# Evaluation of Deep Learning Strategies for Nucleus Segmentation in Fluorescence Images

## Authors

Juan C. Caicedo[1], Jonathan Roth[1,2], Allen Goodman[1], Tim Becker[1], Kyle W. Karhohs[1], Claire McQuin[1], Shantanu Singh[1], Fabian Thies[2], Anne E. Carpenter[1,*]

*1 Imaging Platform -- Broad Institute of MIT and Harvard, Cambridge, MA. USA.*
*2 Institute of Computational Biology -- German Research Center for Environmental Health, Germany.*
** corresponding author: anne@broadinstitute.org*

## Abstract

Identifying nuclei is often a critical first step in analyzing microscopy images of cells, and classical image processing algorithms are still commonly used for this task. Recent studies indicate that deep learning may yield superior accuracy, but its performance has not been evaluated for high-throughput nucleus segmentation in large collections of images. We compare two deep learning strategies for identifying nuclei in fluorescence microscopy images (U-Net and DeepCell) alongside a classical approach that does not use machine learning. We measure accuracy, types of errors, and computational complexity to benchmark these approaches on a large data set. We publicly release the set of 23,165 manually annotated nuclei and source code to reproduce the results. Our evaluation shows that U-Net outperforms both pixel-wise classification networks and classical algorithms. Although deep learning requires more computation and annotation time than classical algorithms, it improves accuracy and halves the number of errors.

# Main

Image analysis is a powerful tool in cell biology to collect quantitative measurements in time and space with precision, speed, and sensitivity. From image-based assays to high-content screening [1,2], microscopy images have led to understanding genetic perturbations, pursuing drug discovery, and phenotyping cells in biomedical applications and cell biology research [3,4]. The most widely used quantitative imaging technique in biological laboratories is fluorescence imaging; with automation it can easily produce terabytes of primary research data [5]. Accurate and automated analysis methods are key to successfully quantify relevant biology in such large image collections.

One critical step in quantifying fluorescent images is often the identification of the nucleus of each cell with a DNA stain, and there is a long history of research efforts to designing and improving nuclear and cellular segmentation [6]. One of the most commonly used strategies for nucleus segmentation is Otsu's thresholding method [7] followed by seeded watershed [8,9], because of its effectiveness, simplicity of use and computational efficiency. Machine learning-based segmentation methods have also been introduced for segmenting cells [10], which typically require annotated examples in the form of segmentation masks or interactive scribbles. Many of these strategies are readily available in various bioimage software packages [11], including open source options such as CellProfiler [12], Ilastik [10], and ImageJ/Fiji [13], facilitating their adoption in routine biological research.

Despite widespread adoption, segmentation tools in biology generally do yield non-trivial amounts of segmentation error. These may silently propagate to downstream analyses, yielding unreliable measures or systemic noise that is difficult to quantify and factor out. There are several causes for segmentation errors. First, existing algorithms have limitations due to the assumptions made in the computational design that do not always hold, such as thresholding methods that assume bimodal intensity distributions, or region growing that expects clearly separable boundaries. Second, the most popular solutions for nucleus segmentation were originally formulated and adopted several decades ago when the biological systems and phenotypes of interest were often simpler; however, as biology pushes the limits of high-throughput cellular and tissue models, natural and subtle variations of biologically meaningful phenotypes are more challenging to segment. Finally, algorithms are usually configured using a few −hopefully representative− images from the experiment, but variations in signal quality and the presence of noise pose challenges to the robustness and reliability of the solution at large scale.

The ideal approach to nucleus segmentation would be a generic, robust and fully automated solution that is as reliable as modern face detection technologies deployed in mobile applications and social networks. The current state of the art in face detection and many other computer vision tasks is based on deep learning [14], which has demonstrated high accuracy, even surpassing human-level performance in certain tasks [15]. Several models based on deep learning have already been proposed for cell segmentation in biological applications, most notably U-Net [16] and DeepCell [17], which are based on convolutional neural networks.

In this paper, we evaluate the performance of these two deep learning-based strategies, and investigate their potential to improve the accuracy of nucleus segmentation in fluorescence images. Expert biologists on our team hand-annotated more than 20,000 nuclei in an image collection of 200 images of the DNA channel from a large image-based chemical screen, sampled from a diverse set of treatments [18]. Our evaluation methodology measures the success of identifying objects, whereas previous studies focused on pixel or boundary accuracy measures [17,19] that may ignore certain object-level errors. We also analyze different types of segmentation errors in each method, computational efficiency, and the impact of quantity and quality of training data for creating deep learning models.

# Results

Identifying nuclei in an image is best framed as an “instance segmentation” problem [20], where the challenge is to find distinct regions corresponding to a single class of objects: the nucleus. Semantic segmentation [21], which splits an image to regions of various classes without requiring objects to be separated, is not helpful for nucleus segmentation because there is only one class, and touching nuclei would not be distinguished from each other. Both of the deep learning strategies evaluated in this paper are cases of instance segmentation that formulate nucleus segmentation as a boundary detection problem.

The boundary detection problem consists of identifying three different types of pixels in an image of nuclei: 1) background, 2) interior of nuclei, and 3) boundaries of nuclei. This formulation simplifies the problem of instance segmentation into a three-class, pixel-wise classification problem (Figure 1). If the boundary mask is correctly predicted, individual instances of nuclei can be recovered from the interior mask using a connected component labeling algorithm [22], thus successfully distinguishing two or more touching nuclei. Note that while we pose this as a pixel-wise classification problem of boundaries, we evaluate the performance on the success of identifying entire objects.

A diverse set of convolutional neural network (CNN) architectures can address pixel-wise classification; here we evaluate two, representing two families of prominent models: DeepCell [17] and U-Net [16] (Online Methods). We use the same preprocessing and postprocessing pipeline when evaluating both CNN models, so differences in performance are explained by architectural choices only.

