Technische Universität München
Department of Mathematics
Mathematical Modeling of Biological Systems

Broad Institute of MIT and Harvard
Imaging Platform

Helmholtz Zentrum München
Institute of Computational Biology

Fluorescence Microscopy Image Segmentation with Deep Learning

Master's Thesis

Jonathan Roth
jonathan.roth@tum.de

Supervisors:	Prof. Dr. Dr. Fabian J. Theis
	Dr. Anne E. Carpenter
	Dr. Juan C. Caicedo
Submission Date:	June 30, 2017

I hereby declare that I have written this Master's Thesis on my own.
No other than the referenced sources were used.

Used code and data are publicly available.

Garching, June 30, 2017

Jonathan Roth

Contents

Abstract

In this Master's Thesis, the segmentation of fluorescence microscopy images with deep neural networks is discussed. In particular, the DNA stain of these multi-channel images is used to segment cell nuclei.

Segmenting nuclei is challenging mainly because of clumped nuclei which are the major reason for why current approaches do not yield satisfying segmentations.

Here we show that using manually annotated data to train deep convolutional neural networks for segmenting nuclei, instances of clumped nuclei can be separated, and the number of undetected nuclei can be reduced from 15% to 6%.

Segmentations are crucial in the analysis of experiments in quantitative biology. Thus, our suggested method, which yields higher quality segmentations, can be used to make more precise measurements from biological images.

Acknowledgements

I want to thank Fabian Theis, Anne Carpenter, and Juan Caicedo for the outstanding supervision of this Master's Thesis, their helpful advice, and the fruitful discussions. Working at the Imaging Platform was always a great experience. I'm especially thankful for Beth Cimini, Kyle Karhohs, and Minh Doan for their support with the annotations, Allen Goodman and Claire McQuin for their assistance with programming, and Mark Bray for providing the CellProfiler segmentation pipeline.

Thanks to my family and friends who made not only this project but also my whole studies a truly unforgettable time.

This work was generously supported by a fellowship within the program "FIT weltweit" of the German Academic Exchange Service (DAAD).

1 Introduction

Morphological profiling as a method of systems biology aims at quantifying changes in the structure of cells as a result of biological perturbations [46, 19]. The basis of this technique is fluorescence microscopy which allows to capture high-resolution images of a large number of cells. These images are used to extract properties of single cells which are summarized in morphological profiles. A statistical analysis can then be performed on these profiles.

To obtain single-cell profiles, microscopy images must be segmented such that morphological features can be derived from single cells. Segmenting cell nuclei from the DNA staining of fluorescence microscopy images is the first step in the process. CellProfiler is a commonly used state-of-the-art software tool for the segmentation of fluorescence microscopy images, and the extraction of single-cell profiles [11].

Due to the high density of cells in fluorescence microscopy images, nuclei often appear clumped. Thus, the main challenge for segmentation algorithms is to separate different instances of nuclei. Traditional segmentation methods such as the seeded watershed algorithm, which is implemented in CellProfiler, are slow and sometimes make errors. In addition, the user needs to be experienced in order to use these algorithms as parameter tuning is required [70, 8]. The goal of this thesis is to overcome these dominant problems in the segmentation of nuclei in microscopy images.

We use a deep convolutional neural network for segmenting nuclei of fluorescence microscopy images. We train the neural network on hand-annotated images and evaluate its performance with object-based metrics.

With the support of experienced biologists, we created hand annotations for 200 images of a compound-profiling experiment. These ground truth annotations are crucial for training the segmentation network, and are the basis of our method which quantifies the quality of the obtained segmentations.

The neural network is trained to predict the probability of belonging to the

outline of a nucleus for each pixel of the input image. A post-processing step transforms these outlines into a segmentation. We obtain the best results when using data augmentation techniques for training the neural network.
To evaluate the performance of our neural network, we refrain from using pixel-based metrics. As computing object-based metrics such as the share of undetected nuclei is more relevant to biologists, we adhere to metrics common in object detection [18].

Structure of this Thesis. The following sections of this thesis are structured as follows: In section 2, we give a brief introduction to the basic concepts used in this project; cytometry, morphological profiling, image segmentation, and convolutional neural networks.

Section 3 covers image segmentation with deep learning in more detail, introduces BBBC022, the data set we're working on, and presents the object-based metric which we use to evaluate the performance of segmentation algorithms.

Our experiments and results are explained in section 4. We evaluate the segmentation performance of CellProfiler before quantifying the quality of segmentations obtained with different deep learning models.
We show that neural networks that were trained on automatically generated data do not yield satisfying segmentations. However, models trained on hand-annotated images surpass the performance of CellProfiler. Finally, we show that we can significantly improve our segmentations with data augmentation.

2 Background

In this section, the main concepts and methods used in this Master's Thesis are explained. Concerning biology, basic concepts in cytometry such as high-content analysis and high-throughput methods are explained as well as morphological profiling based on fluorescence microscopy. On the mathematical side, image segmentation and deep convolutional neural networks (CNNs) are introduced.

2.1 Cytometry

2.1.1 Motivation

Prominent genetic diseases such as cancer or Alzheimer's disease are caused of influenced by the genes encoded by our DNA. A crucial process within each cell is transcribing the DNA into RNA which is translated into proteins. Thus, the existence of a protein within a cell can be linked to a genetic blueprint of the cell. In addition, diseases and natural fluctuations in the cell state highly influence the cell's morphological properties. For example, the size of its nucleus will increase as DNA is replicated during the cell cycle.

Using fluorescent dyes or fluorescence-conjugated antibodies, virtually any cellular component such as the cytoplasm or the cell's nucleus can be fluorescently labeled. Microscopy images of cells thus allows researchers to understand how cells work, to study cellular diseases, and to discover drugs against them.

In fluorescence microscopy, cells are treated with dyes that highlight areas of interest. These dyes are then excited by a light source, typically a laser, and the fluorescence response is captured as an image. Fluorescence microscopy is a popular tool used in cytometry as it is a high-throughput and high-content method. These concepts are explained in the following section 2.1.2.

2.1.2 High-Content and High-Throughput Analysis

Fluorescence microscopy can combine the advantages of high-throughput and high-content analysis. In the following, these two concepts are explained.

High-Content Analysis. In high-content analysis, rich information, such as high-resolution images, is obtained from each sample. Most recent advancements of high-content analysis include super-resolution microscopy and imaging mass spectrometry, which increase the amount of spatial information or number of labels measured in a given sample, respectively [3, 33, 49].

High-Throughput Analysis. In high throughput analysis, a larger number of samples are analyzed quickly, though not necessarily at a single cell resolution [21]. Examples for high-throughput methods are traditional flow cytometry or fluorescence intensity measurement using plate readers that yield a fluorescence response for each well in a plate. Thereby, thousands of cells are contained in each well.

Combining High-Content and High-Throughput Analysis. Some techniques combine the advantages of high-content and high-throughput analysis. Capturing biological images is now highly automated, which enables methods that deliver rich information for a large number of cells at the same time. Figure 1 shows how robots are used to automate the handling of multiwell plates.
Imaging flow cytometry and fluorescence microscopy are examples for methods that allow high-throughput and high-content analysis at the same time.

- **Imaging Flow Cytometry**
  An imaging flow cytometer captures high resolution images of each single cell passing through a flow channel and thus using dyes, one can obtain spatial information about substances of interest [30, 4, 17]. Sample images are shown in Figure 2.

- **Fluorescence Microscopy** In fluorescence microscopy, high resolution microscopes capture the fluorescence response of multiple cells

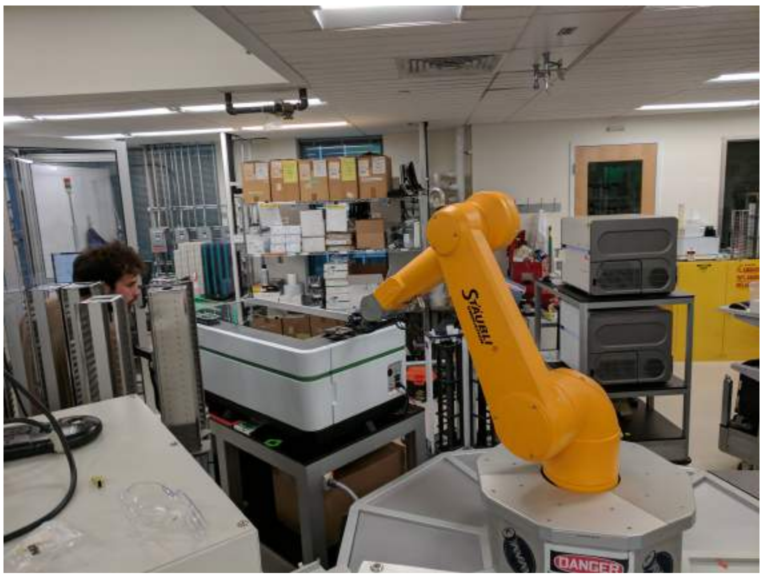


Figure 1: A robot is used to automate fluorescence microscopy in a high-throughput wet lab. This enables monitoring fluorescence responses over a long period of time in regular time intervals, and high-throughput experiments as shown here. The robot feeds a 384-well plate to a fluorescence microscope.

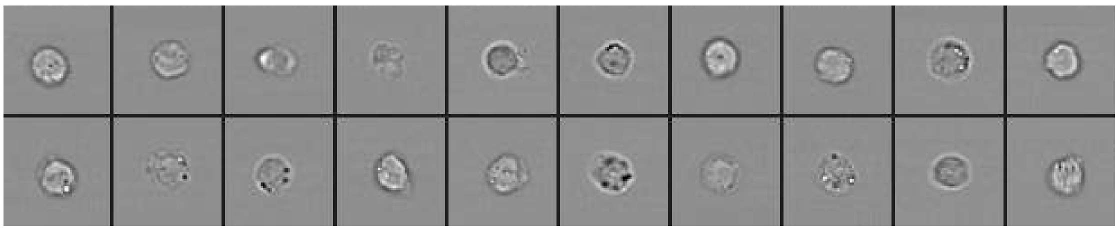


Figure 2: Images of cells captured with an imaging flow cytometer. The images were obtained with an ImageStreamX device from MilliporeSigma by collaborators of the Imaging Platform.

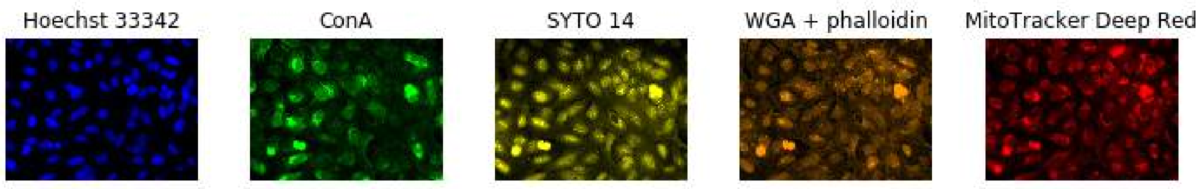


Figure 3: In fluorescence microscopy, usually multiple different stains are captured. These images were generated with images in BBBC022. The colors correspond to the wavelengths of the light the dyes emit.

in one well at once. Images typically depict hundreds of cells. This method's spatial resolution is very high and thus morphological details are visible for each cell. This is why this method is used in morphological profiling which is covered in section 2.2. Compared to imaging flow cytometers, fluorescence microscopes provide higher resolution images, are more sensitive and allow higher magnifications.

Cell profiling experiments based on fluorescence microscopy and the Cell Painting protocol which is a standard staining strategy for fluorescence microscopy experiments are explained in more detail in Section 2.2.

2.1.3 Applications

Analysis of fluorescence microscopy images, which this thesis aims to improve, is a critical step in many biological applications [9]. In this chapter, two popular examples for applications are presented.

Drug Discovery. To find new drugs against a given disease, sick cells are treated with a chemical which could be a potential drug against the disease. This experiment is performed with thousands or even millions of different compounds in parallel using multi-well plates. The cells' reactions to those compounds are observed and compounds that do not seem beneficial to the cure of the disease are ruled out.

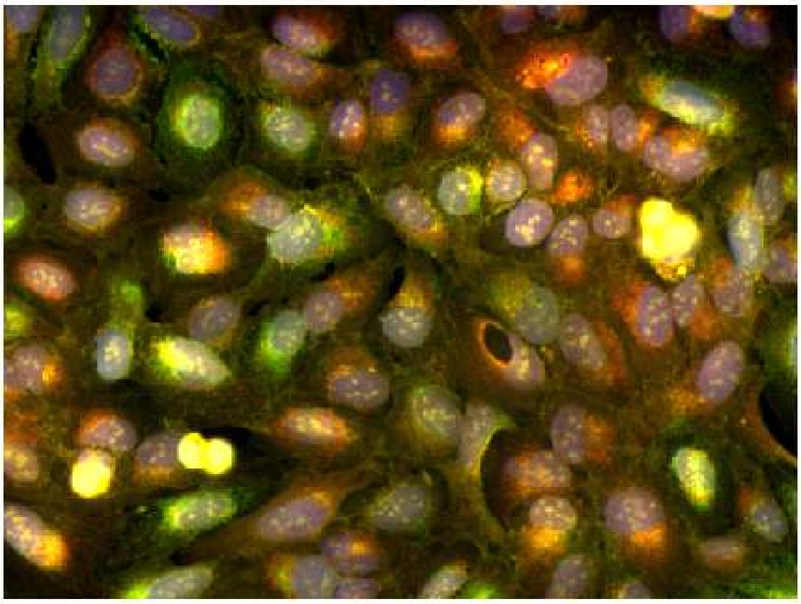


Figure 4: Different fluorescence response channels can be combined into highly multiplexed images.

Methods from cytometry such as cell profiling, which is explained in Section 2.2 are used to measure the compounds' influence on the cells [55].

Drug Repurposing. Testing drugs and getting them approved is a very long and expensive process for pharmaceutical companies. Thus, there is strong interest in reusing drugs that were already approved for one disease to treat others. A well-known example is sildenafil, which is a drug used against hypertension. It was discovered that sildenafil is also an efficient treatment against erectile dysfunction, and is now marketed as Viagra by Pfizer [62].

Generally speaking, drug repurposing is used to find new targets for well-known drugs. In drug repurposing experiments based on cytometry, the known drug's effect on cells is measured. A drug can be tested on multiple cell lines that model different diseases. It is then to be evaluated which of the diseases are likely to be cured by the compound [2].

The Drug Repurposing Hub is a project promoting a systematic search for new applications for existing drugs. It provides a large library of over 5 000 compounds and information about their targets, usage, and mechanism of action [7, 14].

2.2 Morphological Profiling with Fluorescence Microscopy

In this chapter, we give a more detailed introduction to morphological cell profiling with fluorescence microscopy images, which is the key application relevant to this project.

The goal of each profiling experiment is to obtain a set of features for each cell that describes its phenotype. This set of features is called the cell's profile. Cell profiles can be analyzed and compared to each other. In a drug screening experiment for example, profiles of cells that were treated can be compared against profiles of cells in the control set to quantify important cellular changes. For example, cell profiles can reveal a cell's disease state or can be used for classification in cell states such as phases of the cell cycle or hematopoietic differentiation [39].

Profiles can be of very different kinds. Examples include expression profiles that quantify the transcription of genes and morphological profiles that quantify the shape of the cell and its compartments. Fluorescence microscopy is an important tool in morphological profiling and captures the images used to obtain morphological cell profiles.

2.2.1 Experiment Setup

Assay Preparation. The cells of interest are usually grown in standard culture flasks and then placed into plates, which consist of multiple wells each holding cells. Usually, plates contain 96 or 384 wells. Figure 5 shows a collection of plates that are part of a screening. In many experiments, cells in different wells are treated with different compounds. The high degree of

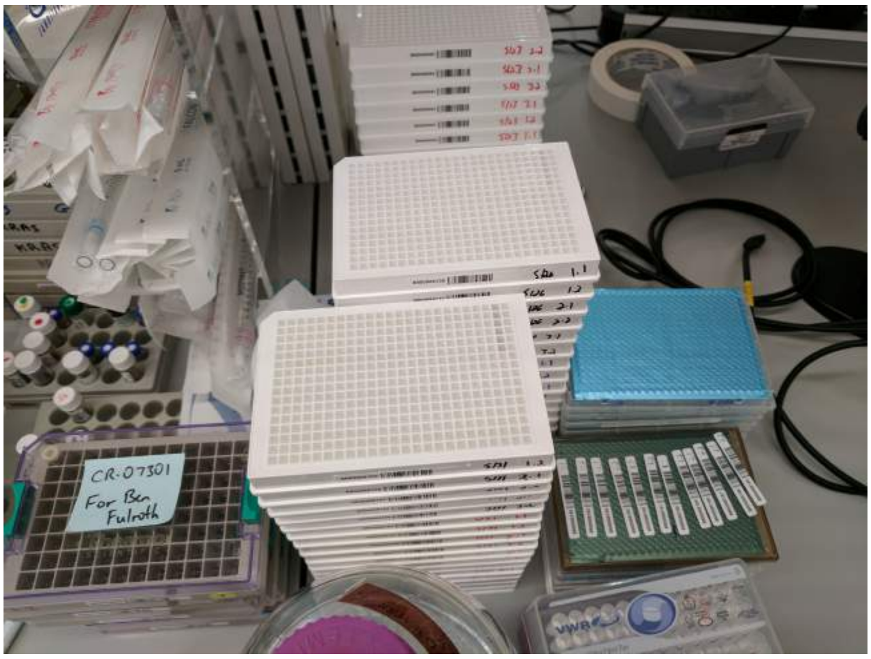


Figure 5: 384-well plates stacked on top of each other in a wet lab.

automation with liquid handling robots supports researchers in putting the cells in the wells and treating them with compounds.

Staining. Different dyes are used to stain different compartments of the cell. Each fluorescent dye is specified by the substance it binds to, an energy at which it can be excited, and an emission spectrum. An example is shown in Figure 6. By adding the dyes to the cells and exciting them at the required wavelength one can highlight the substances to which the dye binds.
Multiple stains can be used in the same experiment. Usually, for each dye a different channel is created for the resulting image. To ensure that the response of a dye is visible in one channel only, the dyes are excited sequentially and a different wavelength filter for each channel is used. When using multiple stains whose excitement or emission spectra overlap, capturing the fluorescence response from only one dye might be difficult. The unwanted effect of capturing the response of multiple dyes in one channel is called bleed through.
A popular combination of stains is defined in the Cell Painting protocol,

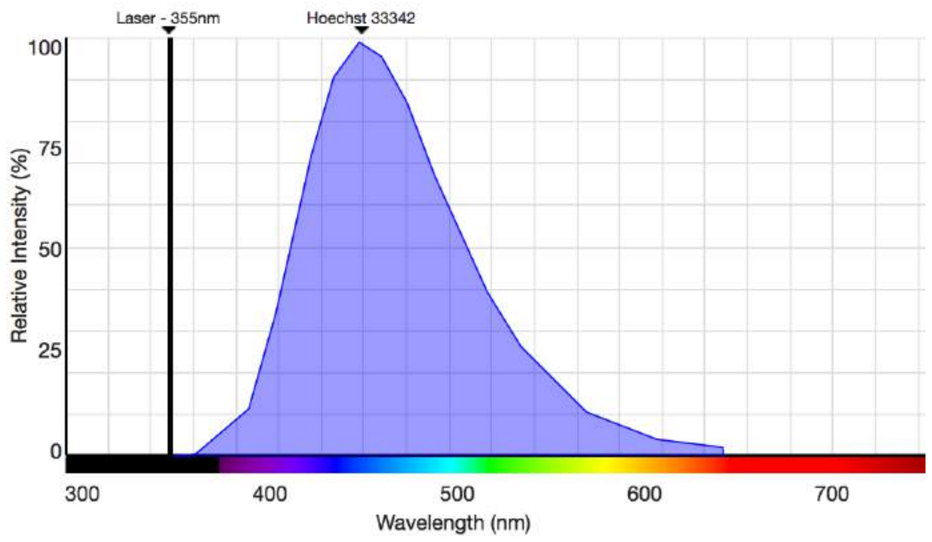


Figure 6: Emission and excitation spectrum of the Hoechst 33342 stain. The dye can be excited at 355nm and emits visible light with a wavelength between 400nm and 600nm. The figure is taken from the product's description website of Thermo Fisher Scientific, a company selling biological dyes [68].

which is covered in Section 2.2.2.

