UNIVERSITY OF TARTU

Institute of Computer Science

Software Engineering Curriculum

Sten-Oliver Salumaa

Convolutional Neural Networks for Cellular Segmentation

Master's thesis (30 ECTS)

Supervisors: PhD Leopold Parts

MSc Dmytro Fishman

Tartu 2018

Convolutional Neural Networks for Cellular Segmentation

Abstract:

There is a persistent demand for work-assisting algorithms in industry. Using present-day technology, it is possible to free people from mundane tasks so they can concentrate on work that requires human skills and flexibility. Deep learning methods can complete tasks that were previously considered hard or even impossible for machines.

One example of this kind of task is segmenting brightfield microscopy images of cells. This work is needed mostly in biomedical laboratories and pharmaceutical companies that must analyse and quantify vast amounts of image data. Current workflows avoid useful brightfield imagery because automatic industrial solutions for segmentation do not exist. Manual annotation is very challenging and time consuming, even for experienced professionals.

The goal of the thesis is to demonstrate that deep learning can solve the task of segmenting challenging brightfield images. The developed solution opens new experimental approaches, saving time and resources for biomedical scientists across the globe.

Keywords:

convolutional neural networks, deep learning, segmentation, biomedicine, image analysis

CERCS: P170

Konvolutsionaalsed tehisnärvivõrgud rakupiltide segmenteerimiseks

Lühikokkuvõte:

Üha enam lülituvad algoritmid töö tegemisel väärtuslikeks abimeesteks. Tänapäevase tehnoloogia toel on võimalik inimesed vabastada lihtsamatest ülesannetest, et nad saaksid keskenduda teistele töödele, mis on arvuti jaoks keerulised. Üks abistavatest tehnoloogiatest on süvaõpe. Selle abil suudavad arvutid lahendada ülesandeid, mida varem peeti arvutite jaoks raskeks või koguni võimatuks.

Üheks selliseks tööks on erevälja rakupiltide segmenteerimine. Seda on tarvis eelkõige biomeditsiinilaborites ning ravimifirmades, mis peavad suurt hulka mikroskoobipilte analüüsima ja kvantifitseerima. Praegused tööprotsessid väldivad ereväljapiltide kasutust, kuna nende segmenteerimiseks pole tööstuslikke lahendusi ning käsitsi töötlemine on keerukas ja aeganõudev.

Magistritöö eesmärgiks on tõestada, et masinõpe suudab lahendada seni masinatele raskete ereväljapiltide segmenteerimise ülesande. Loodud lahendus aitab teadlastel üle maailma katsetada teisi uurimismeetodeid ja säästa palju aega.

Võtmesõnad:

konvolutsionaalsed närvivõrgud, süvaõpe, segmenteerimine, biomeditsiin, pildianalüüs

CERCS: P170

Table of Contents

1. Terms and Notations

Convolutional Neural Network (CNN) is a class of artificial neural networks which work on a feed-forward principle and employ convolution operations; often applied to analysing visual data.

Brightfield microscopy is the simplest form of microscopy where light is either passed through or reflected off a specimen [1].

Fluorescence microscopy is a form of microscopy in which fluorescent molecules are used to mark certain structures, e.g. nucleic acids, which can then be viewed with a dedicated microscope [2].

Fluorescent dying (staining) is a process of applying fluorescent molecules to bind to parts of cells. Fluorescent chemicals start emitting light when excited by a certain wavelength of light.

Nucleus is a cell organelle that is found in most living eukaryotic cells, directing their growth, metabolism, reproduction, and functioning [3].

Biomarker is a biological molecule used as a marker for a substance or process of interest. Biomarkers used in fluorescent microscopy include dyes or stains.

Graphics Processing Unit (GPU) is a specialized hardware component in computers that helps to accelerate image processing. Nowadays it is also used in high-performance computing to speed up calculations on matrices by using parallelization of computations.

Microplate is an experiment container for accelerating high throughput microscopy. It is made of plastic and can contain 24, 96, 384, or 1536 subcompartments in which cells are put into. Instead of changing samples after every imaging in a microscope, up to thousands of images of different cells in microplate compartments can be captured in an automated way.

Semantic segmentation is pixelwise distinguishing of different object classes in an image. When there are multiple instances of one object class they are considered equivalent and assigned the same instance labels.

Instance segmentation is pixelwise distinguishing different objects and their classes in an image. When there are multiple instances of the same object class they are considered separate and given different instance labels.

Compound library is a collection of chemicals in standardized containers optimized for high-throughput screening.

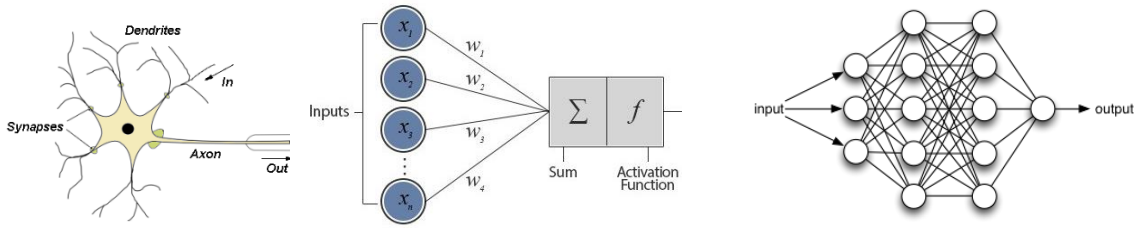
Screening in biomedicine is a strategy of making experiments and observing their results in high throughput using standardized reagents and protocols.

Transfer learning is initializing artificial neural network with weights that have been trained on other data. For example, using weights from a model trained on COCO dataset [4] to initialise a network that segments cellular microscopy images.

2. Introduction

2.1 History of neural networks in image domain

The basic principles of neural networks and deep learning are known for decades. The initial idea behind modern solutions dates to 1958, when Frank Rosenblatt first modelled a computational system inspired by biology. He defined the Perceptron, a computational approximation of how a human neuron works (Figure 1a, Figure 1b). The calculation core receives several inputs, multiplies them with weights, sums them to obtain a total input, and yields an output after passing through an activation function. Perceptrons can iteratively find the best weight configuration to solve a linear classification problem. [5]



a) b) c)

Figure 1. a) Living neuron gets its signals from dendrites through synaptic gates, then processes the signal inside the cell and produces its output in axons [6]. b) Artificial neuron receives inputs $x_1 ... x_n$ from connections (dendrites) which pass through weight multiplication (synapses), are summed (cell body). Then an activation function is applied to the result to generate the output. [7] c) A neural network can be formed by connecting neurons' outputs to other neurons' inputs [8].

The history of hierarchical neural networks in visual perception begins in 1961, when scientists at Harvard Medical School published a paper about mammalian vision mechanisms. They discovered the hierarchical structure of visual cortex where simple neurons are activated when seeing images of straight lines or dots, whereas complex neurons become active when eyes detect more intricate shapes, for example triangles. [9] It gave researchers initial ideas of decision systems in multilevel visual perception.

The first artificial neural network that resembles a convolutional neural network (CNN) is Neocognitron from 1982. It featured a multi-layer design and convolution operations but lacked an efficient supervised learning algorithm [10]. In 1989, CNNs started using

backpropagation-based training and were already capable of recognizing hand-written digits [11].

While development of these systems continued, there were no disrupting applications emerging. Two main obstacles restricted the development of image analysis solutions in particular: lack of computing power and scarcity of labelled training data. With more data and increasingly powerful computers in 2000s, the preconditions started becoming favourable for the success of neural networks for images.

The revolutional era of deep learning started with AlexNet CNN architecture outperforming all previous competitors on the ImageNet image classification challenge in 2012 by a large margin. The task involves classifying images into 1000 different object categories (Figure 2). Krizhevsky et al. successfully trained AlexNet on 1.2 million images on two powerful GPUs, and proved that deep learning can tackle complex problems with successful outcomes. [12]

Figure 2. An example of AlexNet classification on ImageNet dataset [12]. The correct label is shown under each image. The words and bar charts below show the most probable predicted labels by AlexNet for each image with red bars indicating the final output, and size of the bars reflecting the posterior probability assigned by the model.

2.2 Present day in deep learning on images

Modern deep learning is fuelled by large labelled datasets. Training data is now more abundant than ever thanks to increased public, academic, and industrial interest. Furthermore, global data science challenges push the field forward by creating and opening

training data to the public [4, 13]. Due to the abundancy of data, researchers can develop and try out their ideas in numerous different settings.

The impact of neural networks in image processing domain is extensive, and CNNs are the most frequently used architectures. Open source deep learning solutions for extracting information from video frames for autonomous cars already exist [14], and Tesla's, the American electric car manufacturer's lead of AI is talking about integrating deep learning into conventional software solutions as *Software 2.0* [15]. Tesla are in the process of substituting computer vision algorithms in their vehicles with lightweight neural networks.

With increasing computing power there has been an emergence of very deep neural networks with several millions of parameters [16]. Despite taking more time to train and predict they result in state-of-the-art performance in ImageNet classification challenge [17]. What is more, it is possible to use these pretrained networks for other projects using transfer learning [18]. With a fraction of time taking to train the original model it is feasible to slightly retrain the network or a part of it for a different task. For example, a model trained on ImageNet [17] dataset can be partially retrained to classify dog breeds on images.

Deep learning is also used in biology and medicine. For instance, neural networks have been employed to automatically classify protein localisation in yeast cells. They can be utilised to learn about gene function and describe quantitative state of cells. [19] Furthermore, deep learning has been applied on detection of referable diabetic retinopathy disease, a human eye disorder, from images. The model could diagnose the condition nearly as well as human specialists. [20]

2.3 Cellular segmentation challenge

Modern drug discovery relies on vast number of experiments to test candidate compound libraries. Thousands to millions of human cell populations are grown in microplates to test the effect of these potential drugs. The cell populations range in origin, as well as in disease stage or state. Within one experiment, a variety of chemicals and their doses are tested. These assays are used to monitor whether the compounds cure or kill the diseased cells, whether the healthy cells react to compounds in some unwanted way, or whether the compound has any effect at all.

Human mind is subjective when it comes to images. One person might say that the cells from experiment X had shown larger effect than experiment Y. But when another experiment is conducted by a different person making an opposite claim then the two are not easily and objectively comparable. Therefore, there is a need to make imaging experiments quantifiable and thus comparable in an unbiased way.

Perhaps the most important step in image analysis is segmenting and annotating structures (Figure 3). Using segmented images offers a wide range of information: nuclei count, number of cells, median and average of nuclei's 2D projection area, and intensity distribution in case of fluorescent images [21]. Nuclear shape and morphology are in many cases indicative of cell well-being. By identifying separate nuclei, we could extract any of abnormal shapes or sizes, cluster them by some metric, monitor their migration and intercellular sociology [22]. Furthermore, extracting segmented regions of interest (ROIs) from the input image lays groundwork for cell phenotyping in subsequent processing steps [23].

Quantifying and analysing images by annotating them cell-by-cell on a pixel level is easily doable if there are only a few images to deal with. The problem starts to get unmanageable when the data load increases, however.