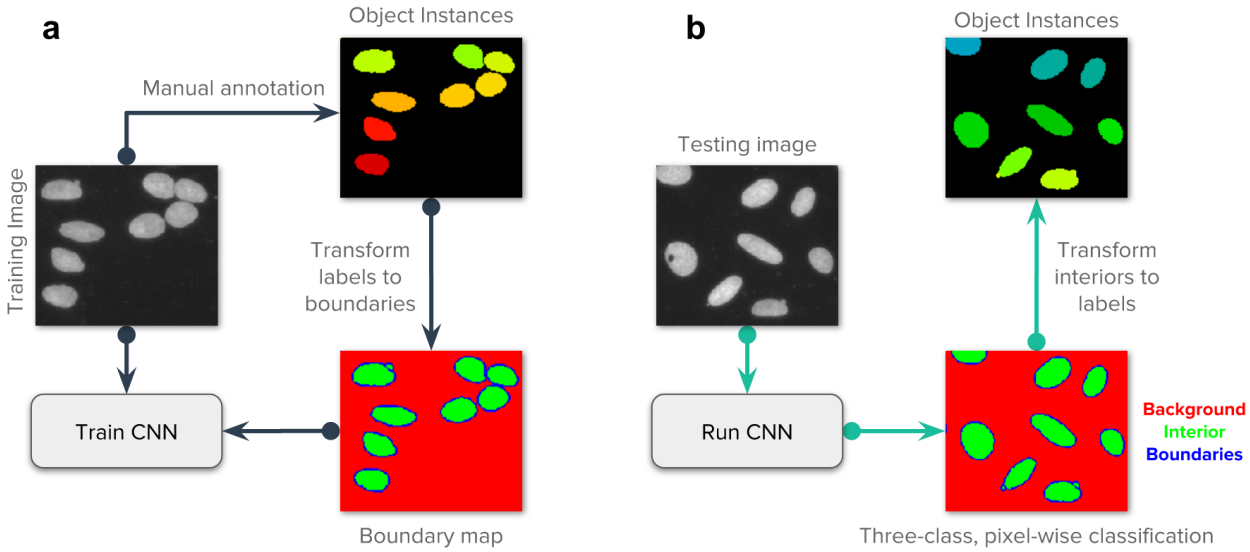


**Figure 1.** ***Strategy of the evaluated deep learning approaches.*** *Our main goal is to follow the popular strategy of segmenting each nucleus and micronucleus as a distinct entity, regardless of whether it shares the same cell body with another nucleus. It is generally possible to group nuclei within a single cell using other channels of information in a post-processing step if the assay requires it. a) Images of the DNA channel are manually annotated, labeling each nucleus as a separate object. Then, labeled instances are transformed to masks for background, nucleus interior and boundaries. A convolutional neural network (CNN) is trained using the images and their corresponding masks. b) The trained CNN generates predictions for the three class classification problem. Each pixel belongs to only one of the three categories. In post-processing, the predicted boundary mask is used to identify each individual instance of a nucleus.*

## Deep learning improves nucleus segmentation accuracy

Overall, we find that deep learning models exhibit higher accuracy than classical segmentation algorithms, both in terms of the number of correctly identified objects, as well as the localization of boundaries of each nucleus (Figure 2). We evaluate these properties using the F1 score (the harmonic average of precision and recall) averaged over increasingly stringent thresholds of overlap between the ground truth and prediction. U-Net and DeepCell obtained higher average F1 scores, yielding 0.85 and 0.78 respectively, versus 0.74 and 0.72 for the advanced and basic CellProfiler pipelines selected as a baseline (see Online Methods). This 20% improvement is a significant margin when experiments are run at large scale with thousands of images. U-Net yields a higher average F1 score across larger thresholds (Figure 2a), indicating that the boundaries of objects are more precisely mapped to the correct contours compared to the other methods.

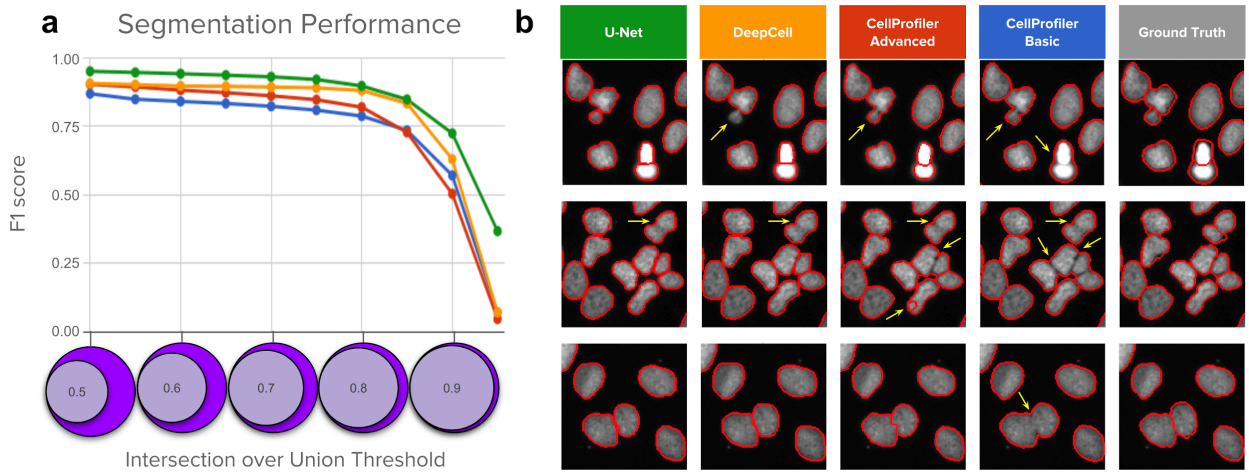


**Figure 2.** ***Segmentation performance of four strategies compared against ground-truth expert segmentations.*** *a) Average F1 score vs. nucleus coverage for U-Net (green), DeepCell (yellow), CellProfiler advanced (red), and CellProfiler basic (blue). The y axis is average F1 score (higher is better), which measures the proportion of correctly segmented objects. The x axis represents intersection-over-union (IoU) thresholds as a measurement of how well aligned the ground truth and estimated segmentations must be to count a correctly detected nucleus. Higher thresholds indicate stricter boundary matching. Notice that average F1 scores remain nearly constant up to IoU=0.80; at higher thresholds, performance decreases sharply, which indicates that the proportion of correctly segmented objects decreases when stricter boundaries are required to count a positive detection. b) Example segmentations obtained with each of the four evaluated methods sampled to illustrate performance differences. Segmentation boundaries are in red, and errors are indicated with yellow arrows.*

The most common errors for all methods are merged objects, which occur when the segmentation fails to separate two or more touching nuclei (yellow arrows in Figure 2b). Deep learning strategies tend to reduce this type of error (more in Figure 3c) and provide tighter and smoother segmentation boundaries than those estimated by global Otsu thresholding and declumping, which is at the core of the baseline CellProfiler pipelines for nucleus segmentation.

Qualitatively, nucleus boundaries predicted by the U-Net appear to define objects better than those produced by human annotators using an assistive annotation tool, which can introduce boundary artifacts (Online Methods). Neural nets can learn to provide edges closer to the nuclei with fewer gaps and better-delineated shapes, despite being trained with examples that have such boundary artifacts, showing ability to generalize beyond noise. Overcoming the limitation of assisted annotations is a major strength of this approach because fixing boundary artifacts by hand in the training data is very time consuming. We suspect that the accuracy drop observed in the segmentation performance plot at IoU=0.85 (Figure 2a) may be partly explained by inaccurate boundaries in ground truth annotations, i.e. improved segmentations may be unfairly scored at high thresholds.

