Master thesis

Segmentation of White Blood Cells Using Deep Learning

Desiré Nilsson and Sophia Grimmeiss Grahm

18 December, 2019

Supervisors:
Kalle Åström, LTH
Kent Stråhlen, CellaVision AB

LUNDS UNIVERSITET

Abstract

The white blood cell count and differential is an important part of diagnosing a number of medical conditions. Instead of doing this by manual microscopy, CellaVision's technology has automated the process of finding and classifying white blood cells. To support a diagnosis it is desired that the system can produce cytoplasm-to-nucleus-ratio. This ratio is calculated from a segmented image where the pixels are labelled as background, cytoplasm, or nucleus. The system used today, using active contours, does not always produce perfect segmentations for all cells, and it would therefore be beneficial to improve the segmentation. Using machine learning, we have constructed a network for segmenting white blood cell images. This model, with some small modifications, produces both binary (cell and background) and multi-class (cytoplasm, nucleus and background) segmentations. The model is a U-net inspired by work previously done on other similar segmentation tasks. The network reached an IoU of 93.9% in the binary case, and in the muli-class case 82.8% and 94.5% for the cytoplasm and nucleus respectively. The main challenges were to separate neighbouring cells and cells in a cluster.

Over all the network performed better than the active contour method in difficult images, and in cases where neither were good, the network was usually better. If the network was trained more on images that are difficult to segment, the resulting segmentations of these images could be improved.

Acknowledgement

We would first like to thank our supervisor Kalle Åström for all the valuable ideas, suggestions and delimitations. It was a great help both to expand and limit the content of this project.

We would also like to thank CellaVision for giving us this opportunity, and more importantly all the people at CellaVision that welcomed us. First and foremost our supervisor Kent Stråhlén for supporting us and answering all our questions about cells and their segmentations. We would also like to thank him for providing us with material and for the feedback on our report. Further we would like to thank Benny Klein for producing our data, explaining it to us and providing material for our writing process. Worth mentioning is also Åse Sykfont Snygg who explained the staining process to us, and Ida Wagnström who helped us get started with the coding. Thanks also to the people whom we shared an office with for the company, help and all the nice chats that had nothing to do with cells or segmentation.

Finally we would also like to thank our family and friends that helped us throughout our education. Special thanks to Emil Johansson for all the encouragement, notes, code examples and emotional support.

Contents

Glossary

Abbreviations

WBC = White Blood Cell
RBC = Red Blood Cell
CNR = Cytoplasm-to-Nucleus-Ratio
RGB = Colour images, containing three colour channels Red, Green and Blue

Cell biology

Leukocyte = White blood cell
Erythrocyte = Red blood cell
Thrombocyte = Platelet

Nucleus = The part of the cell containing DNA
Cytoplasm = The part of the cell that is not nucleus, containing all apparatus for the cell to survive and function properly
Vacuole = A fluid filled organelle in the cytoplasm

Machine learning

Batch = A subset of the input data, used to train on smaller sets to decrease memory usage
FCN = Fully Connected Network, a network where all nodes in a layer is connected to all nodes in the previous layer
CNN = Convolutional Neural Network, a network with convolutional layers
ReLU = Rectified Linear Unit, an activation function

Ground truth = Segmentation of a cell considered to be correct
Hyperparameter = A parameter set when designing the network, before training
Convergence = When the network have found a set of parameters that will not be changed significantly even if trained longer
Overfit = When a network has adapted to much to the training data that it can not generalise to other input

Active contour = The segmentation method currently used in CellaVisions system
U-net = The network architecture used in this report

Evaluation

IoU = Intersection over Union, a measure of how well a segmentation matches ground truth
IoU_b = Intersection over Union but with border pixels between classes excluded from the measure

1 Introduction

Human blood contains three types of blood cells. The most common is red blood cells (RBC), or erythrocytes, which transport oxygen. The other ones are platelets, also called thrombocytes, which are important for clotting, and white blood cells (WBC), or leukocytes, which are a vital part of the immune system. There are many different types of leukocytes, the five most common types are lymphocytes, monocytes, basophils, neutrophils and eosinophils. These are normally found in a healthy persons blood in different concentrations and they each play a different role in the body's defence against infections and disease, and differ not only in function but also in appearance. Counting how many of the different types there are present in a blood sample is a vital part of diagnosing a number of illnesses as their relative proportions indicate different types of problems [3].

One of the first steps in blood analysis is to analyse the sample in a cell counter. The cell counter automatically calculates the concentration of white blood cells, red blood cells and platelets. If an abnormality is detected, the sample is flagged for a process called the white blood cell differential. This is the process of counting and classifying leukocytes, which is often done manually by studying a blood smear through a light microscope. The WBC count and differential is time consuming and requires skilled staff, but as modern technology has made automatic blood analysis possible, fully or partially automated solutions have become increasingly common. This makes it possible for skilled staff to put their time elsewhere [4].

With automated analyses more information could also be extracted. One parameter of interest, which is hard to estimate by only looking at the cells, is the cytoplasm-to-nucleus-ratio (CNR) of the white blood cells. Like most cells in the body the leukocytes consists of a nucleus, which contains DNA, and cytoplasm, which contains all apparatus the cells needs to live. To obtain this ratio the cell must be segmented, which is a procedure of determining which pixels belongs to which part of the image, in this case which pixels are a WBC and which are background.

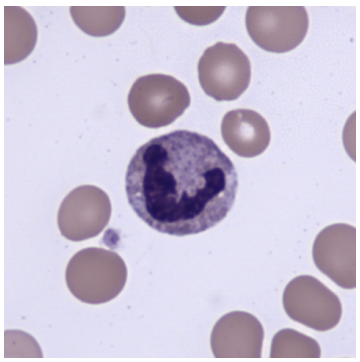
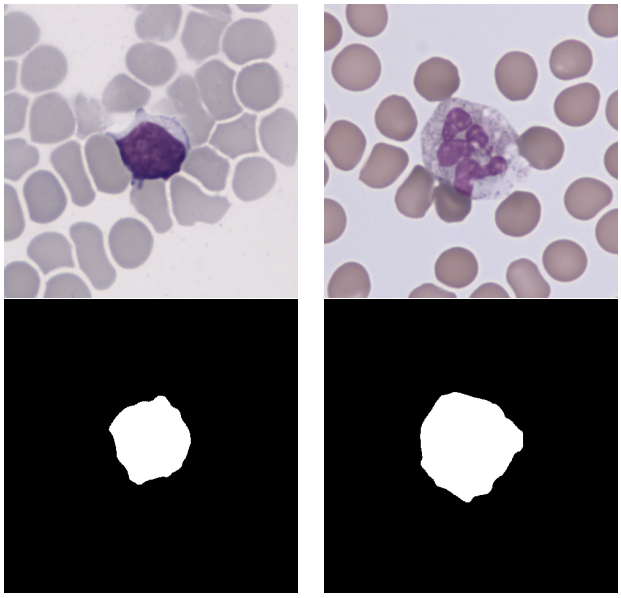


Figure 1.1 – An example of a cell image, containing a centred WBC, multiple RBCs and a small platelet.

1.1 About CellaVision and Automatic Differential

CellaVision's main product is automated cell differentiation. They offer many applications, such as red blood cell characterisation and body fluid differential, but the main feature is the white blood cell differentiation. To perform the differentiation a blood sample is prepared by placing a drop of peripheral blood on a glass slide and smearing it with a spreader slide. This can be done automatically

or manually. The slide is then air dried and fixed with either methanol or ethanol. In order to distinguish the different cell types, the slide is stained with a Romanowsky stain such as May-Grünwald Giemsa, Wright, or Wright-Giemsa. The system places the sample under a microscope and identifies a monolayer, i.e. an area where cells are mostly non-overlaping but not too far apart. Then a pre-locater finds where the white blood cells are located, and must distinguish what is a WBC and what is not. This may sound easy, but varying image quality, the use of different staining techniques as well as the presence of colour stains, artefacts, and fragments of broken cells sometimes makes it difficult even for the human eye. There are also blood components, such as thrombocytes, that might be similar to white blood cells in appearance but are not as relevant for the differential. When the interesting cells are located the system camera takes a picture of it. An example of such an image is presented in Figure 1.1. The images are taken in 100x magnification in Bayer format which is then converted to standard colour images (RGB). The system performs a segmentation and a pre-classification. All images and their respective cell classes are then presented to the operator who reviews and verifies the result.



(a) Large segmentation. (b) Small segmentation.

Figure 1.2 – Examples of segmentations produced by the active contours method. This method sometimes includes RBCs and excludes weakly coloured areas of the cytoplasm.

The current segmentation method in CellaVision's system is based on a snake with area-based external energy. A classic snake is an active contour model, which is based on the idea to fit a closed curve to the edges of the object to segment. This is done by choosing a closed curve and letting it expand until it reaches the object outline [8]. For the most part, this method performed well and the segmentations have been accurate. There were some instances however, where the method had a hard time differentiating between cell and background. A common problem was images with a large amount of erytrocytes where the snake often expanded into adjacent red blood cells, see Figure 1.2a. Another issue was cells where the cytoplasm had a colour similar to the background, as in Figure 1.2b. In these cases, the method did not include the brighter parts in the segmentation. There was also a problem with images containing multiple cells.