Image Capturing. Fluorescence microscopes are used to capture the images. The microscopes are equipped with light sources to excite the dyes, optics that magnify the sample, and high resolution cameras which capture the images. Depending on the experiment, the plates may be fed to the microscope by a robot as seen in figures 1 and 7. The fluorescence microscopy data set that we are working on was captured with an ImageXpress Micro microscope from Molecular Devices [50].

2.2.2 Cell Painting

For morphological profiling, it is crucial to highlight as many of the cell's structural elements as possible. Cell Painting uses six different dyes to highlight eight cellular components [6]. As two of the dyes have the same excitation and emission peak, they are imaged in a common channel. The

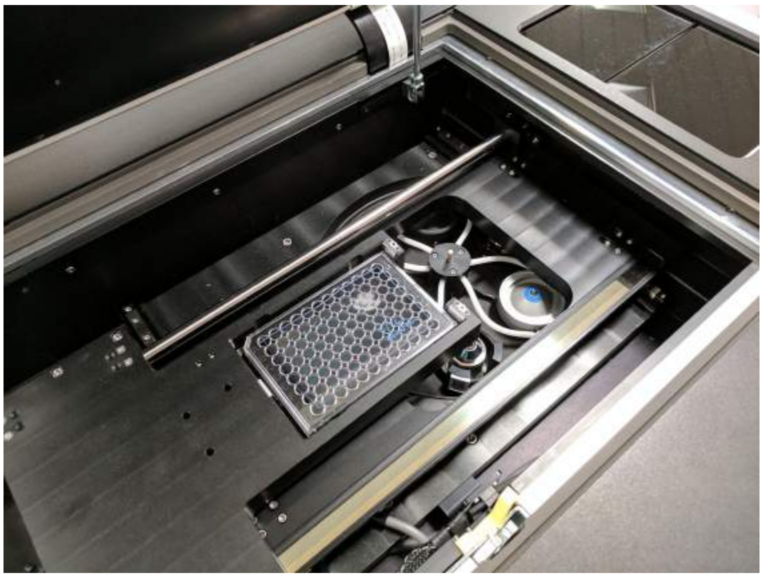


Figure 7: A 96-well plate is placed in top of the objective lens of a Opera Phenix fluorescence microscope from PerkinElmer [57].

dyes used are summarized in Table 1. Cell Painting thereby is unbiased in the sense that the selection of dyes is independent from the experiment setup.

Cell Painting was developed at the Imaging Platform and is optimally designed for the usage in morphological profiling [60]. The data set we are working with is a morphological profiling experiment and uses the Cell Painting protocol.

2.2.3 Analysis Pipeline

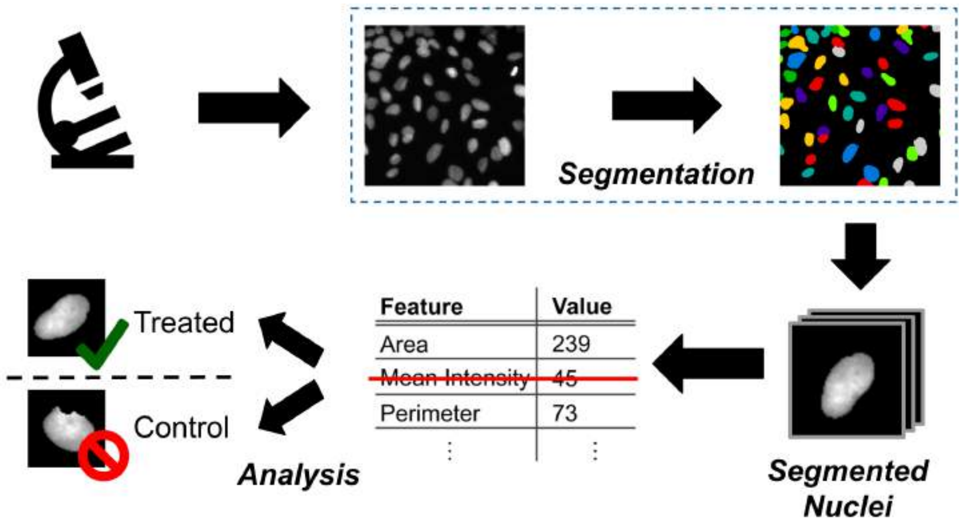
In order to calculate morphological profiles from microscopy images, several steps are necessary which are summarized by figure 8:

- **Cell Segmentation**
  As a microscopy image may depict multiple cells, these must be detected and segmented, such that an analysis can yield single cell measurements. This step is the main focus of this thesis.

- **Feature Measurement**
  Given the images depicting a single cell, these must be analyzed for

Dye name	Excitation	Emission	Component
Hoechst 33342	387	417-477	Nucleus
Concanavalin A (Alexa Fluor 488)	472	503-538	Endoplasmic reticulum
SYTO^{TM}14	531	573-613	Nucleoli, cytoplasmic RNA
Phalloidin (Alexa Fluor 568)	562	622-662	F-actin cytoskeleton
WGA (Alexa Fluor 555)	562	622-662	Plasma membrane
MitoTracker Deep Red	628	672-712	Mitochondria

Table 1: Dyes, their excitation wavelength, their emission spectrum, and the components highlighted in a Cell Painting assay. For conjugate dyes, the fluorescent marker is written in brackets. Wavelengths of excitation and emission are given in nm.



Feature	Value
Area	239
~~Mean Intensity~~	~~45~~
Perimeter	73
⋮	⋮

Figure 8: An overview of the analysis pipeline of morphological profiling.

features to best describe them. An example feature is the size of the cell's nucleus. In a typical setting, thousands of features are calculated per cell, yielding a morphological profiles for each.

- **Feature Selection**
  As features might be redundant, negligible for the downstream analysis or just too numerous, feature selection might be required. Often, feature selection is performed to avoid the curse of dimensionality.[1]

- **Profile Analysis**
  As the final step in the analysis pipeline, data analysis can detect changes in morphology by comparing profiles. A simple analysis task could be the classification of cells into treated and control cells based on their cell profile.

The software tool CellProfiler covers the first two steps; object recognition and feature measurement [11]. In the Imaging Platform, Cytominer is used for pooling of single cell features to per-well profiles, feature selection and feature extraction [66]. For analysis, CellProfiler Analyst is used [37, 15]. It offers data exploration, data visualization and supervised classification features.
In our research, we focus on the object detection task. Many software packages have image segmentation functions, including CellProfiler. This thesis covers a method to improve these segmentations.

2.3 Image Segmentation

As discussed in section 2.2.3, cell segmentation is an important step in the pipeline of morphological profiling experiments [44]. Whereas image segmentation in general describes only the partitioning of an image into subsets, one can further distinguish between semantic segmentation and instance segmentation.

[1]The curse of dimensionality describes the phenomenon that some data analysis techniques developed for low-dimensional data fail for high-dimensional data sets. A popular example is the k-nearest neighbors algorithm.

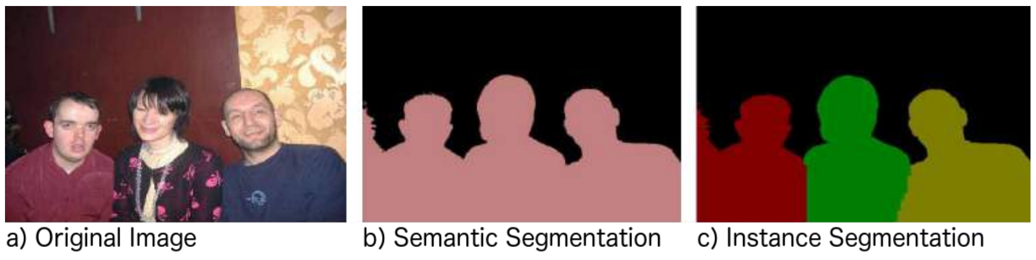


Figure 9: The difference of semantic and instance segmentation is crucial when objects of the same class overlap or adjoin. In this case, instance segmentation is able to tell them apart, semantic segmentation is not. Figure adapted from [42].

2.3.1 Semantic Segmentation

Semantic segmentation describes the unique assignment of pixels to classes. Thus, objects in the image are detected as connected regions of pixels that were assigned to the same class by the segmentation algorithm. Semantic segmentation can thus be seen as a pixel classification task.

The object representation capabilities of semantic segmentation is limited. When it is required to separate overlapping objects, represent objects that are not connected, or segment occluded objects, other segmentation strategies need to be used.

2.3.2 Instance Segmentation

On the other hand, instance segmentation assigns pixels to objects [28]. Two instances of the same class can thus be represented by different objects. This enables the separation of objects of the same class even though they are touching in the image. Figure 9 visualizes this difference to semantic segmentation.

Instance segmentation is especially hard when a lot of instances of the same class are depicted. In popular instance segmentation challenges such as Microsoft COCO, these cases are ignored for the result evaluation [43].

2.3.3 Image Segmentation in CellProfiler

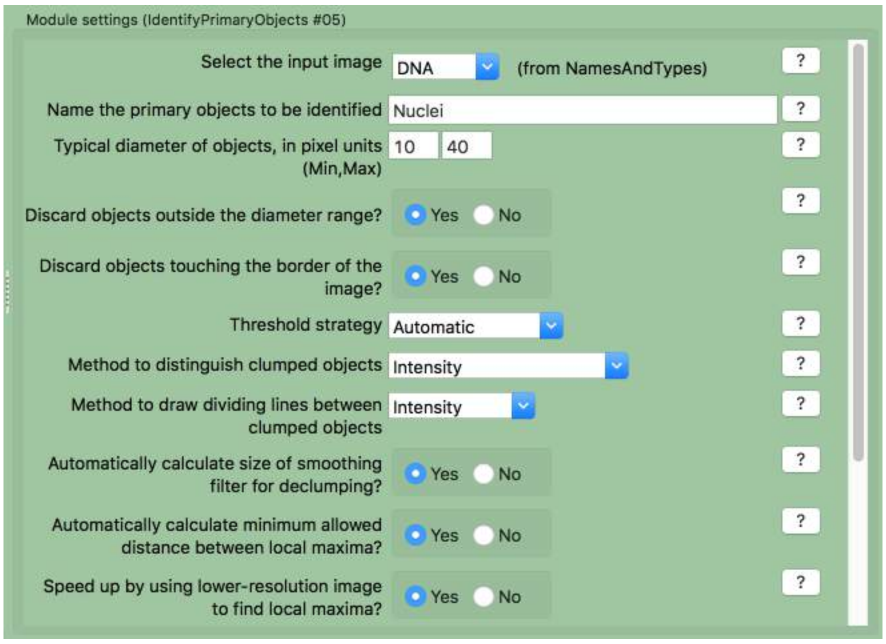
As mentioned earlier, the goal of this project is to improve the segmentation of nuclei in the DNA channel. We compare the performance of our segmentation algorithms against the current segmentations obtainable with CellProfiler which is the standard approach. CellProfiler does a combination of semantic and instance segmentation for nuclei. It classifies pixels into foreground and background pixels and then assigns the foreground pixels to an object instance. CellProfiler's current implementation for segmentation is implemented in the module `IdentifyPrimaryObjects` and consists of three steps:

1. Using thresholding, foreground regions are identified. Thus, CellProfiler distinguishes between pixels that belong to nuclei and background pixels.

2. Clumped nuclei are identified and segmented. For the segmentation, a seeded watershed algorithm is used [47]. The user can choose to either apply it on the distance transform of the thresholded image or on pixel intensity values. Other traditional computer vision methods such as image smoothing are used to improve the segmentation results.

3. Nuclei whose features do not fall into user-specified ranges are discarded. For example, CellProfiler offers to discard nuclei that are close to the image's boundary, or nuclei below a certain size.

The segmentation quality of this approach is highly affected by the choice of parameters for the used algorithms. Thus, the user's input is required for choosing the parameters. The interface for the parameter selection of the module is shown in figure 10.
For parameter optimization, users usually start off with a given set of parameters and adapt these empirically based on the obtained segmentation. The user can then optimize the parameters in an iterative process. As ground truth annotations are usually lacking, the user has no quantitative segmentation quality measure and is relying on visual inspection only.



Module settings (IdentifyPrimaryObjects #05)

Select the input image	DNA (from NamesAndTypes)
Name the primary objects to be identified	Nuclei
Typical diameter of objects, in pixel units (Min,Max)	10 40
Discard objects outside the diameter range?	[✓ Yes] No
Discard objects touching the border of the image?	[✓ Yes] No
Threshold strategy	Automatic
Method to distinguish clumped objects	Intensity
Method to draw dividing lines between clumped objects	Intensity
Automatically calculate size of smoothing filter for declumping?	[✓ Yes] No
Automatically calculate minimum allowed distance between local maxima?	[✓ Yes] No
Speed up by using lower-resolution image to find local maxima?	[✓ Yes] No

Figure 10: The module `IdentifyPrimaryObjects` in CellProfiler offers a lot of flexibility.

This segmentation process has several downsides that we are trying to tackle:

- The traditional computer vision algorithms used within the segmentation pipeline lack computational efficiency when applied to large image sets [51]. Parallelization is thus required for all but the smallest experiments.
- As users usually choose the parameters such that a handful of images is segmented in a satisfying way, no segmentation performance guarantee can be given for images that the user has not used for parameter optimization.
- As the choice of parameters is very involved, it requires a lot of expertise with the software and can be quite time consuming.

We want to address these issues by segmenting nuclei with convolutional neural networks which are explained in section 2.4. Artificial neural networks can be easily parallelized and are thus fast in execution, they generalize well when trained without overfitting the training data and they do not require manual parameter tuning.

2.4 Convolutional Neural Networks

In this section, we give a short introduction to convolutional neural networks. We start with a brief explanation of Feed Forward Neural Networks before covering the single-layer perceptron and how it relates to logistic regression. We then generalize to the multi-layer perceptron and finally introduce convolutional neural networks.

2.4.1 Feed Forward Neural Networks

The term *artificial neural network* describes systems of parametrized mathematical operations that are applied on some input data in order to obtain output values. These mathematical operations can be seen as synapses between intermediate results which in turn are called neurons due to the similarity to the function of the brain. It is to be noted that biological neural networks indeed inspired the design of artifical neural networks, but the function is still different. One central difference between both models is that biological nerve cells have spikes in their action potential at discrete time points whereas artificial neural networks use continuous values to describe the excitement of a neuron.
The models presented here are all feed-forward neural networks which is an artificial neural network whose operations do not form loops. Recurrent neural networks are artificial neural networks that contain loops and are not covered here.

2.4.2 Single-Layer Perceptron

Idea. A basic task in supervised machine learning is the binary classification of data. Given two classes A and B and data points $x \in R^d$ that belong to exactly one of these classes, the goal is to learn a model that classifies new data points correctly [20].

The simplest form of classification is finding a hyperplane $w^T x + b = 0$ in the d-dimensional space that separates the data. Thus, one wants to find weights W and a bias b such that $w^T x + b > 0$ for $x \in A$ and $w^T x + b < 0$ for $x \in B$. Thus, the model summarizes as follows:

$$f(x) = \operatorname{sgn}\{w^T x + b\}$$

Without loss of generality, if $f(x) = 1$, x is classified as belonging to class A and belonging to class B if $f(x) = -1$. This model is called a single-layer perceptron [67].

Distinction from Logistic Regression. In contrast to logistic regression, a single-layer perceptron does a hard assignment of the data points to one of the two classes. Logistic regression performs a soft assignment in the sense that a probability $p \in (0, 1)$ for the data point belonging to class A is predicted.

$$P(Y = c|x, \Theta) = \operatorname{Ber}(c|\sigma(x, \Theta))$$

Ber hereby denotes the Bernoulli distribution, Θ summarizes the model parameters w and b. The logistic function σ is defined as follows:

$$\sigma(x, \Theta) :- \frac{1}{1 + e^{-(w^T x + b)}}$$

However, the model of logistic regression is similar to a single-layer perceptron. Assuming a maximum a posteriori estimator, a linear decision boundary $w^T x + b = 0$ can be obtained at $p = \frac{1}{2}$. As the logistic function is applied on the distance to this hyperplane, the model's predicted probability of the data point belonging to class A becomes more degenerate with increasing distance to the hyperplane [36].

Fitting. The parameters of a single-layer perceptron can be optimized with Rosenblatt's perceptron learning algorithm. It tries to minimize the sum D of the distances d_i between the decision boundary and the misclassified data points $x_i : i \in M := \{i|\hat{y}_i \neq y_i\}$. As the distance between a data point and a hyperplane can be easily calculated as $d_i = w^T x + b$, the loss function of the training algorithm can be written as follows:

$$D = -\sum_{i \in M} y_i \cdot (w^T x_i + b)$$

Thereby, y_i denotes the true label for x, thus $y_i = 1$ if $x \in A$ and $y_i = -1$ if $x \in B$. For this loss function, the gradient can be easily calculated and gradient descent can be performed:

$$\frac{\partial D}{\partial w} = -\sum_{i \in M} y_i \cdot x_i$$
$$\frac{\partial D}{\partial b} = -\sum_{i \in M} y_i$$

In practice, online learning is performed by updating the model's weights after each classification [54]:

$$\begin{pmatrix} w \\ b \end{pmatrix} \leftarrow \begin{pmatrix} w \\ b \end{pmatrix} + \rho * \begin{pmatrix} y_i \cdot x_i \\ y_i \end{pmatrix}$$

The learning rate ρ can be used to control the speed of convergence. It can be shown that the model is guaranteed to converge if the data is linearly separable as depticted in figure 11 [67].

As this condition is not necessarily fulfilled, a non-linear decision boundary can be obtained by expanding the model to multiple layers and introducing non-linear activation functions in intermediate layers.

2.4.3 Multi-Layer Perceptron

Idea. As already discussed in section 2.4.2, introducing non-linearities in models sometimes is required. A multi-layer perceptron (MLP) is a parallel and sequential concatenation of single-layer perceptrons. Two consecutive layers of an MLP are connected by a series of parallel single-layer perceptrons. The layers are stacked sequentially and instead of applying the sign

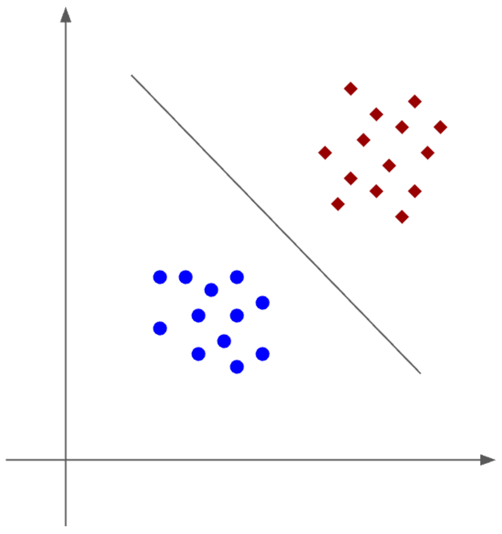


Figure 11: The two clusters of data points are linearly separable.

function to the output of each layer as in a single-layer perceptron, non-linear activation functions are used. The transformed outputs of a given layer serve as the input data for the subsequent layer. A simple MLP is visualized in figure 12.