## Deep learning excels at correct splitting of adjacent nuclei

Deep learning methods make fewer segmentation mistakes compared to classical pipelines, effectively correcting most of their typical errors (Figure 3a). Here, an error is defined as when a nucleus in the ground truth is missed in an estimated segmentation mask after applying a

minimum IoU threshold of 0.7. By this metric, U-Net achieves an error rate of 8.1%, DeepCell 14.0%, advanced CellProfiler 15.5% and basic CellProfiler 20.1%. These results are consistent with the evaluation of accuracy performed at multiple IoU thresholds, indicating that U-Net obtains significantly better performance.

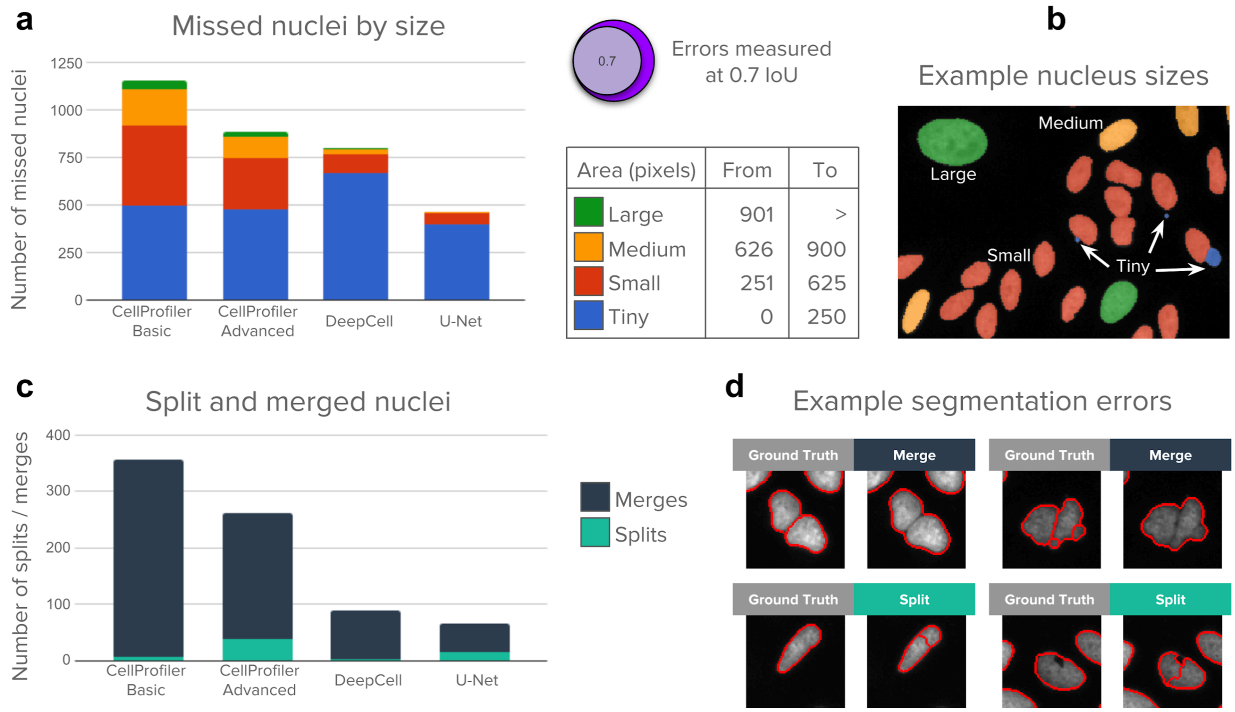


**Figure 3.** ***Analysis of segmentation errors (missed, splits, merged objects).*** *The 5,720 nuclei in the test set were used in this analysis. a) Counts of missed nuclei by object size (see table). Missed objects in this analysis were counted using an IoU threshold of 0.7, which offers a good balance between strict nucleus coverage and robustness to noise in ground truth annotations. b) Example image illustrating sizes of nuclei. c) Counts of merged and split nuclei. These errors are identified by masks that cover multiple objects with at least 0.1 IoU. d) Example merges and splits.*

To understand the performance differences among the evaluated methods, we categorized missed objects by size (Figure 3a, b) and segmentation errors by type (merges vs. splits) (Figure 3b, c). An object is missed when the segmentation does not meet the minimum IoU threshold criterion. A merge is counted when one estimated mask is found covering more than one ground truth mask. Similarly, a split is counted when a ground truth mask is being covered by more than one estimated mask. Note that splits and merges are a subset of the total number of errors, and partially overlap with the number of missed objects. That is, some splits and all merges result in one or more missing objects, but not all missing objects are a result of a split or merge.

Deep learning corrects almost all of the errors made by classical pipelines for larger nuclei. We also note that all methods usually fail to capture tiny nuclei correctly (generally, micronuclei, which are readily confounded with debris or artifacts, and represent about 15% of all objects in the test set) (Figure 3a). Interestingly, although they make almost the same number of mistakes (14.0% vs 15.5%), DeepCell tends to accumulate errors for tiny nuclei only while the advanced

CellProfiler pipeline tends to make errors across all sizes (Figure 3b). Tiny nuclei may be particularly hard to segment using the boundary detection approach because most of the pixels are encoded in the boundary, and very few pixels are left for the interior which is used to produce segmentation masks. This might be addressed in part by increasing image resolution, either optically or computationally (Discussion).

Both deep learning approaches are effective at recognizing boundaries to separate touching nuclei and correct typical error modes of classical algorithms: merges and splits (Figure 3c and 3d). Split errors produced by the advanced CellProfiler pipeline reveal a trade-off when configuring the parameters of classical algorithms: in order to fix merges we have to accept some more splits. A similar situation happens with U-Net: it has learned to separate clumped nuclei very effectively because the boundary class has 10 times more weight in the loss function (Online Methods), which at the same time forces the network to make some splits to avoid the cost of missing real boundaries.

## More training data improves accuracy and reduces errors

Using U-Net models only –given their better performance and faster running times– we found that training with just two images performs more accurately than an advanced CellProfiler pipeline (Figure 4a). This is consistent with previous reports on DeepCell [17] and U-Net [16], which were designed to learn from few images by processing patches and using data augmentation. Since training a convolutional neural network requires the additional effort of manually annotating example images for learning, limiting the investment of time from expert biologists is valuable.

Providing more annotated examples improved segmentation accuracy and reduced the number of errors significantly (Figure 4). Accuracy improves with more data, gaining a few points of performance as more annotated images are used, up to the full 100 images in the training set (Figure 4a). We found little difference in this trend whether using basic data augmentation vs. using extra augmentations based on elastic deformations (Online Methods).