1.2 Aim

This project aims to produce a neural network, more specifically a U-net, that can correctly segment WBCs from cell images. It should segment the WBC as a whole in a binary image by classifying pixels as cell or background. It should also perform a multi-class segmentation where the cytoplasm and nucleus are segmented into differently coloured areas in the same image. This is to be used to extract the CNR which, combined with other medical information, is an interesting measure in diagnostics.

1.3 Related Work

Various types of microscopic cell images have previously been shown to be possible to segment with satisfying results using neural networks. In [1], S. Akram et al. (2016) tested a convolutional neural network on three different data sets. Although neither were from peripheral blood smears, the results indicate that it is possible to segment cell images with neural networks. In [15], F. Xing et al. (2019) used many different neural networks for nucleus detection on large-scale pathology images from a wide variety of organs.

A now commonly used network to segment a number of cell types is the U-net structure. It was first proposed by O. Rönneberger et al. (2015) in [12] and has been widely used since. In the study, an IoU (Intersection over Union) score of 92% was presented. However, this study used images of HeLa cells taken with electron microscope and light microscope. In [16], Z. Zhou et al. (2018) studied microscopy images of cell nuclei among others, with some different models including the U-net. The microscopy images had an IoU score of 91% using U-net. In [7], J. Isaksson et al. (2017) used the U-net model on images of prostate tissue. They achieved a result of 88 % on cytoplasm pixels and 72% on nuclei.

In [9], G. Moallem et al. (2017) segmented white blood cells with an accuracy of about 80% using a level-set algorithm and in [13], F. Sadeghian et al. (2009) used a snake to segment white blood cells and achieved an accuracy of 78%.

In short, there is plenty of research on segmentation with neural networks, and also many studies on segmentation using U-nets. However, as to the authors knowledge there are no studies done on segmentation of white blood cells in peripheral blood smears using U-nets.

2 Neural Networks

The design of neural networks is originally inspired by neurons in the human brain. The idea is that many inputs in some way are combined and weighted to form the output. We will now continue to explain the components of our neural network. For the interested reader, we would recommend [5] and [6] for further reading.

A simple network structure is shown in Figure 2.1. It is made up of a number of nodes, drawn as circles, which are connected by edges, drawn as arrows. The nodes on the left side are input, which could be pixels in an image or words in a sentence. The node to the right are output, which again can have a number of formats, such as pixel value or a number. There could be one, two, or many inputs and outputs, and they do not have to be equally many or of the same type. For example, an input image could result in an output number representing a label. Between input and output, the information is processed in the hidden part of the network, which can be a single layer, or many hundreds. Each edge has a weight associated to it, and each node has a bias. The output, y_i, from a node with m input nodes, each with the output x_j, can be described by the formula

$$y_i = f\Big(\sum_{j=1}^{m} (w_{i,j} \cdot x_j) + b_i\Big),$$

where $w_{i,j}$ is the weight corresponding to the input node x_j, and b_i is the bias in node y_i. In this expression, f is an activation function which allows the output to be non-linear in regards to the input, which is desired when using a neural network. The use of a liner network is unnecessary since there are easier ways to deal with linear dependencies. During what is called training, the network will update the values of the weights and biases. Training is done by giving the network input and corresponding correct output. The correct output is collected in the ground truth set, which is to be considered the ideal answer. How the network should update the weights is determined by a loss function. This function is usually formulated to describe the error between the output and the ground truth, which is minimised during the training process.

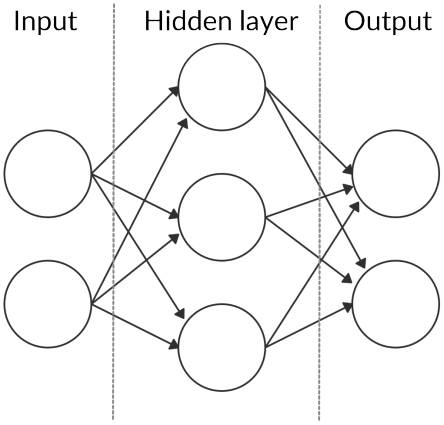


Figure 2.1 – A simple model of a neural network.

There are numerous types of neural networks, for example recurrent networks and autoencoders. One commonly used for classification is the fully connected network (FCN). In this type of network,

all nodes in one layer are connected to all nodes in the next layer, like the example in Figure 2.1. When using a FCN, the number of parameters increase rapidly when adding new layers, which makes them very computationally demanding. Therefore it is common to use a convolutional neural network (CNN).

2.1 Convolutional Neural Networks

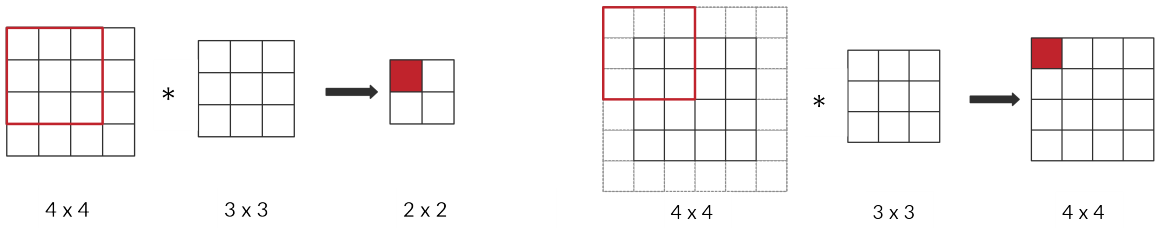
To replace the fully connected layers and efficiently reduce the amount of network parameters, the convolutional neural network uses convolutional and pooling layers. These networks also usually have an activation function, and sometimes batch normalization and dropout layers which will be explained shortly. In a network classifying into categories, the convolutional and pooling layers would be followed by a number of fully connected layers that eventually produces the output prediction. This structure works well for classification tasks since the resolution and dimensionality of the feature map is reduced in each step, resulting in a single feature vector containing probabilities for each class. In segmentation tasks, where the desired output is an image instead of a category, this output vector or the previous feature maps must be used to generate a segmented output image.

2.1.1 Convolution

The convolutional layers in a network combine local information from surrounding pixels when calculating a pixel's value in the next layer. This is done using a kernel, or filter, which has a pre-determined size, describing how large the area of interest is, or in other words, how many of the surrounding pixels that should be taken into consideration. In a convolutional layer, the content of the kernel is what is learned. The kernel is moved across the image with a stride, which describes how many pixels the kernel should be shifted. The final size of a $n \times m$ image with a kernel size $f_n \times f_m$ and stride k is

$$\left(\frac{n - f_n}{k} + 1\right) \times \left(\frac{m - f_m}{k} + 1\right),$$

but it is sometimes beneficial to keep the size of the image. This can for example be done by wrapping, reflecting or replicating the edge pixels but is commonly done by adding zeros outside of the image, called zero-padding. To keep the original size, padding is done with $(f - 1)/2$ zeros, where f again is the kernel size. The resulting sizes after a convolution, signed $*$, for an arbitrary sized image and kernels are shown in Figure 2.2, where the zeros are shown as dotted elements.



(a) Convolution without padding.

(b) Convolution with padding.

Figure 2.2 – Convolution with a 3×3 kernal and stride 1, with and without padding. Notice that the original size is preserved when padding is used.

2.1.2 Max-pooling

Pooling is a commonly used operation between convolutional layers. It is a way to down-sample and reduce dimensionality, while still keeping the most important information. This is, like the convolution,

done using a kernel and a stride. The max-pooling kernel sized 3×3 in Figure 2.3 takes all pixels in the red area into consideration and only saves the maximum value, in this case the 8. Here the stride is 2×3, meaning that the kernel is moved 2 steps to the right, and now becoming the blue area. After becoming the yellow area, the kernel can can no longer be moved to the right, and it restarts and move 3 pixels down from where it started, becoming the area in green.

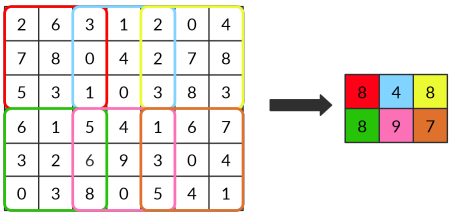


Figure 2.3 – Visualization of max-pooling, with kernel 3×3 and stride 2×3.

As the example in Figure 2.3 shows, the stride and kernel does not have to be square, but usually are. The most commonly used in max-pooling is a kernel of size 2x2 with a stride of 2 in both directions, which halves the size. This kernel and stride are used in our network.

2.1.3 Batch Normalization

When using large amounts of data during training, the data is often divided into smaller parts, called batches. These are propagated through the network one by one in order to decrease the required memory usage. Given a batch, the normalization is done by modifying the output from a prior activation layer and is used to make the learning faster and more robust. In practice this is done by subtracting the mean and dividing by the standard deviation of the output. By doing this, the weights between the hidden layers vary less and all weights have the same influence. Batch normalization also allows a layer to depend a little less on other layers to learn.

2.1.4 Activation Functions

Activation functions are used in a network to transform a node's input to its output. Using non-linear activation functions is essential since it introduces non-linearity into the network, which is important when learning complex behaviour. There are many types of activation functions, the ones used here are softmax, sigmoid and ReLU.

