OpSeF-IV.

AUTHOR CONTRIBUTIONS

TR designed and developed the OpSeF, and analyzed the data. RH and PH designed and developed the AnnotatorJ integration. TR wrote the manuscript with contributions from all authors.

FUNDING

Research in the Scientific Service Group Microscopy was funded by the Max Planck Society. PH and RH acknowledge the LENDULET-BIOMAG Grant (2018-342) and from the European Regional Development Funds (GINOP-2.3.2-15-2016-00006, GINOP-2.3.2-15-2016-00026, and GINOP-2.3.2-15-2016-00037).

ACKNOWLEDGMENTS

We would like to thank Thorsten Falk and Volker Hilsenstein (Monash Micro Imaging) for testing of the software and critical comments on the manuscript and the IT service group of the MPI Heart and Lung Research for technical support. We gratefully acknowledge Varun Kapoor (Institut Curie), Constantin Pape (EMBL), Uwe Schmidt (MPI CBG), Carsen Stringer (Janelia Research Campus), and Adrian Wolny (EMBL) for advice. We thank anyone who made their data available, in particular: the Broad Bioimage Benchmark Collection for image set BBBC027 (Svoboda et al., 2011) and BBBC038v1 (Caicedo et al., 2019) and Christoph Möhl (DZNE). We appreciate the referee's valuable comments. This manuscript has been released as a pre-print at bioRxiv (Rasse et al., 2020).

REFERENCES

Akram, S. U., Kannala, J., Eklund, L., and Heikkilä, J. (2016). "Cell segmentation proposal network for microscopy image analysis," in *Deep Learning and Data Labeling for Medical Applications*, eds G. Carneiro, D. Mateus, L. Peter, A. Bradley, J. M. R. S. Tavares, V. Belagiannis, et al. (New York, NY: Springer International Publishing), 21–29. doi: 10.1007/978-3-319-46976-8_3

Albarqouni, S., Baur, C., Achilles, F., Belagiannis, V., Demirci, S., and Navab, N. (2016). AggNet: deep learning from crowds for mitosis detection in breast cancer histology images. *IEEE Trans. Med. Imaging* 35, 1313–1321. doi: 10.1109/TMI.2016.2528120

Arganda-Carreras, I., Kaynig, V., Rueden, C., Eliceiri, K. W., Schindelin, J., Cardona, A., et al. (2017). Trainable Weka Segmentation: a machine learning tool for microscopy pixel classification. *Bioinformatics* 33, 2424–2426. doi: 10.1093/bioinformatics/btx180

Bankhead, P., Loughrey, M. B., Fernandez, J. A., Dombrowski, Y., McArt, D. G., Dunne, P. D., et al. (2017). QuPath: open source software for digital pathology image analysis. *Sci. Rep.* 7:16878. doi: 10.1038/s41598-017-17204-5

Berg, S., Kutra, D., Kroeger, T., Straehle, C. N., Kausler, B. X., Haubold, C., et al. (2019). ilastik: interactive machine learning for (bio)image analysis. *Nat. Methods* 16, 1226–1232. doi: 10.1038/s41592-019-0582-9

Blin, G., Sadurska, D., Portero Migueles, R., Chen, N., Watson, J. A., and Lowell, S. (2019). Nessys: a new set of tools for the automated detection of nuclei within intact tissues and dense 3D cultures. *PLoS Biol.* 17:e3000388. doi: 10.1371/journal.pbio.3000388

Bykov, Y. S., Cohen, N., Gabrielli, N., Manenschijn, H., Welsch, S., Chlanda, P., et al. (2019). High-throughput ultrastructure screening using electron microscopy and fluorescent barcoding. *J. Cell Biol.* 218, 2797–2811. doi: 10.1083/jcb.201812081

Caicedo, J. C., Goodman, A., Karhohs, K. W., Cimini, B. A., Ackerman, J., Haghighi, M., et al. (2019). Nucleus segmentation across imaging experiments: the 2018 data science bowl. *Nat. Methods* 16, 1247–1253. doi: 10.1038/s41592-019-0612-7

Chessel, A., and Carazo Salas, R. E. (2019). From observing to predicting single-cell structure and function with high-throughput/high-content microscopy. *Essays Biochem.* 63, 197–208. doi: 10.1042/EBC20180044

Çiçek, Ö, Abdulkadir, A., Lienkamp, S. S., Brox, T., and Ronneberger, O. (2016). "3D U-net: learning dense volumetric segmentation from sparse annotation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, eds S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, and W. Wells (New York, NY: Springer International Publishing), 424–432. doi: 10.1007/978-3-319-46723-8_49

Cireşan, D. C., Giusti, A., Gambardella, L. M., and Schmidhuber, J. (2012). "Deep neural networks segment neuronal membranes in electron microscopy images," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, (Lake Tahoe, Nev: Curran Associates Inc.).

Cireşan, D. C., Giusti, A., Gambardella, L. M., and Schmidhuber, J. (2013). "Mitosis detection in breast cancer histology images with deep neural networks," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013*, eds K. Mori, I. Sakuma, Y. Sato, C. Barillot, and N. Navab (Berlin: Springer), 411–418. doi: 10.1007/978-3-642-40763-5_51

de Chaumont, F., Dallongeville, S., Chenouard, N., Herve, N., Pop, S., Provoost, T., et al. (2012). Icy: an open bioimage informatics platform for extended reproducible research. *Nat. Methods* 9, 690–696. doi: 10.1038/nmeth.2075

Haubold, C., Schiegg, M., Kreshuk, A., Berg, S., Koethe, U., and Hamprecht, F. A. (2016). Segmenting and tracking multiple dividing targets using ilastik. *Adv. Anat. Embryol. Cell Biol.* 219, 199–229. doi: 10.1007/978-3-319-28549-8_8

He, K., Gkioxari, G., Dollar, P., and Girshick, R. (2017). "Mask R-CNN," in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, (Venice: IEEE).