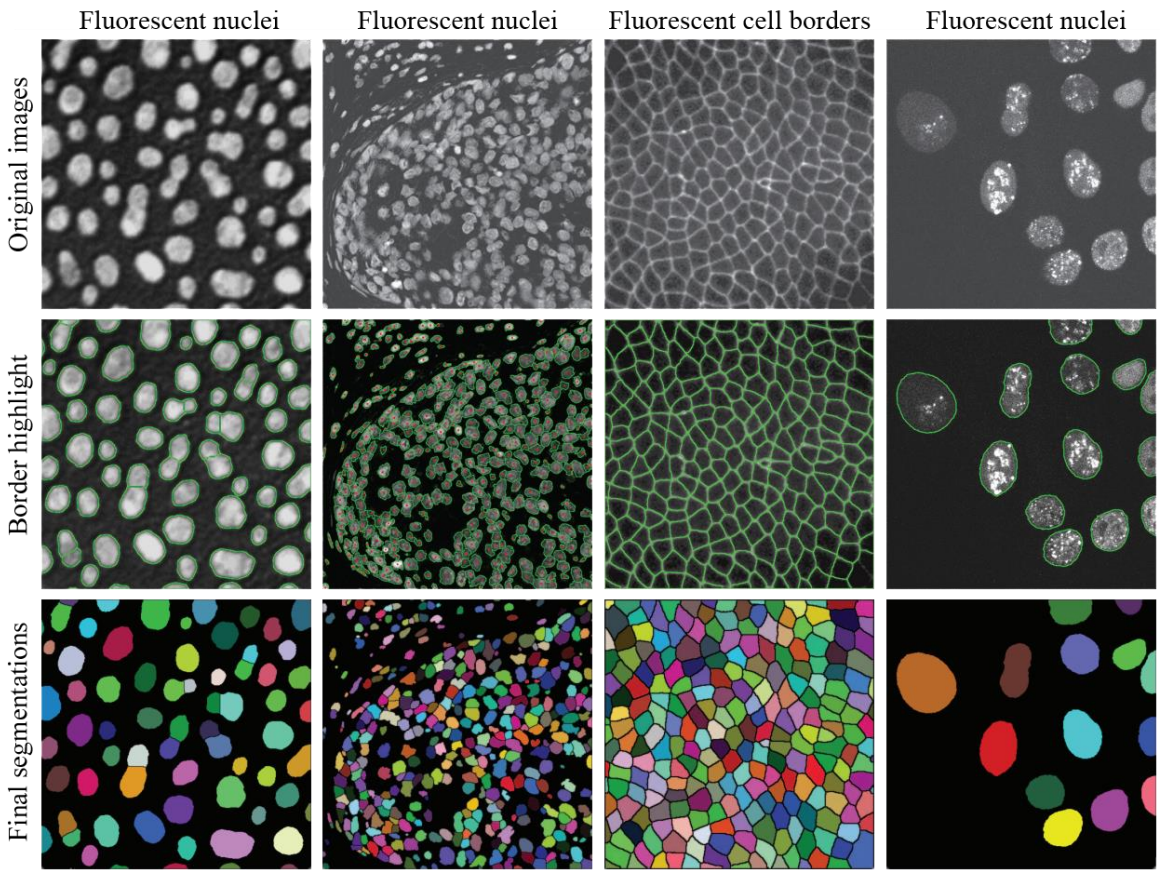


Figure 3. Fluorescent nuclei and cell segmentation. Columns represent image modalities and rows show different segmentation processing steps. [22]

Modern high-throughput microscopy systems can take hundreds of thousands of images of biological experiments using automation [21]. As the amount of data increases, the biologists find themselves short-handed when it comes to analysing the results of the experiments. Various software solutions exist to help the scientists with this task. For example, CellProfiler [24] is a free and open-source software package designed to help biologists quantify a sample phenotypic profile from image data. Furthermore, Harmony [25] software by PerkinElmer is a more advanced commercial product that includes many additional features.

2.4 The value of brightfield images

Most cellular-level imaging solutions need biomarkers to make nuclei or other cellular structures stand out in the images and to make segmentation possible for people and software. Even if fluorescent dyes are used, a user is still needed to adjust the settings for

the segmentation algorithms on an image batch basis. The latter can take time from a few minutes up to several hours.

Using fluorescent staining is helpful for segmentation but unfortunately it has drawbacks. Firstly, inserting biomarkers into cells takes time. Cell plate samples must be prepared specially for imaging. It generates additional work for laboratory specialists and the dyes take a while to end up in the intended region. For example, biomarkers used for nuclear staining in fluorescence microscopy need time to be absorbed by the cell and the nucleus. [26]

Secondly, cell staining changes the cell state. During fluorescent sample preparation cells are in most cases fixed into place with a chemical, then their membranes are opened with another substance. Finally, the fluorescent molecule is introduced to the sample. These steps kill the cell and further experiments on the same plate are not possible. Although it is more challenging to do, sometimes fluorescent dyes are inserted into living cells. The stain's interference with cell's inner chemistry changes its state and as such renders further experiment investigation inaccurate or difficult. [27]

Thirdly, fluorescent stains experience an effect called photobleaching. When intense single-wavelength light is shined upon the samples the dye light emitting intensity quickly fades due to excited molecules reacting to intracellular chemicals. Because of this fluorescent samples can be viewed for only a certain time. [28]

Finally, using a fluorescent dye comes with an opportunity cost of using the same emission spectrum for another readout. Four, and in some specialised setups, six different colours can be distinguished using standard fluorescent molecules. If information from some stains could be replaced by another readout, the channel could be used to extract richer information from the cells.

Brightfield microscopy images are made using natural light without biomarkers (Figure 4). They are hard to annotate using classic image analysis software, since the cells are nearly transparent and low contrast [29]. In this type of image is hard to find nuclei from even for a human.

However, brightfield images are easier to acquire and much faster to prepare since no extra chemicals are added. For the same reason, they are cheaper to make because no additional reagents are used. In addition, possibility of using brightfield microscopy for quantitative image analysis enables using same cells for time-series analysis without needing extra

samples to account for cells that would be killed with fluorescent pre-processing. Without stress-inducing additional dye molecules we can observe cells in a more natural environment and get better results from tests.

Automated industrial brightfield image quantification has the potential to save thousands of hours of unnecessary work. Thus, pharmaceutical companies are interested in a solution that helps to extract more information from brightfield images. This would save their employees' time and thus bring monetary savings and increase companies' efficiency. Additional cost reduction would come from using less reagents and cell samples.

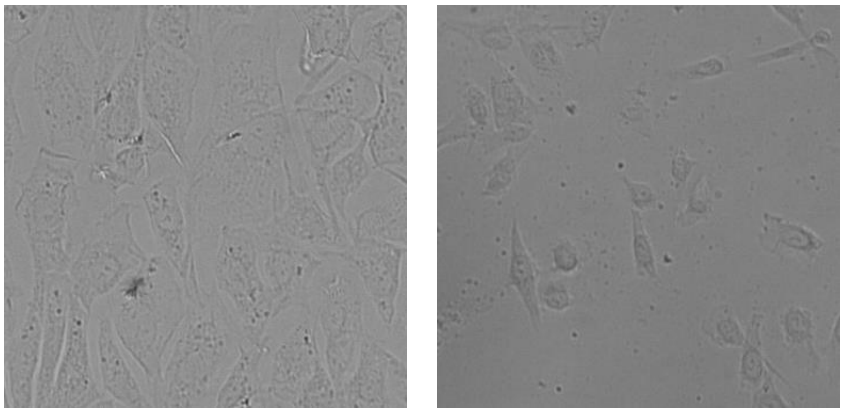


Figure 4. Cropped examples of brightfield images of cells.

University of Tartu was approached by PerkinElmer [30], a global corporation focused on human and environmental health, with aspirations on automatic brightfield segmentation. Their head of image processing was interested whether deep learning solutions could further develop the field of cellular annotation. The main goal of this cooperation is to integrate machine learning models into Harmony software as additional segmentation tools.

The aim of this thesis is to create and explore proof-of-concept deep learning models that could solve the nuclei segmentation problem in brightfield images. In addition, research is focused on keeping computational cost low since PerkinElmer's objective is to run the algorithms on desktop computers once the models are incorporated to their segmentation software.

I would like to thank Leopold Parts who took me aboard this project, showed me life and world-class science in England, provided me with advice, support, and a great sense of humour all the way through this journey. My big thanks go Dmytro Fishman who sparked my interest in this project with his great presentation skills. He was always ready to help me whenever I needed it and stayed easy-going and great to work with from start until the end.

3. Methodology

3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of artificial neural networks used in deep learning whose main calculation operation is convolving filters over matrices. This paragraph introduces main principles behind its work mechanisms.

Convolutions

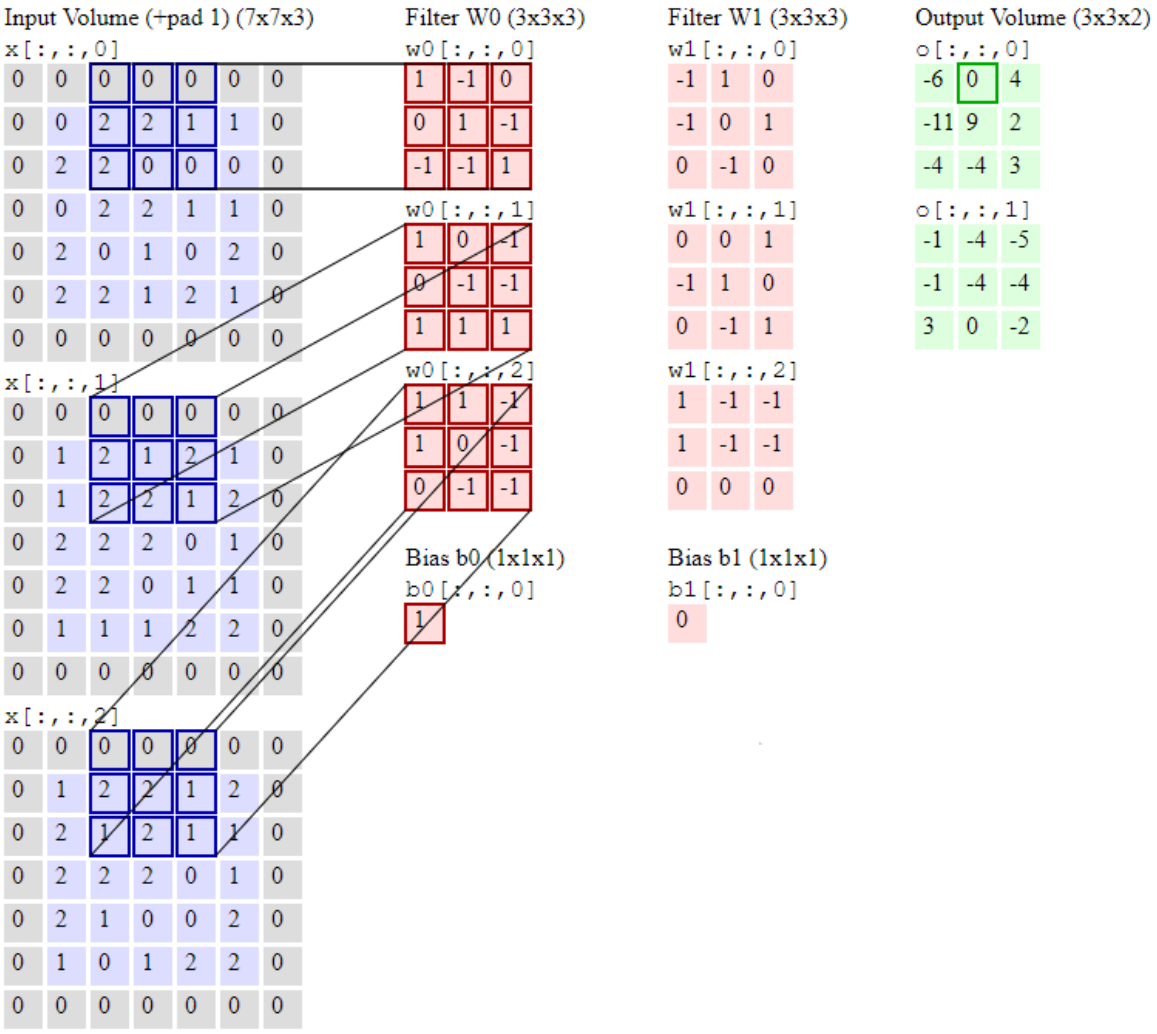


Figure 5. CNN convolution operation [31]. Light blue matrices represent three colour channels on the input image. Red matrices are filters that are convolved over the input volume with a step size of two. Current filter location is shown by dark blue 3x3 patches. Green matrices are the calculated feature maps. Dark green square on the output illustrates the spatial location where the dot product result between input and filter is currently being written.

Images are matrices with each individual element in the matrix representing one pixel intensity in one acquisition channel (usually RGB). Figure 5 shows an input image of 5x5 pixels with three channels – red, blue, green. It is extended with zeros on its edges (zero-padded) to 7x7 to adjust the final output size as given in Equations 1 and 2. The network convolves a filter, which is another matrix, over each of the input channels with a certain step size called stride. At each of the filters' locations a dot product between the filters and the coinciding input arrays are calculated. Now all the dot product results are summed and a bias is added to produce a final score in the corresponding location in the output volume, also known as the feature map. The convolution operation rolls the filter across the input image horizontally and vertically, until all the input image has been convolved with a filter. In this case there are two separate filters that produce two feature maps as outputs.

The output shape W_2 x H_2 x D_2 depending on the input shape W_1 x H_1 x D_1 can be calculated with the following equations where F is the filter's side length, P the amount of zero-padding on the input image's edges, K the number of filters, and S the stride.

$$W_2 = \frac{W_1 - F + 2P}{S} + 1 \tag{1}$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1 \tag{2}$$

$$D_2 = K \tag{3}$$

Filters

Filters in Figure 5 are called neurons in CNNs. Each number in the filter matrix is a weight to be learned during training time. Every filter has the same number of channels as its immediate input. For example, input image with three channels of shape 256x256x3 pixels can have filters of 5x5x3 sliding and computing dot products over it. Filters in hidden layers (Figure 6) have as many channels as the layers before them had filters. In each spatial location the dot products for each input channel are calculated and summed together.