Segmentation errors are reduced significantly with more annotated examples, by roughly half (Figure 4b), but as above, even training with two images produces results better than the advanced CellProfiler baseline. Touching nuclei particularly benefit from more training data, which helps to reduce the number of merge errors. The trend for split errors is to increase with more data as an effect of learning to recognize difficult boundaries; however, this represents a very small fraction of the total number of errors that are still fewer than the number of splits made by the advanced CellProfiler pipeline.

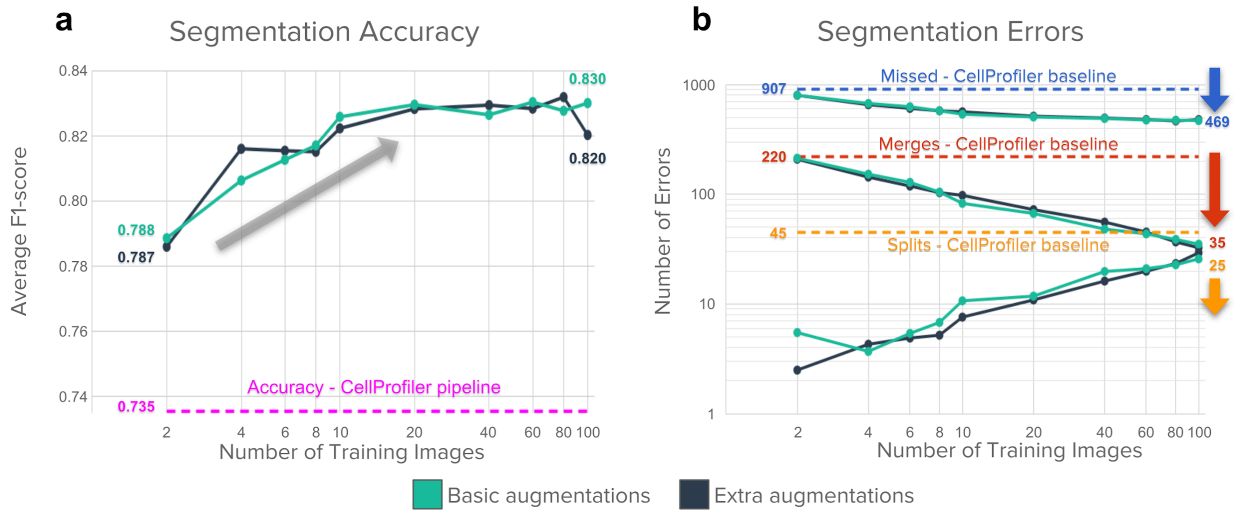


**Figure 4.** ***Impact of the number of annotated images used for training a U-Net model.*** *Basic augmentations include flips, 90 degree rotations and random crops. Extra augmentations include the basic plus elastic deformations. a) Accuracy improves as a function of the number of training images, up to a plateau around 20 images (representing roughly 2,000 nuclei). b) Segmentation errors are reduced overall as the number of training images increases, but the impact differs for merges vs. splits. The advanced CellProfiler pipeline is shown as dotted lines throughout. Results are reported using the validation set to prevent over-optimizing models in the test set (holdout). For all experiments, we randomly sampled (with replacement) subsets (n = 2, 4, 6, 8, 10, 20, 40, 60, 80, 100) of images from the training set (n=100) and repeated 10 times to evaluate performance. Data points in plots are the mean of repetitions. Although the percent overlap between the random samples increases with increasing sample size, and is 100% for n=100, we nonetheless kept the number of repeats fixed (=10) for consistency.*

## Providing a variety of training images improves generalization

We found that training with images that exhibit different types of noise produces models that transfer better to other sets (Figure 5a). In contrast, training on an image set that has homogeneous acquisition conditions does not transfer as well to other experiments (Figure 5b). We tested U-Net models trained on one image set and evaluated their performance when transferring to another set. In one case we took images of nuclei from a prior study ("Van Valen's Set")[17], representing different experiments and including representative examples of diverse signal qualities, cell lines, and acquisition conditions (Figure 5d). In the other case we used the BBBC022 image collection which exhibits homogeneous signal quality and was acquired under similar technical conditions (Figure 5c).

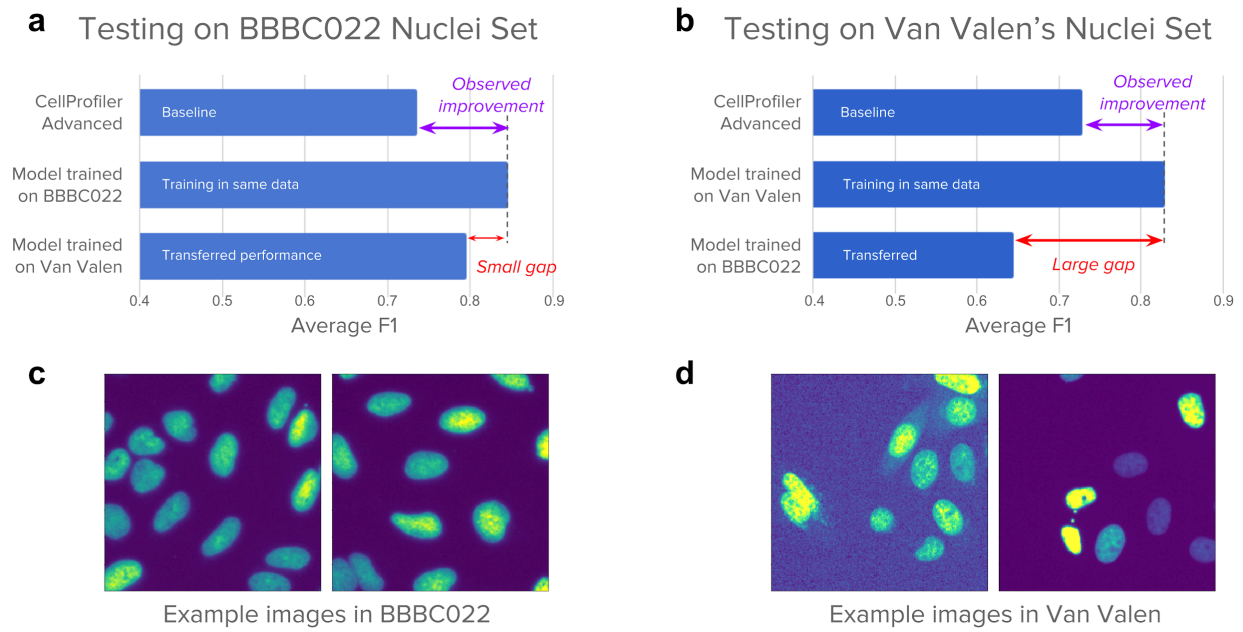


**Figure 5.** ***Signal quality is the main challenge when transferring models across experiments:*** *Performance differences when models are trained and evaluated in different experiments. a) Models evaluated on the BBBC022 test set, including a U-Net trained on the same set, another U-Net trained on Van Valen's set, and a CellProfiler pipeline. The results indicate that transferring the model from one screen to another can bring improved performance. b) Models evaluated in Van Valen's test set, including CellProfiler baselines adapted to this set (Online Methods), a U-Net trained on the same set, and another U-Net trained on BBBC022. The results illustrate the challenges of dealing with large signal variation. c) Example images from BBBC022 showing homogeneous signal with uniform background. d) Example images from Van Valen's set illustrating various types of realistic artifacts, such as background noise and high signal variance. Number of training images: 100 in BBBC022 and 9 in Van Valen. Number of test images: 50 in BBBC022 and 3 in Van Valen.*