The rectified linear unit, more commonly known as ReLU, is described as

$$f_r(x) = \max(0, x),$$

which sets the output to zero if the input value x is negative, but is kept unchanged if x is positive. The ReLU function is shown in Figure 2.4a.

The sigmoid function is often used when performing a binary classification. It is described as

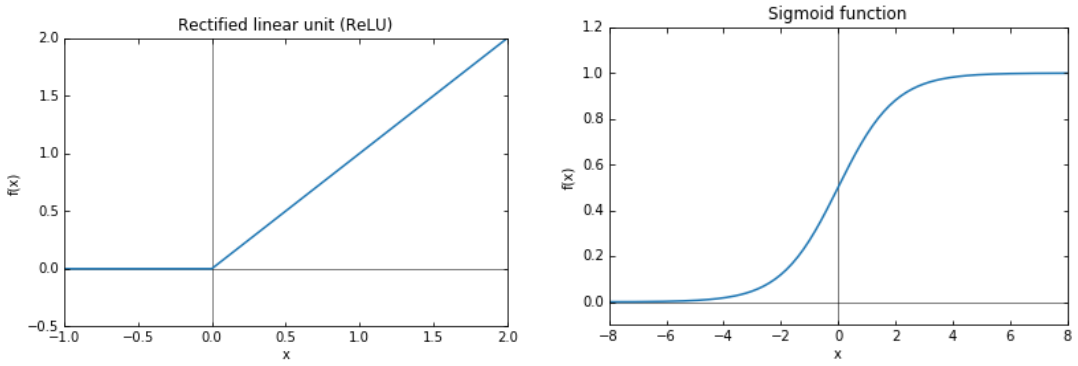
$$f_{sig}(x) = \frac{1}{1 + e^{-x}},$$

and is shown in Figure 2.4b. It contracts all values to the interval between 0 and 1.

The softmax function is a generalization of the sigmoid function and is used when having a multi-class classification, meaning there are more than two classes. This function is a described as

$$f_{soft}(x)_i = \frac{e^{x_i}}{\sum_j^C e^{x_j}},$$

where C is the number of classes and i is the class of interest. Like the sigmoid function, it scales the outputs for the different classes to all be between 0 and 1, but also makes sure that the outputs adds up to 1 so that they can be interpreted as probabilities.



(a) The ReLU function.

(b) The Sigmoid function.

Figure 2.4 – Two of the activation functions used in our model. Note that the scales are not the same.

2.1.5 Dropout

One major problem with deep neural networks is the risk of the network overfitting. This occurs when the network is adapting to the training data without being able to generalise the results. If an overfitted network is used, it will perform well on the images in the training data but not on other images. One can say that the network learns features specific for the training data by heart. Using dropout is a way to avoid this problem. This method works by randomly inactivating nodes by setting the weights to zero on all the outgoing edges, which has the same effect as simply removing the node from the network during training. This means that a node cannot trust that a neighbouring node catches the interesting structure but must itself learn important content. This prevents the network from learning structures that are specific for the training data and therefore makes it better at generalising. When designing the network, a hyperparameter is set to decide the likelihood of dropping a node. The higher the parameter, the more nodes will be dropped. Dropout is also a way of training different nets "at the same time", since the network design changes during training [14]. However, since the network is changing, the time until convergence, when the weights no longer change, is longer.

2.1.6 Augmentation

Augmentation is a commonly used technique to make the model more robust and to expand the training set without having to actually acquire more data. This is typically done with image modification like zooming, shifting or rotating the image so that the network will be trained on data with varying features. Augmentation also prevents overfitting as the input has more variation. By providing the data this way, the network should be able to correctly segment an object regardless of whether it is perfectly centred, has a different size than usual or is slightly deformed.

2.2 U-net

The U-net structure, first described in [12], consists of an encoder and a decoder part. The encoder, or contraction part, is a downsampling which extracts features from the input, similar to a CNN. The decoder, or expanding part, is an upsampling that translates the features back to a format similar to

the original input. This could be considered to be a way of copying information, but in the end only the most important parts of the input are desired as output. This is why the network is designed in a way that it is forced to learn and "copy" only the most important information. In the U-net the downsampling consists of convolutional and pooling layers and the upsampling is a combination of upconvolution and convolutional layers.

In addition to these encoder and decoder parts, the U-net also has a concatenation part. It allows information to pass the encoder part without having to make features out of it. This means that the network learns both information by being forced to create features in the deep part, but can also utilise the information in the features of the more shallower parts. An example of a U-net structure is shown in Figure 2.5.

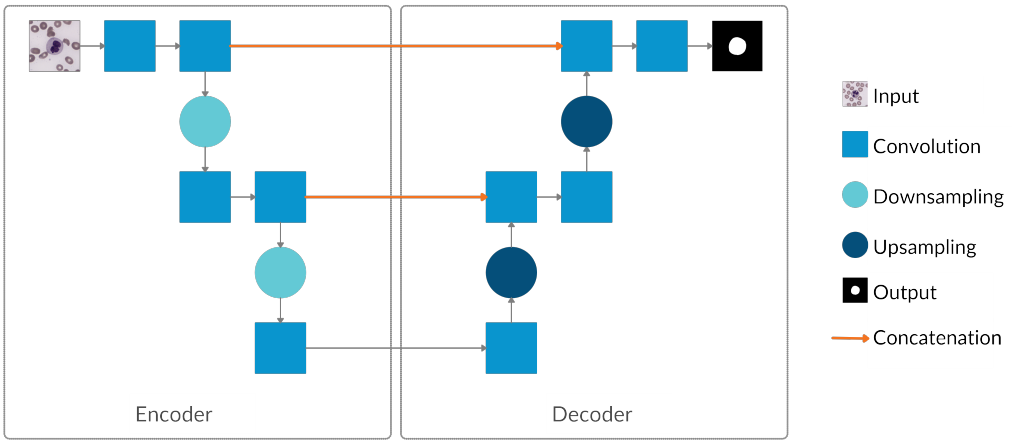


Figure 2.5 – An example of a simple U-net structure. The left part extracts features from the image and the right side creates an output image from these features.

2.2.1 Upconvolution

Upconvolution is a way of increasing resolution in the decoder part of the U-net. This is done with up-sampling, a process visualised in Figure 2.6 below. Starting from a small image, the 2x2 image to the left, the desired output is a larger image, containing the same information. First the rows are copied and inserted below the original row, as illustrated inside the dotted frame. Then the same is done with the columns, resulting in a doubling in size to the right. After the up-sampling step, a padded convolution is done, described in section 2.1.1.

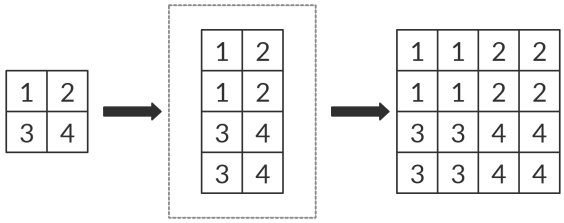


Figure 2.6 – Visualisation of up-sampling. This is used to increase the image resolution.

2.3 Specifics for Training

When training a network, it must somehow measure how well it performs, and what it should do to improve. This is the purpose of the loss function and the optimiser.

2.3.1 Loss Functions

The loss function is the network's tool to determine how well it performs. There are multiple types of loss functions and in our network we use binary cross-entropy loss when classifying binary images (cell and background) and categorical cross-entropy loss when segmenting into multiple classes (cytoplasm, nucleus and background). The binary cross-entropy loss is described as

$$CE = -\sum_{i=1}^{C=2} t_i \log(f_{sig}(s_i)) = -t_1 \log(f_{sig}(s_1)) - (1 - t_1) \log(1 - f_{sig}(s_1)),$$

where f_{sig} is the sigmoid function described in section 2.1.4. The parameter s_1 is the output from the network, and $f_{sig}(s_1)$ is the probability that the pixel is a cell pixel, taking a value between 0 and 1. The second parameter, t_1, is the ground truth value. This has a value of either 0 or 1 and it describes if the pixel is a cell pixel or not, according to the correct segmentation described in the ground truth. When the pixel truly is a cell pixel, meaning $t_1 = 1$, the loss is low if $f_{sig}(s_1)$ is close to 1 and high if $f_{sig}(s_1)$ is close to 0, see Figure 2.7. This function results in a high loss if the network is very sure about a wrong answer, in this case when $f_{sig}(s_1)$ is very close to 0.