Neural networks are comprised of layers. The two filters with their biases in Figure 5 can be referred to as a layer without an activation function. Figure 6 shows how a simplified

three-layer network is comprised. It has an input layer with three nodes which in our case represent the channels of the input image, hidden layers with red nodes which represent inner filters, and green output nodes correspond to final filters that output predicted feature maps.

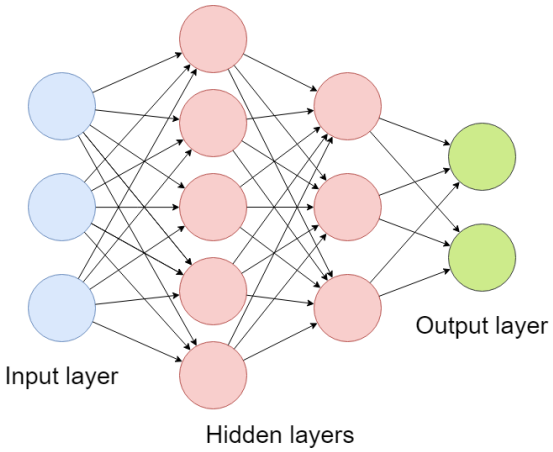


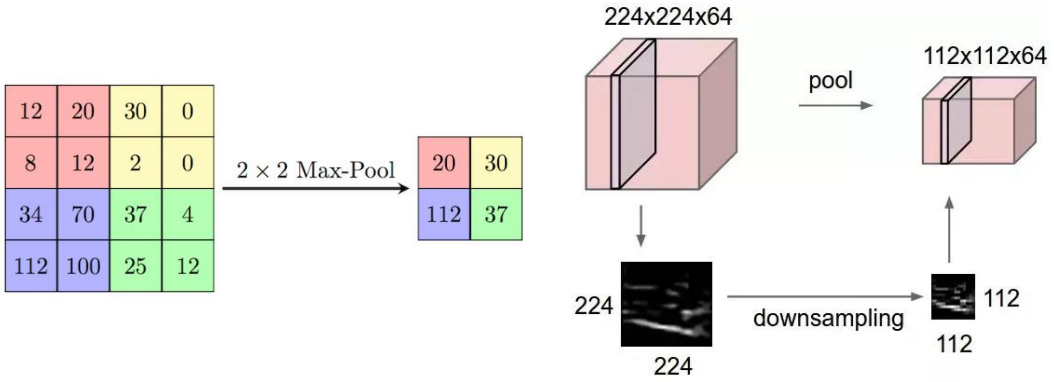
Figure 6. Three-layer neural network with two hidden layers and one output layer.

Activation function

In Figure 5, we can see outputs of one convolutional layer. To introduce non-linearity into the network, activation functions are used. For this work, Rectified Linear Units (ReLUs) were used as activations on the outputs of a convolutional layers. They have proved to be more efficient for neural networks' inside layers or hidden layers than sigmoid or tanh activations in many cases [32]. ReLUs calculate $max(0, x)$ where x represents the output matrix after convolution. After this operation all negative numbers in feature map matrices will be assigned zeros and other values are left unchanged. Applying activation function gives us activation maps, one for each filter. These are passed on to the next layer in the network.

Max pooling

Max pooling is an operation that downscales the input volume by a factor. It can be considered as a special case of filters. Instead of producing the dot product with the underlying matrix it instead takes the area at hand and only extracts one value, the maximum from it. For example, if a 2x2 region in the input matrix consists of numbers 1, 4, 2, and 3 then in the output of this pooling operation we get only number 4. Pooling filters move like convolution filters but in most cases the stride value is equal to pooling window side length. Figure 7 illustrates max-pooling operation with a filter size of 2x2 and stride 2.



a) b)

Figure 7. Max-pooling operation on a) pictorial and b) image-based examples using a pool size of 2x2 and stride of 2 [33].

The biggest advantage of pooling is reduction of computational cost. Images are data-heavy and thus are slow to convolve through in big networks. Downsampling them inside the network allows to introduce more filters while not suffering from training slowing down.

Dropout layer

Dropout layers have been proved to help to reduce overfitting in CNNs [34]. Dropout works by randomly not using a neuron during training time with a fixed turn-off probability. It discourages some neurons becoming too dominating in the neural network which helps the network to generalize better to unseen test data.

Output layer

This is the last layer of a neural network. In classification models it is usually a fully connected layer. This means that every activation map value from previous layer has a weighted connection to each output layer node. There are as many output nodes as there are possible classes.

In case of segmentation models, the last layer is usually convolutional. If one channel segmentation map is required then the last layer is a single convolution filter, usually of size 1x1 with a stride of one. This calculates the weighted sum of previous layer's output maps and reduces the image depth to one, outputting a single channel segmentation.

3.2 Training

The purpose of training is to minimize the difference between network's output and ground truth data. It is also required that the model would produce accurate results when presented with data that was not used during training phase.

Backpropagation

Backpropagation is the main algorithm behind the training process of a CNN. It was first used in a CNN in 1989 [11] for handwritten character and digit recognition and has since been the backbone for training modern architectures. In a network each neuron has its weights which influence the final output. Backpropagation algorithm compares the network output with the expected output or ground truth and calculates an error measure. From that result it backtracks the change or gradient of every step in the computational process using the chain rule partial derivatives. In the end of this process we can see how much each value in filters affected the final loss of the network. Based on these gradients, there is a parameter or weight update conducted depending on the step size and the optimizer method of the network. After taking a step, the whole process of forward-feeding an image batch through the network is repeated until some stopping criteria is met.

Optimizer

Optimizer is the algorithm that decides how the parameter update takes place. The simplest option is to use the standard Stochastic Gradient Descent that updates individual values in each filter based on their respective gradients and a constant learning rate. In current work, a more modern optimizer Adam [35] is used. Although recent work has suggested that Adam optimizer might not be the best option for overall generalization and convergence performance [36], in practice it converges faster than some alternatives [35].

Loss function

Loss function is a way of calculating the error between ground truth and the network's output. In classification problems we can compare the network's output class probability vector with ground truth. In segmentation tasks we can measure the differences between each pixel in output and ground truth masks.

Batch training

Usually the dataset size is bigger than available memory capacity and hence cannot be passed through a network all at once. Batch training means only using a subset of the data at a time while doing a forward pass during training.

Stopping criterion

There are two main reasons to stop training process. Firstly, the network has reached its smallest error rate and the loss function on validation data does not change anymore. In this case the network has converged. Secondly, the training loss is getting substantially smaller than validation loss. This indicates that the model is overfitting and further training decreases model's performance on unseen data.

3.3 Network architectures tested

Due to restrictions to the thesis format, employed artificial neural networks are described here without in-depth technical details. Literature references for each network are provided for additional information.

DeepCell architecture with patch approach

Microscopy image segmentation can be transformed into a classification task, thus making it possible to leverage on the vast existing experience and information available. Each pixel in an image can be thought of as a centre of its surrounding area. We can extract thousands of small square crops from an image this way and label them based on whether the central pixel represents a nucleus or background. A neural network can be trained to classify the central pixel of each of these patches using its surroundings as context information (Figure 8). Training data is kept balanced by selecting the same number of patches from different classes. In our case, around 1,000 nucleus and background labelled patches are selected from each image for training, resulting in 3878812 patches in the training dataset and 969624 in the validation dataset.

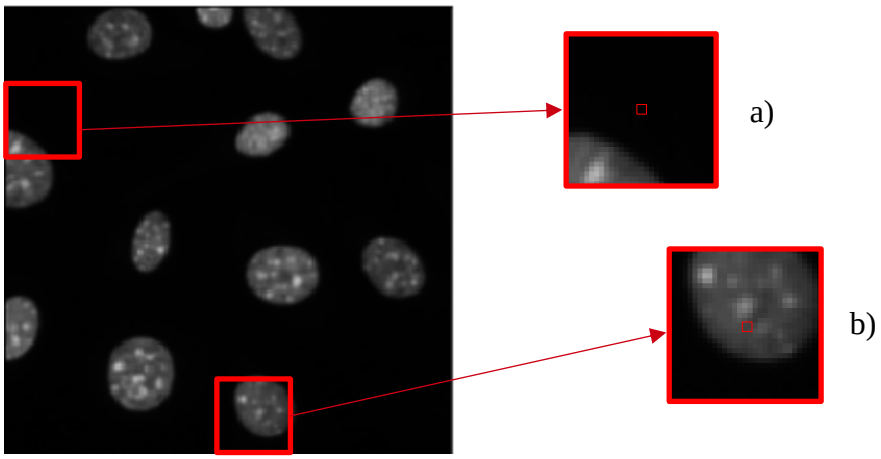


Figure 8. Breaking down an image to separate 31x31 pixel patches for training. Patch a) gets a label of a background pixel while b) is assigned a nucleus pixel label.

During prediction, a patch is generated for every pixel of the input image. To complete the surrounding area for pixels near the edges of the image, image is extended by reflecting its pixels near its edges. An example of reflective padding is illustrated in Figure 11. Each created patch is passed through the network to be classified. After classification, the segmentation is generated by taking each central pixel from the classified patches and labelling it with a certain colour depending on its class. For example, white is used for nuclei and black for background class. Central pixels are put back into their original locations and a fully segmented image is returned.

For an image of size 1080x1080 we must create 1,166,400 patches and classify each of them. Due to the computation-heavy nature of this approach, it is relatively slow to segment the whole image.

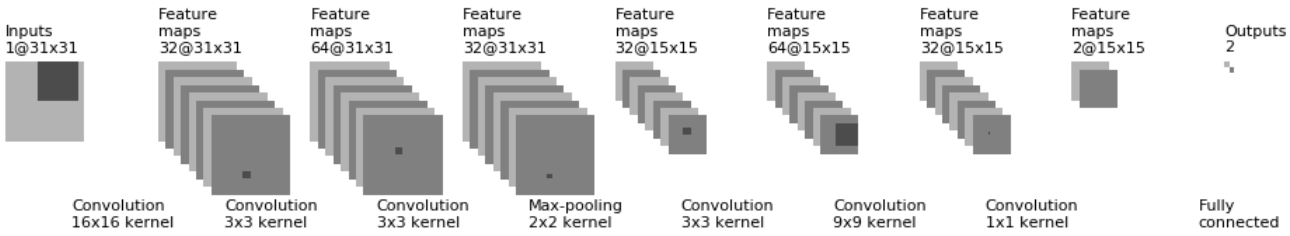


Figure 9. DeepCell architecture [37] for classifying each pixel in the image into nucleus or background. The rectangles denote individual feature maps, and inputs are passed through the network from left to right, with operations defined below the image applied at each stage. The number and dimensions of feature maps are provided above individual layers.

The DeepCell network features six convolutional layers with one fully connected layer in the end to produce a classification output for the patch (Figure 9).

U-net architecture

The U-net architecture [38] was specifically developed for biomedical image segmentation. It has a smaller computational cost due to lacking redundant calculations of overlapping patches compared to the patch based approach. Instead of predicting the image pixel-by-pixel with patches, entire image is forwarded through a CNN only once to get a segmentation. In the output there will be a ready-made segmentation map for the image. In this thesis, an implementation of U-net [38] is used to test whether whole image segmentation in a cellular setting would be feasible.