Activation Function. Layers which are neither the input nor the output layer are called hidden and denoted with sequential indices starting at h_1. The output layer is denoted with $\hat{y}$, the input layer denoted with x. An MLP with four layers as depicted in figure 12 can then be written as follows:

$$\begin{aligned}\hat{y} &= g_3(W_3h_2 + b_3)\\ h_2 &= g_2(W_2h_1 + b_2)\\ h_1 &= g_1(W_1x + b_1)\end{aligned}$$

Hereby, the activation functions are denoted with g_1, g_2 and g_3. In practice, the same activation function is used for all hidden layers. A very popular example is the ReLU function [52]:

$$\mathrm{ReLU}(z) = \max\{0, z\}$$

Depending on the task, a special activation function is used for the output layer. In binary classification, the output $\hat{y}$ is supposed to model the probability of the data point x belonging to a class c:

$$P(Y = c|x, \Theta) = \mathrm{Ber}(c|\hat{y})$$

Hereby, $Y := \{0, 1\}$ denotes the set of all possible classes. To ensure a valid probability distribution, the logistic function is used as an activation function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Whereas in binary classification only one probability value is predicted, a different activation function must be used in multi-class classification. The softmax activation function ensures the validity of the probability distribution:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_i e^{z_i}}$$

For regression problems, a different formulation is used:

$$P(Y = y|x, \Theta) = \mathcal{N}(y|\hat{y}, \sigma^2)$$

Whereas the outputs of classification models are restricted to the interval $[0, 1]$, the regression model can take arbitrary values. Thus, the identity function is commonly used as the activation function for the last layer.

All activation functions need to be differentiable almost everywhere as this property is required by the back propagation algorithm used to fit the model parameters.

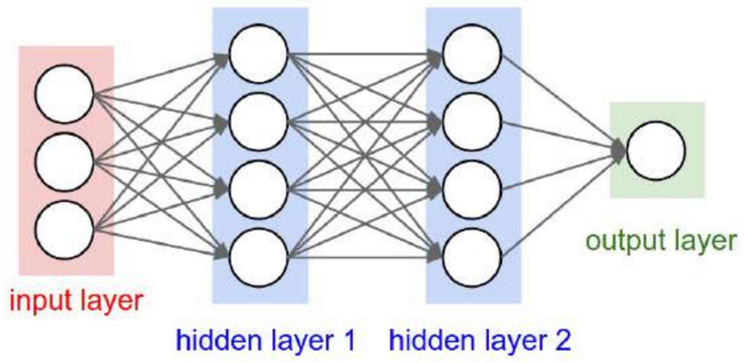


Figure 12: A simple multi-layer perceptron with four layers. There are nine single-layer perceptrons in total. Four perceptrons are used between the input and the first hidden layer, four between the two hidden layers, and one perceptron calculates the output value. Reprinted from [41]

Loss Function. A loss function is used to evaluate the network's error. Given a tuple of a data point in the training set with its true output (x, y), the loss functions is used to penalize the deviation of the predicated output $\hat{y}$ from y.

In general, maximum likelihood estimation is performed which is equivalent of minimizing the negative log-likelihood [24]:

$$l(\Theta) := -\log P(\texttt{data}|\Theta)$$

For classification problems the negative log-likelihood simplifies to a categorical cross-entropy loss. The formulation for problems with n different classes is as follows:

$$\mathcal{L}(y, \hat{y}) = -\sum_{i \in [n]} y_i \log(\hat{y}_i)$$

For a two class problem for which only one probability y is predicted, this can be written as

$$\mathcal{L}(y, \hat{y}) = -y \cdot \log(\hat{y}) - (1 - y) \cdot \log(1 - \hat{y})$$

For regression problems, minimizing the mean squared error is equivalent to maximizing the log likelihood:

$$\mathcal{L}(y, \hat{y}) = \|\hat{y} - y\|_2$$

Back Propagation. The advanced structure of an MLP does make fitting it to the training data more difficult compared to fitting a single-layer perceptron. The basic concept, stochastic gradient descent, is still used. However, the computation of the gradient is different. The gradient is computed in two steps. In a forward pass, values of all neurons are computed for a given data point. In addition, an error value is computed. In a second step, the partial derivatives of the loss function with respect to the models parameters are computed. This requires all used activation functions and the loss function to be differentiable. This second step is called the backward pass as the gradient is calculated by exploiting the chain rule. Gradients of layers closer to the output layer have to be computed first before the computation of the gradients of parameters in preceding layers. Going back to the example in figure 12, the chain rule for differentiation of the loss with respect to a parameter in the weight matrix of the first layer W_{1ij} is applied as follows:

$$\frac{\partial \mathcal{L}}{\partial W_{1ij}} = \frac{\partial \mathcal{L}}{\partial y} \cdot \frac{\partial y}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial W_{1ij}}$$

In modern deep learning frameworks, the analytical derivatives of all used activation functions and loss functions are implemented. Thus, gradient computation is efficient.

2.4.4 Convolutions in Neural Networks

Traditional computer vision models heavily rely on image filters. A very popular example are edge detectors such as the Canny filter [10]. Another example is a Gaussian filter that is frequently applied as a preprocessing step in image analysis.

All these filters have in common that they can be written as a convolutional operation of an image I with a kernel K. In the one-dimensional continuous space, the convolutional operation is well known from signal processing. A signal $x(t)$ is convolved with a weighting function $w(t)$:

$$s(t) = (x * w)(t) := \int x(a) \cdot w(t-a)\, da$$

In one-dimensional discrete space, one can write:

$$s(t) = (x * w)(t) := \sum_{a=-\infty}^{a=\infty} x(a) \cdot w(t-a)$$

As microscopy images by nature are two-dimensional signals and filters are of limited size, we use a 2D convolution here:

$$S(i,j) = (I * K)(i,j) := \sum_m \sum_n I(m,n) \cdot K(i-m, j-n)$$

In neural networks, the kernel flipping is not important, so in machine learning frameworks the following definition of convolution is implemented:

$$S(i,j) = (I * K)(i,j) := \sum_m \sum_n I(i+m, j+n) \cdot K(m,n)$$

The usage of filters in neural networks is implemented by parameter sharing. Thus, the implementation of a convolutional neural network (CNN) is equivalent with adding special constraints on the weight matrices of fully connected layers.

The usage of convolutions in artificial neural networks has multiple advantages. First, it mimics traditional computer vision algorithms and the functional mechanics of the primary visual cortex in human brains. Second, the number of parameters used is drastically reduced by parameter sharing as size of the filters used is small in typical settings. Finally, learning filters makes the neural network invariant to translation of the input data. This obviously is a desired property when dealing with images. For example, in image classification, a network should not need to remember the exact position of on object, but rather the significant features of it independent from its position on the image.

Convolutional neural networks are designed such that they learn multiple filters in each layer. Thus, for each layer, the input can be seen as an image

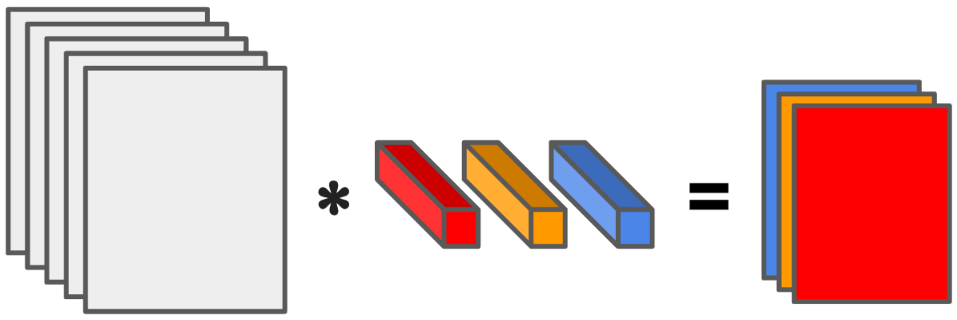


Figure 13: An input image with five channels is convolved with three small filters. Thus, the output image has three channels. The output images are smaller in resolution than the input images if the region in which the definition of convolution is valid is used only. Padding with zero can extend the valid region such that the output image has the same resolution as the input.

with a certain number of channels. Filter kernels that range over all channels of the input image are learned and the output image consists of an image with one channel per filter learned. Figure 13 illustrates one convolutional layer.

3 Methods

3.1 Image Segmentation with Deep Learning

3.1.1 Image Understanding

As we have seen in section 2.3, image segmentation has many different facets and is not a trivial problem to solve. Thus, many different deep learning models were developed in recent years to tackle different image segmentation tasks.
Deep learning models evolved from image classification [41]. In image classification, a whole image is given one single label. A popular example for an image classification task is the classification of the images in the ImageNet data set [63, 29].

A next step in image understanding is the localization of objects within an image. Usually, the location of an object is given with bounding boxes [64].

Pixel-wise segmentation is the most involved task in image understanding. Many different papers have been published for instance segmentation and semantic segmentation tasks [65, 56, 27]. A popular approach for semantic segmentation that we pursue here, is to treat semantic segmentation as pixel classification. For this project, we consider two different formulations. In the first formulation, we classify each pixel as either nucleus, boundary or background pixel with multinomial classification [40]. We call this formulation the 3-class formulation. In a second formulation, we use binary classification to predict whether a given pixel belongs to the boundary of a nucleus. This formulation is referred to as boundary formulation.

3.1.2 U-net

We want to use a convolutional neural network based on the architecture of `U-net`, a network used for semantic segmentation of microscopy images [13, 61].
U-net has a multi-level architecture and uses skip connections. It consists of

a downscaling branch and an upscaling branch.
In the downscaling branch, each step consists of two convolutional layers followed by a downsampling operation. Downsampling operations are operations that reduce the data dimension by summarizing features. The downsampling operation is a max pooling layer which is an operation that summarizes each neighborhood of two by two neurons with it's maximum value. Thus, it reduces the dimension of the data by a factor of $\frac{1}{4}$.

Each step in the upscaling branch also consists of two convolutional layers followed by an upsampling operation which doubles the output layer's image resolution by repeating each neuron's value twice over both output image dimensions. So called skip connections are used which are operations that merge the output of the of the last convolutional layer of each step in the downsampling branch onto the output of the upsampling layer of the same resolution in the upsampling branch. Finally, The output of the upsampling branch is the input for a sequence of three additional convolutional layers. Figure 14 visualizes the network.

3.1.3 Adaptions to U-net

In our model, we introduce a small number of adaption to U-net:

- To save memory during the training process, we use a smaller input image resolution of 256 by 256 pixels.

- The output of our model has three channels in our 3-class formulation and one channel in our boundary formulation.

- As in U-net, we use filters of size three by three. However, we prefer a model in which the input resolution matches the output resolution. This makes handling the data much easier. Our experiments have shown that U-net still provides outstanding results when the input image of each convolutional layer is padded with zeros of width one. In this case, the input resolution matches the output image resolution.

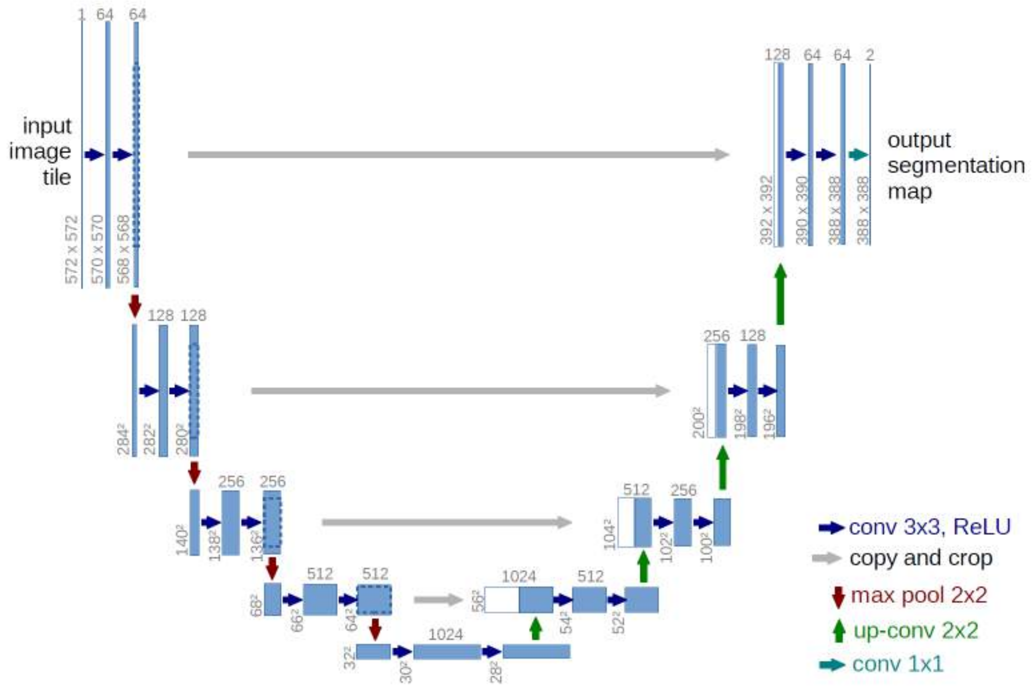


Figure 14: The architecture of U-net. The naming of this network is inspired by its shape in this visualization. On the left, the downsampling branch consists of four steps. The upsampling branch on the right has four steps too. The skip connections in gray merge intermediate layers to the input images of each step in the upsampling branch as additional channels. Figure reprinted from [61].

With this model, we ignore that convolutions at the image boundary are strictly speaking not valid.

- To account for the smaller input image size, we only use three steps in both branches. We hence use only three max pooling layers and three upsampling layers.
- Batch normalization is used after each convolution as a regularization technique and to avoid the vanishing gradient problem. A momentum value for the moving average computation of 0.9 is used [35].
- We observed that the RMSprop optimizer with a learning rate of 10^{-4} worked best for all models [32]. Recommendations from [41] were followed for the optimization of hyperparameters.

The trunk of the network for both formulation is the same and is outlined in table 2. For the 3-class formulation, we add a convolutional layer with three layers and a softmax activation the ensure the validity of the predicted probability distribution. For the boundary formulation, we add a convolutional layer with a single channel and the sigmoid function as an activation.

3.1.4 Implementation

We use the toolbox *Keras* to implement our variant of U-net [12]. Training and inference is performed with neural network framwork *TensorFlow* and monitored with *TensorBoard* [1]. For preprocessing the data we make use of the libraries *scikit-image*, *NumPy*, and *Pandas* [69, 71, 48]. For visualization, we use *Matplotlib* [34].

We save our data and code on a cluster available at the Imaging Platform. On the same cluster, we train or models making use of one of the available graphic cards:

- Nvidia GeForce GTX TITAN X
- Nvidia TITAN X (Pascal)
  Two GPUs of this type are available on the machine.

Layer Name	Layer Type	Output Shape
input_1	InputLayer	(256, 256, 1)
conv2d_1	Conv2D	(256, 256, 64)
conv2d_2	Conv2D	(256, 256, 64)
max_pooling2d_1	MaxPooling2D	(128, 128, 64)
conv2d_3	Conv2D	(128, 128, 128)
conv2d_4	Conv2D	(128, 128, 128)
max_pooling2d_2	MaxPooling2D	(64, 64, 128)
conv2d_5	Conv2D	(64, 64, 256)
conv2d_6	Conv2D	(64, 64, 256)
max_pooling2d_3	MaxPooling2D	(32, 32, 256)
conv2d_7	Conv2D	(32, 32, 512)
conv2d_8	Conv2D	(32, 32, 512)
up_sampling2d_1	UpSampling2D	(64, 64, 512)
merge_1	Merge	(64, 64, 768)
conv2d_9	Conv2D	(64, 64, 256)
conv2d_10	Conv2D	(64, 64, 256)
up_sampling2d_2	UpSampling2D	(128, 128, 256)
merge_2	Merge	(128, 128, 384)
conv2d_11	Conv2D	(128, 128, 128)
conv2d_12	Conv2D	(128, 128, 128)
up_sampling2d_3	UpSampling2D	(256, 256, 128)
merge_3	Merge	(256, 256, 192)
conv2d_13	Conv2D	(256, 256, 64)
conv2d_14	Conv2D	(256, 256, 64)

Table 2: Trunk of the architecture used in our model. The batch normalization layers used after each convolutional layer are not depicted in the table. In addition, one last layer is added, depending on the problem formulation. In total, the model's trunk has 14 convolutional layers and less than eight million parameters.

- Nvidia GeForce GTX 1080 Ti

All our code is publicly availble on GitHub. The code can be cloned from the following repository:

`https://github.com/jr0th/segmentation.git`

3.2 The BBBC022 Data Set

3.2.1 Description

The BBBC022 data set is a compound profiling experiment using the Cell Painting assay [45]. It was run for 1600 bioactive substances on human U2OS cells. The cells were stained with six different dyes using the Cell Painting protocol described in section 2.2.2. Each compound was used to treat cells in 4 different wells and in addition, 64 wells per plate were used as a control experiment. In total, 20 384-well plates were used.
As 9 different sites were captured per well, the data set contains images of $20 \cdot 384 \cdot 9 = 69\,120$ sites. As two dyes (WGA and phalloidin) share the same wavelength for excitation and emission, 5 different wavelengths were captured. However, we only use the DNA staining for our experiments.
In the analysis of this data set, the compounds were clustered using morphological features of the cells that were treated with it. It was shown that compounds which were similar in protein targets and chemical structure were likely to end up in the same cluster. This emphasizes that Cell Painting can be used in combination with CellProfiler to extract meaningful morphological features from fluorescence microscopy images.
Further information about the experiment can be found in the corresponding paper [26].

3.2.2 Segmentation with CellProfiler

We want to compare the nucleus segmentations obtained with the deep learning model to outputs obtained with the traditional segmentation algorithms

Parameter	Value
Typical diameter of objects in pixel, Min	15
Typical diameter of objects in pixel, Max	150
Threshold strategy	Global Otsu
Method to distinguish clumped objects	Shape
Maxima suppression distance in pixel	20

Table 3: Parameters for the module `IdentifyPrimaryObjects` in the CellProfiler pipeline used for segmenting BBBC022

currently implemented in CellProfiler. In addition, we want to use CellProfiler to generate training data for the deep learning model.

Therefore, we segment not only a test set consisting of 50 images but the DNA channel of all images contained in BBBC022 with a CellProfiler pipeline specifically designed for BBBC022. This simplistic pipeline contains the following modules:

- `LoadData`
  Loads data from disk.
- `IdentifyPrimaryObjects`
  Performs the segmentation. Parameters such as the thresholding method are set here and summarized in table 3.
- `ConvertObjectsToImage`
  Converts the segmented nuclei from `IdentifyPrimaryObjects` into a mask that can be stored as an image.
- `SaveImages`
  This module saves the mask to file.

The main parameters used for the segmentation are listed in table 3. The pipeline contains all further details and is publicly available in our GitHub repository under `code/segmentation.cppipe`.

3.2.3 Building a Ground Truth Data Set

Annotation Purpose. We hand-annotate 200 images from BBBC022 for the following reasons.

- Ground truth data is needed to evaluate the performance of a segmentation algorithm. We use ground truth annotations to evaluate both the performance of the deep learning model and the performance of CellProfiler.
- Data for training and validating the deep learning model is needed.

The 200 annotated images are split into a training set of 100 images, a validation set of 50 images and a test set of 50 images. The test set is used for performance evaluation only. This ensures that the deep learning model has not seen any of these images during the training process.

Hand-Annotating 200 Images.

Choosing Images for Annotation. We chose 200 images out of 69 120 available images in such a way that for each compound there is at most one single annotated image. This minimizes the correlation between the images in the training, validation and test data sets. The publicly available metadata of BBBC022 contains the information about which well was treated with which compound.