A model trained only on the 9 diverse images of Van Valen's set generalizes well to test images in BBBC022, improving performance over the baseline (Figure 5a) and reaching comparable performance to the model trained on BBBC022 training images. Note that training a network on images of BBBC022 improves performance with respect to the CellProfiler baseline. The transferred model does not fix all the errors, likely because the number of training examples is limited. Nevertheless, the transferred performance indicates that it is possible to reuse models across experiments to improve segmentation accuracy.

A transfer from the more homogenous BBBC022 set to the more diverse Van Valen set is less successful: a model trained with 100 examples from the BBBC022 set fails to improve on the test set of Van Valen's images despite the availability of more data (Figure 5b). This demonstrates the challenges of dealing with varying signal quality, which is a frequent concern in high-throughput and high-content screens. The large gap in performance is explained by varying signal conditions (Figure 5c, d): because the U-Net did not observe these variations during training, it fails to correctly segment test images.

The CellProfiler pipelines also confirm the difficulty of handling noisy images. A single pipeline cannot deal with all variations in Van Valen's test set, requiring the adjustment of advanced settings and the splitting of cases into two different pipelines (Online Methods). In BBBC022, a

single pipeline works well due in part to the homogeneity of signal in this collection; the errors are due to challenging phenotypic variations, such as tiny nuclei or clumped objects.

## Deep learning needs more computing and annotation time than classical methods

Although we found the performance of deep learning to be favorable in terms of improving segmentation accuracy, we also found that this comes at higher computational cost and annotation time. First, deep learning requires significantly more time to prepare training data with manual annotations (Figure 6a). Second, deep learning needs the researchers to train a model and tune its parameters (Figure 6b), usually with special hardware. Third, when a model has been trained, it is slower to run on new images than classical algorithms (Figure 6c). However, running times can be accelerated using graphic cards, which makes the technique usable in practice (Figure 6d).

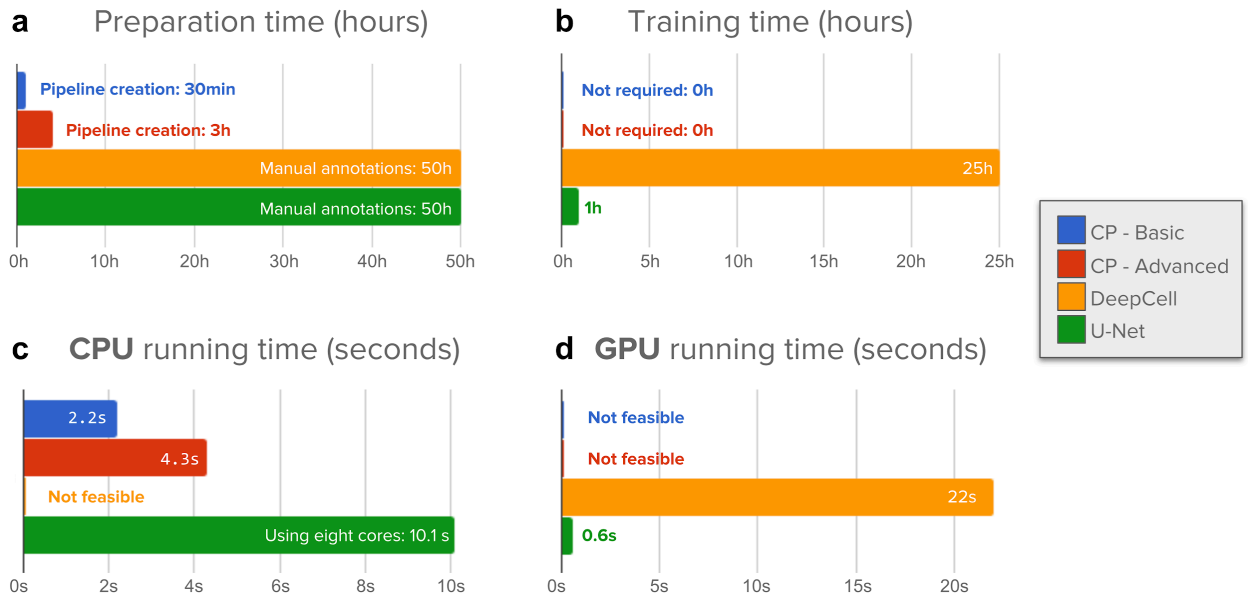


**Figure 6**. ***Evaluation of the time needed to create annotations, train, and run segmentation models.*** *a) Preparation time measures hands on, expert time annotating images or creating CellProfiler pipelines. Manually annotating 100 training images with about 11,500 nuclei requires significantly longer times. b) Neural networks need to be trained while CellProfiler pipelines do not need additional processing. Training algorithms were run on a single NVIDIA Titan X GPU. DeepCell trains an ensemble of 5 models, which was used in all other evaluations. c) CellProfiler pipelines and trained neural networks are run on new images using only CPU cores to measure the computational cost of segmenting a single image. Deep learning needs significantly more resources to accomplish the task. d) Deep learning models can be accelerated using GPUs, which have thousands of computing cores that allow algorithms to run operations in parallel. This reduces significantly the elapsed time, making it practical and even faster than classical solutions. The GPU used in this evaluation is a single NVIDIA Titan X GPU.*

We observed that the time invested by experts for annotating images is significantly longer than configuring CellProfiler segmentation pipelines (Figure 6a). We estimate that manually annotating 100 images for training (~11,500 objects) takes 50 hours of work using an assisted-segmentation tool (Online Methods). In contrast, a basic CellProfiler pipeline can be calibrated

in 15 to 30 minutes of interaction with the tool, setting up a configuration that even users without extensive experience nor computational expertise could complete. CellProfiler is very flexible and allows users to add more modules in order to correct certain errors and factor out artifacts, creating an advanced pipeline that can take from 1 to 3 hours.