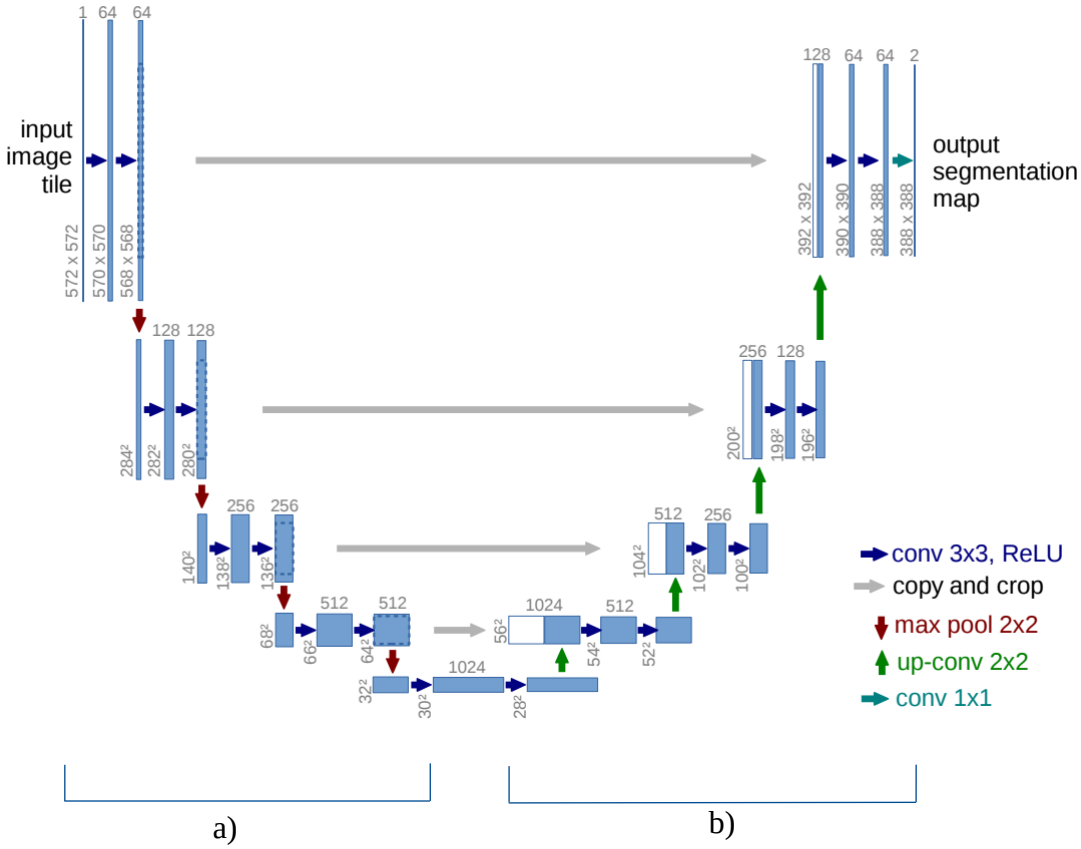


Figure 10. U-net architecture [38]. Blue boxes represent calculated feature maps from the passing image. White boxes correspond to feature maps that were calculated in a) the encoder path and concatenated with b) the decoder path. The up-conv layers represent a special convolution operation that increases the activation map size. The network holds in total 23 convolutional layers.

This architecture consists of an encoder and a decoder paths (Figure 10). Skip connections are built at different levels to pass on activation maps from the encoder path to decoder

paths. It helps to combine less processed input image information with more processed feature maps.

U-net architecture is only usable with images that are the same size. The original U-net uses image crops of size 572x572 as inputs. The actual image sections are of size 388x388 but need reflective padding on the edges (Figure 11) because after every convolution layer the feature map dimensions diminish when not using padding on the edges (Equations 1 and 2). In case of larger inputs not designed for this network, the image can be separated into smaller parts which can be segmented independently and combined. In the original U-net paper multiple passes through the network are required. Our dataset is 1080x1080 pixels and the goal is to pass it through the network as fast as possible. Thus, image cropping is not used.

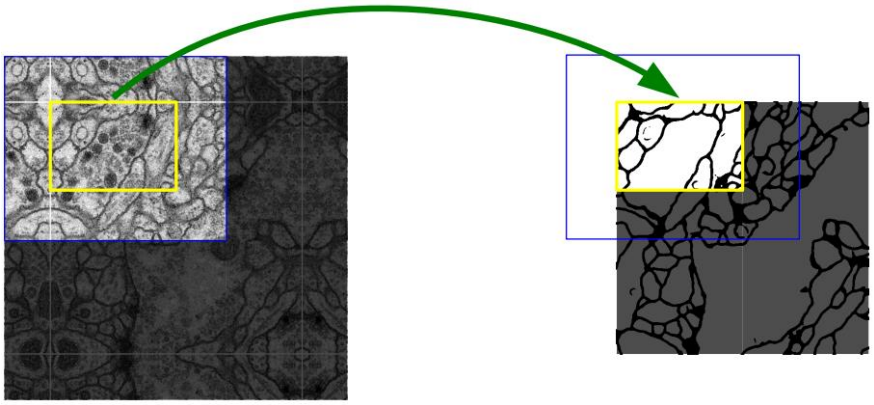


Figure 11. Reflective padding added to input images in the original U-net paper to account for dimension reduction in convolution layers [38]. In our implementation, this is mitigated with Keras library's 'same' padding function which makes sure that the feature map has the same width and height after a convolution layer.

U-net model is changed in implementation compared to the original design in the paper to increase its performance. The original U-net implementation has more than 31 million parameters or weights that are used in the filters during learning and predicting. Segmentation and training times suffer from big models because all these numbers are used for calculations when predicting or training.

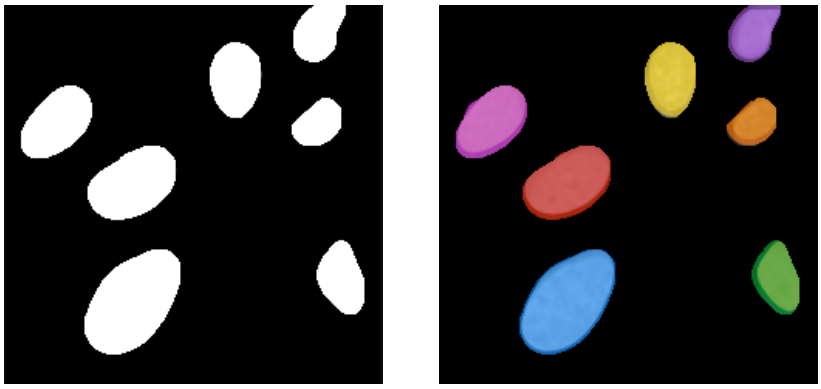
The original U-net architecture was changed to accommodate new data dimensions of 1080x1080 and increase speed. Two additional levels of layers were added to encoder and decoder parts of the network and filter numbers were trimmed on each layer. To further decrease parameters, transposed convolution layers, which were used in the U-net paper, were replaced with upsampling layers that resize feature maps without using learned

parameters. Our final network has 1,219,009 parameters, over 25 times less than the original version. Fewer parameters means smaller model and less calculations which, in turn, leads to higher segmentation speed.

During training of our U-net no transfer learning is applied. The network is trained from scratch on 2016 images and validated on 504 images.

Mask-RCNN architecture

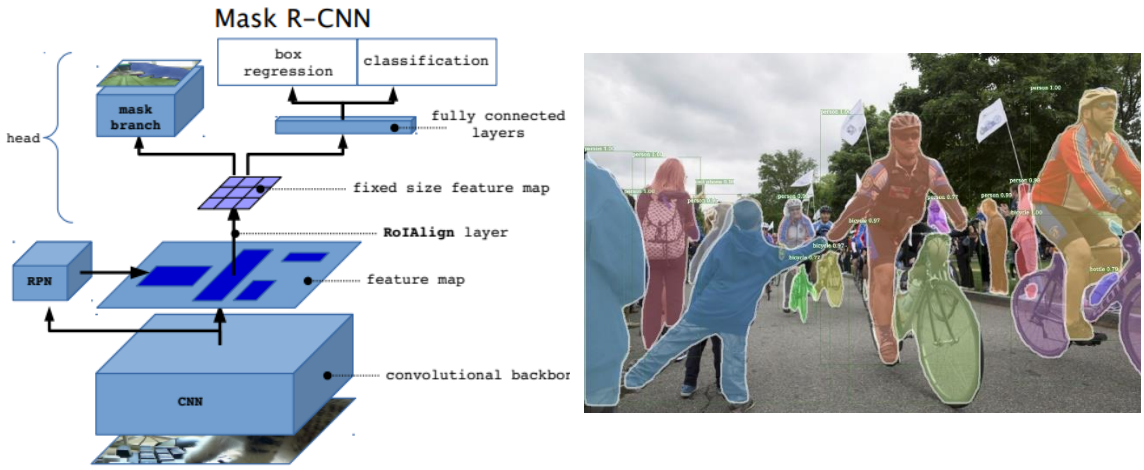
Mask-RCNN can detect objects in the image, segment them, and provide bounding boxes [39]. Since its outputs are already self-contained items (Figure 12b) they are easy to quantify and study on an object-by-object basis further down the image analysis pipeline.



a) b)

Figure 12. Difference between a) semantic and b) instance segmentation. Semantic segmentation assigns each pixel a class label not recognizing separate objects. Instance segmentation treats each nucleus as a distinct object. Images by Dmytro Fishman.

Mask-RCNN relies on three different previously published object detection and classification models: R-CNN [40] , Fast-RCNN [41], and Faster-RCNN [42]. Earlier groundwork was done to find out how well CNNs could be used for object detection and classification. Mask-RCNN added segmentation to detected objects (Figure 13b).

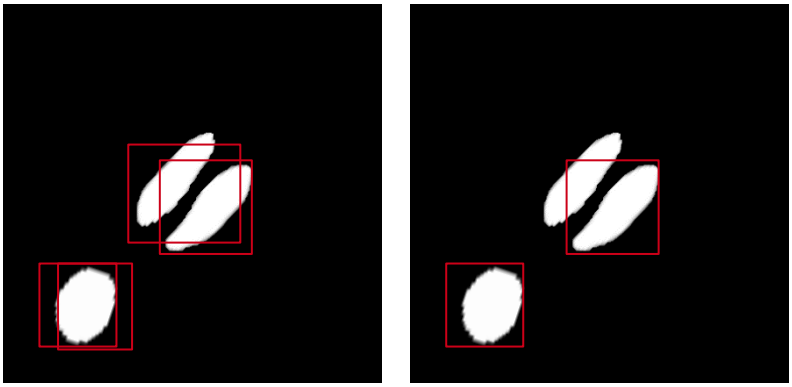


a) b)

Figure 13. Mask-RCNN a) architecture [43] and b) instance segmentation example highlighting individual instances of people and bikes [44].

The architecture (Figure 13a) consists of a convolutional backbone that processes the input image and outputs a feature map on its last layer. In the original paper, ResNet architecture [45] is used in conjunction with Feature Pyramid Network [46] to form the backbone.

A region proposal network generates coordinates for potential objects in the feature map. It also prunes away bounding box proposals which are overlapping (Figure 14) or have low probabilities of being objects. Next, the extracted objects are squashed into a fixed size feature map that can be processed by the mask branch, box regressor, and classifier network (Figure 13a). For every feature map the mask branch with a convolutional network generates a segmentation. Separately, fully connected layers classify and provide bounding box coordinates for the object.



a) b)

Figure 14. A pictorial example of a region proposal network a) proposing and b) removing substantially overlapping bounding boxes. In one case, a nucleus is undetected after bounding box pruning. In the other case, the algorithm avoids double detection. These images are different from actual feature maps since the output on which region proposal network operates in the used implementation is a more abstract feature map with a shape of 32x32x2048 [47].

Current thesis uses a Mask-RCNN implementation by Matterport [48]. Mask-RCNN is a complex architecture, essentially consisting of four separate networks. Due to its intricate design it is challenging to train. A simple end-to-end training of all layers is not the best option in practice. There are numerous techniques how to approach the training regarding which layers to choose. A solution that works well is to first initialize the network backbone with weights pretrained on COCO dataset [4]. Next, it is beneficial to train mask head and fully connected layers, after which the whole network could be trained. To get better results, the whole network can be additionally trained with a lowered learning rate.

3.4 Deep learning software

Deep learning related work on the thesis was conducted using Keras with Tensorflow backend.

Keras

Keras is an open source high-level wrapper library for neural network frameworks. Its main features include user friendliness, modularity, and ease of extensibility [49]. Keras is written in Python which makes for an easy to understand source code. It is one of the most used high-level wrappers for Tensorflow. Developing neural networks is quicker when using it for many commonly used layer declarations. The wrapper provides means to build networks,

load pre-trained weights, pre-process data, augment data, asynchronously feed training data and much more.

Tensorflow

Keras is flexible with regards to using different backend frameworks for carrying out the computation-heavy tasks. One those frameworks is Tensorflow which was used in this thesis.

Tensorflow is meant for high-performance numerical computations across various computing hardware. To achieve its performance levels, it uses optimized code for specific hardware platforms. For instance, on NVIDIA GPUs Tensorflow can use cuDNN library [50] for computation-heavy tasks.

Computation environment

University of Tartu's High Performance Computing Center aims to build and develop the required infrastructure for scientific computing [51]. It serves two GPU nodes, one of which (falcon2) was used for the thesis, and has the following hardware specifications [52]:

- 2 x Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz (48 cores total)
- 256GB RAM
- 5TB of local SSD storage
- 8 x NVIDIA Tesla P100 GPUs with 12GB of VRAM each; only a single GPU was used at a time for training

3.5 Dataset

The data used in this thesis was provided by PerkinElmer under a cooperation contract with the University of Tartu. The dataset consists of 3024 different microplate images of 1080x1080 pixels that contain cells from seven different cell lines: HeLa, HepG2, HT1080, A549, MCF7, NIH-3T3, MDCK. To stain the nuclei, Hoechst 33342 dye was used, which emits fluorescence upon binding double stranded DNA. Each image has two channels (human-visible brightfield; fluorescence) corresponding to different image modalities acquired from the same sample, and one channel with label masks (Figure 15). The ground truth segmentations were generated with existing PerkinElmer’s Acapella software based on the fluorescence channel images.