Ensuring Highest Annotation Accuracy. As we require the ground truth annotations to be very accurate, we asked experts for their support with the image annotations. Two biologists and a medical doctor marked the boundaries of nuclei on printouts of all 200 images that were to be annotated. An example of a printout is give in Figure 15
The marked printouts were used to annotate nuclei in an annotation tool [25]. The output of the annotation tool is a mask that assigns each pixel to a class. Background pixels were annotated with 0, pixels belonging to nuclei were annotated with positive integers in such a way that touching nuclei were

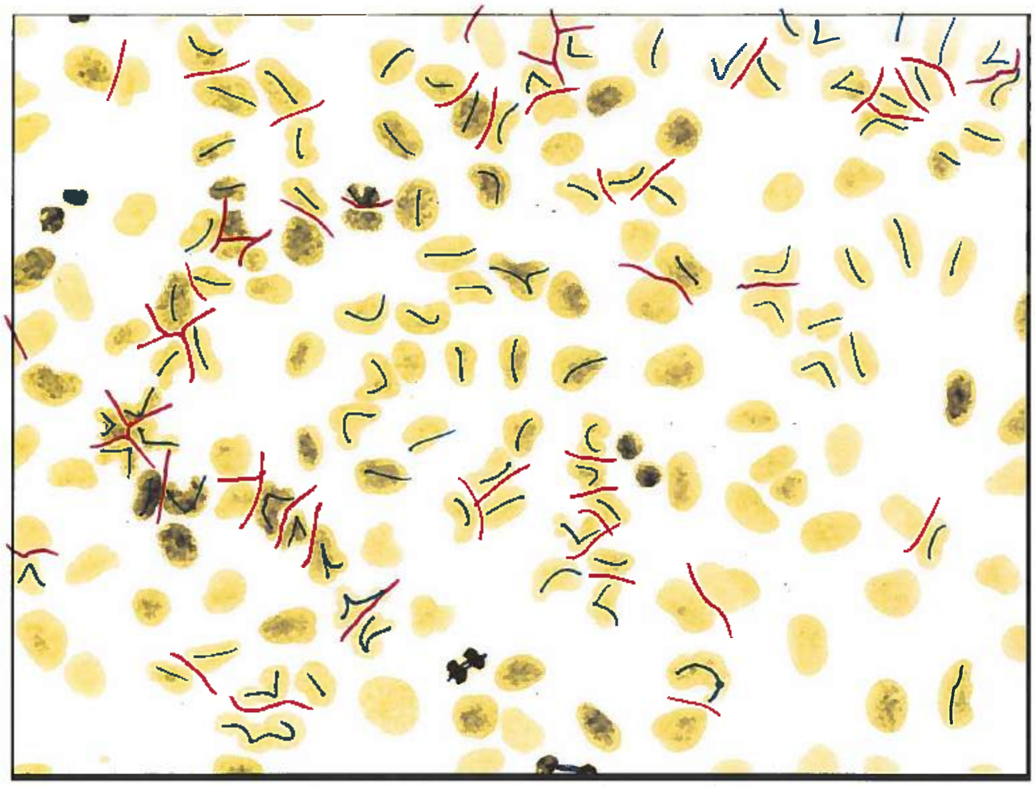


Figure 15: A printout of a fluorescence microscopy image was annotated by an expert. For clumped nuclei, red lines separate two nuclei, a blue line connects regions that belong to the same nucleus. The image was processed with the open source image processing software GIMP for better readability. The colors were inverted and a yellow tint was added [23].

assigned different numbers. Figure 16 depicts a fully annotated image in the annotation tool.
Annotating a printout took the experts roughly 10 minutes per image. An additional 15 minutes were spent on creating the masks with the annotation tool.
We filter out small objects, and thus micronuclei, for the experiments as they are not annotated consistently and the evaluation metrics for the segmentation introduced in section 3.3 are heavily influenced by the number of objects detected, independent of their size. The filter requires every object in the annotation to cover an area of at least 100 pixels which is larger than a typical micronucleus in the data set and smaller than a typical cell. Since micronuclei have biological significance, a different segmentation strategy should be considered for experiments in which they are to be analyzed.

3.3 Computing Splits and Merges to Compare Segmentations

To benchmark our model, we compare the segmentations obtained with CellProfiler and those obtained with deep learning models against the ground truth segmentations. As a test set, 50 hand-annotated images are used that the deep learning models are not allowed to use during the training process.

This section describes a method we implemented to compare two segmentations. This approach is commonly used in computer vision for evaluation of instances segmentation algorithms and we adapted it to objectively measure the quality of segmentations [43, 72, 18].

3.3.1 Motivation

Segmenting a fluorescence microscopy image in foreground and background regions is solved relatively well by thresholding the image on pixel intensity values. However, splits and merges of nuclei are common errors in segmentations of DNA stains of crowded cell populations. These two type of errors are most troublesome to biologists.

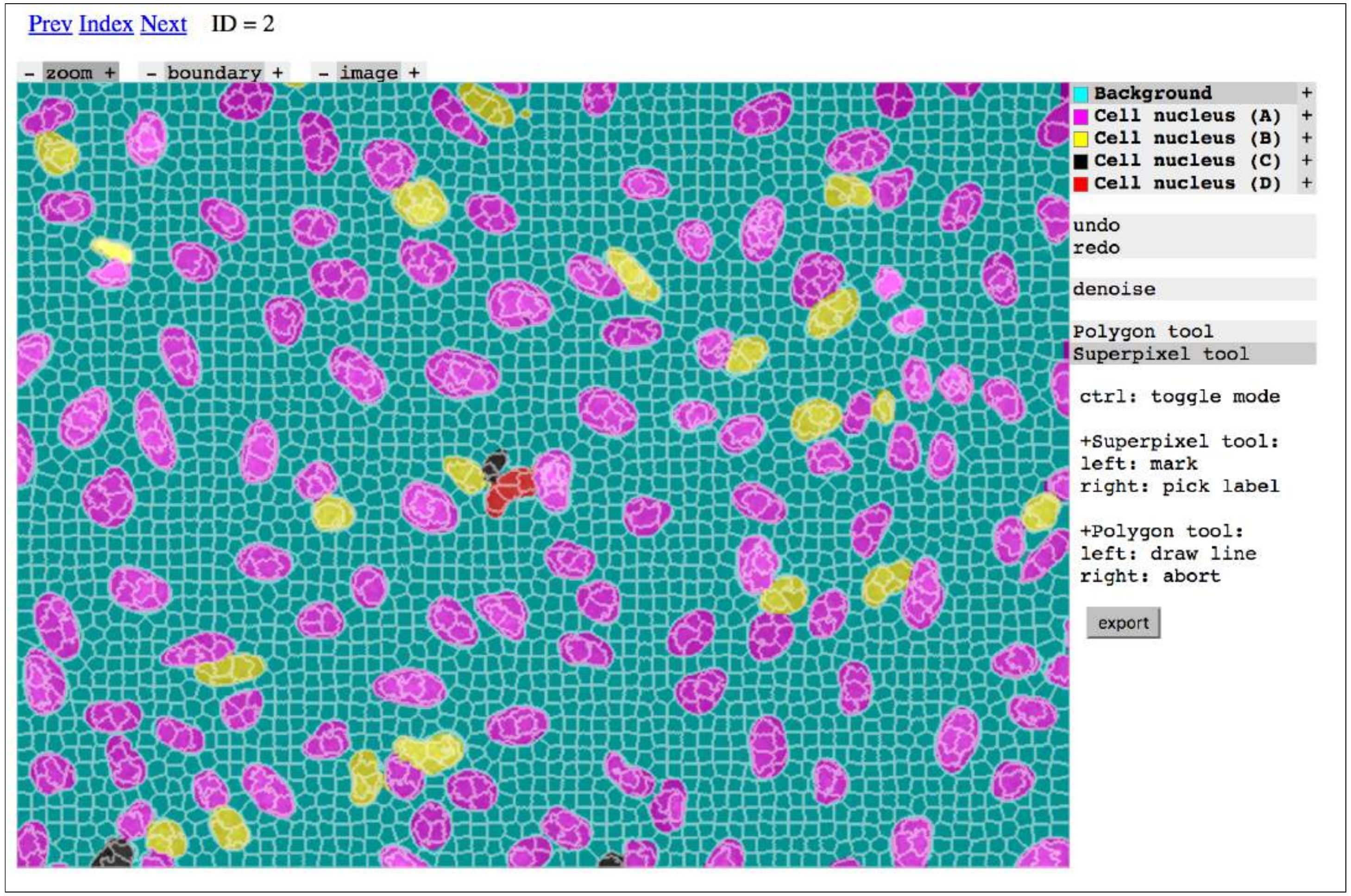


Figure 16: An annotation tool was used to create a mask for the image. Different colors in the annotation tool represent different integers that the pixels are assigned to. One can observe that touching nuclei are annotated with different integers.

- Split

  A split is an oversegmentation of a nucleus. Even though the image just depicts one nucleus, the method's segmentation detects two or more nuclei that combined usually cover the area of the real nucleus in the ground truth. Examples of splits are visualized in figure 17.

- Merge

  A merge is an undersegmentation of multiple nuclei. The method finds only one nucleus whereas there are multiple touching nuclei in the image. A typical merge that occurred during training of our deep learning models is depicted in figure 18 with the image and its ground truth annotations as a reference in figures 19 and 20 respectively.

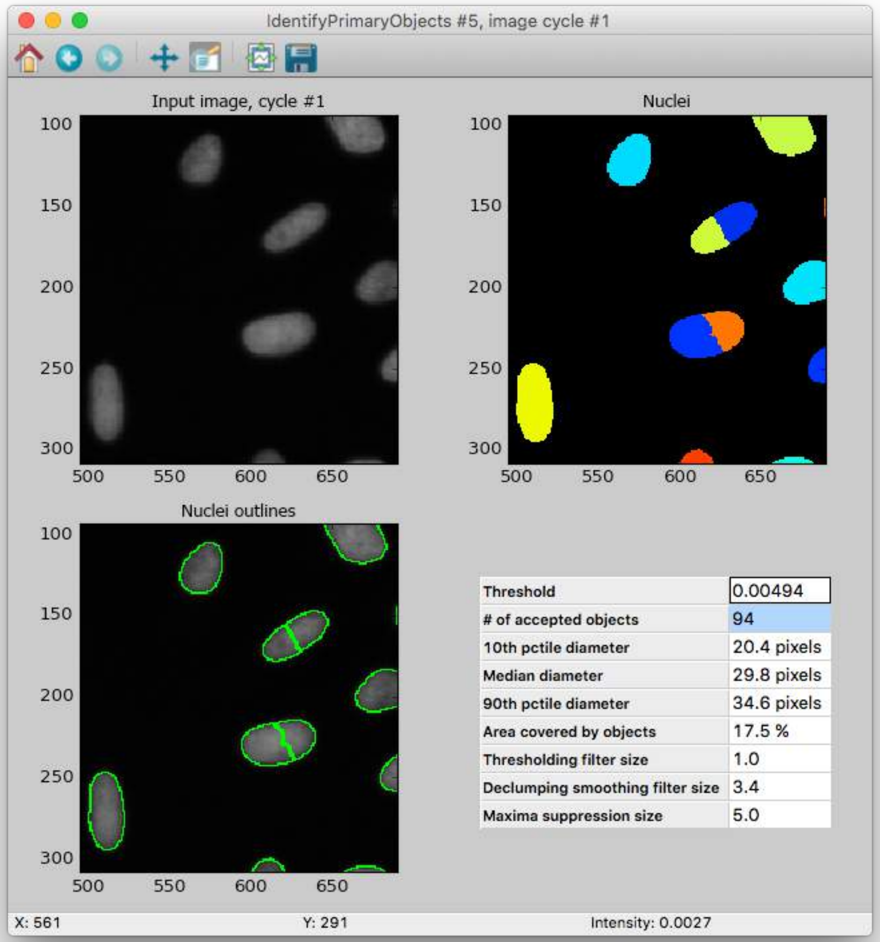
Thus, by calculating the number of split and merge errors made by a method compared to the ground truth, we can estimate its performance.
We generalize splits and merges to *overdetections* and *underdetections*. Overdetections are cases where the method predicts a nucleus that is not present in the ground truth data. An underdetection occurs if a nucleus present in the ground truth data is not detected by the segmentation algorithm.

3.3.2 Implementation

To implement this kind of error measure, we agreed on a function that has the method's segmentation and a ground truth segmentation as an input and outputs the number of overdetections and underdetections.
Our implementation is publicly available and can be found in the GitHub repository in the file `metrics.py` in the directory `/code/helper/`.

The function `compare_two_labels` checks each pair (n_{gt}, n_{meth}) consisting of a nucleus n_{gt} in the ground truth segmentation S_{gt} and a nucleus n_{meth} of the method's segmentation S_{meth} for overlap. The overlap of the two nuclei is defined by the intersection over union [53]:

$$\text{IoU} = \frac{\#\text{pixels in } n_{gt} \cap n_{meth}}{\#\text{pixels in } n_{gt} \cup n_{meth}}$$



Threshold	0.00494
# of accepted objects	94
10th pctile diameter	20.4 pixels
Median diameter	29.8 pixels
90th pctile diameter	34.6 pixels
Area covered by objects	17.5 %
Thresholding filter size	1.0
Declumping smoothing filter size	3.4
Maxima suppression size	5.0

X: 561 Y: 291 Intensity: 0.0027

Figure 17: Splits suggested by a CellProfiler pipeline with bad parameter tuning. The nuclei in the image center are erroneously split into multiple parts as observed not only in the mask image on the top right but also with the nuclei outlines in the bottom left visualization.

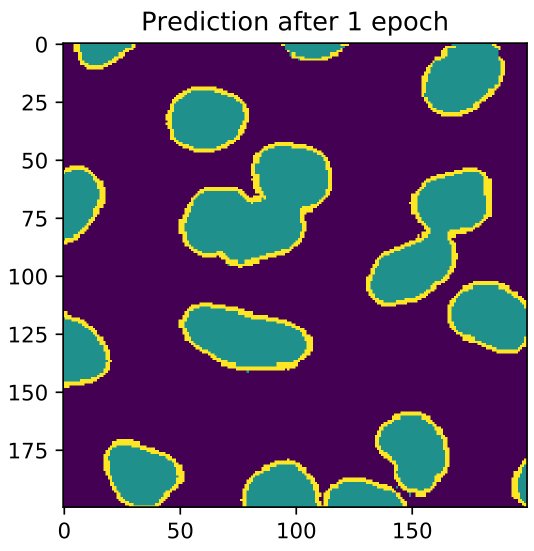


Figure 18: Merge predicted by deep learning model. Two predicted nuclei close to the image center actually consist of multiple nuclei.

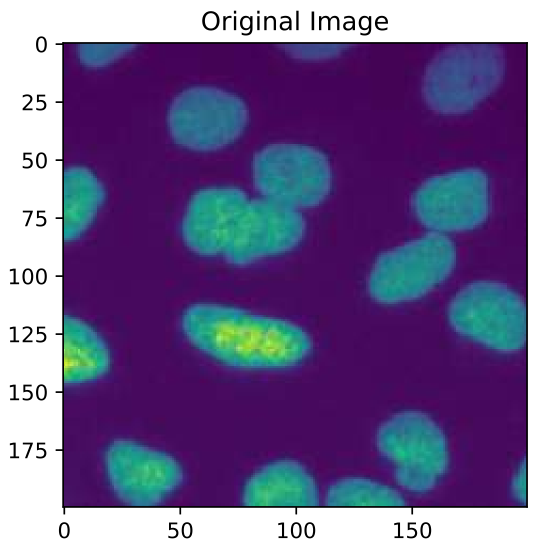


Figure 19: Raw image used for segmentation. Nuclei touch frequently in fluorescence microscopy images and segmenting them correctly is not trivial.

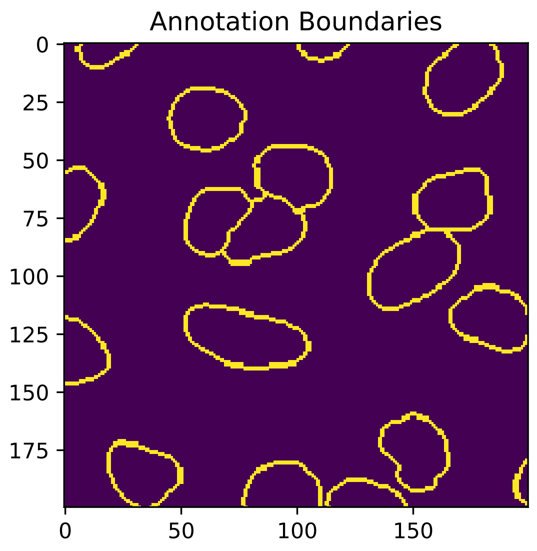


Figure 20: Boundaries of the corresponding ground truth segmentation according to an expert.

If the IoU exceeds $\frac{1}{2}$, we consider the corresponding pair as a true positive detection, a match between the algorithm's segmentation and the ground truth segmentation. Obviously, a nucleus, no matter which segmentation it is contained in, can at most be matched with one other nucleus in the other segmentation as it can not overlap by more than $\frac{1}{2}$ with more than one nucleus.

Finally, an unmatched nucleus in S_{gt} is an underdetection, an unmatched nucleus in S_{meth} is an overdetection.

To compare segmentation algorithms across data sets, we consider the algorithms' task as object detection and thus calculate object-level precision and recall metrics [72]. Precision represents the share of matches among all detected objects, the recall measures the share of matches among all nuclei in the ground truth annotation:

$$\text{Precision} = \frac{\#\text{matches}}{\#\text{nuclei in algorithm's segmentation}}$$

$$\text{Recall} = \frac{\#\text{matches}}{\#\text{nuclei in ground truth annotation}}$$

To summarize these two metrics in a single scalar, we make use of the F_1 score which is the harmonic mean of precision and recall [5]:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F_1 score gives us a proxy for the performance of a segmentation algorithm; the higher the better. To get more insights about the F_1 score, we use bootstrapping to evaluate the robustness of this measure [59]. We draw 20 out of all 50 test images for 10 000 iterations and calculate the F_1 score for each sampled subset. We report the mean F_1 score and its standard deviation. Precision and recall are not computed with bootstrapping, a statistic across the whole test data is reported.
In the following section, we benchmark the algorithms against the ground truth annotations with this method.

4 Experiments and Results

This section summarizes the results we obtained for training the CNN, how we optimized it, and how its segmentation performance compares against the quality of segmentations obtained with CellProfiler. We thereby use the method described in section 3.3. We train the deep learning models on 100 images and perform validation on 50 images for which ground truth annotations were created as discussed in section 3.2.3. The remaining 50 images of the ground truth data set were used as the test set.

First, we evaluate the segmentation performance of CellProfiler on the test set for BBBC022. Next, we present the results obtained with deep learning models that were trained on CellProfiler segmentations of the training images. As these segmentations are considered noisy, we improve our model by training on hand-annotations. We show that one can obtain better segmentations than CellProfiler when training a deep learning model on a relatively small hand-annotated data set consisting of 100 images depicting about 10 000 cells. We show results of this model in both formulations introduced in section 3.1.1, the 3-class formulation and the boundary formulation.
Finally, we show that we can increase the model's performance by using data augmentation.
All benchmarking experiments can be found in the GitHub repository under `/experiments/` and are structured in the following way. `<Model>` hereby stands for the concrete model which is evaluated.

- **Model to Segmentation**
  A notebook named `<Model>_2_label` performs the prediction of the deep learning model and creates the segmentations for the test images. This obviously is not needed for benchmarking CellProfiler. We used CellProfiler's output directly in this case.

- **Segmentation Comparison**
  The obtained segmentations are compared against the ground truth annotations in a notebook called `<Model>_vs_GT`.

- **Visualizing Errors**
  In the notebooks called `visualize_errors`, we visualize the errors made by the model with error images. The error images are to be read as follows. Pixels that belong to an underdetected nucleus are colored in brown, pixels belonging to an overdetection in light blue. In the special case in which a pixel is part of an underdetection and an overdetection, it is colored in pink. This is often encountered where merges occur.