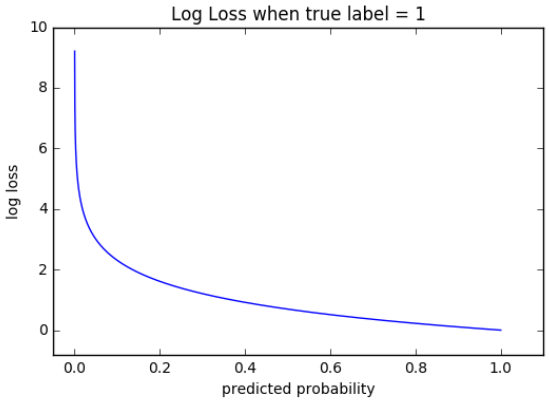


Figure 2.7 – Log-loss when the true label is one. Notice that the loss increases when the prediction is close to zero. Image from [2]

The other loss used in our network is the categorical cross-entropy which is described by

$$CE = -\sum_{i}^{C} t_i \log(f_{soft}(s)_i),$$

where C is the number of classes, t_i is the ground truth label for class i, and $f_{soft}(s)_i$ is the output from the network, using softmax, considering class i. This is similar to the equation for the binary cross-entropy when we have two classes, that is when $C = 2$, except that $f(s)$ is a sigmoid in the binary loss and a softmax the categorical loss. As the ground truth label is zero for all values except one, $t_i = t_p$, we can write out the entropy as

$$CE = -\log(\frac{e^{s_p}}{\sum_j^C e^{s_j}}),$$

where we simply replace the f with the softmax function.

2.3.2 Optimiser

To enhance performance of the network an optimiser is used. This is a function used to optimise parameters, which in this case means finding the optimal combination of the weights. There are many different optimisers, which all work a little differently. Most commonly used are optimisers based on gradient decent. The basic idea with this method is to first compute in which direction a set of weights should be changed according to the loss function, meaning which should be increased and which should be decreased. This is commonly done by differentiation, calculating a gradient that indicates how the weights should be changed to reduce the loss as fast as possible. The second step is to actually change the value in this direction by a small amount. How small is decided by a hyperparameter called learning rate. The optimiser used in our network is called Adam, which stands for adaptive moment estimation. It works by adding a fraction of the last gradient to the current one. In this way, the convergence is little faster when weights are differently distanced from their optimum.

In Figure 2.8 an example with stochastic gradient decent and two parameters, θ_0 and θ_1, is shown. The loss function, $J(\theta_0, \theta_1)$, is the surface coloured according to a scale where red is high and blue is low. Starting from the black cross near the top, the gradient of how to lower the loss as fast as possible is calculated to be the black line. Then a step is taken, with length according to the learning rate, and the next position is the next black cross. This is then repeated until the weights are adjusted to the dark blue concavity. Though this is a nice and quite common visualisation, it is important to note that the weight changes are made in the θ_0-θ_1 plane below which in turn results in a change in J-direction.

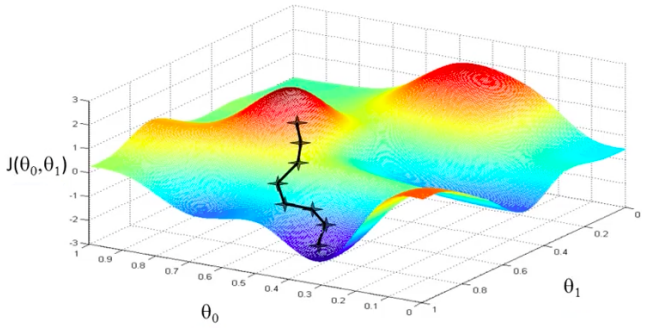
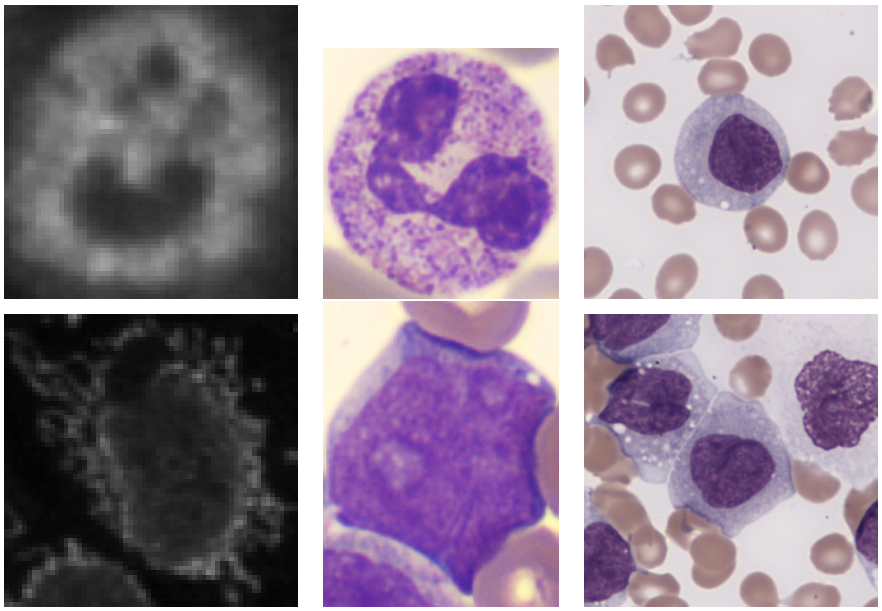


Figure 2.8 – Visualisation of the optimisation procedure. The parameters θ_i are updated to decrease the loss, $J(\theta_0, \theta_1)$. Image from [11].

3 Data

When developing the network, we first started working with a smaller data set with a size of 64 x 64 pixels, see Figure 3.1a. These black-and-white images were used to get started with the segmenting, and had no segmentation of the nucleus. The second data set consisted of RGB images with varying dimensions. As can be seen in Figure 3.1b, the cell constitutes most of the image and in many cases it stretched beyond the extent of the image. These images were used to develop the algorithm and try different models. It was also used to develop our evaluation methods, described in Section 4.2, and an interface for segmenting images by hand. Examples from the third and final data set used to produce the result are included in Figure 3.1c.



(a) Data set 1. (b) Data set 2. (c) Data set 3.

Figure 3.1 – Examples of images from the three data sets used when developing the network.

3.1 Data Specifics

The final data consisted of microscopic images containing one or more white blood cells and a corresponding black-and-white segmentation of the cell and nucleus, generated by the system described in Section 1.1. The images originate from samples stained with Wright-Giemsa, and were taken using CellaVision's system, producing images with a size of 480x480 pixels and a resolution of 10.2 pixels/µm. Since this data set contained exploded cells, colour residue from the staining and other artefacts which are not clinically relevant, some images were removed. The data was then divided into training, validation and test parts, containing 70%/15%/15% of the total amount of data respectively, resulting in a partition of 1856/398/398 images respectively. To make the model more robust and to expand the data set without having to manually segment more data, we chose to do augmentation to the training set. This was done by rotation, shifts in both height and width, shear, zoom, and horizontal flip.

3.2 Generating Ground Truth

For the binary ground truth the provided black-and-white segmentation of the cell was used in most cases, when the active contour method gave a satisfactory result. Images for which the system did not produce a good segmentation were more or less altered. Small corrections where done only on the faulty part but more severe errors where re-segmented entirely by free hand drawing.

For the multi-class segmentation, the provided segmentation of the nucleus was corrected in the same way as the cell segmentation. The already corrected cell segmentation for the binary case was then combined with the nucleus segmentation to form an RGB image with background as blue, cytoplasm in green and nucleus in red. Both sets of ground truth thus contain segmentations done both by the system and by hand.

3.3 Challenges in Segmentation

The nucleus of the cell is usually a well defined area, clearly separated from other nuclei in adjacent cells and from other similar components like RBCs. It is usually well stained and is therefore often easy to segment. However, the segmentation of cytoplasm or whole cells was not always straight forward and many questions about what should be included or not were raised during the process of generating the ground truth.

3.3.1 Multiple Cells

The active contour method was designed to not include cells that were not fully included in the image. Ideally, there should be one image of each cell where that cell is centred and other close-by cells may be in the picture. For this reasons, there may be multiple cells in an image and a cell might be included in many images. The classification and segmentation of the image should only be based on the cell that is centred. This raised the question of whether our method should only segment the middle cell, all cells that were fully included in the image, or all cells even if some are partly outside the image.

3.3.2 Smudge Cells

Another challenge was the presence of smudge cells and cells where the cytoplasm was barely visible. Smudge cells are leukocytes that have been ruptured during the preparation of the blood smear and appear as only a nucleus that may be slightly or very smudged. Smudge cells have long been seen as an artefact with no clinical relevance, but have been proved to predict survival in patients with Chronic Lymphocytic Leukemia [10]. It is therefore important to detect and correctly classify these cells, but not as important to have a correct segmentation. In some cases, the cytoplasm is still visible around the nucleus, but there is rarely a clear outline. In these cases, it may be correct to only segment the nucleus, since it has an easily defined boundary and the cell outline might not be a reliable measurement after breaking. However, in images of non-ruptured cells with low contrast or where the cytoplasm colour is similar to the background the network should correctly segment the cell outline. The similar appearance of these cells provide difficulties to generate satisfying segmentations.

3.3.3 Platelets

Thrombocytes, also called platelets, have been another source of discussion. Platelets are usually significantly smaller than leukocytes, in which case they can be easily classified as background, but sometimes large platelets can be mistaken for white blood cells during pre-location. If the sample has started to coagulate, there might also be clusters of platelets, also mistaken for a WBC. These are interesting to detect, but since the segmentation is not of interest to perform CNR it does not really matter how well it is segmented. In addition it is usually difficult to determine which platelets are a part of the cluster and therefore the ground truth for such an image is very hard to determine.

More difficulties arise when a platelet is close to a WBC as segmentation methods usually include the platelet, which causes error in the CNR.