The dataset is unbalanced with regards to the number of background and nuclei pixels. Considering the whole dataset there are 87.9 % background pixels and 12.1 % nuclei pixels.

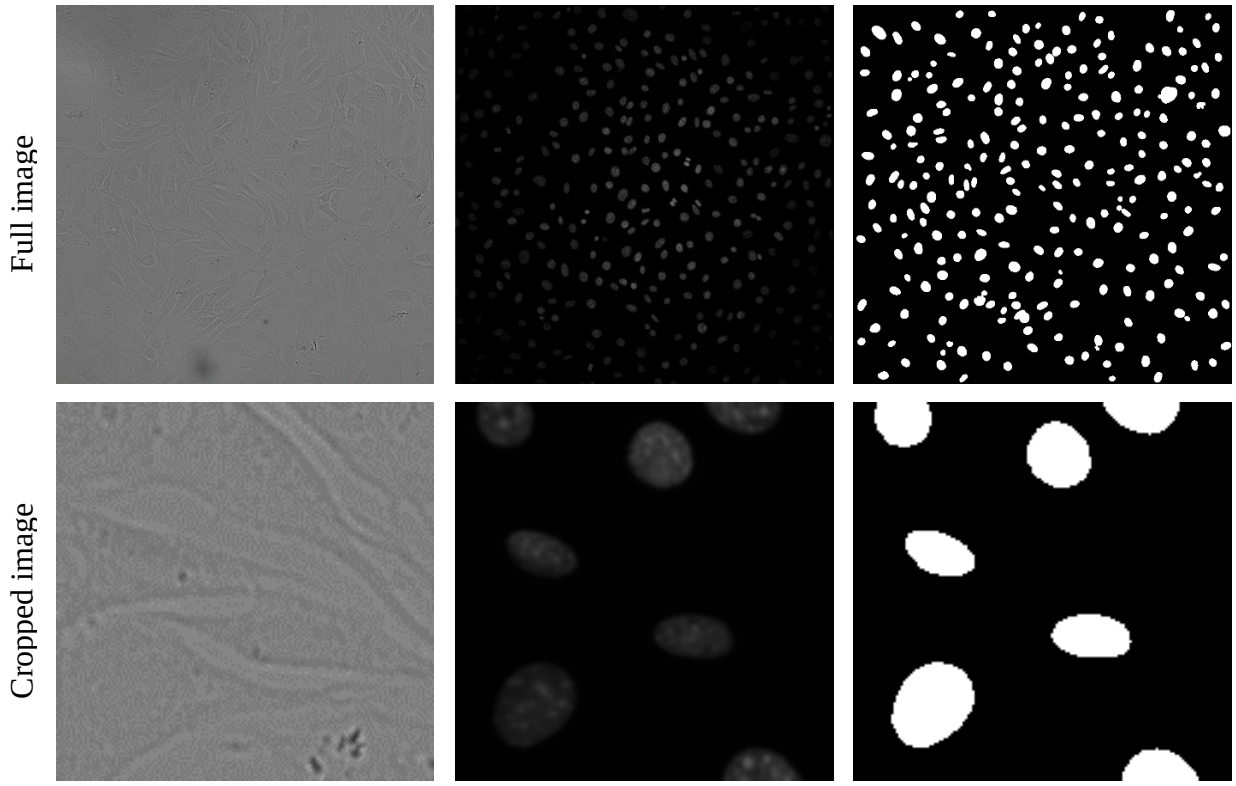


Figure 15. Dataset example of a) brightfield image, b) fluorescence channel on nuclear-stained plate, and c) binary labelled mask of nuclei.

Pre-processing

Some adjustments to the raw dataset are made to increase pipeline speed for training and inference processes. The training, validation, and test image sets are separately concatenated into a Numpy array file and saved on disk. For fluorescent images, pixel intensities are normalized between 0 and 255 to convert to 8-bit format.

Brightfield image pixel values are normalized between 0 and 255 and then contrast is applied using the PIL library's ImageEnhance [53] module in Python. This increases pixel values that are higher and decreases lower ones, hence making cellular structures stand out more. The reasoning behind it is that with the initial normalized images U-net struggled to get past a certain accuracy. After applying pre-processing, our network managed to converge to better validation accuracy.

4. Results

The goal of PerkinElmer is to come up with a solution for segmenting brightfield images. As this modality is more complicated to analyse, fluorescent images are worked on initially to test whether deep learning can solve simpler challenges. We approach the segmentation challenge step by step, testing a range of increasingly complex architectures.

4.1 DeepCell architecture with patch approach

Previous work on fluorescent images was done using DeepCell architecture and patch approach for segmentation by researchers from University of Tartu, Wellcome Sanger Institute and University of Cambridge. DeepCell had proved accurate and relatively simple to implement and train on fluorescent images and was therefore implemented first. Despite having 504 test images available, 200 were used for DeepCell tests due to technical problems. We confirmed that this patch based approach is highly accurate on fluorescent images, with only 2.9% pixel-level error (Table 1).

Table 1. Speed and accuracy of DeepCell architecture on two-class segmentation.

	Fluorescent images	Brightfield images
Segmentation time per image on a GPU	104.5 s	104.2 s
Overall test accuracy on a test set of 200 images	97.1 %	91.8 %
Nucleus pixel accuracy	92.4 %	72.3 %
Background pixel accuracy	97.7 %	94.4 %
Nucleus pixel precision	84.1 %	63.2 %
Nucleus pixel recall	92.4 %	72.3 %

To visually verify the results, example segmentations are provided in Figure 16. There is little to no differences between the ground truth and our segmentations. In addition, when PerkinElmer was generating ground truth they decided to filter out nuclei on the edges of

the image because their algorithm was more uncertain when segmenting those. Our solution can recall nuclei on edges as well as anywhere else in the image. This suggests that our 97.1% accuracy is not higher because in some cases it performs better than the ground truth.

Segmenting time per image is a little over 100 seconds. Spending that much time per image is slow but the speed can be further improved. It was shown by University of Tartu PhD student Daniel Majoral that overlapping convolution calculations are redundant and can be further optimized. He demonstrated that the segmentation time could be shortened to just 2.55 seconds on a GPU.

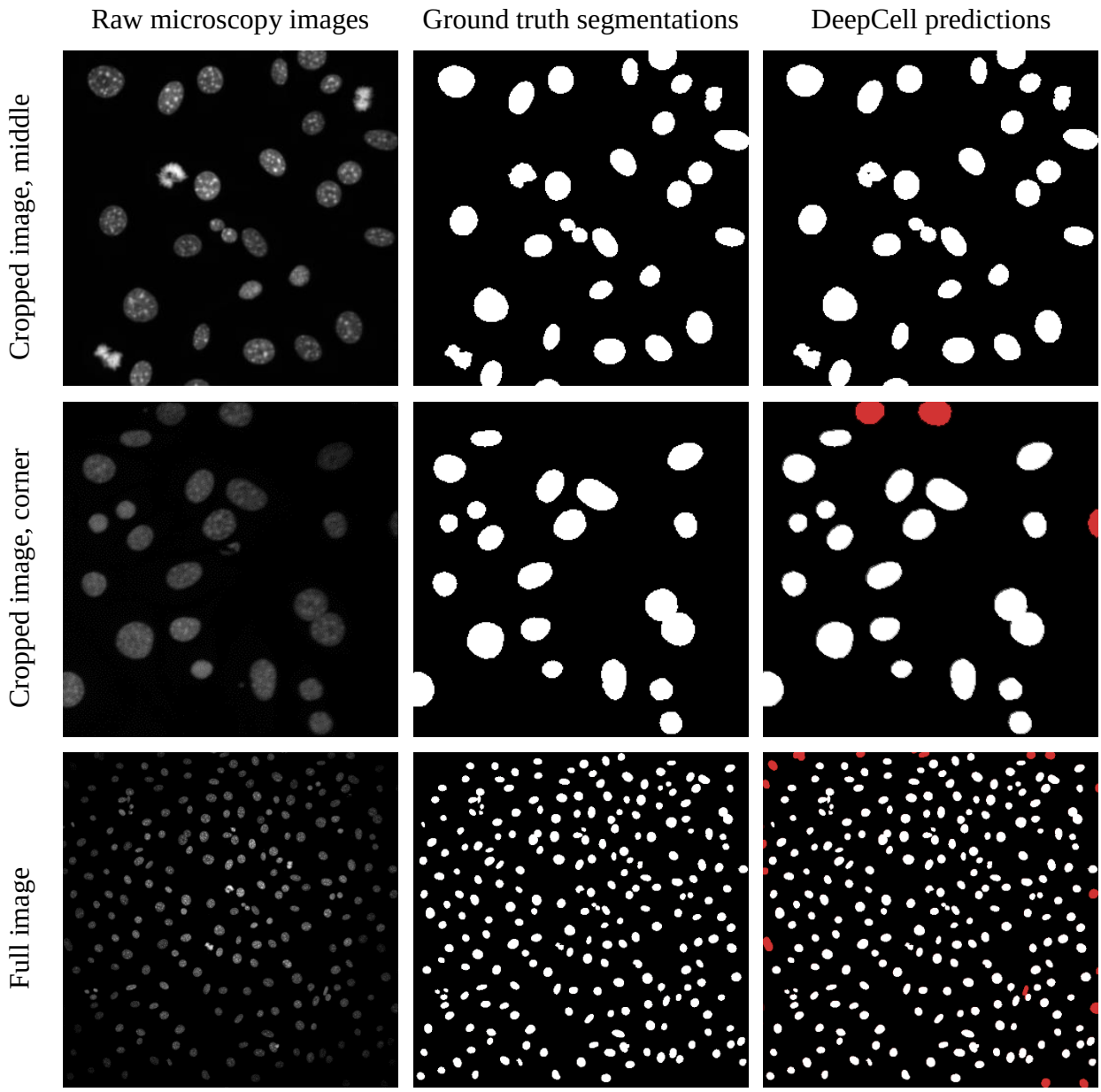


Figure 16. Binary prediction on fluorescent images using DeepCell architecture with patch method. Red in the predictions column indicates nuclei that did not appear in the ground truth generated by PerkinElmer.

Next, we tested whether patch approach can handle the more complex brightfield images. The process of generating training data stayed the same. In this case only the input fluorescent images were swapped for brightfield images and the network was trained again from scratch.

While DeepCell with patch technique had good accuracy on fluorescent images, performance dropped to 91.8% accuracy for brightfield (Table 1), with the network's

segmentations very different from ground truth (Figure 17). With a more complex image modality, DeepCell missed many nuclei and falsely predicted extranuclear debris.

This network converges after 9 epochs on fluorescent image data, with each epoch lasting 870 seconds. On brightfield images, it takes 25 epochs for the loss to plateau.