4.1 Benchmarking CellProfiler

As discussed in section 3.2.2, we use CellProfiler to segment a test set of 50 images of BBBC022 which were also hand-annotated. We use the calculation of splits and merges described in section 3.3 to evaluate the quality of these segmentations. By bootstrapping 10 000 samples from our test set results, we compute statistics for the F_1 score. We obtain a mean F_1 score of 0.90 with a standard deviation of 0.013. The precision across the whole test set 95%, the recall 85%. This means that 15% of all nuclei in the ground truth data set are undetected and 5% of all detected objects are not annotated in the ground truth segmentation. Figure 21 visualizes the distribution of the F_1 score.

Figure 22 shows a ground truth segmentation of a test image, Figure 23 a segmentation obtained by CellProfiler, Figure 24 the corresponding error image.

4.2 Learning on Automatically Generated Annotations

As obtaining manual annotations of images is time-consuming, we evaluate the performance of a model which was trained on automatically generated annotations. We therefore train a CNN on segmentations generated by CellProfiler. In addition we are interested in whether a CNN can actually provide better segmentations than those on which it was trained.

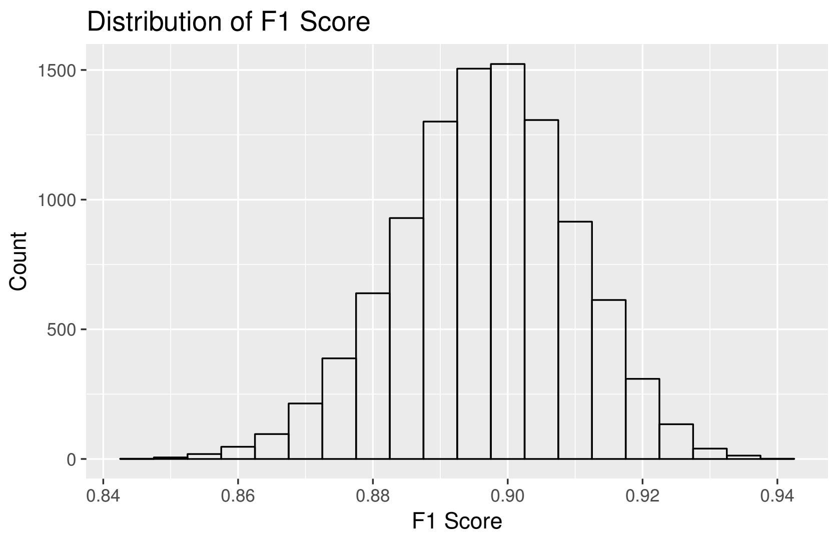


Figure 21: The distribution of the F_1 score for the quality of the segmentations obtained with CellProfiler. We sampled 20 of all 50 test images 10 000 times.

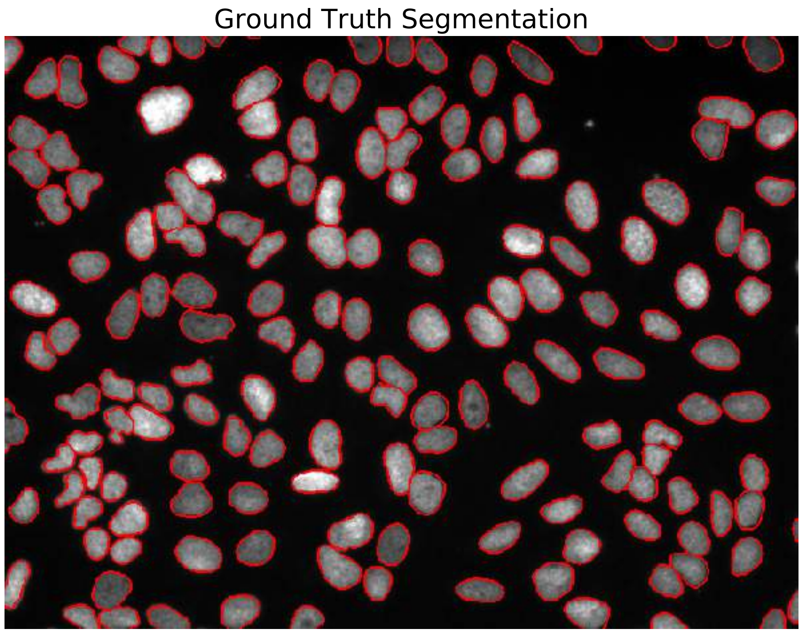


Figure 22: Ground truth segmentation. A full image (BBBC022, plate 20589, well D20, site 3) is depicted here with its full resolution of 696x520 pixels.

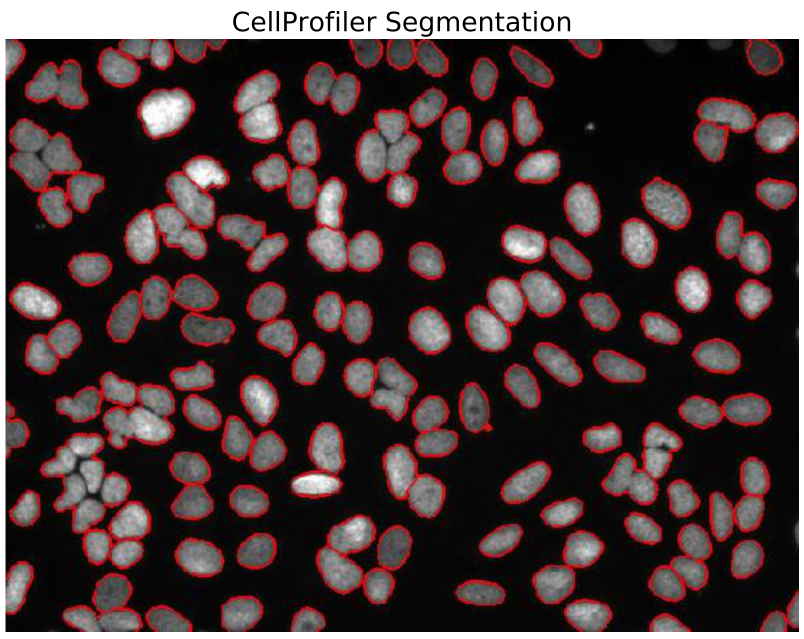


Figure 23: Segmentation obtained with CellProfiler. It shows merges for clumped cells. These merges are detected as multiple underdetections and an overdetection. This is the same input image as in figure 22 (BBBC022, plate 20589, well D20, site 3).

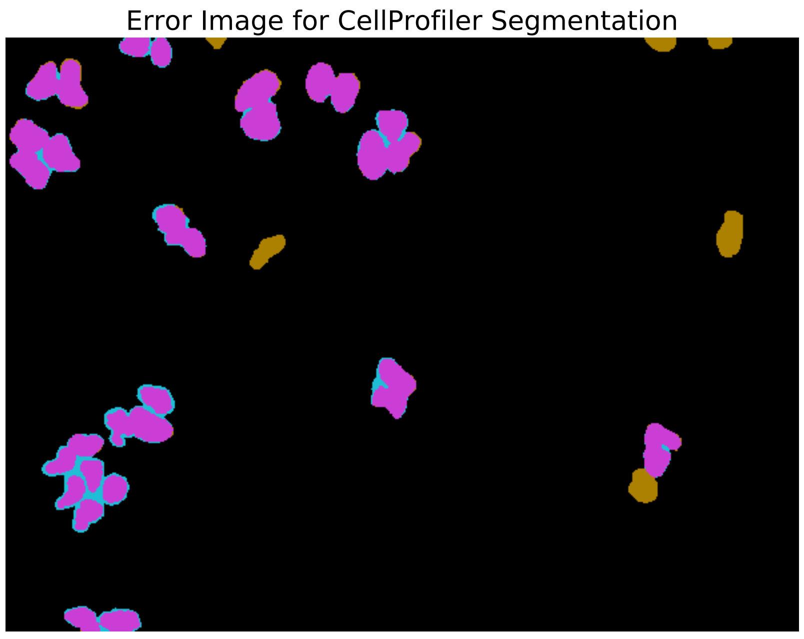


Figure 24: Error image for CellProfiler segmentation of the same image (BBBC022, plate 20589, well D20, site 3). Cells in brown are underdetections, cells in blue are overdetections, and pixels that are contained in overdetections and underdetections are colored in pink.

Experiment Setup. We formulate the segmentation problem as a 3-class problem introduced in section 3.1.1 which predicts the probability of each pixel belonging to either a nucleus, a nucleus' boundary or the image's background. We use the images from the training and validation set, but we do not use their ground truth annotations for training. Instead, we use segmentations generated by CellProfiler as described in section 3.2.2. For test purposes, we compared against the ground truth segmentations. The segmentations generated by CellProfiler are preprocessed in such a way that the boundaries around nuclei have a width of 2 pixels. We heavily make use of the `skimage.segmentation` library [69]. To generate training data of size 256 by 256 pixels, we use four non-overlapping crops from each image. Thus, in total we train our model on 400 patches of the required input size.

The post processing step is trivial for the 3-class formulation. As the network predicts a probability distribution across the three classes for each pixel, we use the class for which the network predicted the highest probability. We ignore the background and boundary channel, and connected components in the nucleus channel are detected nuclei. We filter connected components which are smaller than 100 pixels as discussed in section 3.2.3 to not take into account micronuclei. A further optimization that can be performed is dilating the detected nuclei by half of the boundary width. This is crucial for optimizing the intersection over union with the ground truth annotation.

This model uses the softmax function introduced in section 2.4.3 as the activation function for the last layer and categorical cross-entropy as the loss function. The hyperparameters are chosen as described in section 3.1.3. The classification accuracy gives us an additional metric on the model's performance. We want to mention here, that both, categorical cross-entropy and classification accuracy are pixel-based metrics and not object-based, even though object-based metrics are more relevant to biologists. Thus, we use differentiable metrics for training the network but switch to the method described in section 3.3 to evaluate the model's performance

Results. Pixel-wise, we can achieve a classification accuracy close to 100% on the test set and a validation accuracy of more than 98% as visualized in figure 25. Even though the accuracy is high, the segmentation results on the test set are not satisfying as many merges occur. We conclude that this is because the classification of only few pixels, namely those between clumped nuclei, defines whether split or a merges error occur. These few crucial are misclassified by the network and we see a large number of merges occur with this model as visualized in figure 27. We see that classification accuracy thus is not a good metric for the performance of the model as discussed in section 3.3. The learning curve is visualized in figure 26. Training this network takes roughly two hours for 200 epochs.

The object-based metrics show a precision of 94%, a recall of 80% and a mean F_1 score of 0.86 with a standard deviation of 0.018. We conclude that this model performs worse than CellProfiler.

4.2.1 Using Boundary Boost

To improve the segmentations generated by the network, we changed the post processing step. In this experiment, we multiply the probability of a pixel belonging to a boundary with a factor before taking the argmax over the three class probabilities to assign pixels to classes. This increased the number of boundary pixels. As a consequence, more pixels between two clumped nuclei were assigned to the boundary class and thus we achieved fewer merges. A boundary boost factor of 100 worked best in our experiments.

Using this boundary boost factor, we could slightly improve the performance of this model. We achieved a similar precision of 94%, an improved recall of 82% and thus a better F_1 score of 0.88 with a standard deviation of 0.015. However, this performance is still worse than the performance of CellProfiler measured in section 4.1.

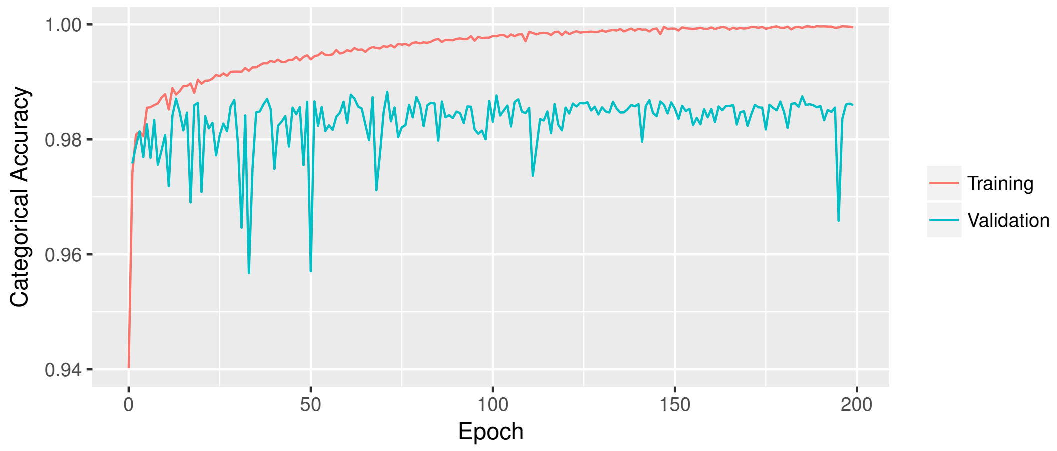


Figure 25: The categorical accuracy of the 3-class model trained on automatically generated segmentations.

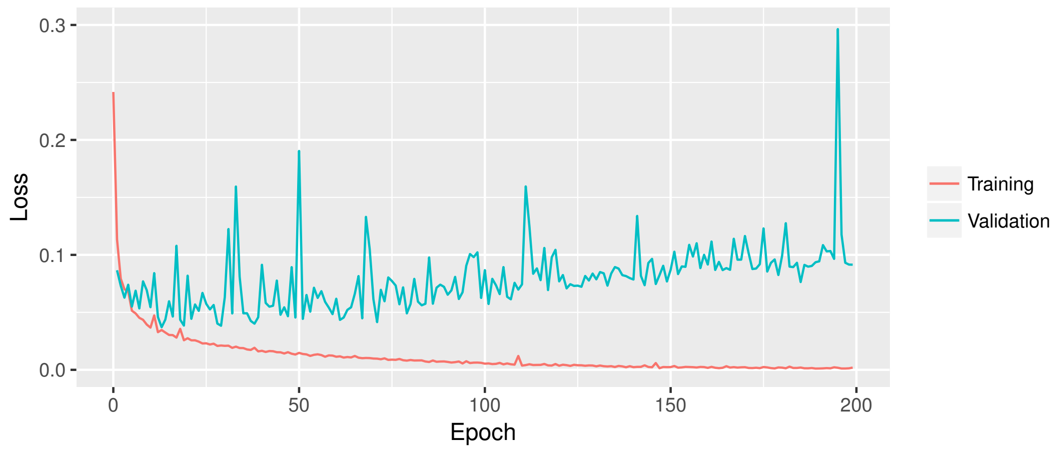


Figure 26: The learning curve of the 3-class model discussed in section 4.2. One can see that overfit occurs already after a few epochs.

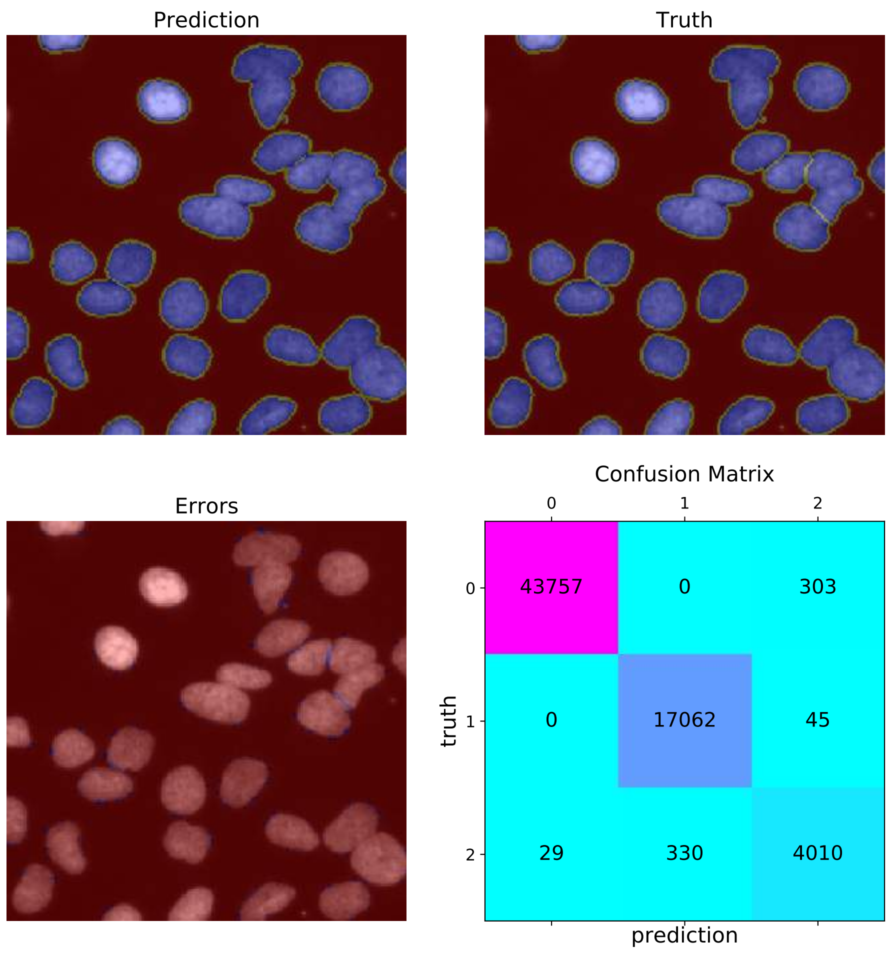


Figure 27: The segmentations of the model trained on CellProfiler segmentations shows a large number of merged nuclei even though the confusion matrix on the right shows a low classification error. The true labels are CellProfiler segmentations here and thus also show merged nuclei. Pixels labeled with 0 are background pixels, 1 encodes nuclei and 2 boundaries. In the error image, one can clearly observe that the network does not predict boundaries between nuclei even though the CellProfiler segmentation does.

4.2.2 Changing the Training Set Size

Previous works about segmentation with deep learning state that very little training data is needed to achieve satisfying results [70, 61].

To test this assumption in our context, we trained CNNs on training data sets with different sizes in further experiments. Training sets with 10, 100, 1 000 and 69 120 images were used.

We observed that increasing the size of the training data set did not yield significantly better results in the sense that the network still yielded segmentations with merge errors. Furthermore, when reducing the data set size, we observed that the model overfits the training data. We thus train our CNNs on hand-annotated data instead of automatically generated segmentations in the following experiments.

4.3 Removing Noise with Hand Annotations

As observed in the previous experiment, training on CellProfiler's segmentations does not lead to a network that can provide improved segmentations. We also observed that adding more training data or using a more sophisticated preprocessing function did not significantly improve the results.

Thus, in a next step we remove the noise in the training data by using the ground truth annotations of 100 hand-annotated training images and 50 hand-annotated validation images in our following experiment.

Experiment Setup. Analogously to the previous experiment described in section 4.2, we use the 3-class formulation here. The annotations from the annotation tool are processed in such a way that between nuclei and boundary, as well as between touching nuclei, there's a boundary of width 2 pixels. The hyperparameters remain unchanged and we refrain from using boundary boost. Also, training time remains about two hours.

Results. Using hand-annotated data for training, we can surpass the performance of CellProfiler with a mean F_1 score of 0.91 ± 0.011 with a precision of 95% and a recall of 87%.
Similar learning curves as when training on automatically generated annotations can be observed.

4.4 Predicting Nuclei Outlines

As we've seen in section 4.2, the pixels classified as boundary or background are ignored in the transformation of the network's output to a segmentation when using the 3-class model. Common errors occurring with this approach are merge errors which are due to the lack of boundary pixels between clumped nuclei. To resolve these errors, predicting only the outline of nuclei is a promising approach.