3.3.4 Other Difficulties

Other difficulties when segmenting a white blood cell is faded or blurred edges that are hard for a network to separate from the background. The same problem happens when a vacuole, which looks like a bubble with the same colour as the background, is at the very edge of a cell, and the segmentation gets an indent.

Sometimes a WBC is squeezed in between a cluster of RBCs, generating protrusions and a very jagged edge, and some WBCs have protrusions without evident reason. Other cells have similar colour to the surrounding RBCs which is problematic as the network excludes what it believes to be part of a red blood cell.

3.4 Our Classification

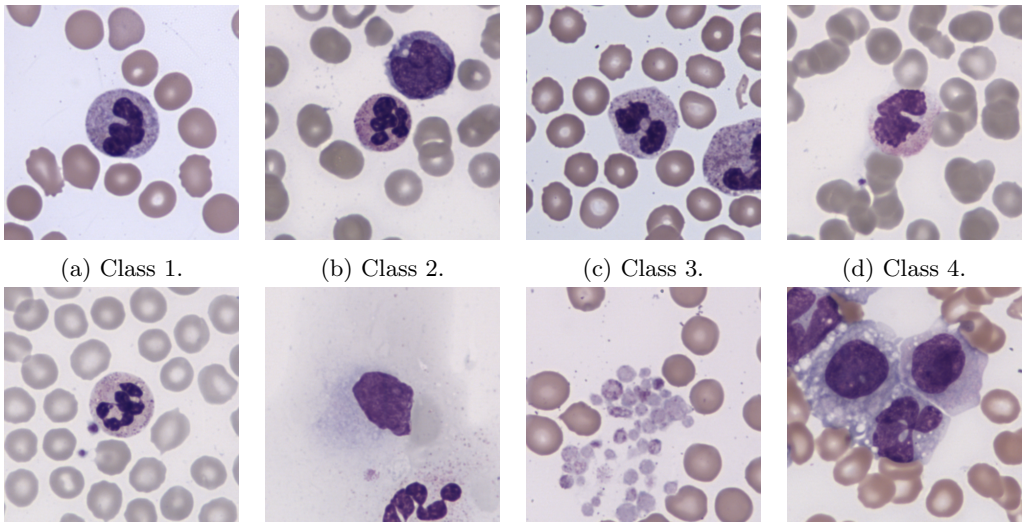
We have divided our data into 8 classes based on segmentation difficulty in order to facilitate evaluation of our method. A higher class number corresponds to a more challenging image to segment. These classes are not related to the classification of leukocytes used during the white blood cell count and differential. In Table 3.1 it is displayed how well represented the different classes are in the test and train set. Examples of images from the different classes can be seen in Figure 3.2 and are explained as follows:

1. Easy cells - images with a single cell that is easy to segment.
2. Multiple cells - two or more, clearly separated cells that are fully contained in the image.
3. Half cells - some cells are partially outside the image, but clearly separated from one another.
4. Difficult cells - images with cells that are difficult to segment. Other cells or platelets may be present.
5. Adjacent platelets - images with platelets adjacent to the cell.
6. Smudge cells - images with smudge cells. Other cells or platelets may be present.
7. Platelets - platelet cluster or single platelets, may contain other cells.
8. Adjacent cells - images with two or more adjacent cells. The adjacent cell might be partly outside the image.

Some of the images could belong to multiple classes, for example a smudge cell with other cells beside it. In these cases, the images were classified in the class with the highest number, accounting for the most difficult feature.

Data set	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8
Test (%)	68.1	1.3	2.8	8.0	7.8	5.8	3.0	3.3
Train (%)	67.7	0.8	4.1	9.5	5.8	4.9	2.4	4.8

Table 3.1 – Distribution of cell images in test and training data.



(a) Class 1. (b) Class 2. (c) Class 3. (d) Class 4.

(e) Class 5. (f) Class 6. (g) Class 7. (h) Class 8.

Figure 3.2 – Examples of images in different classes.

4 Method

4.1 Network Design

The network we designed is a U-net which is greatly inspired by networks previously used for segmentations, in particular [7]. It was built using Keras, which is a deep learning library for Python, that works with GPU processing. It runs on top of Tensorflow, which is a library for developing machine learning models. The structure of the network design for binary segmentation is shown in Figure 4.1.

To reduce time and computer memory, the input is re-scaled images sized 128x128 pixels. The contracting part consists entirely of convolutions with a 3×3 kernel, followed by batch normalizations and ReLUs, all three combined and shown as blue boxes. The down-sampling is done with max-pooling with a 2×2 kernel which halves the size, shown in light blue circles. In the bottom layer, after the first convolution, a second convolution is done with a kernel of 1×1, shown as dark red box.

The expanding part is done with upconvolutions with up-sampling factor 2×2, meaning that both the number of rows and number of columns are doubled, shown with dark blue circles. Then, again, the expanding part are mostly convolutions with 3×3 kernel, followed by batch normalization and ReLUs, again blue boxes. But now, information from the contracting part is added with concatenation, the orange arrows. When reaching the top a dropout layer with hyperparameter set to 0.5, green diamond, is added before the final convolution with 1×1 kernel and sigmoid activation function, shown as light red box, which produces the output. All convolutions are padded which means that the size does not change between layers except for the max-pooling and upconvolutions. As there are equally many max-pooling layers as upconvolutions, the output will also be 128x128 pixels.

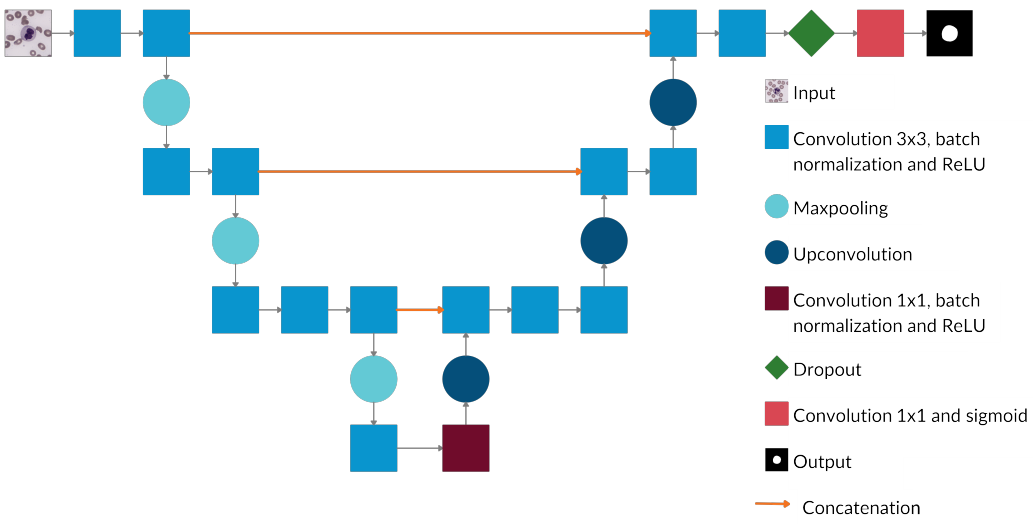


Figure 4.1 – The network design for binary cell segmentation.

This network uses the Adam optimiser with learning rate 0.0001. The loss function is a binary cross-entropy loss since the problem is a pixel-wise binary classification.

In addition to this network to produce the binary segmentation, it was slightly modified to also be able to handle multi-class segmentation where the cytoplasm and nucleus are separate classes. To do this the last convolution before the output is changed from sigmoid to instead be a softmax activation, and having 3 output channels to generate RGB images. The optimiser is still Adam with learning rate 0.0001, and the loss is categorical cross-entropy as the classification is no longer binary.

4.2 Evaluation

To evaluate the performance of our network, several evaluation methods where used.

4.2.1 Pixel-by-Pixel Accuracy

The first, and probably most intuitive one is pixel-by-pixel accuracy. Each pixel of the produced segmentation is compared to the ground truth segmentation, and the accuracy is then calculated by

$$\frac{1-(P_{FP}+P_{FN})}{P_{\text{tot}}} = \frac{P_{TP}+P_{TN}}{P_{\text{tot}}},$$

where P_{FP} is the number of false positives, meaning pixels that is background in the ground truth but cell in the segmentation, and P_{FN} is the opposite, the number of false negatives, pixels that is cell in ground truth but background in the segmentation. On the other side of the equality, P_{TP} is the number of true positives, pixels that should be cell according to the ground truth and is also cell in the segmentation. The number of true negatives, pixels that are correctly classified as background, is denoted P_{TN}. The the total number of pixels in the image is P_{tot}. In other words, the accuracy is the percentage of correctly classified pixels. This measurement is simple and easy to understand, but is not very informative when a large majority of the pixels all belong to one class. For example, an image with only 10% of the pixels being cell would get a 90% accuracy when classifying all pixels as background.