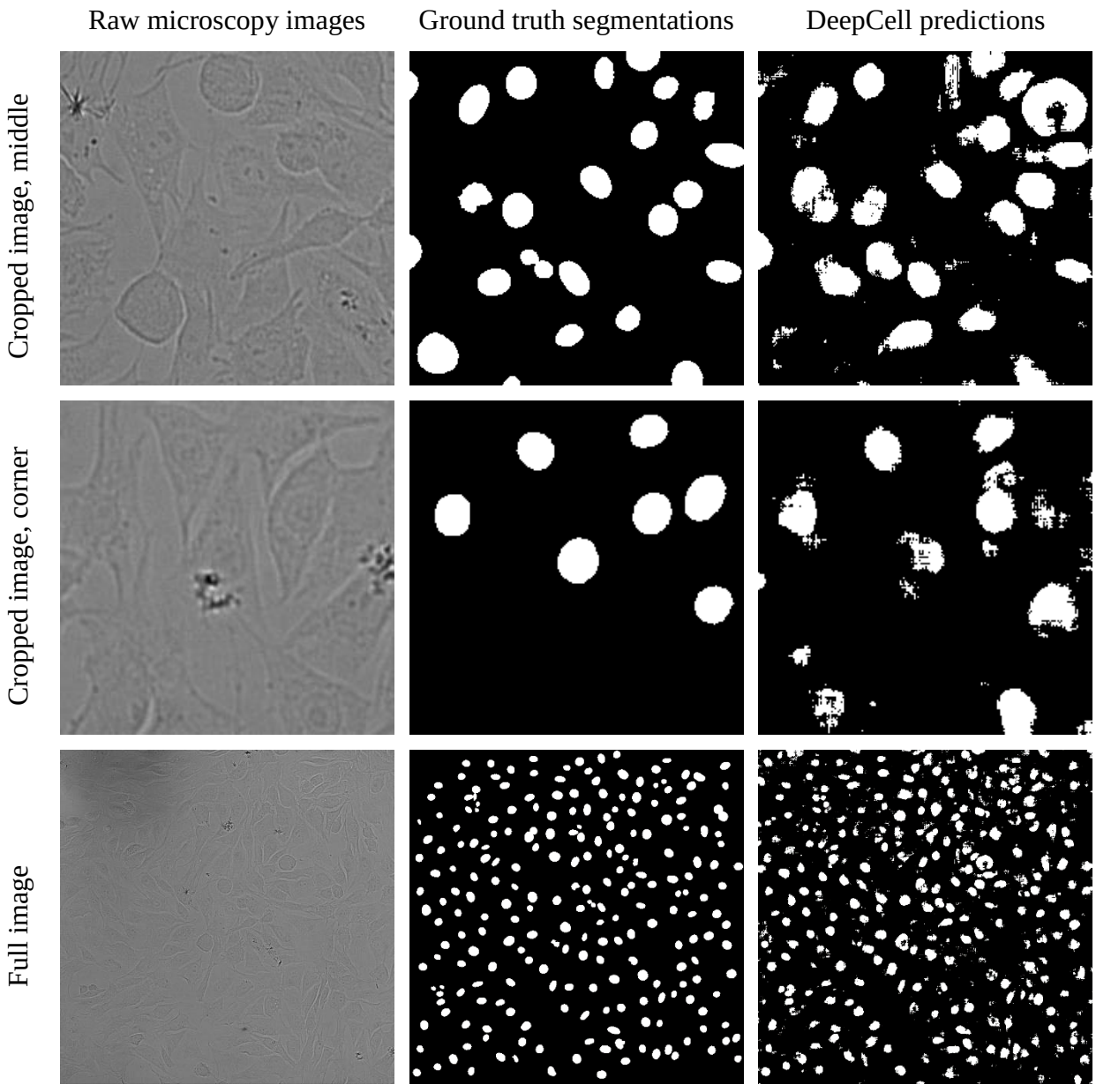


Figure 17. Binary prediction on brightfield images using DeepCell architecture with patch method.

Although the method could be further refined, we decided not to continue because of its main drawback, low speed. Even after optimizing it proved to be too slow, with 2.55 seconds required per image using a GPU. In addition, the patch generation process takes three seconds per 1080x0180 image with a CPU. The results show that the patch method cannot

meet the industrial demands of one second per image on a desktop computer. We thus looked for other architectures that would perform better on brightfield modality.

4.2 U-net architecture

We first tested U-net, which has been successfully applied in several bioimaging challenges. It converges after 20 epochs of each taking 1200 seconds. U-net was 96.3 % accurate on brightfield images (Table 2), and almost thousand times faster compared to DeepCell patch method, taking 0.109 seconds to segment a 1080x1080 image. U-net inputs do not need additional time-consuming generating of hundreds of thousands of patches from input media.

Table 2. Speed and accuracy reports by U-net segmentation on brightfield images.

	Brightfield images
Prediction time per image	0.109 s
Overall test accuracy on a test set of 504 images	96.3 %
Nucleus pixel accuracy	82.4 %
Background pixel accuracy	98.1 %
Nucleus pixel precision	85.8 %
Nucleus pixel recall	82.4 %

Pre-processing steps were essential for successful segmentation. Without contrast enhancement, the network predicted all pixels to be background.

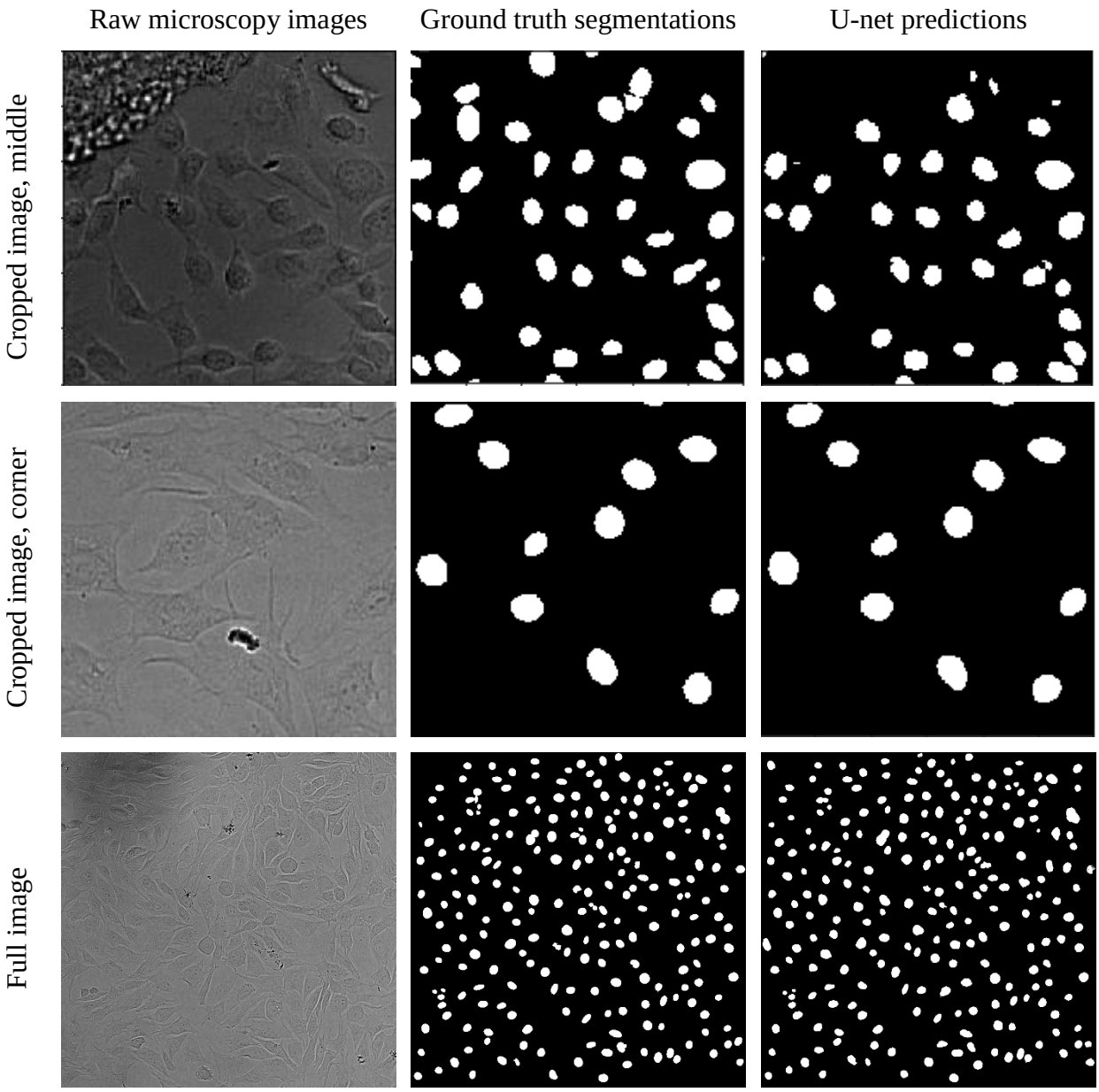


Figure 18. Binary prediction on brightfield images using U-net architecture on full images.

Qualitatively, small debris is occasionally predicted as nuclei (Figure 18, top row). Although the network generated errors there, they are easy to fix with conventional algorithms. For example, every white blob can be extracted from the mask and their surface area in pixels calculated. Then, based on size, the outlier pixel regions may be removed to increase accuracy.

The U-net model is small enough to be incorporated into future software products without inflating the software package size. Its final unpacked size is 14.1 MiB.

4.3 Mask-RCNN architecture

Both DeepCell and U-net are tools for semantic segmentation. Usually, their outputs need to be further processed by conventional algorithms to separate nuclei for additional analysis. Mask-RCNN produces a separate mask for every nucleus from its input image which reduces postprocessing efforts and complexity (Figure 12).

On average, training takes 920 seconds per epoch of 600 images. Current best model is trained for 120 epochs in total.

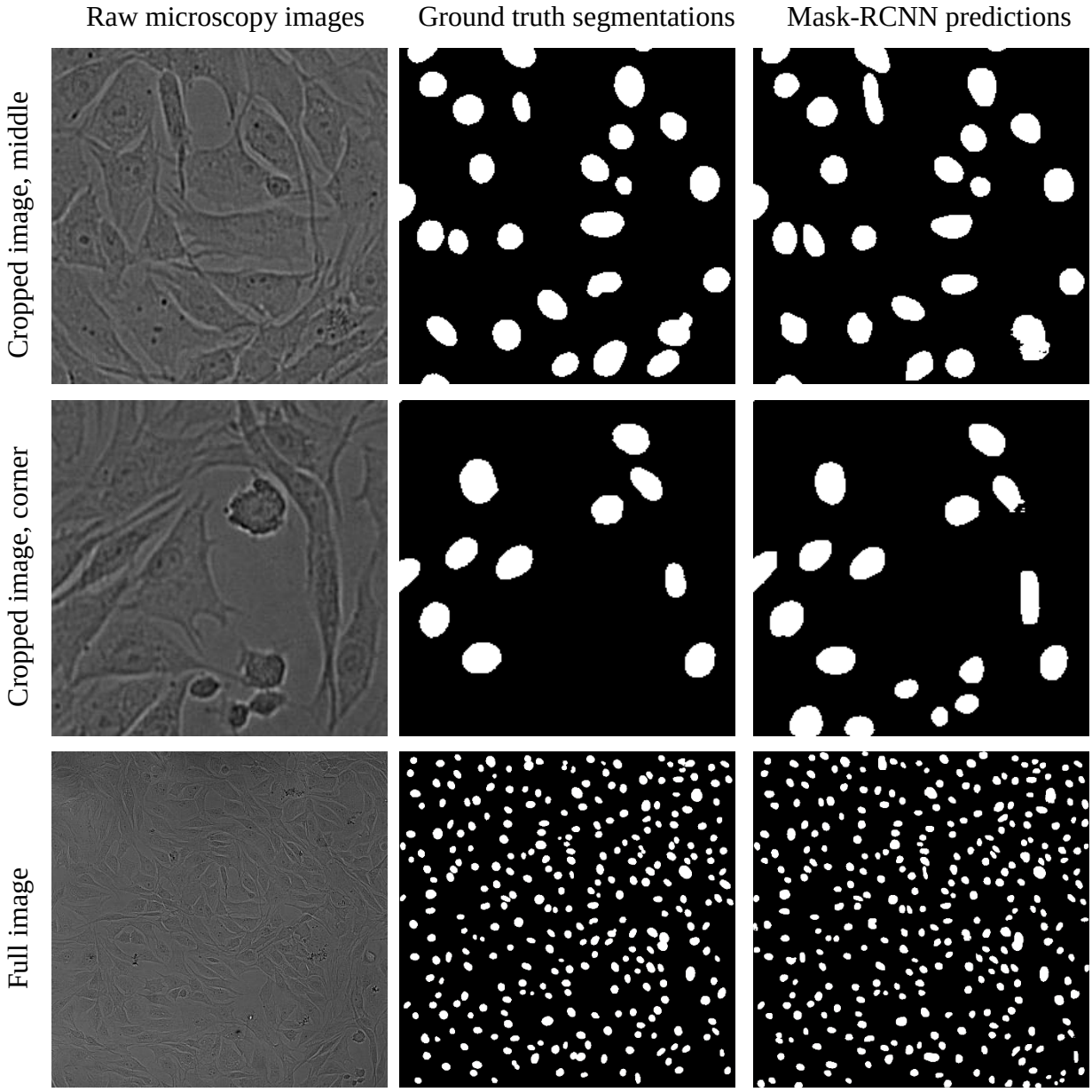


Figure 19. Binary prediction on brightfield images using Mask-RCNN architecture on full images. Output instance segmentations are flattened to semantic segmentations for easier visual comparison with ground truth.