Hoffman, D. P., Shtengel, G., Xu, C. S., Campbell, K. R., Freeman, M., Wang, L., et al. (2020). Correlative three-dimensional super-resolution and block-face electron microscopy of whole vitreously frozen cells. *Science* 367:eaaz5357. doi: 10.1126/science.aaz5357

Hollandi, R., Diósdi, Á, Hollandi, G., Moshkov, N., and Horváth, P. (2020a). AnnotatorJ: an ImageJ plugin to ease hand-annotation of cellular compartments. *Mol. Biol. Cell* 31, 2179–2186. doi: 10.1091/mbc.E20-02-0156

Hollandi, R., Szkalisity, A., Toth, T., Tasnadi, E., Molnar, C., Mathe, B., et al. (2020b). nucleAIzer: a parameter-free deep learning framework for nucleus segmentation using image style transfer. *Cell Syst.* 10, 453–458.e6. doi: 10.1016/j.cels.2020.04.003

Holzinger, A., Haibe-Kains, B., and Jurisica, I. (2019a). Why imaging data alone is not enough: AI-based integration of imaging, omics, and clinical data. *Eur. J. Nucl. Med. Mol. Imaging* 46, 2722–2730. doi: 10.1007/s00259-019-04382-9

Holzinger, A., Langs, G., Denk, H., Zatloukal, K., and Muller, H. (2019b). Causability and explainability of artificial intelligence in medicine. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 9:e1312. doi: 10.1002/widm.1312

Houle, D., Govindaraju, D. R., and Omholt, S. (2010). Phenomics: the next challenge. *Nat. Rev. Genet.* 11, 855–866. doi: 10.1038/nrg2897

Jones, T. R., Carpenter, A. E., Lamprecht, M. R., Moffat, J., Silver, S. J., Grenier, J. K., et al. (2009). Scoring diverse cellular morphologies in image-based screens with iterative feedback and machine learning. *Proc. Natl. Acad. Sci. U.S.A.* 106, 1826–1831. doi: 10.1073/pnas.0808843106

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., et al. (2016). "Jupyter Notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, eds F. Loizides and B. Schmidt (Amsterdam: IOS Press).

Kreshuk, A., and Zhang, C. (2019). Machine learning: advanced image segmentation using ilastik. *Methods Mol. Biol.* 2040, 449–463. doi: 10.1007/978-1-4939-9686-5_21

Lamprecht, M. R., Sabatini, D. M., and Carpenter, A. E. (2007). CellProfiler: free, versatile software for automated biological image analysis. *Biotechniques* 42, 71–75. doi: 10.2144/000112257

Lang, P., Yeow, K., Nichols, A., and Scheer, A. (2006). Cellular imaging in drug discovery. *Nat. Rev. Drug Discov.* 5, 343–356. doi: 10.1038/nrd2008

Lotufo, R., and Falcao, A. (2000). "The ordered queue and the optimality of the watershed approaches," in *Mathematical Morphology and its Applications to Image and Signal Processing*, eds J. Goutsias, L. Vincent, and D. S. Bloomberg (Boston, MA: Springer), 341–350. doi: 10.1007/0-306-47025-x_37

McQuin, C., Goodman, A., Chernyshev, V., Kamentsky, L., Cimini, B. A., Karhohs, K. W., et al. (2018). CellProfiler 3.0: next-generation image processing for biology. *PLoS Biol.* 16:e2005970. doi: 10.1371/journal.pbio.2005970

Milletari, F., Navab, N., and Ahmadi, S. (2016). "V-Net: fully convolutional neural networks for volumetric medical image segmentation," in *Proceedings of the 2016 Fourth International Conference on 3D Vision (3DV)*, (Stanford, CA: IEEE), 565–571.

Moeskops, P., Viergever, M. A., Mendrik, A. M., Vries, L. S. D., Benders, M. J. N. L., and Išgum, I. (2016). Automatic segmentation of MR brain images with a convolutional neural network. *IEEE Trans. Med. Imaging* 35, 1252–1261. doi: 10.1109/TMI.2016.2548501

Najman, L., and Schmitt, M. (1996). Geodesic saliency of watershed contours and hierarchical segmentation. *IEEE Trans. Pattern Anal. Machine Intell.* 18, 1163–1173. doi: 10.1109/34.546254

Neumann, B., Walter, T., Heriche, J. K., Bulkescher, J., Erfle, H., Conrad, C., et al. (2010). Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes. *Nature* 464, 721–727. doi: 10.1038/nature08869

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: machine learning in python. *J. Machine Learn. Res.* 12, 2825–2830.

Rajchl, M., Lee, M. C. H., Oktay, O., Kamnitsas, K., Passerat-Palmbach, J., Bai, W., et al. (2017). DeepCut: object segmentation from bounding box annotations using convolutional neural networks. *IEEE Trans. Med. Imaging* 36, 674–683. doi: 10.1109/TMI.2016.2621185

Rasse, T. M., Hollandi, R., and Horváth, P. (2020). OpSeF IV: open source python framework for segmentation of biomedical images. *bioRxiv*[Preprint] doi: 10.1101/2020.04.29.068023

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: convolutional networks for biomedical image segmentation. *arXiv*[Preprint] Available online at: https://arxiv.org/abs/1505.04597 (Accessed March 15, 2020),

Ruifrok, A. C., and Johnston, D. A. (2001). Quantification of histochemical staining by color deconvolution. *Anal. Quant Cytol. Histol.* 23, 291–299.

Schindelin, J., Rueden, C. T., Hiner, M. C., and Eliceiri, K. W. (2015). The imageJ ecosystem: an open platform for biomedical image analysis. *Mol. Reprod. Dev.* 82, 518–529. doi: 10.1002/mrd.22489

Schmidt, U., Weigert, M., Broaddus, C., and Myers, G. (2018). "Cell detection with star-convex polygons," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, eds A. F. Frangi, J. A. Schnabel, C. Davatzikos, C.

Alberola-López, and G. Fichtinger (Cham: Springer International Publishing), 265–273. doi: 10.1007/978-3-030-00934-2_30

Schneider, C. A., Rasband, W. S., and Eliceiri, K. W. (2012). NIH Image to ImageJ: 25 years of image analysis. *Nat. Methods* 9, 671–675. doi: 10.1038/nmeth.2089

Schorb, M., Haberbosch, I., Hagen, W. J. H., Schwab, Y., and Mastronarde, D. N. (2019). Software tools for automated transmission electron microscopy. *Nat. Methods* 16, 471–477. doi: 10.1038/s41592-019-0396-9

Sieb, C., Meinl, T., and Berthold, M. R. (2007). Parallel and distributed data pipelining with KNIME. *Mediterr. J. Comput. Netw.* 3, 43–51.