Training deep learning models takes substantial computing time on GPUs, while CellProfiler pipelines do not need any additional training or post-processing (Figure 6b). In our study, the deep learning models under evaluation (Online Methods) are big enough to need a GPU for training, but light enough to be trained in a few hours. In particular, a U-Net can be trained in a single Nvidia Titan X GPU in just one hour, while DeepCell takes 25 hours (an ensemble of 5 models as suggested in the original work [17], which required 5 hours each in our experiments). Also, training models may need preliminary experiments to calibrate hyperparameters of the neural network (e.g. learning rate, batch size, epochs), which adds more hands-on time.

When segmenting new images using CPU cores, deep learning models are slower than CellProfiler pipelines (Figure 6c). The computational complexity in terms of space (memory) and time (operations) of a convolutional neural network is proportional to the number of layers, the number of filters, and the size of images. As these architectures get deeper and more complex, they involve more operations to produce the final result, and thus require more computing power. This is in contrast to classical segmentation algorithms whose thresholding and filtering operations have relatively limited computing requirements that scale with the size of images. Even with 8 times more cores, a U-Net takes 10.1 seconds to segment a single image, which results in about 20 times more computing power requirements than the advanced CellProfiler pipeline. CellProfiler pipelines are run in a single CPU core and take 2.2 and 4.3 seconds for the basic and advanced pipelines respectively.

Using GPU acceleration can significantly speed up the computations of deep learning models, making them very usable and efficient in practice (Figure 6d). Segmenting a single image with a U-Net model takes only 0.6 seconds on a Nvidia Titan X GPU, improving computation times by a factor of 16X. Note that no batching was used for prediction, which can accelerate computation of groups of images even further. This result shows that a deep learning model is faster than classical algorithms when using appropriate hardware. The DeepCell model takes significantly more computing time even with GPU acceleration, which is explained by the patch-based design of the model not fully exploiting parallel hardware. However, DeepCell is more memory efficient and could process larger images without tiling.

# Discussion

Our results show that deep learning strategies improve segmentation accuracy and reduce the number of errors significantly as compared to baselines based on classical image processing algorithms. In our benchmark, U-Net provided the best performance in all the tests that measured accuracy and error rates. Despite requiring significant annotation effort and

computational cost, deep learning methods can have a positive impact on the quality and reliability of the measurements extracted from fluorescence images.

## Improved accuracy

The results show that U-Net outperforms DeepCell in all our benchmarks. This can be explained by several architectural differences between the two models: U-Net makes special use of skip connections, which are known to improve training stability, speed, and accuracy of convolutional neural networks [23]. In contrast, DeepCell follows a simpler design with a stack of convolutional layers. Also, the U-Net architecture has a total of 7.7 million parameters versus 2.5 million for DeepCell, which translates in more learning capacity in the U-Net. More parameters may make the model prone to overfitting; however, we did not observe evidence of this being a problem in our experiments with a single dataset, i.e., test performance did not drop significantly with respect to training and validation when running evaluations in the BBBC022 set.

The analysis of errors indicates that deep learning can fix most of the segmentation errors observed in classical algorithms, especially merges. One special type of error that represents a challenge for both deep learning models is the segmentation of tiny nuclei. If needed for a given application, this could be solved by increasing the resolution of images, either during acquisition [24] or with computational methods such as resizing images to make objects look bigger. Alternatively, different loss functions adapted to this problem might be designed.

## Training data

In our evaluation, the amount of training data was shown to be an important factor to reduce the number of errors. Our results confirm that training a neural network with only a few images is enough to get improved performance relative to non-deep learning baselines. However, in order to improve accuracy and leverage the learning capacity of deep learning models, more data is required. Importantly, a neural network can also be reused across experiments, as long as the training data incorporates variations in morphological phenotypes as well as variations in signal quality and acquisition conditions. We argue that a single deep learning model might be constructed to address all the challenges of nucleus segmentation in fluorescence images if a diverse database of annotated examples were to be collected to incorporate these two critical axes of variation. We advocate for collecting that data collaboratively from different research labs, so everyone will benefit from a shared resource that can be used for training robust neural networks. We have begun such an effort via the 2018 Data Science Bowl https://www.kaggle.com/c/data-science-bowl-2018/.

## Computational cost

We observed that U-Net is faster than DeepCell when using GPU acceleration despite U-Net's largernumber of parameters. By design, DeepCell processes images using a patch-based approach for single pixel classification, which involves redundant computations and does not

take full advantage of GPU parallelism. CellProfiler pipelines do not require acceleration, but are limited by the accuracy of the underlying algorithms.

Deep learning models generally run a higher computational cost. GPUs can be useful in microscopy laboratories for accelerating accurate neural network models; if acquisition or maintenance is prohibitive, cloud computing allows laboratories to run deep learning models using remote computing resources on demand. Adopting these solutions will equip biologists with essential tools for many other image analysis tasks based on artificial intelligence in the future.

# Online Methods

## Experimental Design

The main dataset in this paper, BBBC022, was split in three subsets, using 50% of the images for training, 25% for validation, and 25% for testing (holdout set). All optimization steps of the deep learning strategies were tuned using the training and validation sets, and the reported results are obtained with final evaluations using the holdout test set (unless otherwise indicated).

We evaluate four segmentation strategies, two based on deep learning, and two baseline pipelines based on the CellProfiler software. The deep learning strategies are U-Net [16], representing a family of neural networks originally designed for segmentation; and DeepCell [17], representing another family of neural networks for patch-based, pixel-wise classification. The baseline segmentation pipelines include an advanced settings mode and a basic settings mode.

With trained deep learning models and calibrated CellProfiler pipelines, we proceeded to segment all images in the test set. To evaluate and compare the estimated masks we use the average F1 score and modes of error.

## Image Collection

The image set is a high-throughput experiment of chemical perturbations on U2OS cells, comprising 1,600 bioactive compounds [18]. The effect of treatments was imaged using the Cell Painting assay [25] which labels cell structures using six stains, including Hoechst for nuclei. From this image collection, we randomly sampled 200 fields of view of the DNA channel, each selected from a different compound. By doing so, phenotypes induced by 200 distinct chemical perturbations were sampled.

The original image collection is part of the Broad Bioimage Benchmark Collection, with accession number BBBC022, and publicly available at https://data.broadinstitute.org/bbbc/BBBC022/. We will contribute the set of manually annotated nuclei to this collection.

## Expert Annotations

Each image in the sampled subset was reviewed and manually annotated by PhD-level expert biologists. Annotations were made to label each single nucleus as a distinguishable object, even if nuclei happen to be clumped together or appear to be touching each other. Nuclei of all sizes and shapes were included as our goal was to densely annotate every single nucleus that can be recognized in the sampled images, regardless of its phenotype. In this way, a wide variety of phenotypes was covered, including micronuclei, toroid nuclei, fragmented nuclei, round nuclei, and elongated nuclei, among others [18].