Table 3. Speed and accuracy of Mask-RCNN segmentation on brightfield images

	Brightfield images
Prediction time per image	5.6 s
Overall test accuracy on a test set of 504 images	94.4 %
Nuclei pixel accuracy	70.5 %
Background pixel accuracy	97.7 %
Nucleus pixel precision	80.7 %
Nucleus pixel recall	70.5 %

Mask-RCNN's 94.4% overall accuracy did not outperform U-net (Table 2, Table 3). Qualitative analysis of the output images shows that Mask-RCNN does not predict small extranuclear debris (Figure 19). In contrast, it has problems with low nucleus pixel recall rate (Table 1). A possible reason for this is that the ResNet feature extractor is not trained well enough or is incapable of detecting nuclei in such modality. The other possible culprit might be the algorithm that dismisses proposal boxes. When two nuclei are close in a way that their bounding boxes overlap a lot, then one box and hence its future segmentation is removed (Figure 14). This processing step is designed into the architecture to avoid generating several proposals for a single object in the feature map, but its assumptions are not always valid in our application.

In addition, Mask-RCNN is more than fifty times slower than U-net (5.6 s vs 0.1 s, respectively). This is especially important given all times provided in this thesis are produced on a GPU, and would be much larger on CPUs. Although the measurements vary from experiment to experiment, a rough estimate is that running the Mask-RCNN prediction on a single image would take nearly a minute on a CPU.

5. Discussion

U-net brightfield segmentation results were demonstrated to and discussed with potential future users. PerkinElmer clients who work with nuclei detection and segmentation expressed intention to apply developed models for their brightfield experiments as soon as possible.

Despite all the progress achieved while working on this thesis, there are still many ideas that did not fit into the scope of current thesis. A few ideas came from a nuclear segmentation focused Kaggle competition and after its end in its discussion forums [54]. Kaggle participants revealed methods they used to get high scores on the leader board. A few of those can also be used to further develop brightfield segmentation solution.

From semantic segmentation to instance segmentation

The U-net method proposed in this thesis for tackling brightfield image analysis produces semantic segmentations. For quantitative analysis we would need every nucleus segmented as a separate object in the image. In the dataset our U-net was trained on, most of the nuclei are separate blobs and can be separated using simple computer vision algorithms. In cellular images with more overlapping cells or in cases where nuclei are close together (e.g. polynuclear cells), the current solution will perform worse, as it lacks the ability to separate adjacent objects by drawing boundary lines.

One idea that can fix the above-mentioned problem is to modify initial training data to make the network predict nuclear pixels and the border between adjacent nuclei. A loss weight map (Figure 20) derived from ground truth masks can force a network to learn to keep cells separate from each other on the image [38]. It is computed as

$$w(x) = w_c(x) + w_0 \times \exp\left(-\frac{\left(d_1(x) + d_2(x)\right)^2}{2\sigma^2}\right) \quad (4)$$

where x is a pixel value to be computed, w_c is a weight map that is balancing class frequencies, d_1 and d_2 are distances to nearest and the second nearest cell border, w_0 is a scalar value for the weight range, which in case depicted on Figure 20 is 10, and σ is picked to be 5 pixels [38]. Each pixel on the map is assigned a value based on ground truth masks. Pixel is assigned a bigger value when it is close to two different cell borders and a smaller

value when it is far from borders. This technique is used in the original U-net paper for separating cells but it can be applied for nuclei, too.

After a forward pass through the network the output activation map is multiplied with the loss map and only then the final loss is calculated. This multiplication before loss calculation adds the loss weight map to the computation graph and thus makes possible for backpropagation algorithm to find which filter values contributed to the loss in critical border areas, and adjust them accordingly. In a way, this added multiplication layer acts as a filter that has the same size as its input feature map. However, the parameters of this weight map are different for every image passing the neural network and are not learnable thus cannot be changed by backpropagation.

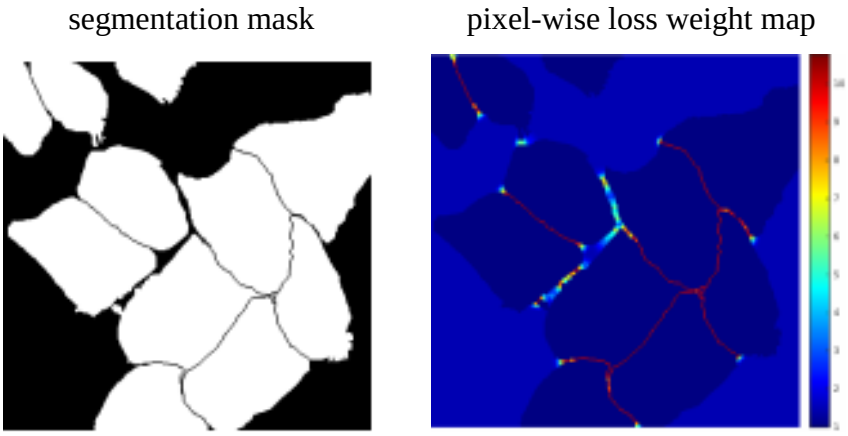


Figure 20. Segmentation mask of cells and a loss weight map generated from it [38].

Instead of using the loss weight map, predicting two separate channels is possible. We could generate a separate discrete border annotation channel for the ground truth masks. Border masks will only be generated between touching objects, not on the outer edges of them, as otherwise the network would easily learn to generate separating lines in non-ambiguous places, for example outer edges of objects, and perform badly in critical places, between adjacent nuclei. Furthermore, the width of nuclei separating border areas would be bigger as the sizes of nuclei they are separating increases. In practice, it proves to be harder to separate bigger nuclei that are close to each other. [55]

This idea was proposed by one of the winners of the Data Science Bowl 2018 [55], team "Topcoders". They trained the network with two different mask channels with one containing nuclei and the other one internuclear borders. The network predicts objects and borders separately and these two prediction channels are combined into one image (Figure 21).

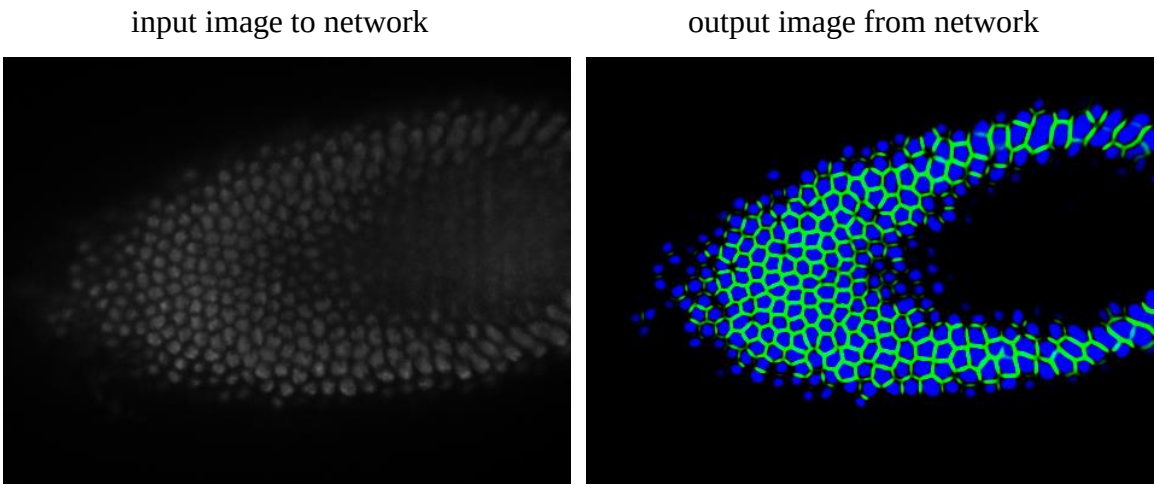


Figure 21. "Topcoders" winning solution example segmentation on Data Science Bowl 2018. [55]

In addition, “Topcoders” did not use a standard U-net architecture, but instead a pre-trained deep encoder Resnet101 as the encoder part for U-net (Figure 10a) [55]. Although this may have benefits in terms of accuracy, it likely also has drawbacks with respect to speed. It would be worthwhile to experiment with this architecture in the future for direct performance comparison with approaches introduced in this thesis.

Toward better memory management and input image handling

The U-net model implemented as part of this thesis only operates on 1080x1080 pixel images. Although it can segment an image in 0.109 seconds, it consumes a lot of memory. On an NVIDIA Tesla P100 GPU with 12 GB of RAM, training and inference allow only a batch size of one or will otherwise run out of memory. In addition, if future images are of different resolutions, this network will fail to process them.

There are a few solutions that allow to mitigate these problems. First, networks could segment on smaller input dimensions thus speeding up the process. To enable processing of big images the inputs can be resized before and after (Figure 22). Although this would lose information and accuracy it would increase speed and consume less memory.

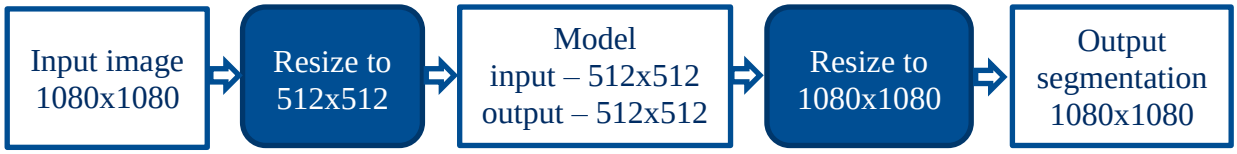


Figure 22. Potential solution for faster segmentation and lowering RAM consumption.

Another possibility is to divide input images into smaller sections to be segmented separately. It would lower overall speed since more passes of the network must be conducted depending on the image and crop sizes. On the other hand, it would keep accuracy the same and reduce memory consumption. After segmentation the smaller parts would be merged together to form a final output (Figure 23). To reduce edge effects, the cropped segments can be partially overlapping.

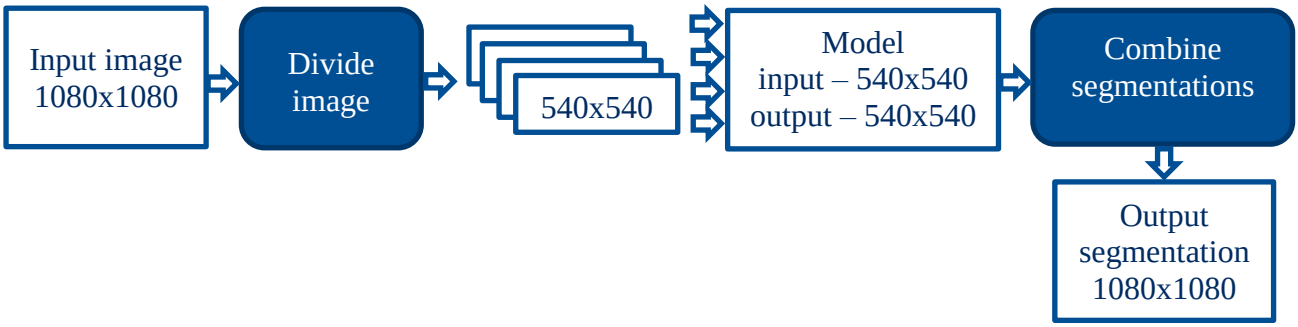


Figure 23. Potential solution for lowering RAM consumption while maintaining high accuracy.

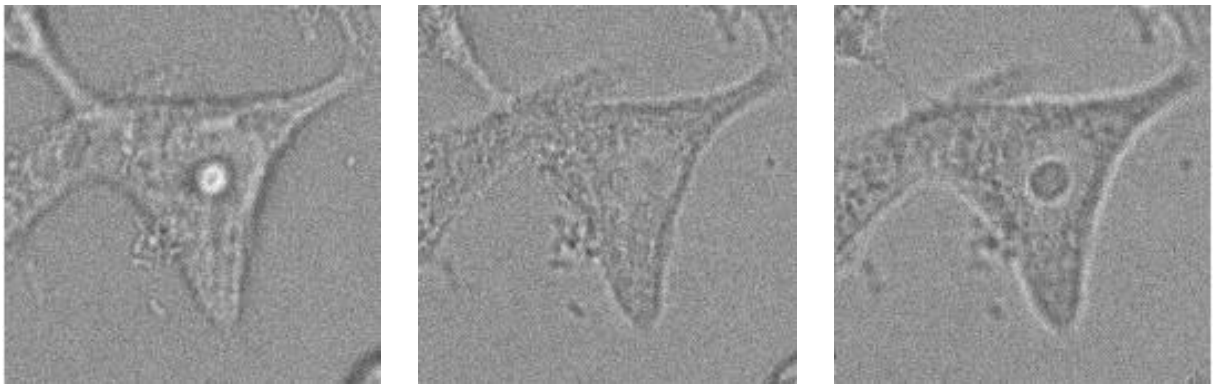
Utilisation of multiple focal planes for richer information

All brightfield experiments in this thesis were carried out on data containing only single focal plane images. In some cases, it is difficult to extract necessary information for segmentation from a single focal plane as shown on Figure 24a. In this example there is a cellular monolayer flipped on top of another one. We can see that cellular structures cannot be distinguished well because of blurriness and distortion. In addition, Figure 24c indicates poor performance of our U-net segmentation on this image.

a) b) c)

Figure 24. A challenging segmentation example. a) Brightfield image has been c) segmented by our developed U-net model but compared to b) ground truth is missing many nuclei.