Stringer, C., Michaelos, M., and Pachitariu, M. (2020). Cellpose: a generalist algorithm for cellular segmentation. *bioRxiv*[Preprint] doi: 10.1101/2020.02.02.931238

Svoboda, D., Homola, O., and Stejskal, S. (2011). *Generation of 3D Digital Phantoms of Colon Tissue*. Berlin: Springer, 31–39.

Swoger, J., Pampaloni, F., and Stelzer, E. H. (2014). Light-sheet-based fluorescence microscopy for three-dimensional imaging of biological samples. *Cold Spring Harb. Protoc.* 2014, 1–8. doi: 10.1101/pdb.top080168

Titze, B., and Genoud, C. (2016). Volume scanning electron microscopy for imaging biological ultrastructure. *Biol. Cell* 108, 307–323. doi: 10.1111/boc.201600024

Ueda, H. R., Erturk, A., Chung, K., Gradinaru, V., Chedotal, A., Tomancak, P., et al. (2020). Tissue clearing and its applications in neuroscience. *Nat. Rev. Neurosci.* 21, 61–79. doi: 10.1038/s41583-019-0250-1

Valuchova, S., Mikulkova, P., Pecinkova, J., Klimova, J., Krumnikl, M., Bainar, P., et al. (2020). Imaging plant germline differentiation within Arabidopsis flowers by light sheet microscopy. *eLife* 9:e52546. doi: 10.7554/eLife.52546

van der Walt, S., Schonberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., et al. (2014). scikit-image: image processing in Python. *PeerJ* 2:e453. doi: 10.7717/peerj.453

Van Valen, D. A., Kudo, T., Lane, K. M., Macklin, D. N., Quach, N. T., DeFelice, M. M., et al. (2016). Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments. *PLoS Comput. Biol.* 12:e1005177. doi: 10.1371/journal.pcbi.1005177

Vidavsky, N., Akiva, A., Kaplan-Ashiri, I., Rechav, K., Addadi, L., Weiner, S., et al. (2016). Cryo-FIB-SEM serial milling and block face imaging: large volume structural analysis of biological tissues preserved close to their native state. *J. Struct. Biol.* 196, 487–495. doi: 10.1016/j.jsb.2016.09.016

Vincent, L., and Soille, P. (1991). Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Machine Intell.* 13, 583–598. doi: 10.1109/34.87344

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods* 17, 261–272. doi: 10.1038/s41592-019-0686-2

Wang, S., Yang, D. M., Rong, R., Zhan, X., and Xiao, G. (2019). Pathology image analysis using segmentation deep learning algorithms. *Am. J. Pathol.* 189, 1686–1698. doi: 10.1016/j.ajpath.2019.05.007

Webster, J. D., and Dunstan, R. W. (2014). Whole-slide imaging and automated image analysis: considerations and opportunities in the practice of pathology. *Vet. Pathol.* 51, 211–223. doi: 10.1177/0300985813503570

Weigert, M., Schmidt, U., Haase, R., Sugawara, K., and Myers, G. (2019). Star-convex polyhedra for 3D object detection and segmentation in microscopy. *arXiv*[Preprint] doi: 10.1109/WACV45572.2020.9093435

Weigert, M., Schmidt, U., Haase, R., Sugawara, K., and Myers, G. (2020). "Star-convex Polyhedra for 3D object detection and segmentation in microscopy," in *Proceedings of The IEEE Winter Conference on Applications of Computer Vision (WACV)*, (Piscataway, NJ: IEEE), 3655–3662.

Whitehead, L. (2020). *Tweet: Correct. It's the Pre-Trained "Versatile" Model. (on use of StarDist)*. Available online at: https://twitter.com/DrLachie/status/1224207976350633985 (accessed February 5, 2020).

Williams, E., Moore, J., Li, S. W., Rustici, G., Tarkowska, A., Chessel, A., et al. (2017). The image data resource: a bioimage data integration and publication platform. *Nat. Methods* 14, 775–781. doi: 10.1038/nmeth.4326

Wolny, A., Cerrone, L., Vijayan, A., Tofanelli, R., Barro, A. V., Louveaux, M., et al. (2020). Accurate and versatile 3D segmentation of plant tissues at cellular resolution. *eLife* 9:e57613. doi: 10.7554/eLife.57613

Zhang, W., Li, R., Deng, H., Wang, L., Lin, W., Ji, S., et al. (2015). Deep convolutional neural networks for multi-modality isointense infant brain image segmentation. *Neuroimage* 108, 214–224. doi: 10.1016/j.neuroimage.2014.12.061

Zuiderveld, K. (1994). "Contrast limited adaptive histogram equalization," in *Graphics gems IV*, ed. P. S. Heckbert (Cambridge, MA: Academic Press Professional, Inc), 474–485. doi: 10.1016/b978-0-12-336156-1.50061-6

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

IV.

SCIENTIFIC REPORTS
nature research

There are amendments to this paper

OPEN

Test-time augmentation for deep learning-based cell segmentation on microscopy images

Nikita Moshkov[1,2,3], Botond Mathe[1], Attila Kertesz-Farkas[3], Reka Hollandi[1] & Peter Horvath[1,4*]

Recent advancements in deep learning have revolutionized the way microscopy images of cells are processed. Deep learning network architectures have a large number of parameters, thus, in order to reach high accuracy, they require a massive amount of annotated data. A common way of improving accuracy builds on the artificial increase of the training set by using different augmentation techniques. A less common way relies on test-time augmentation (TTA) which yields transformed versions of the image for prediction and the results are merged. In this paper we describe how we have incorporated the test-time argumentation prediction method into two major segmentation approaches utilized in the single-cell analysis of microscopy images. These approaches are semantic segmentation based on the U-Net, and instance segmentation based on the Mask R-CNN models. Our findings show that even if only simple test-time augmentations (such as rotation or flipping and proper merging methods) are applied, TTA can significantly improve prediction accuracy. We have utilized images of tissue and cell cultures from the Data Science Bowl (DSB) 2018 nuclei segmentation competition and other sources. Additionally, boosting the highest-scoring method of the DSB with TTA, we could further improve prediction accuracy, and our method has reached an ever-best score at the DSB.