We created a web-based annotation tool to complete single-object masks for all images. Our user interface allows the expert to zoom in and out to double check details, and also presents the annotation mask overlaid on top of the original image using a configurable transparency layer. Importantly, our annotation tool is based on assisted segmentation based on superpixels, which are computed based on intensity features to facilitate user interactions. The source code for the annotation tool is publicly available in the following repository: https://github.com/0x00b1/annotation-prototype/.

## DeepCell

DeepCell [17] is a CNN-based strategy to segment images of cells, using a patch-based, single pixel classification objective. The network architecture has seven convolutional layers, each equipped with a ReLu nonlinearity [26] and batch normalization [27]; three max-pooling layers to progressively reduce the spatial support of feature maps; and two fully connected layers; totalling about 2.5 million parameters for training. This architecture has a receptive field of 61x61 pixels, which is the approximate area needed to cover a single cell, and produces as output a three-class probability distribution for the pixel centered in the patch.

In our evaluation, we use the recommended configuration reported by Van Valen et al. [17], which was demonstrated to be accurate on a variety of cell segmentation tasks, including mammalian cell segmentation and nuclei. Their configuration include training an ensemble of five replicate networks to make predictions in images. The final segmentation mask is the average of the outputs produced by each individual network. The ensemble increases processing time, but can also improve segmentation accuracy. The settings of the DeepCell system that we used in our experiments can be reproduced using the following Docker container: https://hub.docker.com/r/jccaicedo/deepcell/.

## U-Net

The U-Net architecture [16] resembles an autoencoder [28] with two main sub-structures: 1) an encoder, which takes an input image and reduces its spatial resolution through multiple convolutional layers to create a representation encoding. 2) A decoder, which takes the representation encoding and increases spatial resolution back to produce a reconstructed image as output. The U-Net introduces two innovations to this architecture: First, the objective function is set to reconstruct a segmentation mask using a classification loss; and second, the convolutional layers of the encoder are connected to the corresponding layers of the same resolution in the decoder using skip connections.

The U-Net evaluated in our work consists of eight convolutional layers and three max pooling layers in the encoder branch, and eight equivalent convolutional layers with upscaling layers in the decoder branch. All convolutional layers are followed by a batch normalization layer, and the skip connections copy the feature maps from the encoder to the decoder. The receptive field of

the U-Net is set to 256 x 256 pixels during training, which can cover a large group of cells at the same time. This architecture has a total of 7.7 million trainable parameters.

We adapted the objective function as a weighted classification loss, giving 10 times more importance to the boundary class. We apply basic data augmentation during training, including random cropping, flips, 90 degree rotations, and illumination variations. Also, we apply additional data augmentation using elastic deformations, as discussed by the authors [16]. The training parameters for this network were tuned using the training and validation sets, and the final model is applied to the test set to report performance. The source code of our U-Net implementation can be found in https://github.com/carpenterlab/unet4nuclei/.

## Evaluation metrics

Measuring the performance of cell segmentation has been generally approached as measuring the difference between two segmentation masks: a reference mask with ground truth objects representing the true segmentation, vs. the predicted/estimated segmentation mask. These metrics include root-mean-square deviation [29], Jaccard index [17], and bivariate similarity index [19], among others. However, these metrics focus on evaluating pixel-wise segmentation accuracy only, and fail to quantify object-level errors explicitly (such as missed or merged objects).

In our evaluation, we adopt an object-based accuracy metric that uses a measure of area coverage to identify correctly segmented nuclei. Intuitively, the metric counts the number of single objects that have been correctly separated from the rest using a minimum area coverage threshold. The metric relies on the computation of intersection-over-union between ground truth objects T and estimated objects E:

$$IoU(T, E) = \frac{T \cap E}{T \cup E}$$

Consider $n$ true objects and $m$ estimated objects in an image. A matrix $C_{n \times m}$ is computed with all $\mathrm{IoU}$ scores between true objects and estimated objects to identify the best pairing. This is a very sparse matrix because only a few pairs share enough common area to score a non-zero $\mathrm{IoU}$ value. To complete the assignment, a threshold greater than 0.5 $\mathrm{IoU}$ is applied to the matrix to identify segmentation matches. In our evaluation, we do not accept overlapping objects, i.e., one pixel belongs only to a single nucleus. Thus, a threshold greater than 0.5 ensures that for each nucleus in the ground truth there is no more than one match in the predictions, and vice versa. At a given threshold, the object-based segmentation F1 score is then computed as:

$$F_1 = \frac{2TP}{2TP + FN + FP}$$

where $t$ is the evaluated $\mathrm{IoU}$ threshold. We compute the average F1 score across multiple thresholds, starting at $t = 0.50$ up to $t = 0.95$ with increments $\Delta t = 0.05$. This score summarizes the quality of segmentations by simultaneously looking at the proportion of correctly identified objects as well as the pixel-wise accuracy of their estimated masks. Our evaluation metric is similar in spirit to other evaluation metrics used in computer vision problems, such as object detection in the PASCAL challenge [30] and instance segmentation in the COCO challenge

[20,31]. One important difference of these metrics and ours is that our problem considers a single object category (the nucleus), and therefore, it is more convenient to adopt the F1 score instead of average precision.

In our evaluation, we also measure other quality metrics, including the number and type of errors to facilitate performance analysis [32]. The following are different types of errors that a segmentation algorithm can make: false negatives (missed objects); merges (under-segmentations), which are identified by several true objects being covered by a single estimated mask; and splits (over-segmentations), which are identified by a single true object being covered by multiple estimated masks. We identify these errors in the matrix $C$ of $\mathrm{IoU}$ scores using a fixed threshold for evaluation, and keep track of them to understand the difficulties of an algorithm to successfully segment an image.

## Baseline Segmentation

We use CellProfiler 3.0 [33] pipelines to create baseline segmentations. CellProfiler was used as a baseline over other tools because it offers great flexibility to configure multi-step image processing pipelines that connect different algorithms for image analysis, and it is widely used in biology labs and high-throughput microscopy facilities. The pipelines are configured and tested by an expert image analyst using images from the training set, and then run in the validation and test set for evaluation. We refer to two CellProfiler pipelines for obtaining baseline segmentations: basic and advanced.

The basic pipeline relies only on the configuration of the module IdentifyPrimaryObjects, which is frequently used to identify nuclei. The module combines thresholding techniques with area and shape rules to separate and filter objects of interest. This is the simplest way of segmenting nuclei images when the user does not have extensive experience with image analysis operations, yet it is complete enough to allow them to configure various critical parameters.