4.2.2 Intersection over Union

The second method is Intersection over Union, or IoU. As the name implies, this is a quotient where the numerator is the intersection of pixels that are classified as cell in the ground truth image and the generated segmentation and the denominator is the union of pixels classified as cell in either one or both segmentations. More specifically it is produced by

$$\text{IoU} = \frac{P_{TP}}{P_{FP}+P_{TP}+P_{FN}},$$

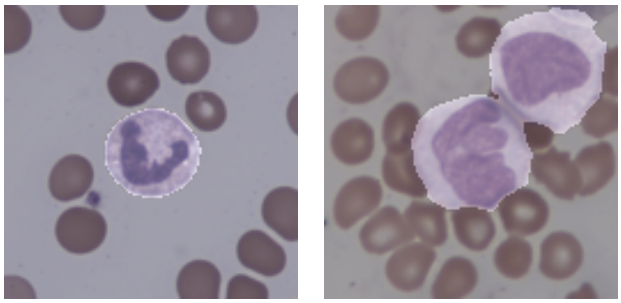
where P_{TP}, P_{FP} and P_{FN} are the same as in the accuracy measurement above. For the multi-class segmentation, an IoU score is generated for each class individually.

As the ground truth is made using the active contour method, the IoU will in all non-handsegmented cases give 100% match. To reduce this bias we used a strategy inspired by [7]. In this approach, a border which is not included in the IoU measure is added between classes. This measure is denoted IoU_b. This means that segmentations from our network with only a thin line of incorrectly classified pixels at the edge where the segmentation changes from one class to another will have the same IoU_b as the active contour segmentation.

4.2.3 Visualisation

In addition to the numerical evaluation methods, the results were visualised to easier determine which errors were made and how serious they were.

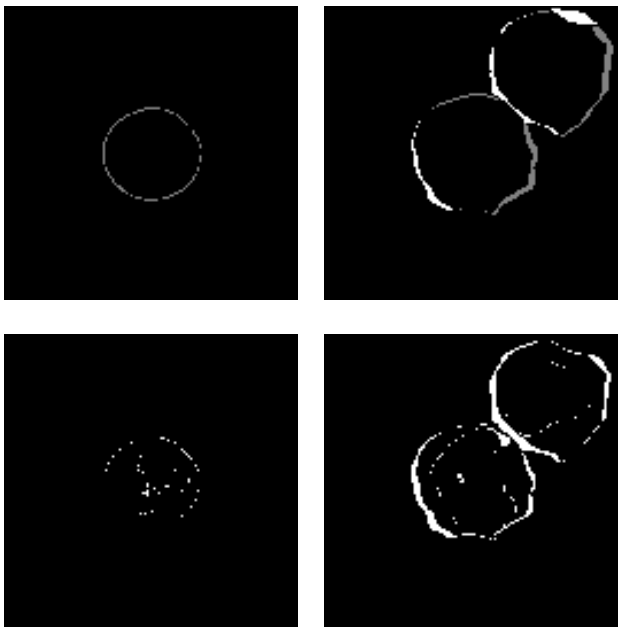
To study how well the generated segmentation matched the original image, a faded version of the binary segmentation was overlapped with the image. This method proved to be difficult to use for evaluation as it was hard see small differences, see Figure 4.2.



(a) Class 1. (b) Class 2.

Figure 4.2 – Example of the visual evaluation using overlap for images with easily segmented cells (class 1) and multiple cells (class 2).

Instead an evaluation based on the ground truth segmentation was used, where images displaying the difference between the ground truth and produced segmentation are made. For the binary classification both true positives and true negatives, meaning all correctly classified pixels, are displayed in black, false positives in white and false negatives in grey, see top of Figure 4.3. For the multi-class case, the visualisation is again black for all correctly classified pixels but white for all incorrectly classified. A coloured version of of false negative and false positives for all three classes would not facilitate evaluation. These are shown at the bottom of Figure 4.3.



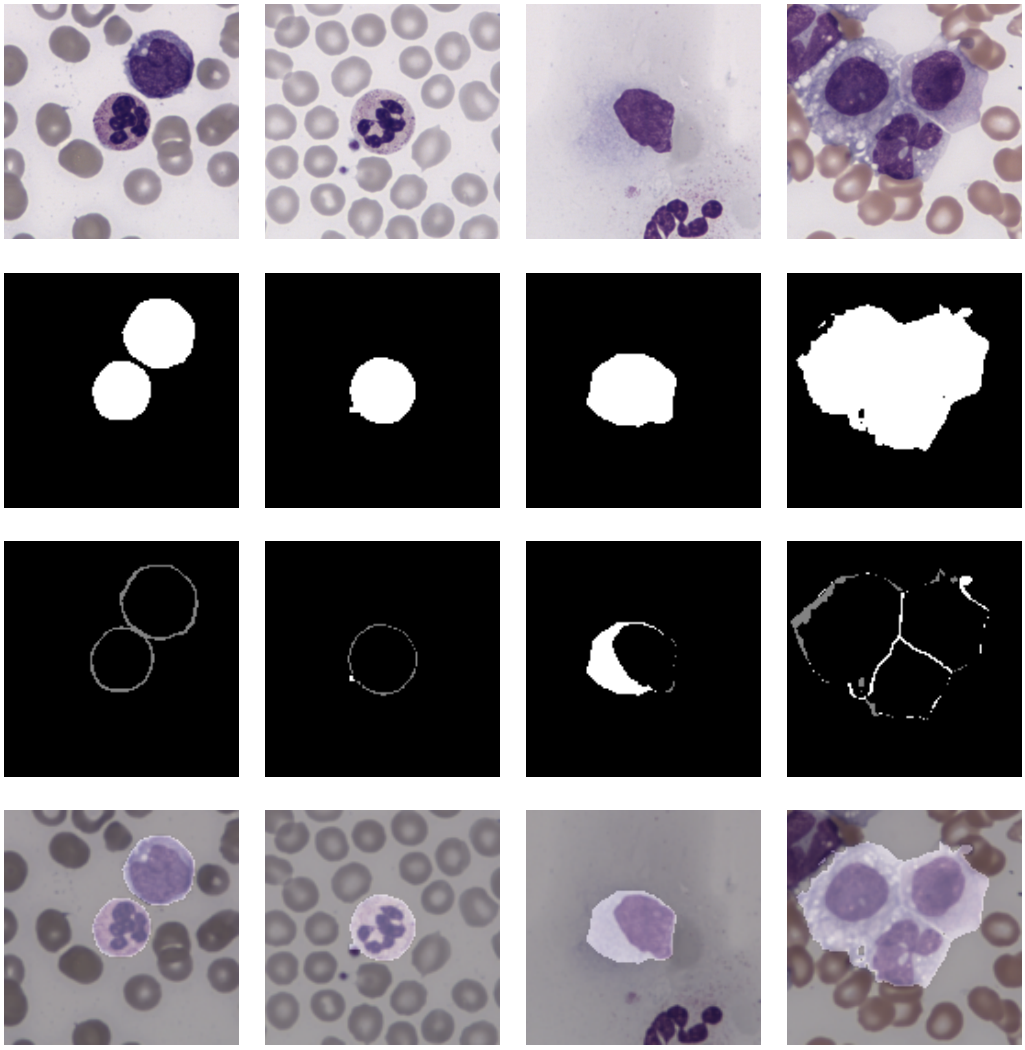
(a) Class 1. (b) Class 2.

Figure 4.3 – Example of the visual evaluation using ground truth segmentation. The top two images are from a binary evaluation and the bottom two from a multi-class evaluation.

5 Results

5.1 Binary Segmentation

Our binary segmentation method achieved an accuracy score of 99.5% and a mean IoU score of 93.9%. Examples of resulting segmentations and evaluation images for test images with multiple cells, adjacent platelets, smudge cells and cell clusters (class 2, 5, 6, and 8) can be seen in Figure 5.1. The result from active contour and ground truth are show in Figure A.1 in appendix.



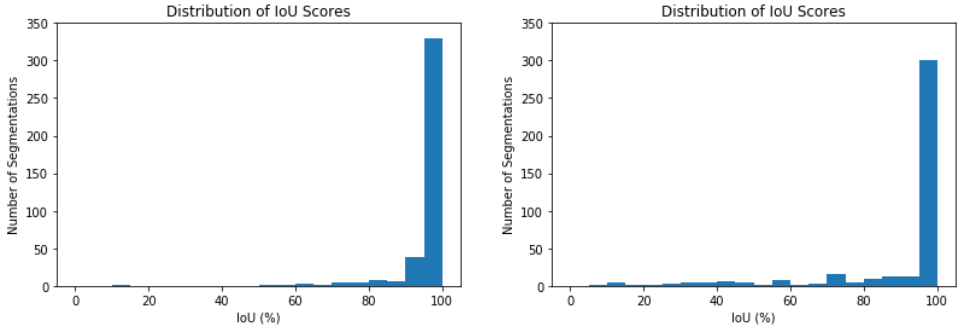
(a) Class 2. (b) Class 5. (c) Class 6. (d) Class 8.

Figure 5.1 – Examples of our resulting segmentations (second row), evaluation (third row), and overlap images (last row) for class 2, 5, 6 and 8. Notice that the segmentations for class 2 and 5 are fairly accurate while class 6 and 8 were more difficult to segment.