In addition, with the previously discussed models, we observed that the CNN had no difficulties with classifying background and nucleus pixels correctly. Errors were mainly due to the misclassification of boundary pixels. Thus, we force the network to focus on outlines of nuclei only by removing the other classes and using the boundary formulation discussed in section 3.1.1. Thus, in this and the following experiment, the probability for a pixel to belong to an outline of a nucleus is predicted.

4.4.1 Reformulation of the Segmentation Problem

Outline Prediction as Classification Problem. As the boundary pixels are not naturally present in the data, but artificially generated, we tried multiple different variants for generating these. Instead of generically having a boundary width of two pixels as in the previous experiments, we consider experiment setups with boundary widths of two, four, six and eight pixels. The `skimage.morphology` library was used for that purpose [69].

For this model, we use the binary cross-entropy as a loss function and as already mentioned in section 2.4.3, we use the sigmoid function as the activation function of the last layer. Again, we report pixel-wise classification accuracy as an additional metric.

Outline Prediction as Regression Problem. In addition to this classification problem, we also framed the outline prediction formulation as a regression model. In this problem, we allowed the annotations to be continuous instead of categorical and thus represent a probability with which a pixel belongs to a nucleus' outline. The closer a given pixel is to the boundary, the higher we set its probability to belong to it. This boundary was created in a preprocessing step as a linear combination of the boundaries with two, four, six and eight pixels. The coefficients of the linear combinations were chosen in such a way that the probability of a pixel belonging to the outline follows a Gaussian bell in the distance of the pixel to the boundary. It is thereby assumed that the boundary is in between the two pixels of the boundary of width two. The Gaussian bell was scaled such that the two pixels adjoint to the boundary had a probability of 1 to belong to the boundary.

For this model, we keep the setup as previously defined for the classification problem. The only difference is that the mean squared error is used as a loss function.

4.4.2 Getting Segmentations from Outlines

As the outline prediction model does not distinguish between background pixels and pixels that belong to a nucleus, the postprocessing step must map all non-boundary pixels to either the background class or the nucleus class. For the final segmentation, only pixels belonging to nuclei are of relevance. This section discusses how the postprocessing function can detect nuclei given a probability map for outlines.

We tried several methods to transform a probability map into a segmentation. All methods threshold the probability map obtained by the CNN at $p = \frac{1}{2}$. A sample probability map is outlined in figure 28.

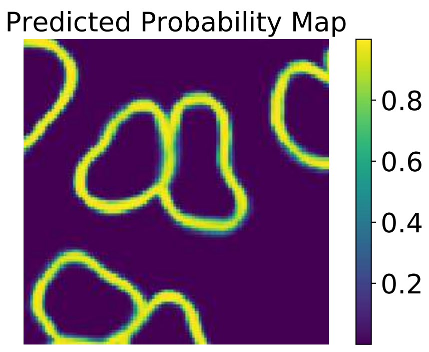


Figure 28: A probability map for boundary pixels predicted by the CNN. One can observe that the neural network is able to detect clumped nuclei.

- An intuitive approach is to threshold the original image and subtract the thresholded probability map and use the remaining foreground pixels as segmented nuclei. This would separate clumped nuclei if the pixels in between them are correctly classified as boundary. However, this would also classify debris and other artifacts that are assigned as foreground by the thresholding method and correctly undetected as nuclei by the deep learning method.

- A second method we considered is finding connected components among pixels that are not classified as boundary and filtering the largest connected component as background. This method was unsatisfactory because it failed for cases where the background was not connected, which is often the case at the image's boundaries.

- We implemented a method that is very similar but uses a more involved filtering strategy. Among all connected components in the pixels not classified as boundary, those whose mean image intensity is above a threshold are defined as nucleus. The threshold can be learned using the training data. Figure 29 shows the histogram for the DNA channel on fluorescence microscopy images from which we learned a threshold of 50 for this purpose using an 8 bit encoding that yields values ranging from 0 to 255.

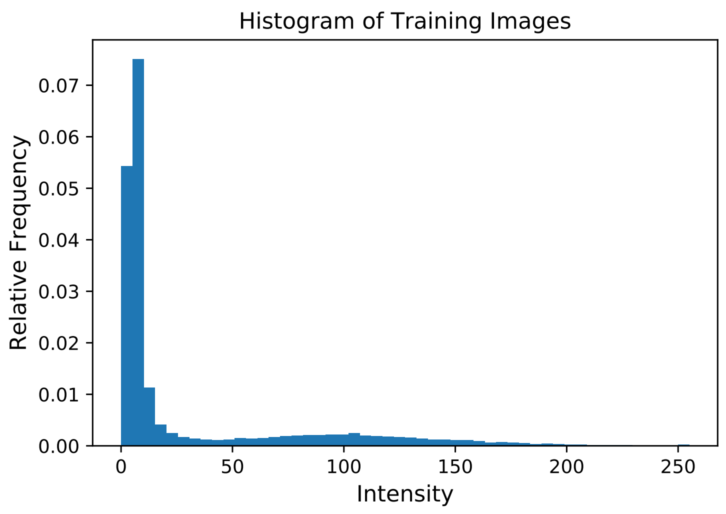


Figure 29: The histogram for intensity values of the images contained in our training set. Images were normalized before being processed. One can easily separate background and foreground pixels. We used a threshold of 50 in our analysis.

Again, objects smaller than 100 pixels are removed to exclude micronuclei. To further optimize the postprocessing, one could grow the detected regions by half of the boundary width analogously to the postprocessing step used with the 3-class formulation.

4.4.3 Improved Segmentations with Outline Prediction

Classification Problem. For a boundary width of two pixels, one can observe very sparse (5%) signal in the thresholded probability map. Thus, sometimes the outlines in the thresholded image are not closed due to small errors. These open outlines lead to underdetections. The visualization of test data revealed that this is the main reason for many underdetections. Figure 30 displays this.

We observe a better performance for the models with higher boundary width. Their segmentations do not show many unclosed contours. Among the thick boundaries, the model using a thickness of 4 delivered the best segmentations

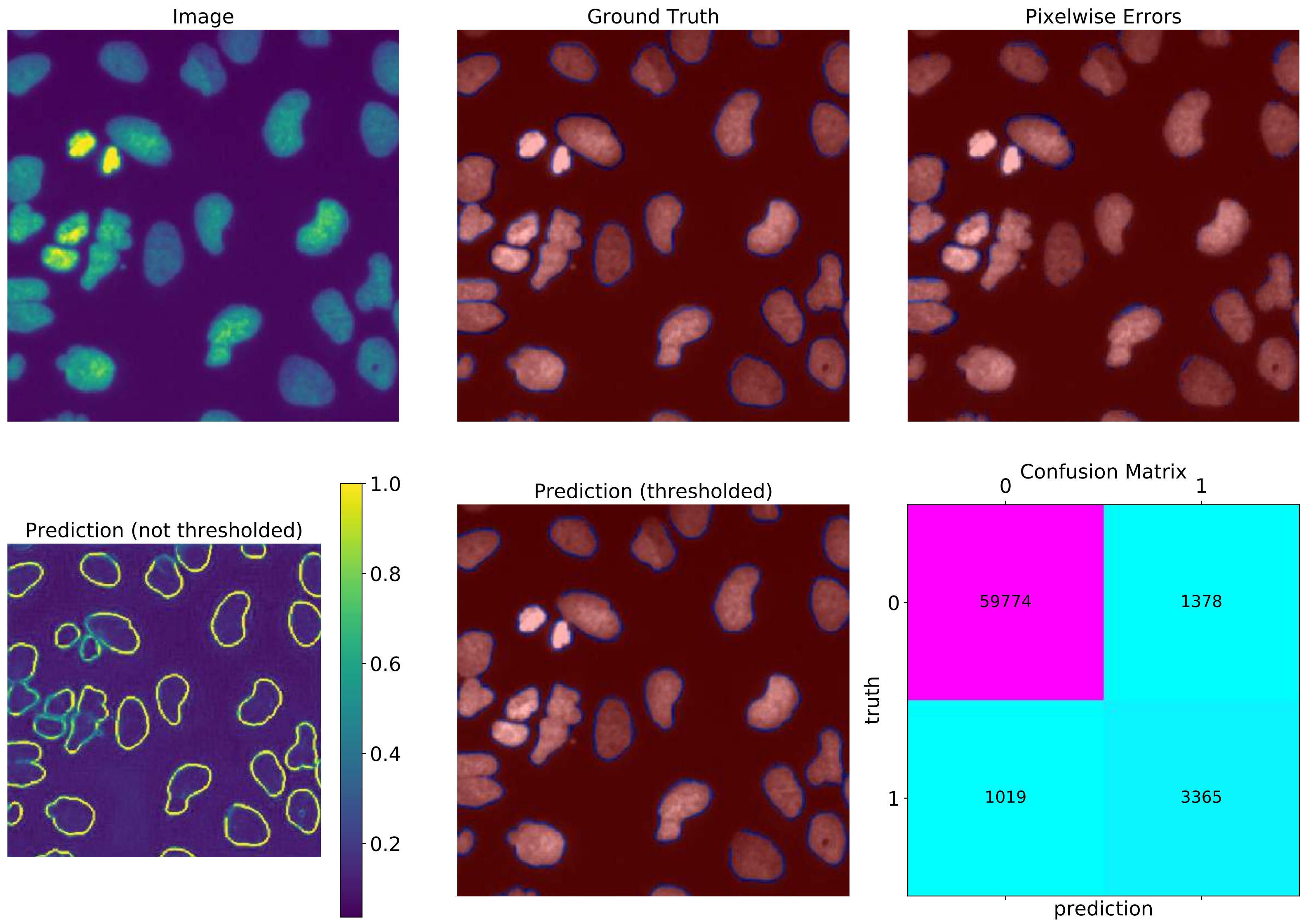


Figure 30: Some outlines of nuclei are not closed in the prediction of the deep learning model trained on boundaries with width two as can be seen in the bottom center plot. Even though the confusion matrix on the bottom right shows a high pixel-wise accuracy, the open contours result in underdetections as non-boundary pixels of a nucleus are connected to the background. Also, a separation of clumped nuclei is not achieved.

with a mean F_1 score of 0.92 with a standard deviation of 0.010. Thereby, the precision is 96%, the recall over the whole test set is 88%.

Similar to the 3-class models, we can achieve a training accuracy close to 100% and a validation accuracy of more than 97% as depicted in figure 31. Figure 32 shows the loss curves of the model trained on outlines of nuclei. For this model, the time needed for training is also roughly two hours on A Nvidia TITAN X (Pascal), the fastest GPU available to us.

Regression Problem. Besides being more complex, the model treating the outline detection problem as a regression problem performs worse than the models trained with a categorical boundary. Thus we drop this approach and focus on optimizing the outputs obtained with the model using a boundary width of four pixels in the next section.

4.5 Improving Performance with Data Augmentation

In a last optimization step, we change the training procedure in such a way that for each training step, we sample a random crop from all training images which is in addition rotated by a multiple of 90° and flipped randomly.
This approach seemed promising to us, as we achieved very high pixel-wise accuracy with the previous models. However, a common error on the test set for this models is the misclassification of boundary pixels between clumped nuclei as pixels belonging to a nucleus. On the training set, these pixels were correctly classified as boundary, which is an indicator for a model overfit on the training data. Thus, by using data augmentation, we avoid this overfit, and show that boundaries between nuclei get correctly classified as boundaries with this technique. Figures 33 and 34 show that much less overfit occurs when using data augmentation.

4.5.1 Models Trained on Hand-Crafted Annotations

Results. For clumped nuclei, we see a drastic performance improvement compared to the model without random sampling of the training crops.

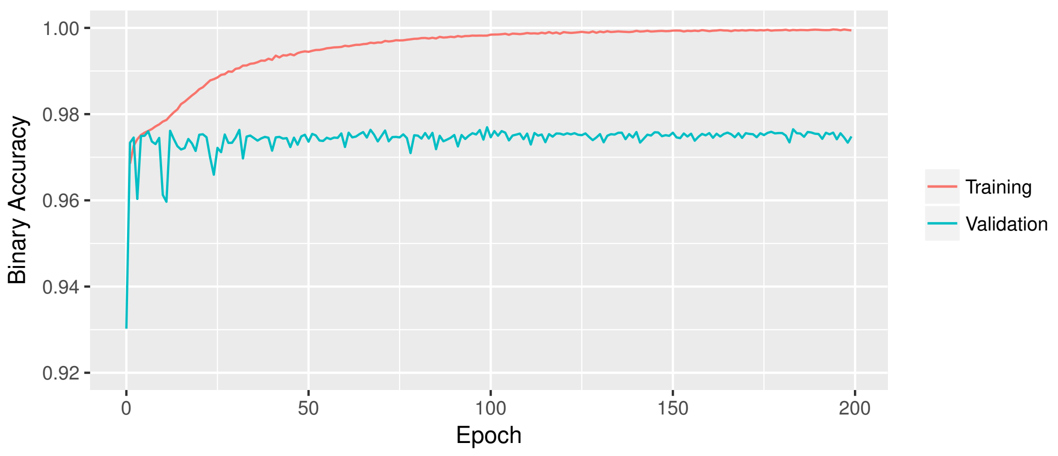


Figure 31: The binary accuracy of the model in boundary formulation trained on hand-annotated data with a boundary width of four pixels.

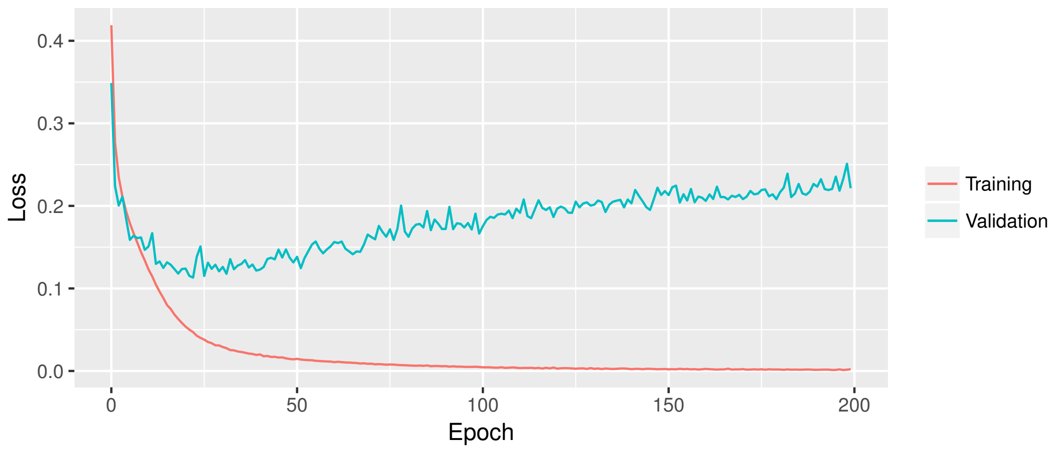


Figure 32: The learning curve for the training process of the CNN using the boundary formulation and hand annotations for training. The model starts overfitting after roughly 10 epochs.

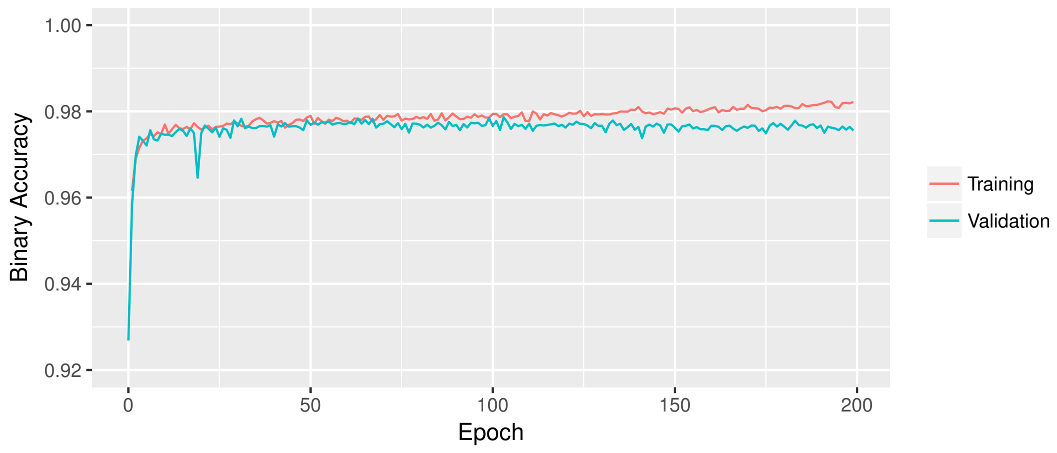


Figure 33: The binary accuracy of the model in boundary formulation with a boundary width of four pixels trained with data augmentation.

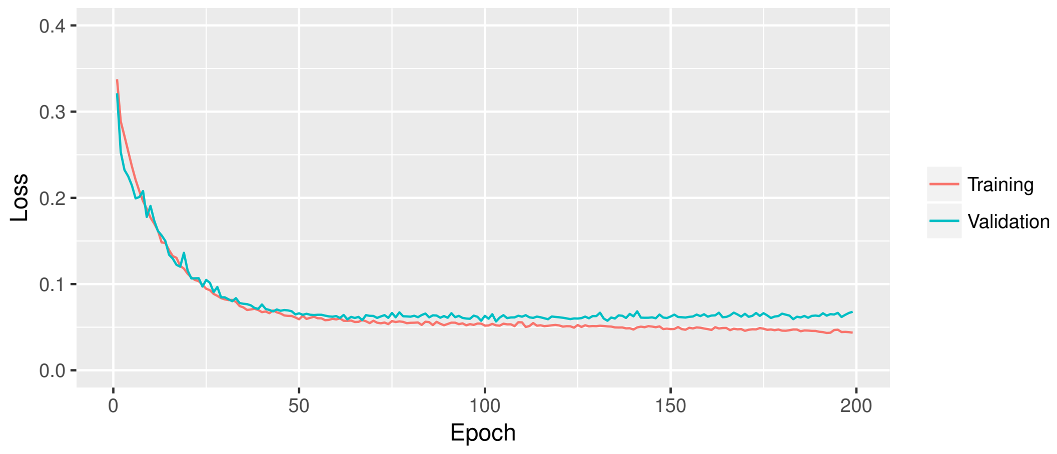


Figure 34: The learning curve for the training process of the CNN using the boundary formulation and data augmentation. We can observe much less overfit than when training the network without data augmentation as visualized in figure 32

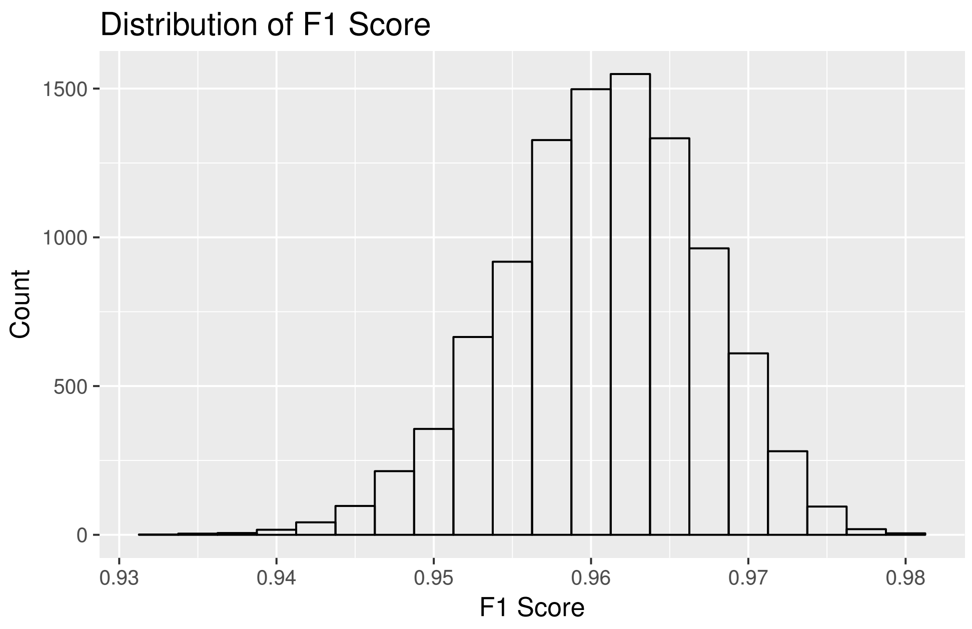


Figure 35: The distribution of the F_1 score of the neural network trained on nucleus outlines with data augmentation. The statistics were obtained with bootstrapping 10 000 subsets of size 20 from all 50 test images.