The advanced pipeline incorporates other modules for preprocessing the inputs and postprocessing the outputs of the IdentifyPrimaryObjects module. In our advanced configuration we included illumination correction, median filters and opening operations, to enhance and suppress features in the input images before applying thresholding. These operations are useful to remove noise and prepare images to the same standard for segmentation using the same configuration. The postprocessing steps include measuring objects to apply additional filters and generate the output masks.

A single pipeline was used for segmenting images in the BBBC022 dataset, while Van Valen's set required to split the workflow in two different pipelines. We observed large signal variation in Van Valen's set given that these images come from different experiments and reflect realistic acquisition modes. Two settings were needed for thresholding, the first for normal single mode pixel intensity distributions and another one for bimodal distributions. The latter is applied to cases where subpopulations of nuclei are significantly brighter than the rest, requiring two

thresholds. We used a clustering approach to automatically decide which images needed which pipeline. The pipelines used in our experiments are released together with the data and code.

# Acknowledgements

We thank Beth Cimini and Minh Doan for their efforts and guidance when annotating the image set used for this research. We also thank Mohammad Robahn and Beth Cimini for fruitful discussions and key insights to design experiments and write the manuscript. Funding was provided by the National Institute of General Medical Sciences of the National Institutes of Health under MIRA award number R35 GM122547 (to AEC).

# Author Contributions

JCC contributed experiments, data analysis, software development, and manuscript writing. JR contributed experiments, data preparation, data analysis and software development. AG contributed experimental design, data preparation and software development. TB contributed experiments, software development and manuscript writing. KWK contributed experiments, data preparation and data analysis. CM contributed data preparation and software development. SS contributed experimental design and manuscript writing. AEC contributed experimental design and manuscript writing.

# References


1. Boutros, M., Heigwer, F. & Laufer, C. Microscopy-Based High-Content Screening. *Cell* **163,** 1314–1325 (2015).
2. Mattiazzi Usaj, M. *et al.* High-Content Screening for Quantitative Cell Biology. *Trends Cell Biol.* **26,** 598–611 (2016/8).
3. Caicedo, J. C., Singh, S. & Carpenter, A. E. Applications in image-based profiling of perturbations. *Curr. Opin. Biotechnol.* **39,** 134–142 (2016).
4. Bougen-Zhukov, N., Loh, S. Y., Lee, H. K. & Loo, L.-H. Large-scale image-based screening and profiling of cellular phenotypes. *Cytometry A* (2016). doi:10.1002/cyto.a.22909
5. Williams, E. *et al.* The Image Data Resource: A Bioimage Data Integration and Publication Platform. *Nat. Methods* **14,** 775–781 (2017).
6. Meijering, E. Cell Segmentation: 50 Years Down the Road [Life Sciences]. *IEEE Signal Process. Mag.* **29,** 140–145 (2012).
7. Otsu, N. A Threshold Selection Method from Gray-Level Histograms. *IEEE Trans. Syst. Man Cybern.* **9,** 62–66 (1979).
8. Beucher, S. & Lantuejoul, C. Use of watersheds in contour detection. in *Proceedings of the International Workshop on Image Processing* (1979).
9. Wählby, C., Sintorn, I.-M., Erlandsson, F., Borgefors, G. & Bengtsson, E. Combining intensity, edge and shape information for 2D and 3D segmentation of cell nuclei in tissue sections. *J. Microsc.* **215,** 67–76 (2004).
10. Sommer, C., Straehle, C., Köthe, U. & Hamprecht, F. A. Ilastik: Interactive learning and segmentation toolkit. in *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro* 230–233 (ieeexplore.ieee.org, 2011).
11. Eliceiri, K. W. *et al.* Biological imaging software tools. *Nat. Methods* **9,** 697–710 (2012).

12. Carpenter, A. E. *et al.* CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome Biol.* **7,** R100 (2006).
13. Schindelin, J. *et al.* Fiji: an open-source platform for biological-image analysis. *Nat. Methods* **9,** 676–682 (2012).
14. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521,** 436–444 (2015).
15. Mnih, V. *et al.* Human-level control through deep reinforcement learning. *Nature* **518,** 529–533 (2015).
16. Ronneberger, O., Fischer, P. & Brox, T. U-net: Convolutional networks for biomedical image segmentation. *Med. Image Comput. Comput. Assist. Interv.* (2015).
17. Van Valen, D. A. *et al.* Deep Learning Automates the Quantitative Analysis of Individual Cells in Live-Cell Imaging Experiments. *PLoS Comput. Biol.* **12,** e1005177 (2016).
18. Gustafsdottir, S. M. *et al.* Multiplex cytological profiling assay to measure diverse cellular states. *PLoS One* **8,** e80999 (2013).
19. Dima, A. A. *et al.* Comparison of segmentation algorithms for fluorescence microscopy images of cells. *Cytometry A* **79,** 545–559 (2011).
20. Hariharan, B., Arbeláez, P., Girshick, R. & Malik, J. Simultaneous Detection and Segmentation. *arXiv [cs.CV]* (2014).
21. Shelhamer, E., Long, J. & Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *arXiv [cs.CV]* (2016).
22. Fiorio, C. & Gustedt, J. Two linear time Union-Find strategies for image processing. *Theor. Comput. Sci.* **154,** 165–181 (1996).
23. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. *arXiv [cs.CV]* (2015).
24. Chen, F., Tillberg, P. W. & Boyden, E. S. Optical imaging. Expansion microscopy. *Science* **347,** 543–548 (2015).
25. Bray, M.-A. *et al.* Cell Painting, a high-content image-based assay for morphological

profiling using multiplexed fluorescent dyes. *bioRxiv* 049817 (2016). doi:10.1101/049817

26. Nair, V. & Hinton, G. E. Rectified linear units improve restricted boltzmann machines. in *Proceedings of the 27th international conference on machine learning (ICML-10)* 807–814 (cs.toronto.edu, 2010).
27. Ioffe, S. & Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. in *International Conference on Machine Learning* 448–456 (jmlr.org, 2015).
28. Hinton, G. E. & Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science* **313,** 504–507 (2006).
29. Gudla, P. R. *et al.* A high-throughput system for segmenting nuclei using multiscale techniques. *Cytometry A* **73,** 451–466 (2008).
30. Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. & Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* **88,** 303–338 (2010).
31. Lin, T.-Y. *et al.* Microsoft COCO: Common Objects in Context. in *Computer Vision – ECCV 2014* 740–755 (Springer International Publishing, 2014).
32. Hoiem, D., Chodpathumwan, Y. & Dai, Q. Diagnosing Error in Object Detectors. in *Computer Vision – ECCV 2012* 340–353 (Springer Berlin Heidelberg, 2012).
33. McQuin, C. *et al.* CellProfiler 3.0: next generation image processing for biology. *PLoS Comput. Biol.* (2018).