As seen in Figure 5.1, our method managed to satisfactory segment cells that were clearly separated and had an easily defined border. When platelets or cells were too close together, it often segmented them as a single cell. It also struggled to find the borders of smudge cells where remains of the cytoplasm was visible around the nucleus as well as to classify vacuoles as part of the cell. The resulting mean IoU and IoU_b scores for both the active contour method and the U-net model are presented in Table 5.1. Here it is seen that the IoU_b score for the U-net was better than for the Active Contours method. The distribution of the IoU_b score on the test images are shown in Figure 5.2. These figures show that almost all segmentations (more than 350/398) produced by the U-net reached a score of above 90% of which most are above 95%. Although most of the active contours segmentations (about 320/398) reach an score of more than 90%, the ones that score below 90% yield scores down to about 30%. The U-net segmentations that do not reach 90% more often get scores around 60-80%.

Method	Accuracy	Loss	IoU (mean)	IoU_b (mean)
U-net	**99.5%**	0.0223	93.9%	**96.7%**
Active contour	**99.5%**	-	**96.2%**	91.0%

Table 5.1 – Comparison between CellaVision's segmentations using active contours and our U-net on binary segmentation. Best result are in bold.



(a) Scores for the U-net segmentations.

(b) Scores for the active contours segmentations.

Figure 5.2 – Distribution of IoU_b scores for the binary segmentation.

As previously mentioned, the data was divided into eight classes depending on how difficult they were to segment. In order to measure how well our method segmented different types of images the IoU_b scores were calculated for the classes. The resulting performance for our network and the active contour method are presented in Table 5.2. Here it is shown that the U-net performed better overall, and especially for class 2, 6, 7, and 8 (multiple cells, smudge cells, platelets and adjacent cells). However both methods struggled with the last three classes.

Class	1	2	3	4	5	6	7	8
U-net (%)	**98.7**	**96.3**	**95.1**	**95.8**	**95.7**	**84.5**	**82.1**	**93.2**
Active contour (%)	94.9	87.0	91.4	93.6	90.9	78.6	36.3	82.8

Table 5.2 – The mean IoU_b score for the different cell classes when performing binary segmentation.

Another way of comparing the two methods is to compare the images with the lowest accuracy for each method. Images where both methods perform well are not as interesting to study since most methods can yield a decent segmentation. When deciding which method to use, it is more interesting

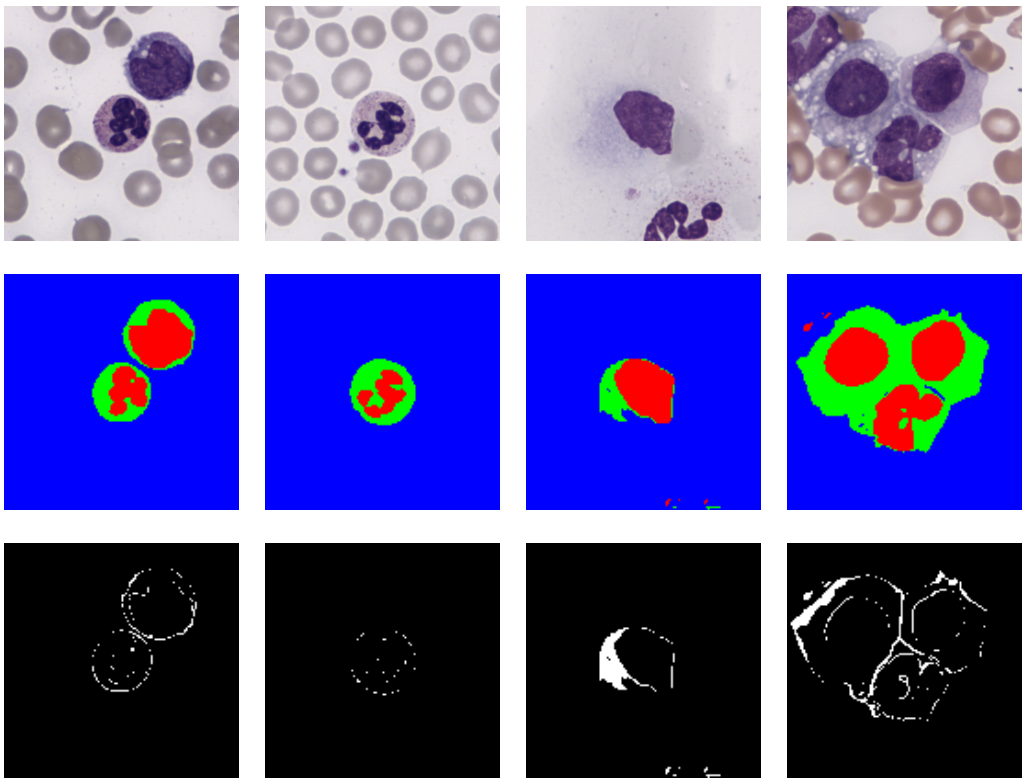
to see which method performs best on images that are difficult to segment. To do this comparison, the N images with the lowest scores for the U-net method, and the N images where the Active Contours method resulted in the lowest scores were chosen. Since the methods often struggled with the same images, there was an overlap of about 50%. These images were then used to calculate the IoU scores. As seen in Table 5.3 the U-net consistently performed better than the active contours method.

IoU Active contour	IoU U-net	Number of images, N	Number of images total
54.2%	**66.0%**	10	15
67.4%	**78.1%**	30	46
73.1%	**82.9%**	50	73

Table 5.3 – Mean IoU scores without considering borders for the N images with the lowest accuracy for each method.

5.2 Multi-class Segmentation

When classifying the images into multiple classes, our method achieved a mean IoU score of 94.5% for nucleus pixels and 82.8% for cytoplasm pixels. The resulting segmentations and evaluation images are shown in Figure 5.3. The result from the active contour method and ground truth is shown in A.2 in Appendix A.



(a) Class 2. (b) Class 5. (c) Class 6. (d) Class 8.

Figure 5.3 – Examples of our resulting segmentations (second row) and evaluation (third row) for class 2, 5, 6 and 8.

Looking at the last row of Figure 5.3, where falsely classified pixels are shown in white and correctly classified pixels in black, it can be seen that the method struggles most with determining the cell outline and not as much with finding the nucleus. The method performs well as long as the cells are far enough apart but has trouble finding the cell outline in smudge cells with remains of cytoplasm around them and separating vacuoles from background. It manages to differentiate adjacent platelets from cells in about half of the images in class 5, one example is shown in Figure 5.3b.

The resulting accuracy and mean IoU_b scores for both the active contour method and the U-net model are presented in Table 5.4. The distributions of these scores are shown in Figure 5.4.

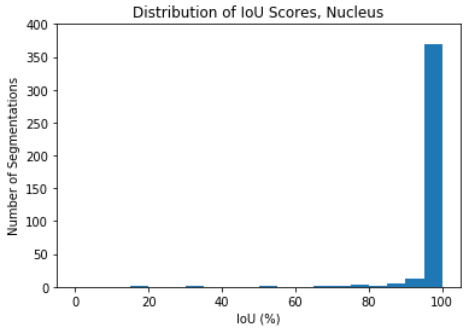
Method	Accuracy	Loss	IoU (mean)	IoU_b (mean)
U-net	98.8%	0.0240	94.5%	98.1%
Active contour	**99.0%**	-	**98.5%**	**98.4%**

(a) Segmentation of nucleus.

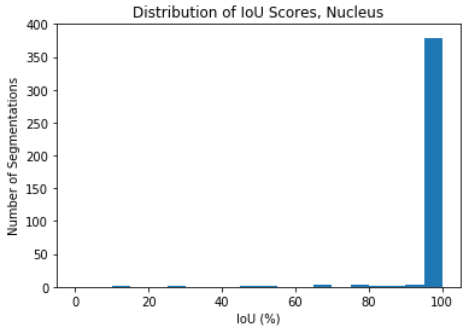
Method	Accuracy	Loss	IoU (mean)	IoU_b (mean)
U-net	98.8%	0.0240	82.8%	88.7%
Active contour	**99.0%**	-	**92.3%**	**91.9%**

(b) Segmentation of cytoplasm.

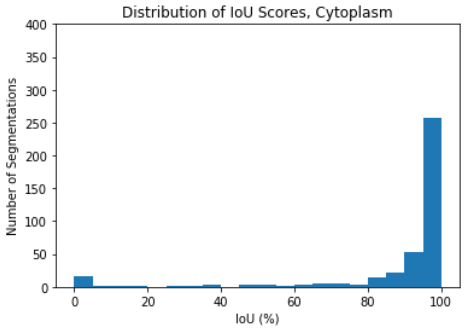
Table 5.4 – Comparison between CellaVision's segmentations using active contours and our U-net on multi-class segmentation.



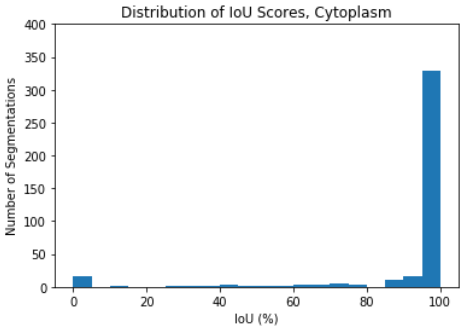
(a) Scores for the nucleus in U-net segmentations.



(b) Scores for the nucleus in active contours segmentations.



(c) Scores for the cytoplasm in U-net segmentations.



(d) Scores for the cytoplasm in active contours segmentations.

Figure 5.4 – Distribution of IoU_b scores for the multi-class segmentations.