As the F_1 score of 0.96 ± 0.006 as shown in figure 35 is higher compared to all other methods, we can conclude that this deep learning method by far outperformed CellProfiler on this test data set. We obtain a precision of 98% and a recall of 94% for the whole test data set. Thus, with this setup, only 6% of nuclei in the ground truth segmentation are missed, and only 2% of detected objects are not annotated in the ground truth data.

For the original image in figure 22, figures 36 and 37 show the segmentation and the error image for the deep learning model respectively.

4.5.2 Improving Models in 3-Class Formulation

Random sampling significantly improved the results for the outline prediction model. We conducted experiments to evaluate if random sampling also improves the performance of a model using the 3-class formulation. In addition, we trained all models using the 3-class formulation with different boundary

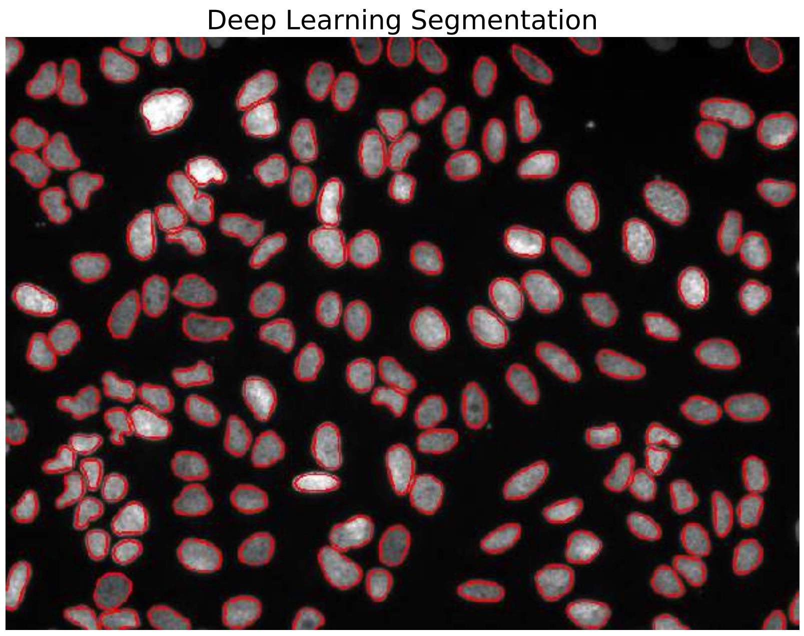


Figure 36: Segmentation obtained with the CNN trained on hand-annotated images with data augmentation (BBBC022, plate 20589, well D20, site 3).

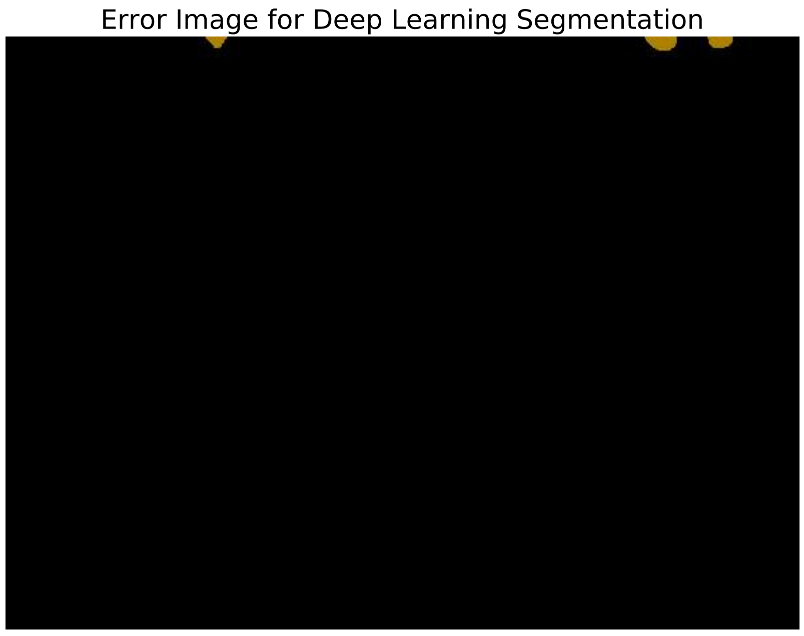


Figure 37: Error image for the deep learning segmentation. One can observe the underdetections on the top image boundary (BBBC022, plate 20589, well D20, site 3). For comparison with the segmentation of the same image with CellProfiler, see figure 23

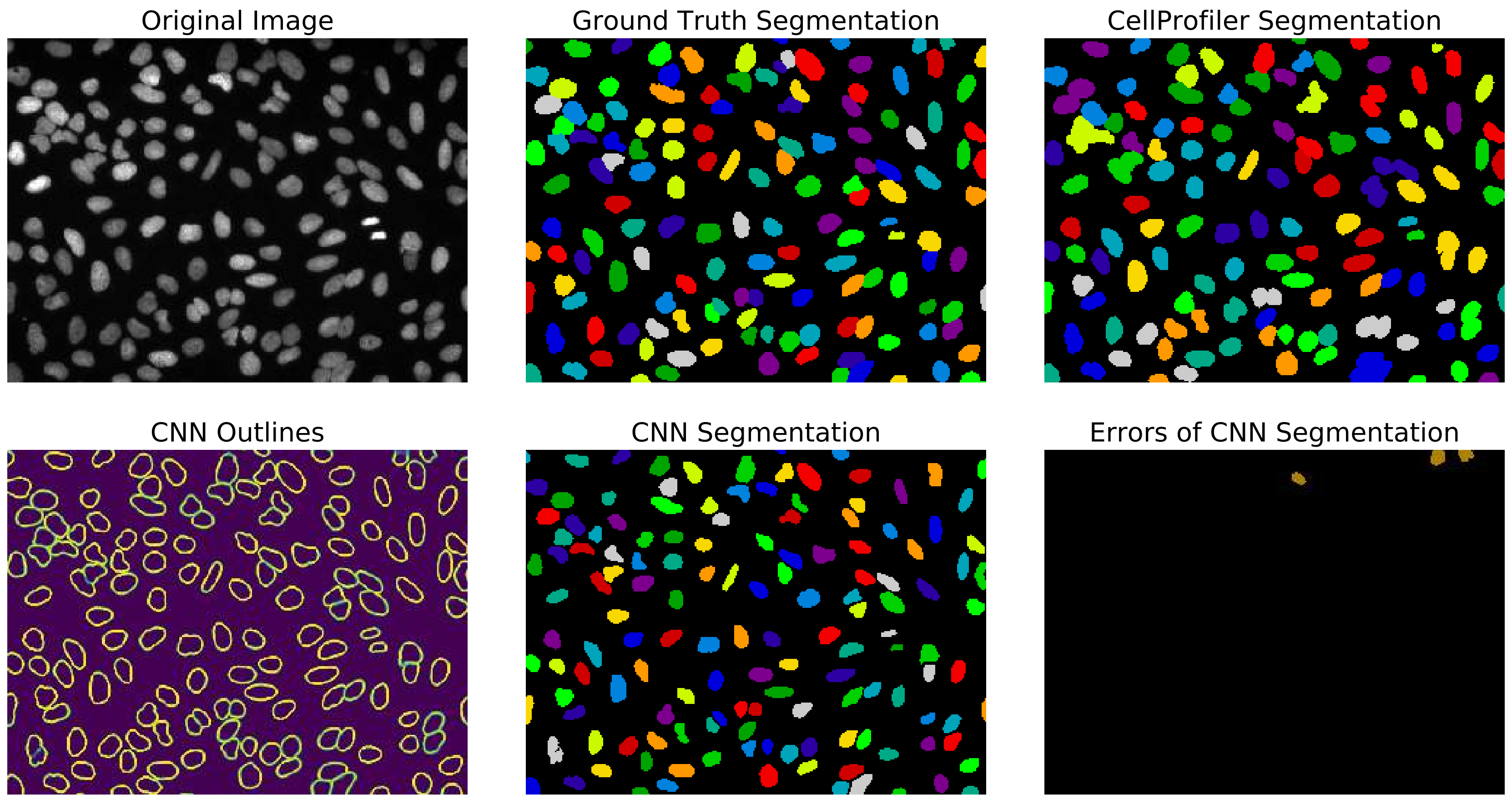

Original Image
Ground Truth Segmentation
CellProfiler Segmentation
CNN Outlines
CNN Segmentation
Errors of CNN Segmentation

Figure 38: Comparison between segmentations obtained with CellProfiler and our best deep learning model for an example image (BBBC022, plate 20646, well C07, site 5).

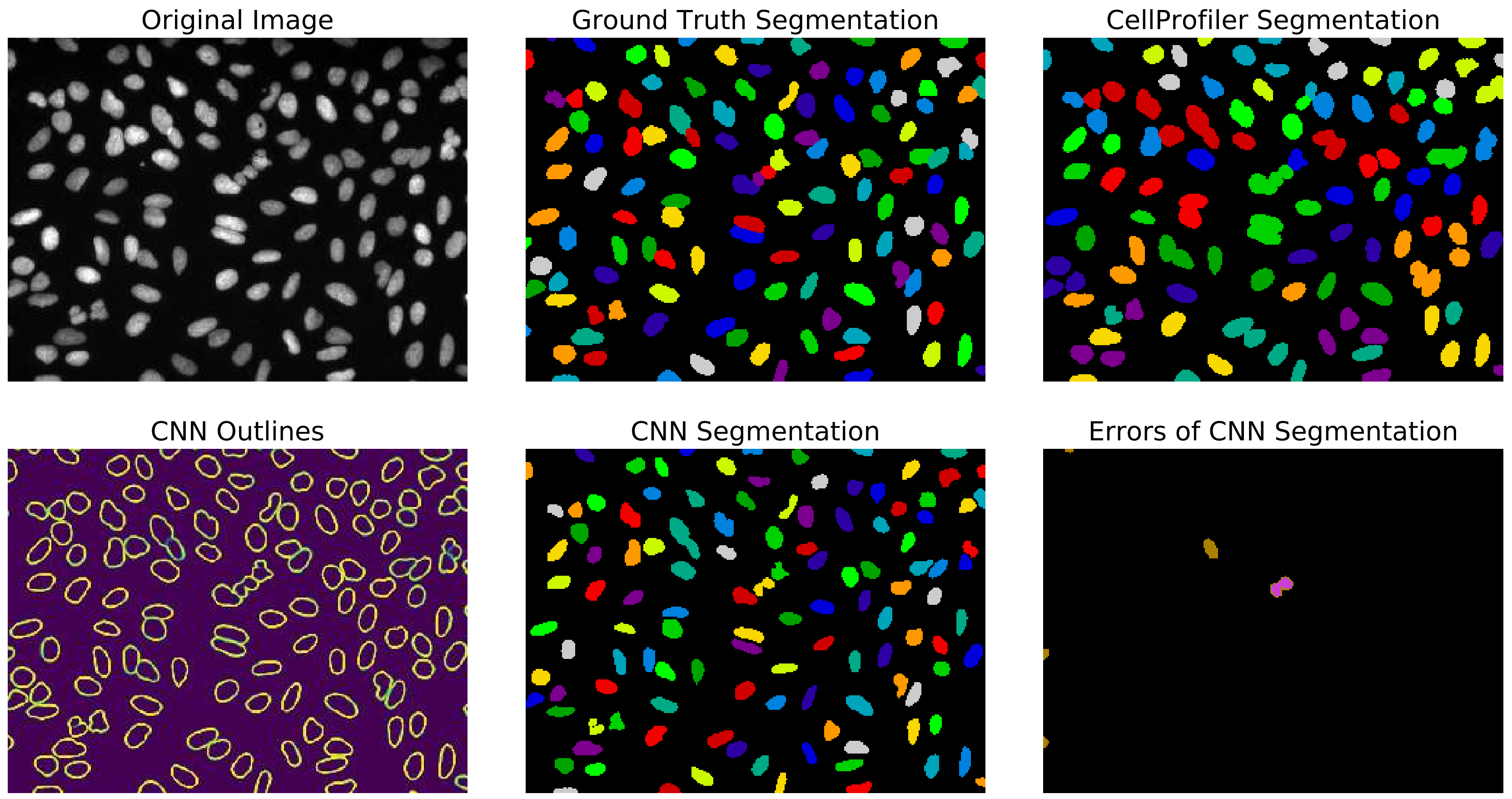

Original Image
Ground Truth Segmentation
CellProfiler Segmentation
CNN Outlines
CNN Segmentation
Errors of CNN Segmentation

Figure 39: Comparison between segmentations obtained with CellProfiler and our best deep learning model for an example image (BBBC022, plate 20592, well C14, site 8).

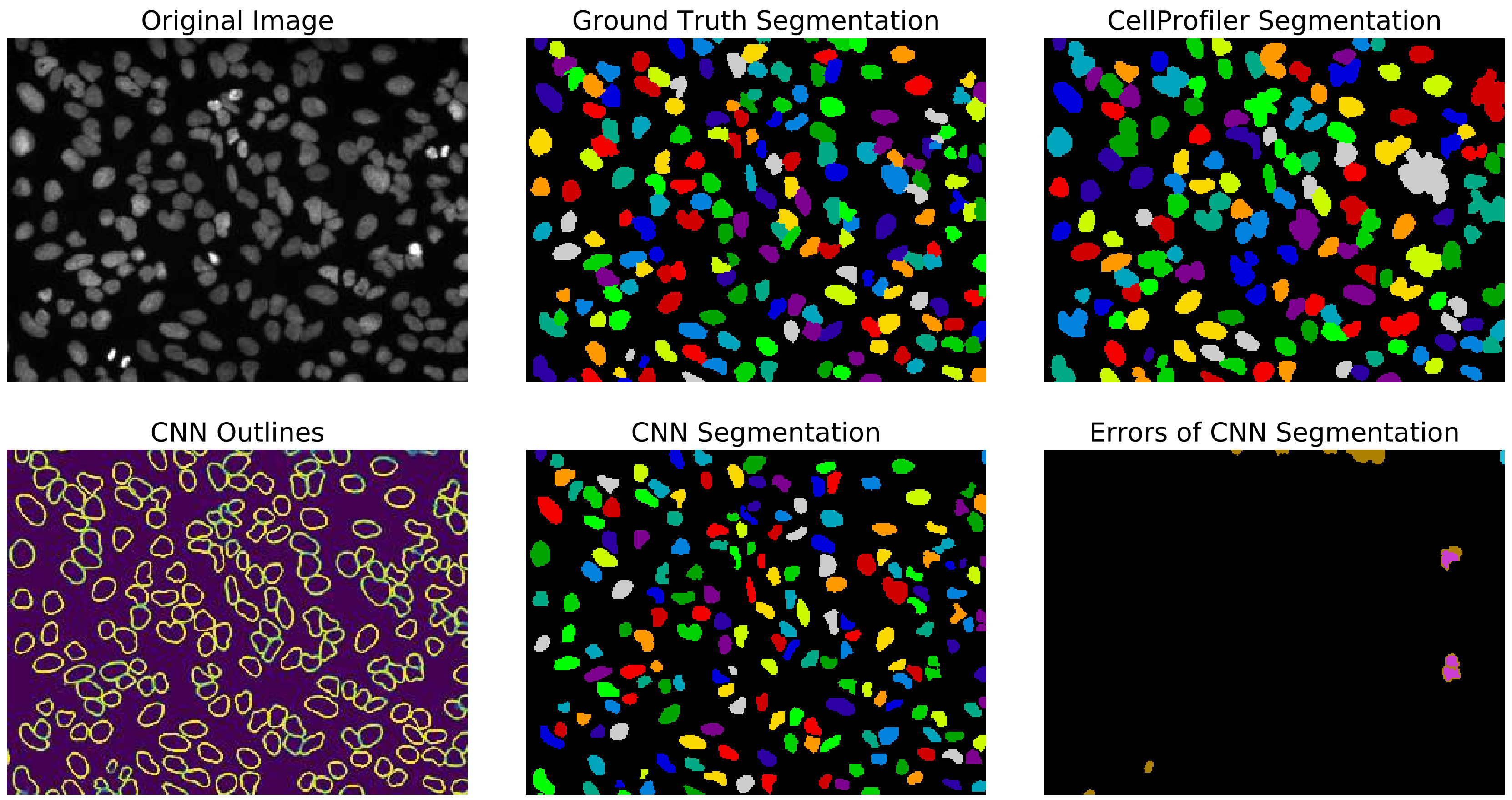

Original Image
Ground Truth Segmentation
CellProfiler Segmentation
CNN Outlines
CNN Segmentation
Errors of CNN Segmentation

Figure 40: Comparison between segmentations obtained with CellProfiler and our best deep learning model for an example image (BBBC022, plate 20596, well I12, site 1).

widths.

Even though we could slightly increase the 3-class model's performance by random sampling, it still performs worse than the boundary detection model with random sampling. The change of boundary width did not have any effect on the performance of the 3-class model.
We thus conclude that training the CNN with boundary formulation and using data augmentation gives us the best results. Figures 38, 39 and 40 show a comparison between segmentations obtained by CellProfiler and our CNN.

4.5.3 Optimizing the Running Time

Experiment Setup. A downside of current segmentation approaches is the computational complexity of the watershed algorithm. As a result, segmenting images with CellProfiler takes multiple seconds. In this section, we show that our model is significantly faster.

The following experiments are conducted on the best-performing version of the model, the network in boundary formulation trained with boundaries of size 4 and random sampling of the training data.

All experiments were carried out on a Nvidia GeForce GTX TITAN X. The IPython notebook used for these experiments can be found in the GitHub repository in the directory `experiments/timing/`.

- **Prediction and Post Processing**

  We tested the computation time required for the prediction of the probability map given an input image. As the post processing step required to get a segmentation from outline probability map is rather involved, the computation time spent on this is measured as well.
  For the post processing, as explained in section 4.4.2, the output of the model is thresholded at $\frac{1}{2}$ and connected components in non-boundary pixels are filtered by their intensity.

Setup	Resolution	Prediction Time	Post Processing Time
Full images	696x530	$9.7 \times 10^{-2}s \pm 3.4 \times 10^{-3}s$	$1.1 \times 10^{-1}s \pm 3.8 \times 10^{-2}s$
Patches	256x256	$2.0 \times 10^{-2}s \pm 8.3 \times 10^{-4}s$	$9.4 \times 10^{-3}s \pm 2.9 \times 10^{-3}s$

Table 4: Running times for model prediction (outlines) and post processing. Mean time measurements and standard deviations are given. One can clearly observe the higher variance in the post processing timing as the post processing is dependent on what is displayed in the image. On the other hand, the prediction time is independent from what is displayed in the image and thus has a low variance. This experiment was carried out with 1000 repetitions and random samples of images across the test data set.

- **GPU Usage**

  To ensure the efficient usage of the GPU during prediction, we analyzed the effect of changing the number of samples that are processed simultaneously. For an efficient usage of processing power and memory, it is required that the processing time needed per image does not change with the number of samples processed simultaneously. Also, this should hold true when exceeding the batch size with which predictions are made.

Results.

- **Prediction and Post Processing** As the model is run in two different setups – training with patches and prediction with full images – we measure computation time for both cases. The time measurement results are summarized in table 4.