Identifying objects at the single-cell level is the starting point of most microscopy-based quantitative cellular image analysis tasks. Precise segmentation of the cell's nucleus is a major challenge here. Numerous approaches have been developed, including methods based on mathematical morphology[1] or differential geometry[2,3]. More recently, deep learning has yielded a never-seen improvement of accuracy and robustness[4–6]. Remarkably, Kaggle's Data Science Bowl 2018 (DSB)[7] was dedicated to nuclei segmentation, and gave a great momentum to this field. Deep learning-based approaches have proved their effectiveness: practically all the teams used some type of a deep architecture in the first few hundred leaderboard positions. The most popular architectures included U-Net[4], originally designed for medical image segmentation, and Mask R-CNN[8], used for instance segmentation of natural objects.

Deep learning approaches for object segmentation require a large, and often pixel-wise annotated dataset for training. This task relies on high-quality samples and domain experts to accurately annotate images. Besides, analysing biological images is challenging because of their heterogeneity and, sometimes, poorer quality compared to natural images. In addition, ground truth masks might be imperfect due to the annotator-related bias, which introduces further uncertainty. Consequently, a plethora of annotated samples is required, making object segmentation a laborious process. One of the techniques utilized to improve the model is data augmentation[9] of the training set. Conventionally, a transformation (i.e. rotation, flipping, noise addition, etc.) or a series of transformations are applied on the original images. Data augmentation has become the *de facto* technique in deep learning, especially in the case of heterogeneous or small datasets, to improve the accuracy of cell-based analysis.

Another option of improving performance relies on augmenting both the training and the test datasets, then performing the prediction both on the original and on the augmented versions of the image, followed by merging the predictions. This approach is called ***test-time augmentation*** (Fig. 1). This technique was successfully used in image classification tasks[10], for aleatoric uncertainty estimation[11], as well as for the segmentation of MRI slices/MRI volumes[12]. A theoretical formulation[12] of test-time augmentation has recently been described by Wang *et al.* Their experiments show that TTA helps to eliminate overconfident incorrect predictions. Additionally, a

[1]Biological Research Centre, Szeged, Hungary. [2]University of Szeged, Szeged, Hungary. [3]National Research University, Higher School of Economics, Moscow, Russia. [4]Institute for Molecular Medicine Finland, University of Helsinki, Helsinki, Finland. *email: horvath.peter@brc.hu

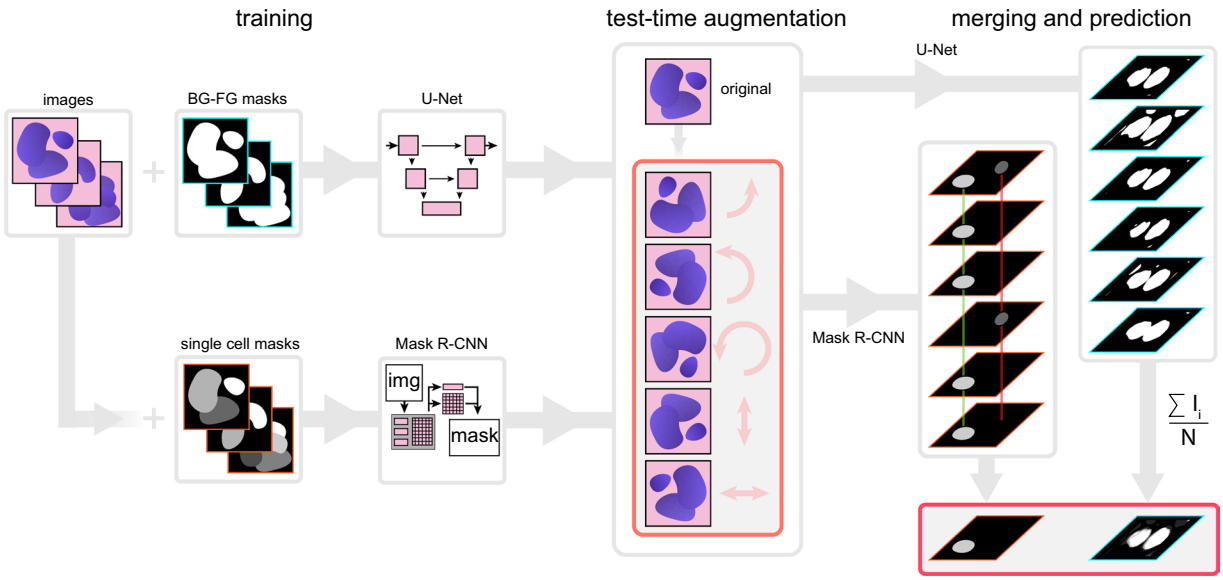


Figure 1. Principle of the proposed test-time augmentation techniques. Several augmented instances of the same test images are predicted, and the results are transformed back and merged. In the case of U-Net, pixel-wise majority voting was applied, while for Mask R-CNN a combination of object matching and majority voting was applied.

framework[13] has also been proposed for quantifying the uncertainty of the deep neural network (DNN) model for diagnosing diabetic retinopathy based on test-time data augmentation. Its disadvantage is increased prediction time, as it is run not only on the original image, but on all of its augmentations as well.

In the current paper we assess the impact and describe cases of utilizing test-time augmentation for deep-learning models trained on microscopy datasets. We have trained deep learning models for semantic segmentation (when the network only distinguishes the foreground from the background, using the U-Net architecture) and instance segmentation (when the network assigns labels to separate objects, using the Mask R-CNN architecture) (Fig. 1). Test-time augmentation has outperformed single instance predictions at each test case, and could further improve the best result of the DSB, as demonstrated by the improvement of the score, changing from 0.633 to 0.644.

Methods

Dataset acquisition and description. We have used two datasets: fluorescent microscopy images (further referred to as 'fluorescent' dataset) and histopathology images (further referred to as 'tissue' dataset). Most of the images have come from the stage 1 train/test data of Data Science Bowl 2018. We also used additional sources[14–20] and other data published in the discussion thread 'Official External Data Thread' (https://www.kaggle.com/c/data-science-bowl-2018/discussion/47572) related to DSB 2018. The images were labelled by experts using the annotation plugins of ImageJ/Fiji and Gimp. Both datasets were divided into three holdout train/test sets: approximately 5%, 15% (6 splits for each, cross-validation), and 30% (further referred to as '5', '15' and '30' in the dataset name, respectively) of uncropped images were held out as the test set. The test sets ('5', first cross-validation split of '15' and '30') did not intersect.