There is additional information about cells on different focal planes. At such scales cellular structures start to function as minuscule lenses. For example, if we are focusing the microscope under the sample we might see brighter spots which indicate nuclei (Figure 25a). Because of their shape they act as converging lenses and increase light density below them if it is shined from above. Modern microscopes can take pictures of brightfield samples by focusing on different heights measured from the sample [56]. Every focal level contains different information. From the middle image in Figure 25 there is hardly any nuclear structure seen and we can see the benefit of other focal planes for nuclear segmentation.



a) b) c)

Figure 25. The same is cell photographed from different focal distances: focus of a) is where the light is converged by the nucleus, b) has focus where the nucleus' converging effect is not visible, and c) is focused at a section where converging light rays from nucleus have diverged after focal point shown in a).

Instead of using neural networks on brightfield images taken from a single focal plane we can use several planes as different image channels. The network could learn to acquire necessary features from these planes and use them to produce its final segmentation. Perhaps this would help to increase nuclei recall in the future.

6. Conclusions and Summary

Microscopy data is being generated at an accelerating pace. Thanks to high throughput microscopy and screening hardware, biologists and pharmaceutical industry can generate more images than ever before. Although development in the image analysis field has been rapid, biologists need additional tools to make their work more efficient.

Annotating nuclei in cellular images gives researchers a plethora of information ranging from counting cell numbers in microplates to laying groundwork for cell phenotyping further down an imaging pipeline. Despite having widely adapted tools for fluorescent image segmentation, for example CellProfiler [24] and Harmony software [25], we still lack industrial solutions for brightfield images.

Brightfield microscopy is faster, cheaper, and less invasive for living cells. For imaging it does not need complex sample preparation or expensive chemicals, and it does not induce cellular stress or death. The task of brightfield image segmentation has not been possible with conventional computer vision algorithms so far. Including deep learning models into cellular sample analysis could open new research methods and save thousands of hours of manual segmentation work. The model that has been implemented as a part of this thesis proves that complex brightfield images can be successfully segmented with machine learning models.

Three different neural network architectures were tested for segmentation: DeepCell, U-net, and Mask-RCNN (Appendix I). DeepCell's pixel-by-pixel classification proved to be very slow and inaccurate on brightfield images. Mask-RCNN solution was on average almost thirty times faster and 2.6 % more accurate. U-net performs the best regarding both accuracy and speed being about fifty times faster than Mask-RCNN, and having 1.9 % fewer pixel-level errors.

We conclude that on brightfield images the U-net model is producing the best results. In addition, U-net model’s size on disk is compact enough to be incorporated into existing software. There are many ways to further develop the proposed segmentation method, for example by optimizing memory consumption, making the system input volume size invariant, and using several brightfield focal stacks to achieve more accurate results. The method developed in this thesis is ready for first tests by PerkinElmer’s software engineers.

7. Bibliography

[1] “Brightfield Microscopy Overview,” Nikon, [Online]. Available: https://www.nikoninstruments.com/en_EU/Learn-Explore/Techniques/Brightfield. [Accessed 14 05 2018].

[2] “Medical Dictionary,” [Online]. Available: https://medical-dictionary.thefreedictionary.com/fluorescence+microscopy. [Accessed 14 05 2018].

[3] “Dictionary,” [Online]. Available: http://www.dictionary.com/browse/nucleus. [Accessed 14 05 2018].

[4] “Common Objects in Context,” [Online]. Available: http://cocodataset.org/#home. [Accessed 14 05 2018].

[5] F. Rosenblatt, “The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain,” *Psychological Review,* 1958.

[6] A. Gołda, “Introduction to neural networks,” [Online]. Available: http://home.agh.edu.pl/~vlsi/AI/intro/. [Accessed 14 05 2018].

[7] L. Jacobson, “Introduction to Artificial Neural Networks - Part 1,” [Online]. Available: http://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-1/7. [Accessed 14 05 2018].

[8] B. Dolhansky, “Artificial Neural Networks: Mathematics of Backpropagation (Part 4),” [Online]. Available: http://briandolhansky.com/blog/2013/9/27/artificial-neural-networks-backpropagation-part-4. [Accessed 14 05 2018].

[9] D. H. Hubel, T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat's visual cortex,” *The Journal of Physiology,* 1962.

[10] K. Fukushima, S. Miyake, “Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position,” *Physica A: Statistical Mechanics and its Applications,* 1982.

[11] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, “Backpropagation Applied to Handwritten Zip Code Recognition,” *Neural Computation,* 1989.

[12] A. Krizhevsky, I. Sutskever, G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1,* 2012.

[13] “Data Science Challenges,” Consortium for Open Medical Image Computing, [Online]. Available: https://grand-challenge.org/All_Challenges/. [Accessed 14 05 2018].

[14] “Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow,” Matterport, [Online]. Available: https://github.com/matterport/Mask_RCNN. [Accessed 14 05 2018].

[15] A. Karpathy, “Software 2.0,” [Online]. Available: https://medium.com/@karpathy/software-2-0-a64152b37c35. [Accessed 14 05 2018].

[16] A. Canziani, A. Paszke, E. Culurciello, “An Analysis of Deep Neural Network Models for Practical Applications,” *arXiv,* 2016.

[17] “ImageNet Challenge,” [Online]. Available: http://www.image-net.org/. [Accessed 14 05 2018].

[18] C. Angermueller, T. Pärnamaa, L. Parts, O. Stegle, “Deep learning for computational biology,” *Molecular Systems Biology,* 2016.

[19] T. Pärnamaa, L. Parts, “Accurate classification of protein subcellular localization from high throughput microscopy images using deep learning,” *bioRxiv,* 2016.

[20] V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, R. Kim, R. Raman, P. C. Nelson, J. L. Mega, D. R. Webster, “Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs,” *Innovations in Healthcare Delivery,* 2016.

[21] M. Mattiazzi Usaj, E. B. Styles, A. J. Verster, H. Friesen, C. Boone, B. J. Andrews, “High-Content Screening for Quantitative Cell Biology,” *Trends in Cell Biology,* 2016.

[22] E. Meijering, “Cell Segmentation: 50 Years Down the Road,” *IEEE Signal Processing Magazine,* 2012.

[23] K. A. D. F. L. P. W. Jones, “Computational biology: deep learning,” *Emerging Topics in Life Sciences,* 2017.

[24] “Cell Profiler Image Analysis Software,” [Online]. Available: http://cellprofiler.org/. [Accessed 14 05 2018].

[25] “Harmony High Content Imaging and Analysis Software,” PerkinElmer, [Online]. Available: http://www.perkinelmer.com/product/harmony-4-6-office-hh17000001. [Accessed 14 05 2018].

[26] “Labeling Nuclear DNA Using DAPI,” 2011. [Online]. Available: http://cshprotocols.cshlp.org/content/2011/1/pdb.prot5556.full. [Accessed 18 05 2018].

[27] E. C. Jensen, “Types of Imaging, Part 2: An Overview of Fluorescence Microscopy,” *The Anatomical Record Advances in Integrative Anatomy and Evolutionary Biology,* 10 2012.

[28] H. Giloh, J.W. Sedat, “Fluorescence microscopy: reduced photobleaching of rhodamine and fluorescein protein conjugates by n-propyl gallate,” *Science,* 1982.

[29] L. L. Drey, M. C. Graber, J. Bieschke, “Counting unstained, confluent cells by modified bright-field,” *Biotechniques,* 2013.

[30] “PerkinElmer global website,” [Online]. Available: http://www.perkinelmer.com/. [Accessed 20 05 2018].

[31] A. Karpathy, “CS231n Convolutional Neural Networks for Visual Recognition,” [Online]. Available: http://cs231n.github.io/convolutional-networks/. [Accessed 14 05 2018].

[32] V. Nair, G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *ICML'10 Proceedings of the 27th International Conference on International Conference on Machine Learning*, 2010.

[33] “Max Pooling,” [Online]. Available: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling. [Accessed 14 05 2018].

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from,” *Journal of Machine Learning Research,* 2014.

[35] D. P. Kingma, J. L. Ba, “Adam: A Method for Stochastic Optimization,” in *ICLR,* 2015.

[36] N. S. Keskar, R. Socher, “Improving Generalization Performance by Switching from Adam to SGD,” *arXiv,* 2017.

[37] D.A. Van Valen, T. Kudo, K. M. Lane, D. N. Macklin, N. T. Quach, M. M. DeFelice, I. Maayan, Y. Tanouchi, E. A. Ashley, M. W. Covert, “Deep Learning Automates the Quantitative,” *Computational Biology,* 2016.

[38] O. Ronneberger, P. Fischer, T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” 2015.

[39] K. He, G. Gkioxari, P. Dollár, R. Girshick, “Mask R-CNN,” *arXiv,* 2017.

[40] R. Girshick, J. Donahue, T. Darrell, J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *arXiv,* 2014.

[41] R. Girshick, “Fast R-CNN,” *arXiv,* 2015.

[42] S. Ren, K. He, R. Girshick, J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *arXiv,* 2016.

[43] Yuthon, “Notes: From Faster R-CNN to Mask R-CNN,” 26 04 2017. [Online]. Available: https://www.yuthon.com/2017/04/27/Notes-From-Faster-R-CNN-to-Mask-R-CNN/. [Accessed 14 05 2018].

[44] “Detectron,” Facebook AI Research (FAIR), [Online]. Available: https://github.com/facebookresearch/Detectron. [Accessed 14 05 2018].

[45] K. He, X. Zhang, S. Ren, J. Sun, “Deep Residual Learning for Image Recognition,” *arXiv,* 2015.

[46] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, “Feature Pyramid Networks for Object Detection,” *arXiv,* 2017.

[47] W. Abdulla, “Splash of Color: Instance Segmentation with Mask R-CNN and TensorFlow,” [Online]. Available: https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46. [Accessed 21 05 2018].

[48] “Matterport Mask-RCNN implementation,” [Online]. Available: https://github.com/matterport/Mask_RCNN. [Accessed 14 05 2018].

[49] “Keras: The Python Deep Learning library,” [Online]. Available: https://keras.io/. [Accessed 14 05 2018].

[50] “NVIDIA cuDNN,” [Online]. Available: https://developer.nvidia.com/cudnn. [Accessed 14 05 2018].

[51] “High Performance Computing Center,” University of Tartu, [Online]. Available: https://www.hpc.ut.ee/en_US/web/guest. [Accessed 14 05 2018].

[52] “High Performance Computing Center's specifications,” University of Tartu, [Online]. Available: https://www.hpc.ut.ee/en_US/web/guest/rocket-cluster?inheritRedirect=true. [Accessed 14 05 2018].

[53] “Python Imaging Library ImageEnhance Module,” [Online]. Available: https://pillow.readthedocs.io/en/3.0.x/reference/ImageEnhance.html#imageenhance-module. [Accessed 18 05 2018].

[54] “Data Science Bowl 2018,” Kaggle Inc, [Online]. Available: https://www.kaggle.com/c/data-science-bowl-2018. [Accessed 14 05 2018].

[55] “[ods.ai] topcoders, 1st place solution on Data Science Bowl 2018,” [Online]. Available: https://www.kaggle.com/c/data-science-bowl-2018/discussion/54741. [Accessed 14 05 2018].

[56] “Opera Phenix™ High-Content Screening System,” [Online]. Available: https://www.perkinelmer.com/lab-solutions/resources/docs/BRO_011289_01_Opera_Phenix.pdf. [Accessed 21 05 2018].

Appendix

I. Combined test results with different architectures for nuclear segmentation

	DeepCell on fluorescence images	DeepCell on brightfield images	U-net on brightfield images	Mask-RCNN on brightfield images
Segmentation time per image on a GPU	104.5 s	104.2 s	0.109 s	5.6 s
Overall pixel accuracy	97.1 %	91.8 %	96.3 %	94.4 %
Nucleus pixel accuracy	92.4 %	72.3 %	82.4 %	70.5 %
Background pixel accuracy	97.7 %	94.4 %	98.1 %	97.7 %
Nucleus pixel precision	84.1 %	63.2 %	85.8 %	80.7 %
Nucleus pixel recall	92.4 %	72.3 %	82.4 %	70.5 %

II. License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Sten-Oliver Salumaa**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:
   1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and
   1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis

Convolutional Neural Networks for Cellular Segmentation,

supervised by Leopold Parts and Dmytro Fishman,

2. I am aware of the fact that the author retains these rights.
3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **21.05.2018**