  One can observe that both prediction and post processing takes significantly longer for larger images. For patches, the post processing time is much lower than the prediction time. For full images, prediction and post processing roughly take equally long. The relative durations of prediction and post processing are depicted in figure 41 for full images and figure 42 for patches. Compared to CellProfiler, which needs multiple seconds for the segmentation of a single full-resolution image,

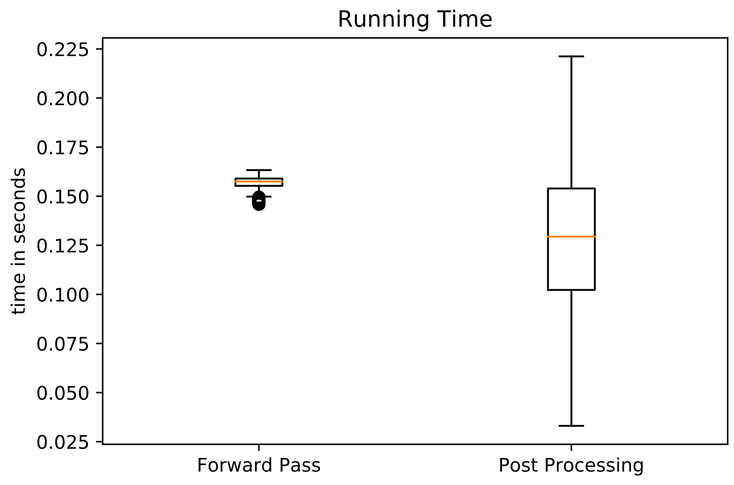


Figure 41: Running time for images of size 696x520. The high standard deviation in the post processing time is visualized. N=1000.

we achieve running times of less than half a second with deep learning models.

- **GPU Usage**

  The prediction time as a function of the size of the processed data set is visualized in figure 43. We did not observe a significant increase or decrease in the prediction time per image with increasing data set size, even when increasing the data set size beyond the batch size, which is expected for an efficient usage of the computing power of the GPU.

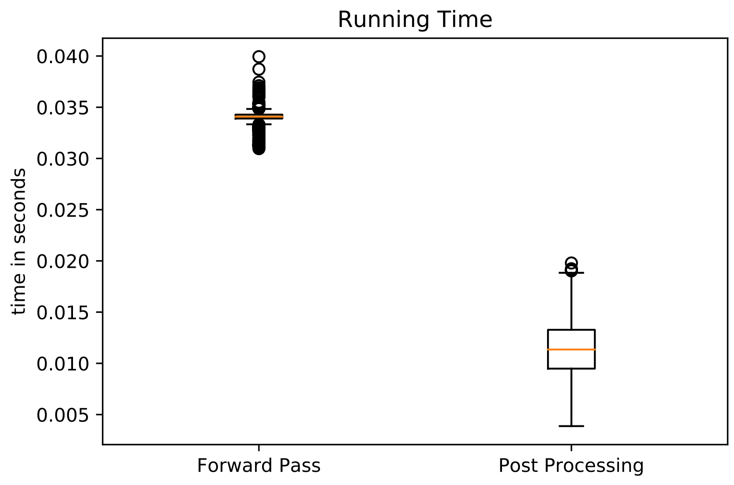


Figure 42: Running time for patches of size 256x256. One can observe that post processing time is much shorter than prediction time in this experiment. N=1000.

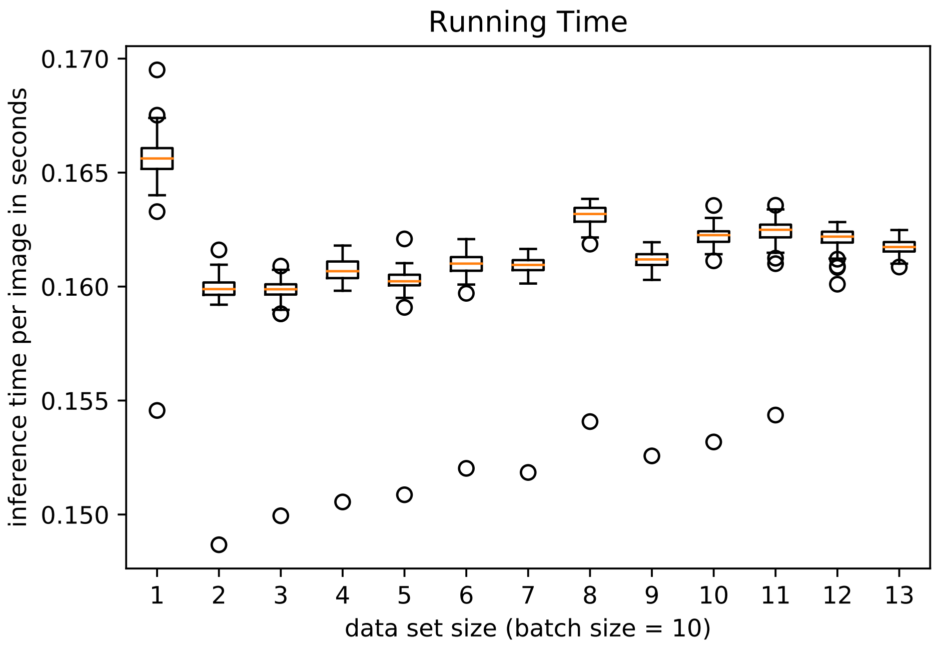


Figure 43: Running time for whole images depending on number of samples simultaneously processed by the GPU. The running time measurements were repeated 100 times for each data set size.

5 Conclusion

Improved Method for Quantifying Segmentations. A method commonly used in object detection was used for comparing segmentations for fluorescence microscopy images. It is an object-based measure and thus much more relevant to biologists than a pixel-based comparison. We used this method to evaluate the performance of different deep learning models for segmentations.

Declumping Nuclei with Semantic Segmentation. Clumped nuclei are common in fluorescence microscopy and thus separating touching nuclei is crucial for segmentation algorithms. By introducing an artificial boundary class between touching nuclei and between nuclei and background pixels, we could separate clumped nuclei using semantic segmentation with convolutional neural networks. The fully convolutional networks we used here allow end-to-end training and can do predictions on an arbitrary input size.

Outperforming Current Segmentation Methods. The neural networks we trained easily surpass the performance of seeded watershed algorithms used with a CellProfiler segmentation pipeline designed by experts. We could lower the share of overdetected nuclei from 5% to 2% and the share of undetected nuclei from 15% to 6%. Table 5 summarizes the performance of all our experiments.

Model	**Training Data** (if applicable)	**Precision**	**Recall**	F_1
CellProfiler		95%	85%	0.90
3-Class CNN (boundary boost)	CellProfiler Outputs	94%	82%	0.88
3-Class CNN	Manual Annotations	95%	87%	0.91
Boundary CNN	Manual Annotations	96%	88%	0.92
Boundary CNN (data augmentation)	Manual Annotations	98%	94%	0.96

Table 5: Summary of the performance metrics of all models presented in section 3.3.

6 Outlook

To further increase the performance of deep neural networks for segmenting nuclei in fluorescence microscopy images, networks need to yield segmentations with even fewer merge errors. Thus, more research should be conducted on cost functions that put more weight on areas where clumped nuclei are depicted in the input image.

In addition, using object detection approaches or instance segmentation instead of semantic segmentation as discussed in this thesis, might yield better results. However, these models are significantly harder to implement and train.

Before algorithms based on neural networks can be made available in software tools, their robustness should be tested. We suggest testing the segmentation performance on different cell types, input image sizes, and microscopy magnifications.

The usability of segmentation algorithms can further be increased as well. Even though the presented method does not require parameter tuning, the user either has to provide annotated images and train a convolutional network, or use a network with corresponding weights trained by others. Both approaches currently require experience with deep learning models. A service that hosts various deep learning models and provides segmentations for a wide variety of images might significantly increase the usage of deep convolutional neural networks for segmentation of fluorescence microscopy images.

References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-Scale machine learning on heterogeneous distributed systems.

[2] Ashburn, T. T., and Thor, K. B. Drug repositioning: identifying and developing new uses for existing drugs. *Nat. Rev. Drug Discov. 3*, 8 (Aug. 2004), 673–683.

[3] Bates, M., Huang, B., Dempsey, G. T., and Zhuang, X. Multicolor super-resolution imaging with photo-switchable fluorescent probes. *Science 317*, 5845 (21 Sept. 2007), 1749–1753.

[4] Blasi, T., Hennig, H., Summers, H. D., Theis, F. J., Cerveira, J., Patterson, J. O., Davies, D., Filby, A., Carpenter, A. E., and Rees, P. Label-free cell cycle analysis for high-throughput imaging flow cytometry. *Nat. Commun. 7* (7 Jan. 2016), 10256.

[5] Bray, M.-A., Fraser, A. N., Hasaka, T. P., and Carpenter, A. E. Workflow and metrics for image quality control in large-scale high-content screens. *J. Biomol. Screen. 17*, 2 (Feb. 2012), 266–274.

[6] Bray, M.-A., Singh, S., Han, H., Davis, C. T., Borgeson, B., Hartland, C., Kost-Alimova, M., Gustafsdottir, S. M., Gibson, C. C., and Carpenter, A. E. Cell painting, a high-content image-based assay for morphological profiling using multiplexed fluorescent dyes. *Nat. Protoc. 11*, 9 (25 Aug. 2016), 1757–1774.

[7] Broad Institute of MIT and Harvard. The drug repurposing hub. `clue.io/repurposing`. Accessed: 2017-6-10.

[8] Buggenthin, F., Marr, C., Schwarzfischer, M., Hoppe, P. S., Hilsenbeck, O., Schroeder, T., and Theis, F. J. An automatic method for robust and fast cell detection in bright field images from high-throughput microscopy. *BMC Bioinformatics 14* (4 Oct. 2013), 297.

[9] Caicedo, J. C., Singh, S., and Carpenter, A. E. Applications in image-based profiling of perturbations. *Curr. Opin. Biotechnol. 39* (June 2016), 134–142.

[10] Canny, J. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell. 8*, 6 (June 1986), 679–698.

[11] 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., Golland, P., and Sabatini, D. M. CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome Biol. 7*, 10 (31 Oct. 2006), R100.

[12] Chollet, F. Keras. `github.com/fchollet/keras`, 2015.

[13] Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., and Ronneberger, O. 3D U-Net: Learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016* (17 Oct. 2016), S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, and W. Wells, Eds., Lecture Notes in Computer Science, Springer International Publishing, pp. 424–432.

[14] Corsello, S. M., Bittker, J. A., Liu, Z., Gould, J., McCarren, P., Hirschman, J. E., Johnston, S. E., Vrcic, A., Wong, B., Khan, M., Asiedu, J., Narayan, R., Mader, C. C., Subramanian, A., and Golub, T. R. The drug repurposing hub: a next-generation drug library and information resource. *Nat. Med. 23*, 4 (7 Apr. 2017), 405–408.

[15] Dao, D., Fraser, A. N., Hung, J., Ljosa, V., Singh, S., and Carpenter, A. E. CellProfiler analyst: interactive data exploration, analysis and classification of large biological image sets. *Bioinformatics 32*, 20 (15 Oct. 2016), 3210–3212.

[16] Eulenberg, P. Extraction of morphological features with artificial neural networks and their relation to the kadanoff renormalization group. Master's Thesis, 30 Dec. 2016.

[17] Eulenberg, P., Koehler, N., Blasi, T., Filby, A., Carpenter, A. E., Rees, P., Theis, F. J., and Alexander Wolf, F. Deep learning for imaging flow cytometry: Cell cycle analysis of jurkat cells. 17 Oct. 2016.

[18] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The pascal visual object classes (VOC) challenge. *Int. J. Comput. Vis. 88*, 2 (1 June 2010), 303–338.

[19] Feng, Y., Mitchison, T. J., Bender, A., Young, D. W., and Tallarico, J. A. Multi-parameter phenotypic profiling: using cellular effects to characterize small-molecule compounds. *Nat. Rev. Drug Discov. 8*, 7 (July 2009), 567–578.

[20] Friedman, J., Hastie, T., and Tibshirani, R. *The elements of statistical learning*, vol. 1. Springer series in statistics Springer, Berlin, 2001.

[21] Fulwyler, M. J. Electronic separation of biological cells by volume. *Science 150*, 3698 (12 Nov. 1965), 910–911.

[22] Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., and Garcia-Rodriguez, J. A review on deep learning techniques applied to semantic segmentation.

[23] GIMP. GNU image manipulation program. `www.gimp.org`.

[24] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning.* MIT Press, 10 Nov. 2016.

[25] Goodman, A. Annotation tool for fluorescence microscopy images.

[26] Gustafsdottir, S. M., Ljosa, V., Sokolnicki, K. L., Anthony Wilson, J., Walpita, D., Kemp, M. M., Petri Seiler, K., Carrel, H. A., Golub, T. R., Schreiber, S. L., Clemons, P. A., Carpenter, A. E., and Shamji, A. F. Multiplex cytological profiling assay to measure diverse cellular states. *PLoS One 8*, 12 (2 Dec. 2013), e80999.

[27] Hariharan, B., Arbel, P., Girshick, R., and Malik, J. Simultaneous detection and segmentation.

[28] He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask R-CNN.

[29] He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1026–1034.

[30] Hennig, H., Rees, P., Blasi, T., Kamentsky, L., Hung, J., Dao, D., Carpenter, A. E., and Filby, A. An open-source solution for advanced imaging flow cytometry data analysis using machine learning. *Methods 112* (1 Jan. 2017), 201–210.

[31] Hilsenbeck, O., Schwarzfischer, M., Loeffler, D., Dimopoulos, S., Hastreiter, S., Marr, C., Theis, F. J., and Schroeder, T. fastER: a user-friendly tool for ultrafast and robust cell segmentation in large-scale microscopy. *Bioinformatics* (22 Feb. 2017).

[32] Hinton, G. CSC321, neural networks for machine learning, lecture slides.

[33] Huang, B., Wang, W., Bates, M., and Zhuang, X. Three-dimensional super-resolution imaging by stochastic optical reconstruction microscopy. *Science 319*, 5864 (8 Feb. 2008), 810–813.

[34] Hunter, J. D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng. 9*, 3 (1 May 2007), 90–95.

[35] Ioffe, S., and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift.

[36] James, G., Witten, D., Hastie, T., and Tibshirani, R. *An Introduction to Statistical Learning: with Applications in R.* Springer Texts in Statistics. Springer New York, 2013.

[37] Jones, T. R., Kang, I. H., Wheeler, D. B., Lindquist, R. A., Papallo, A., Sabatini, D. M., Golland, P., and Carpenter, A. E. CellProfiler analyst: data exploration and analysis software for complex image-based screens. *BMC Bioinformatics 9* (15 Nov. 2008), 482.

[38] Köhler, N. Automatic measurement of the ejection fraction of the human heart with deep learning algorithms on the basis of magnetic resonance imaging. Master's Thesis, 3 Mar. 2017.

[39] Kroiss, M. Using deep neural networks to predict the lineage choice of hematopoietic stem cells from Time-Lapse microscopy images. Master's Thesis, 24 June 2014.

[40] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature 521*, 7553 (28 May 2015), 436–444.

[41] Li, F.-F. CS231n: Convolutional neural networks for visual recognition, lecture notes.

[42] Liang, X., Wei, Y., Shen, X., Yang, J., Lin, L., and Yan, S. Proposal-free network for instance-level object segmentation. *arXiv preprint arXiv:* (2015).

[43] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Lawrence Zitnick, C., and Dollár, P. Microsoft COCO: Common objects in context.

[44] Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., van der Laak, J. A. W. M., van Ginneken, B., and Sánchez, C. I. A survey on deep learning in medical image analysis.

[45] Ljosa, V., Sokolnicki, K. L., and Carpenter, A. E. Annotated high-throughput microscopy image sets for validation. *Nat. Methods 9*, 7 (28 June 2012), 637.

[46] Loo, L.-H., Wu, L. F., and Altschuler, S. J. Image-based multivariate profiling of drug responses from single cells. *Nat. Methods 4*, 5 (May 2007), 445–453.

[47] Malpica, N., de Solórzano, C. O., Vaquero, J. J., Santos, A., Vallcorba, I., García-Sagredo, J. M., and del Pozo, F. Applying watershed algorithms to the segmentation of clustered nuclei. *Cytometry 28*, 4 (1 Aug. 1997), 289–297.

[48] Mc Kinney, W. Data structures for statistical computing in python.

[49] McDonnell, L. A., and Heeren, R. M. A. Imaging mass spectrometry. *Mass Spectrom. Rev. 26*, 4 (July 2007), 606–643.

[50] Molecular Devices. High content imaging devices from molecular devices. `www.moleculardevices.com/systems/high-content-imaging`. Accessed: 2017-6-24.

[51] N. Moga, A., Cramariuc, B., and Gabbouj, M. Parallel watershed transformation algorithms for image segmentation. *Parallel Comput. 24*, 14 (1 Dec. 1998), 1981–2001.

[52] Nair, V., and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (2010), pp. 807–814.

[53] Neverova, N., Luc, P., Couprie, C., Verbeek, J., and LeCun, Y. Predicting deeper into the future of semantic segmentation.

[54] Ng, A. CS 229: Machine learning, lecture notes.

[55] Nichols, A. High content screening as a screening tool in drug discovery. *Methods Mol. Biol. 356* (2007), 379–387.

[56] Noh, H., Hong, S., and Han, B. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1520–1528.

[57] PerkinElmer. Fluorescence microscopes from PerkinElmer. `www.perkinelmer.co.uk/category/high-content-screening-instruments-microscopes`. Accessed: 2017-6-20.

[58] Quan, T. M., Hilderbrand, D. G. C., and Jeong, W.-K. FusionNet: A deep fully residual convolutional neural network for image segmentation in connectomics.

[59] Rizzo, M. L. *Statistical Computing with R.* CRC Press, 15 Nov. 2007.

[60] Rohban, M. H., Singh, S., Wu, X., Berthet, J. B., Bray, M.-A., Shrestha, Y., Varelas, X., Boehm, J. S., and Carpenter, A. E. Systematic morphological profiling of human gene and allele function via cell painting. *Elife 6* (18 Mar. 2017).

[61] Ronneberger, O., Fischer, P., and Brox, T. U-Net: Convolutional networks for biomedical image segmentation.

[62] Roundtable on Translating Genomic-Based Research for Health, Board on Health Sciences Policy, and Institute of Medicine. *Drug Repurposing and Repositioning: Workshop Summary.* National Academies Press (US), Washington (DC), 30 May 2014.

[63] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis. 115*, 3 (1 Dec. 2015), 211–252.

[64] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. OverFeat: Integrated recognition, localization and detection using convolutional networks.

[65] Shelhamer, E., Long, J., and Darrell, T. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell. 39*, 4 (Apr. 2017), 640–651.

[66] Singh, S. Cytominer GitHub repository. `www.github.com/cytomining/cytominer`. Accessed: 2017-6-15.

[67] Theis, F. J. Statistical learning, lecture notes, 3 Feb. 2016.

[68] Thermo Fisher Scientific. Fluorescence SpectraViewer. `www.thermofisher.com`. Accessed: 2017-6-24.

[69] van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T., and scikit-image contributors. scikit-image: image processing in python. *PeerJ 2* (19 June 2014), e453.

[70] 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. Deep learning automates the quantitative analysis of individual cells in Live-Cell imaging experiments. *PLoS Comput. Biol. 12*, 11 (Nov. 2016), e1005177.

[71] Walt, S. v. d., Colbert, S. C., and Varoquaux, G. The NumPy array: A structure for efficient numerical computation. *Comput. Sci. Eng. 13*, 2 (1 Mar. 2011), 22–30.

[72] Wolf, C., and Jolion, J.-M. Object count/area graphs for the evaluation of object detection and segmentation algorithms. *IJDAR 8*, 4 (1 Sept. 2006), 280–296.