We used the same augmentations (horizontal and vertical flip, 90°, 180° and 270° rotations) for training both architectures. The images were cropped to the size of 512 × 512 pixels. Crops from the same image were used only in either the train or test set. Images with a resolution of less than 512 × 512 were resized to that particular size. Sample images are shown in Fig. 2.

Deep learning models and training. These augmented and cropped training data were used to train the models. For each dataset (5, 15 (6-fold cross validation) and 30 holdouts for both fluorescent and tissue images) separate models were trained. Additionally, we also trained U-Net without augmented data to analyse TTA performance on such a network as well (just 1 holdout 15 test set in that case).

Mask R-CNN (implementation[21]) is an extension of Faster R-CNN, the architecture for object detection. Solutions based on Mask R-CNN outperform the COCO 2016 challenge winners, and finished at the third place in Kaggle Data Science Bowl 2018[7]. The architecture of Mask R-CNN incorporates the following main stages: (1) Region proposal network (RPN) to propose candidate bounding boxes. It uses a backbone: a convolutional neural network which serves as a feature extractor. In this implementation it is possible to use *resnet50* or *resnet101* as a backbone, and we used *resnet101*. (2) Network head layers: they predict the class, box offset and an output binary mask for each region of interest (RoI). Masks are generated for each class without competition between the classes.

Following the strategy described by Hollandi *et al.*[5], the network was trained for 3 epochs for different layer groups: first, all network layers were trained at a learning rate of 10^{-3}, then training was restricted to ResNet stage

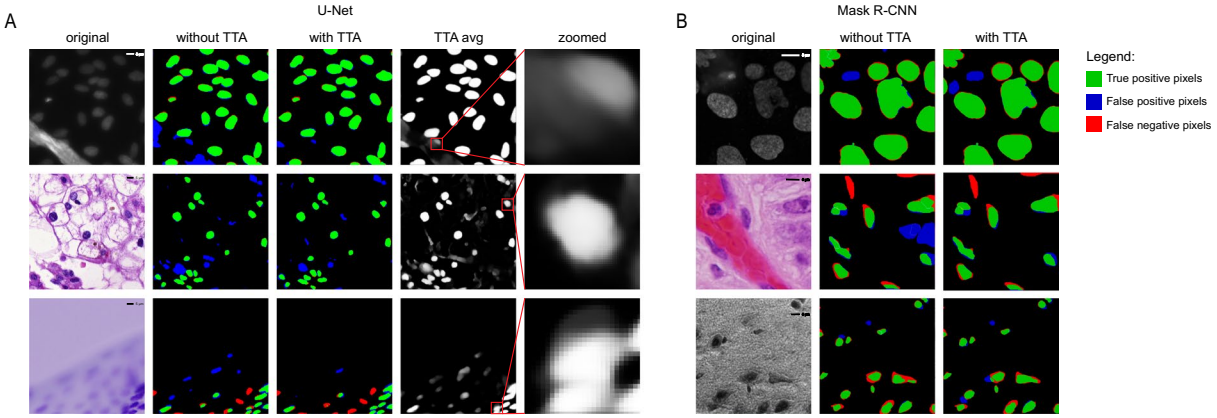


Figure 2. Examples of predictions. (**A**) U-Net predictions. First column - original image, second column - predictions without TTA compared to ground truth, third column - predictions with TTA compared to ground truth. Red indicates false negative pixels, green indicates true positive pixels and blue indicates false positive pixels. Dividing lines: yellow is false positive division of pixels into objects, and cyan is false negative division of pixels into objects. Fourth column - averaged TTA predictions before thresholding, fifth column - zoomed insets from the previous column. (**B**) Mask R-CNN predictions. Columns are the same as the first three columns in (**A**). Images in line 1 are examples of the fluorescent dataset, images in line 2 and 3 are examples of the tissue dataset.

5 (ResNet consists of 5 stages, each with convolution and identity blocks including 3 convolutional layers per block) and head layers at a learning rate of 5×10^{-4}, and finally only the head layers were trained at a learning rate of 10^{-4}. The model was initialized with pre-trained weights (https://github.com/matterport/Mask_RCNN/releases/download/v1.0/mask_rcnn_coco.h5) on the COCO dataset. The loss function of the architecture was binary cross-entropy with ADAM[22] (Adaptive Moment Estimation) solver, batch size 1, the number of iterations being equal to the train set size.

U-Net (implementation[23]) is an architecture originally designed to process biological images, which proved to be efficient, even when utilizing small training datasets. U-Net based solutions won the 2015 ISBI cell tracking challenge[4] and Kaggle Data Science Bowl 2018. Its architecture consists of two main parts: (1) a down-sampling convolution network or encoder by which we obtain the feature representation of the input image, and (2) an up-sampling convolution network or decoder, which produces the segmentation from a feature representation of the input image.

We trained U-Net for 200 epochs at a constant learning rate of 3×10^{-4}, and used a binary cross-entropy loss function with ADAM solver, batch size 1, the number of iterations being equal to the train set size.

Both U-Net and Mask R-CNN implementations are based on the deep learning framework Keras with Tensorflow backend. The training computations were conducted on a PC with NVIDIA Titan Xp GPU, 32 GB RAM and Core-i7 CPU.

Test-time augmentation. Test-time augmentation includes four procedures: augmentation, prediction, dis-augmentation and merging. We first apply augmentations on the test image. These are the same as the augmentations previously applied on the training dataset. We predict on both the original and the augmented images, then we revert the transformation on the obtained predictions; this process is referred to as dis-augmentation. For example, when the prediction was performed on a flipped or rotated image, we restore the obtained prediction to its original orientation. The final merging step is not straightforward in case of Mask R-CNN, as the architecture is instance aware, thus the merging method has to handle instances. We have developed an extended merging method inspired by one of the DSB 2018 solutions[24] (Fig. 1, right). For each detected object from the original image, we find the same detected objects in the augmented images by calculating intersection over union (IoU) between the masks. The minimum IoU threshold used to decide whether the objects found are the same is 0.5. We iterate over all detected objects to find the best match. An object should be present in the majority of the images to be included as a final mask. Next, we check the first augmented image for any remaining unused objects (a possible scenario when an object is not detected in the original image but is detected in any of the augmented ones), and look for matching unassigned objects on other augmentations. Next, we check the second augmented image for detected objects, and perform the same operations. We repeat this process until the majority voting criterion can be theoretically satisfied (in half of the images at a maximum). An average binary object mask is created by majority pixel voting on paired objects.

For U-Net the merging process is straightforward as it is not instance aware, so we simply sum and average all the dis-augmented probability maps. It yields a floating point image that needs to be converted to a binary mask. A simple element-wise thresholding at the value of 0.5 converts the soft masks into binary masks (Fig. 1, right).

Test-time augmentation evaluation. We have evaluated the test-time augmentation model on our test dataset predictions (see the previous section for details) compared to ground truth masks using the following evaluation strategies.

In case of Mask R-CNN we used the same metric as at the Data Science Bowl 2018. It calculates the mean average precision (mAP) at different intersection over union (IoU) thresholds. The thresholds (t) are in the range of [0.5, 0.95] with a step of 0.05. An object is considered true positive when the IoU with ground truth is greater than the threshold, false positive when the predicted object has no associated ground truth object or the overlap is smaller than the threshold, and false negative when the ground truth object has no associated predicted object.

$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Thus, mAP for an image is calculated as follows:

$$\frac{1}{|thresholds|} = \sum_t \frac{TP(t)}{TP(t) + FP(t) + FN(t)}$$

Next, we calculate the average for all images in the test set. The final score is a value between 0 and 1.

U-Net predictions were evaluated using the intersection over union metric, executed at the pixel level. We summed up the prediction and ground truth binary masks, then we simply counted the pixels that are greater than one (i.e. the intersection), and divided the resulting values with the number of pixels greater than zero. The resulting value is a score ranging from 0 to 1.

As described above, we have evaluated the predictions with applying TTA (*merged*) and without applying TTA (*original*). Next, we have evaluated TTA's performance by calculating the difference as $delta = merged - original$.

Results

We have evaluated the performance of TTA on two datasets, named 'Fluorescent' (fluorescent microscopy images) and 'Tissue' (histopathology images) datasets, described in the "Dataset acquisition and description" section in detail. Each of them was split in 3 different ways to have approximately 5% (one holdout set), 15% (cross-validation, 6 splits for each) and 30% (one holdout set) as a test set. By using such versatile data collected from different sources and representing a wide variety of experimental conditions, as well as by the test set splits, we aimed to present the truly general performance of TTA, and demonstrate how robustly it works. Regarding that most of these images were used in a Data Science competition, and some additional images came from other sources, our final datasets are similar to real-world scenarios.

Our choice of the two popular deep learning architectures, Mask R-CNN (yielding instances) and U-Net (semantic segmentation) also served the purpose of testing robustness, as the tasks of semantic and instance segmentation are different, and require different approaches to apply the same method to them. For each dataset/split, we have trained separate U-Net and Mask R-CNN models. Then, we have evaluated the performance of TTA for each model's checkpoint (checkpoints were made for each epoch of training: in case of U-Net, a total of '15' sets, i.e. every 10th epoch was designated as a checkpoint for cross-validation splits 2–6) as described in the "Test-time augmentation evaluation" subsection. Next, we performed statistical tests to assess whether the improvement of the performance is significant.

In the case of Mask R-CNN, TTA on average has provided an improved performance for all dataset splits and for all model checkpoints. The average mAP score *delta* is about 0.01 for all "Fluorescent" and "Tissue_5" sets and 0.02 for the other sets. In all scenarios, TTA has improved the score for most of the images (see Fig. 3 and Supplementary Fig. 1 for cross-validation splits 2–6). Such a *delta* value usually corresponds for better segmentation borders and a reduced rate of false positive or/and false negative detections.

In the case of U-Net, we have evaluated the performance at each epoch during training. For the "Tissue" dataset TTA has demonstrated a performance gain for all epochs. In case of the "Fluorescent" dataset, a slight decline in the performance of TTA was observed during early (first 30–50) epochs, which has turned positive after further training (Fig. 4A,B). After about epoch 50, the performance without TTA was seen to fluctuate without a clear trend in all cases (Fig. 4C,D), while the performance with TTA tended to rise for almost all cases, except in the case of the "Tissue" dataset, where no augmentations were used for training (Fig. 4A). A slight decline or a slight improvement in the score is usually related to cell borders (as the most uncertain regions in the images). In some cases, TTA helps to eliminate artifacts and rarely occurring false positive/false negative objects.

For some images TTA has significantly improved the final prediction. Examples of such cases for both U-Net and Mask R-CNN are shown in Fig. 2.

We have performed Wilcoxon paired test for each dataset/split/checkpoint for the Mask R-CNN results. P-values in all cases have passed the threshold value of 0.05. For U-Net, the test was performed on the means of each 10th epoch (20 vs 20 data points) for each dataset/split. The P-values are shown in Supplementary Table 4.

Applying TTA on the DSB2018 (stage2) test set of images has improved performance significantly, surpassing the best performing method[5] by 0.011 in the DSB scoring metric, which is identical to the mAP used in this paper and the output of which was a set of instance segmented masks (Fig. 5). In the context of data science competitions, when the scores are rather dense, we consider this improvement as significant (difference between 2nd and 1st place on DSB 2018 was only 0.017).

The results without TTA and *delta* values for each set are available as Supplementary Materials (Supplementary Table 1. U-Net when augmentations during training were used, Supplementary Table 2. U-Net when augmentations during training were not used, Supplementary Table 3. Mask R-CNN when augmentations during training were used).

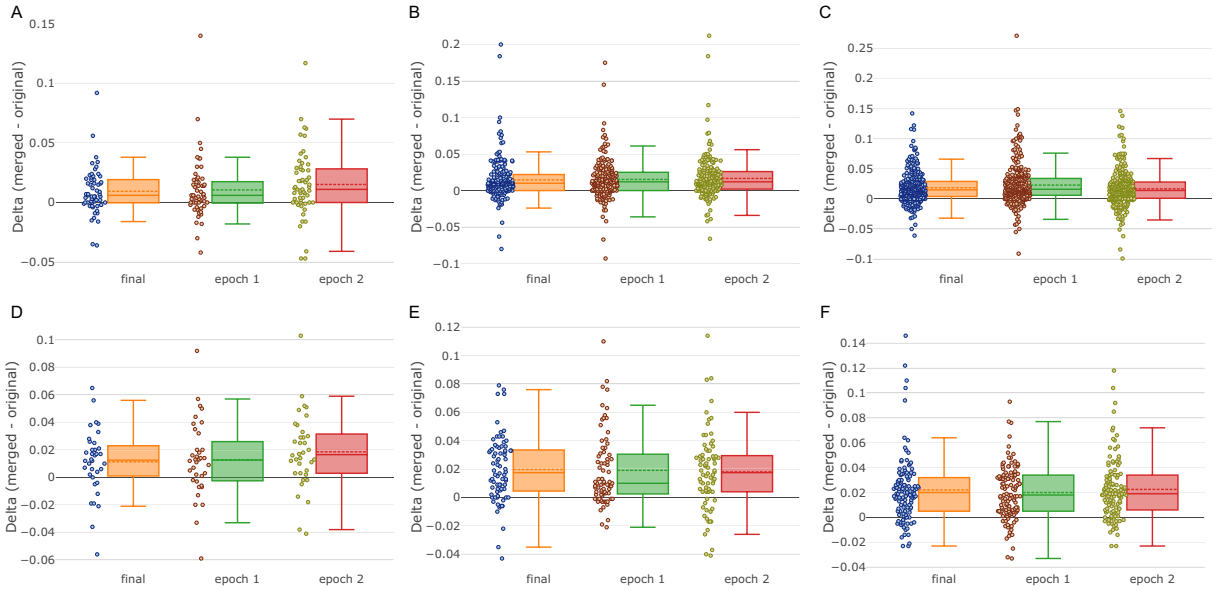


Figure 3. TTA performance for Mask R-CNN. TTA performance (*delta* = *merged* − *original*). Each point represents an image. Dashed line - mean, solid line - median. (**A**) Fluorescent set 5. (**B**) Fluorescent set 15 (cross-validation split 1). (**C**) Fluorescent set 30. (**D**) Tissue set 5. (**E**) Tissue set 15 (cross-validation split 1). (**F**) Tissue set 30. Orange boxplot - the final model (epoch 3), green boxplot - model trained for 1 epoch, red boxplot - model trained for 2 epochs.

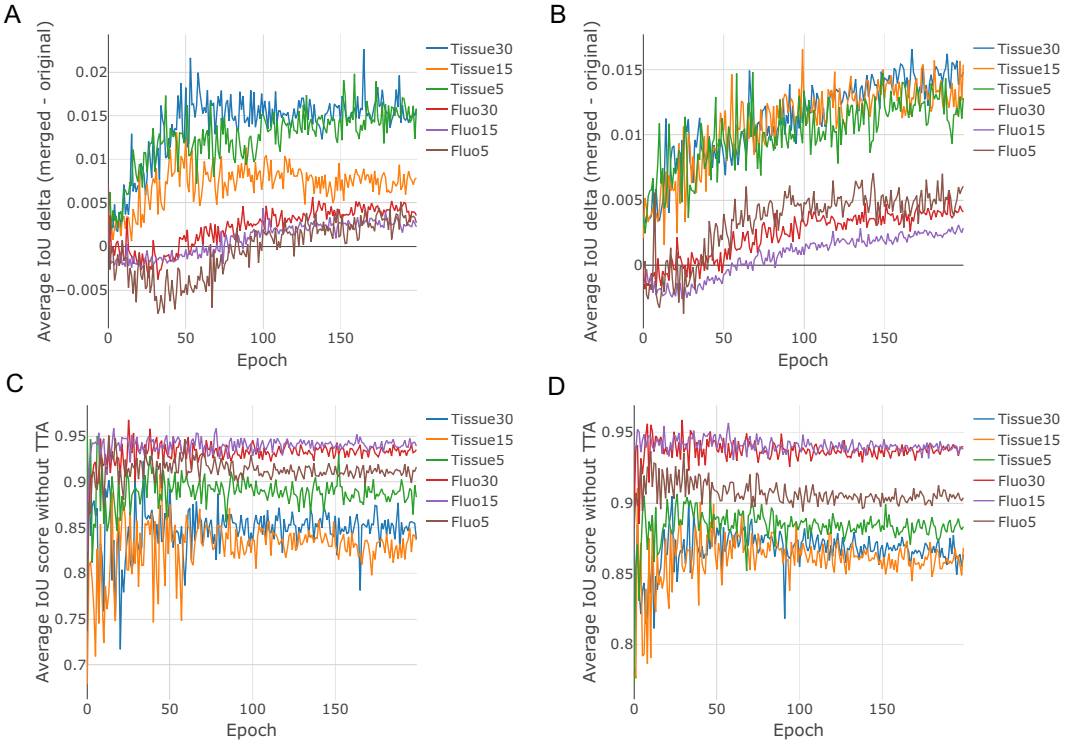


Figure 4. Average performance of TTA for U-Net with different training and test augmentations. (**A**) Average TTA performance trained without augmentations over epochs. (**B**) Average TTA performance trained with augmentations over epochs. (**C**) Average performance without TTA without augmentations during training. (**D**) Average performance without TTA with augmentations during training. Tissue15 and Fluorescent15 stand for the first cross-validation split.

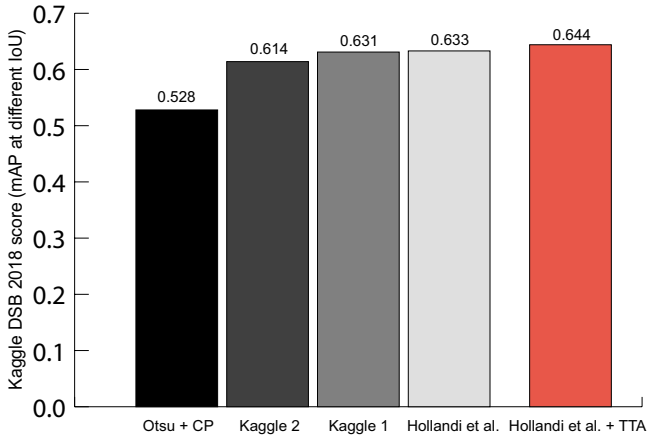


Figure 5. DSB Stage 2 scores for various methods (CellProfiler, Kaggle DSB 2018 2nd and 1st places, Hollandi *et al.*[5] method and the same method with TTA). The red bar shows the highest score.

Conclusions

We have performed experiments to estimate test-time augmentation's performance for two popular deep learning frameworks trained to segment nuclei in microscopy images. Our results indicate that on average TTA can provide higher segmentation accuracy compared to predicting based on the original images only, even though for some images the results might be marginally worse.

TTA mostly affects the objects' borders, but in uncertain cases it can help to fit whole objects (remove false positives or add true positives, especially in case of Mask R-CNN). Overall, in most cases, TTA improves segmentation accuracy. The main use case of TTA is the analysis of uncertain regions in segmentation. However, the high cost of TTA, related to the fact that multiple times more predictions are required for the same object, is also an issue to be considered. Therefore, TTA is mainly recommended for use when the basic cost of prediction is low.

Received: 31 October 2019; Accepted: 20 February 2020;
Published online: 19 March 2020

References

1. Carpenter, A. E. *et al.* CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome Biol.* **7**, R100 (2006).
2. Molnar, C. *et al.* Accurate Morphology Preserving Segmentation of Overlapping Cells based on Active Contours. *Sci. Rep.* **6**, 32412 (2016).
3. Molnar, J., Molnar, C. & Horvath, P. An Object Splitting Model Using Higher-Order Active Contours for Single-Cell Segmentation. *Advances in Visual Computing* 24–34, https://doi.org/10.1007/978-3-319-50835-1_3 (2016).
4. Ronneberger, O., Fischer, P. & Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. *Lecture Notes in Computer Science* 234–241, https://doi.org/10.1007/978-3-319-24574-4_28 (2015).
5. Hollandi, R. *et al.* A deep learning framework for nucleus segmentation using image style transfer. *bioRxiv* 580605, https://doi.org/10.1101/580605 (2019).
6. Dobos, O., Horvath, P., Nagy, F., Danka, T. & Viczián, A. A deep learning-based approach for high-throughput hypocotyl phenotyping. *bioRxiv* 651729, https://doi.org/10.1101/651729 (2019).
7. Caicedo, J. *et al.* Nucleus segmentation across imaging experiments: the 2018 Data Science Bowl. *Nature Methods*, https://doi.org/10.1038/s41592-019-0612-7 (2019).
8. He, K., Gkioxari, G., Dollar, P. & Girshick, R. Mask R-CNN. *IEEE Trans. Pattern Anal. Mach. Intell.*, https://doi.org/10.1109/TPAMI.2018.2844175 (2018).
9. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet classification with deep convolutional neural networks. *Communications of the ACM* **60**, 84–90 (2017).
10. Matsunaga, K., Hamada, A., Minagawa, A. & Koga, H. *Image Classification of Melanoma, Nevus and Seborrheic Keratosis by Deep Neural Network Ensemble.* (2017).
11. Ayhan, M. S. & Berens, P. *Test-time Data Augmentation for Estimation of Heteroscedastic Aleatoric Uncertainty in Deep Neural Networks*. (2018).
12. Wang, G. *et al.* Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neurocomputing* **338**, 34–45 (2019).
13. Ayhan, M. S. *et al.* Expert-validated estimation of diagnostic uncertainty for deep neural networks in diabetic retinopathy detection. *medRxiv* 19002154 (2019).
14. Brasko, C. *et al.* Intelligent image-based *in situ* single-cell isolation. *Nature Communications* **9** (2018).
15. Caicedo, J. C. *et al.* Evaluation of Deep Learning Strategies for Nucleus Segmentation in Fluorescence Images., https://doi.org/10.1101/335216.
16. Caie, P. D. *et al.* High-content phenotypic profiling of drug response signatures across distinct cancer cells. *Mol. Cancer Ther.* **9**, 1913–1926 (2010).
17. Coelho, L. P., Shariff, A. & Murphy, R. F. Nuclear segmentation in microscope cell images: A hand-segmented dataset and comparison of algorithms. *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, https://doi.org/10.1109/isbi.2009.5193098 (2009).
18. Smith, K. *et al.* CIDRE: an illumination-correction method for optical microscopy. *Nature Methods* **12**, 404–406 (2015).
19. Naylor, P., Lae, M., Reyal, F. & Walter, T. Segmentation of Nuclei in Histopathology Images by Deep Regression of the Distance Map. *IEEE Transactions on Medical Imaging* **38**, 448–459 (2019).

20. Kumar, N. *et al*. A Dataset and a Technique for Generalized Nuclear Segmentation for Computational Pathology. *IEEE Trans. Med. Imaging* **36**, 1550–1560 (2017).
21. Matterport. matterport/Mask_RCNN. *GitHub* Available at, https://github.com/matterport/Mask_RCNN. (Accessed: 7th October 2019).
22. Kingma, D. P. & Ba, J. *Adam: A Method for Stochastic Optimization* (2014).
23. Zhixuhao. zhixuhao/unet. *GitHub* Available at, https://github.com/zhixuhao/unet. (Accessed: 7th October 2019).
24. Mirzaevinom. mirzaevinom/data_science_bowl_2018. *GitHub* Available at, https://github.com/mirzaevinom/data_science_bowl_2018. (Accessed: 7th October 2019).

Acknowledgements
N.M., R.H., B.M., and P.H. acknowledge support from the LENDULET-BIOMAG Grant (2018–342) and from the European Regional Development Funds (GINOP-2.3.2-15-2016-00006, GINOP-2.3.2-15-2016-00037), N.M. is supported by the Doctoral School of Interdisciplinary Medicine of the University of Szeged and the Doctoral School of Computer Science at the National Research University, Higher School of Economics. The authors acknowledge an NVIDIA grant for TitanXp GPUs. The authors thank Dora Bokor, PharmD, for proofreading the manuscript.

Author contributions
N.M., B.M., R.H., A.K.-F. and P.H. wrote the manuscript. N.M., B.M. and R.H. performed the experiments. R.H. collected the dataset. R.H. and N.M. prepared the figures for the paper.

Competing interests
The authors declare no competing interests.

Additional information
Supplementary information is available for this paper at https://doi.org/10.1038/s41598-020-61808-3.

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

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

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