These figures show that the scores for nucleus pixels almost always end up between 95-100% for both methods while the cytoplasm scores are more spread out. In Figures 5.4c and 5.4d there are a few segmentations with a score below 5%. These are all cells that do not have any cytoplasm in the ground truth, such as smudge cells or platelets (which are classified as nucleus), which results in 0% score.

As before, it is interesting to see how well the network performed on different types of images. The resulting IoU_b scores for the different classes are presented in Table 5.5. Here it is shown that the methods performed well on different types of images. The U-net method was better at segmenting images with multiple cells, both when they were close together and further apart as well as cells that are difficult to segment. The active contours method performed better for easily segmented cells, images with adjacent platelets and platelet images. Both methods struggled to segment smudge cells.

Class	1	2	3	4	5	6	7	8
U-net (%)	99.0	**98.6**	**94.6**	**98.7**	97.1	98.5	83.1	**94.6**
Active contour (%)	**99.6**	89.1	90.6	97.8	**99.5**	**99.3**	**92.6**	85.6

(a) The mean IoU_b scores for nucleus segmentation.

Class	1	2	3	4	5	6	7	8
U-net (%)	95.1	**90.0**	90.1	**89.3**	92.3	15.8	69.4	**89.6**
Active contour (%)	**98.0**	82.3	**91.6**	85.9	**96.4**	**33.5**	**89.6**	80.3

(b) The mean IoU_b scores for cytoplasm segmentation.

Table 5.5 – The mean IoU_b score for the different cell classes when performing multi-class segmentation.

6 Discussion and Conclusion

The purpose of this project was to construct a neural network that correctly segments white blood cells in microscopic images of blood smears. This was to be done by binary classification and multi-class classification using a U-net. In most cases the network produced very good results with IoU scores around 90%. This is relatively high compared to previous studies, which presented scores between 72% and 92%. It also had a more even IoU distribution and stable performance than the active contours method and generally produced better segmentations for the images that were most difficult to segment correctly. However, there are some limitations with this project, mostly regarding the data and ground truth segmentation since it was based on segmentations produced with the active contours method.

6.1 Data

All our images origin from only a few blood samples and are stained using the same method. This means that there is not much variation among them, neither in colour nor in content. This may result in the network having difficulties generalising to other samples with more variety or differences in staining. Also, as one can see in Table 3.1, the majority of the data is of standard images with one easily defined and easily segmented WBC. There were relatively few difficult images, and only a small number of images that had the same type of difficulties. This means that the network is trained mostly on single, clearly distinctive WBCs during training, but not so many clusters for example, which makes it hard for the network to perform well on these types of images.

The exclusion of images that where falsely selected as WBCs, such as staining artefacts or remains of damaged cells, is not problematic as these images are not clinically interesting, and do therefore not contribute to the data set. These images are taken due to an error made in the pre-location of cells and should ideally not have been included. Since the network is not trained on this type of images we do not know what it would do when presented with them, which it likely would be in a real life setting as pre-location algorithms are not flawless. However, the segmentation the network tries to produce when presented with such an image is not useful or interesting, and how it does it and why is not of value to know. This makes removing these images justified.

6.2 Ground Truth

The active contours method is designed to only segment WBCs that are fully included in the image, which is reasonable as it is not possible to calculate the CNR for parts of cells. We believe that our method would be better at only determining whether a pixel belongs to a WBC or background, regardless of whether a cell is only partially in the image or not, since it seems to have difficulties excluding cells far from the image centre. Ideally, we would like to include all WBCs in the ground truth image, but since it is based on the active contours method, changing the ground truth would require an amount of work that is not within the scope of this project.

Another issue concerning ground truth is its segmentation accuracy. Most of the ground truth is produced with the active contours method and some images are hand segmented. The hand segmented images might not be 100% accurate since it was not done by professionals. As all data was manually reviewed to determine whether it should be corrected or not, some errors may have been missed and might still be present in the ground truth set.

6.3 Evaluation Methods

To evaluate our method in comparison to the active contour method used in the current system, some creative solutions had to be found. The current method naturally scored very high in all easily segmented cells as it actually was the ground truth. However, since the border between cell and background is not perfectly unambiguous, there are many possible segmentations that are equally true, varying only by a few pixels along the cell outline. For example, we see that the resulting segmentations for class 2 and 5 in Figure 5.1 only differ from the ground truth in some of the outermost pixels. This is the case for most of our images since our network seems to consistently make the segmentations a few pixels smaller than the ground truth. Since the ground truth is not segmented with an accuracy of one pixel, it is not possible to say whether our network or the ground truth performs the most correct segmentation in these cases. These segmentations should therefore be considered to have a 100% accuracy. This is the reasoning behind creating the IoU_b measure where border pixels are excluded. As the ground truth is down-sampled to be able to compare it to the segmentation generated by the U-net, this border needs to be about 4 pixels in the original image to become about 1 pixel wide in the down-sampled version. Examples of ground truth images with border pixels are shown in Figure A.3 in Appendix A.

6.4 Network Performance

The network seems to be good at differentiating between cells and background, but struggles with separating WBCs that are close together. This presents a problem when the network finds two or more cells which it segments and counts as one, for which the system then calculates the CNR. When looking at Table 3.1 we see that images containing adjacent cells, class 8, only accounts for 4.8% of the training data and 3.3% of the test data. This means that the network barely has a chance to learn how to segment those images correctly. Even if there where more examples of this kind, the error caused by segmenting nearby cells as one will typically be very small since the border pixels only account for a small part of the image. This makes these types of cells difficult to segment in general, and it is not surprising that neither method performs well on these images. In addition to this, our network might have difficulties differentiating between adjacent cells or platelets due to the down-sizing. Smaller images of the size 128×128 pixels are used as input to the network, which makes the line separating the cells about 1-2 pixels wide. This line is probably blurred after just one down-sampling layer, and would likely not be found unless it had been extracted as a feature during the first two convolutional layers. If these features are found, the concatenation layers are supposed to account for the effects of down-sampling by using feature maps from the uppermost layers. This is however not achieved since there are so few images with adjacent cells and clusters. This difficulty with differentiating between nearby cells might be possible to eliminate if more data with clusters, platelets and nearby cells is used for training or if larger images could be used as input. Another strategy could be to use object detection before segmentation in order to separate individual cells.

The network also does not perform well on images of smudge cells and platelet clusters, class 6 and 7. This does not present a problem though as neither are of interest to segment. As mentioned in Sections 3.3.2 and 3.3.3, we do not calculate a CNR value for either platelets or smudge cells. Platelet clusters are however important to find since they indicate that the blood has started to coagulate. The presence of smudge cells is relevant since it has proven to predict survival in patients with Chronic Lymphocytic Leukemia. Ignoring them might also result in a skewed differential if some cell types are more likely to smudge than others. Therefore, these cells are of interest to find and classify, which is why they where included as data in this project. This means that it does not matter if the network performs well on these cells, and segmentations such as the one in Figure 5.1c can be considered good even though they include some of the remains of the cytoplasm and not only the cell nucleus. These segmentations do however, even though they are to be considered acceptable segmentations, cause a considerable decrease of the IoU score.

6.5 Future Work

Although the network generally performs well, there are some areas that would be interesting for further studies. To enhance network performance, it would be desirable to train a network to learn how to handle cells that are not fully contained in the image and cells with adjacent cells or platelets, which are categorised as class 3, 5 and 8 (see Figure 3.2). This would likely be possible for the network to learn, but would require more training data of that specific type. This could be done by generating training data, either with a generative adversarial network (GAN) that generates images with the same characteristics as its training data, or by merging additional half cells, whole cells and platelets into an image.

In order for the method to be practically useful in a system performing white blood cell count and differentiation, the segmentation of a single image should not take more than 10-20 ms. This limits the depth of the network, the amount of layers, and the image resolution. In this project, we have focused on researching whether it is possible to segment blood smear cell images with deep learning, and how to achieve the best possible segmentations rather than developing a product for clinical use. With this said, it might be possible to decrease the image resolution further or use a smaller model to achieve faster calculations, and still get an acceptable result even if it most likely would produce less accurate segmentations.

6.6 Conclusions

It is possible to segment white blood cells using a neural network with a U-net structure. The network performs well in most images but has difficulties with segmenting nearby cells and excluding cells that are partially in the image. This could however be improved by training the network more on these types of images.

References

[1] Saad Ullah Akram et al. "Cell segmentation proposal network for microscopy image analysis". In: *Deep Learning and Data Labeling for Medical Applications*. Springer, 2016, pp. 21–29.

[2] Algorithmia. *Introduction to Loss Functions*. URL: `https://blog.algorithmia.com/introduction-to-loss-functions`.

[3] Martin Blumenreich. "The white blood cell and differential count". In: *Clinical Methods: The History, Physical, and Laboratory Examinations, 3rd edition*. 1990, pp. 724–727.

[4] CellaVision. *CellaVision Annual report 2018*. 2018.

[5] Konrad Gjertsson. "Segmentation in Skeletal Scintigraphy Images using Convolutional Neural Networks". In: *Master's Theses in Mathematical Sciences*. 2017.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. ISBN: 9780262035613.

[7] Johan Isaksson et al. "Semantic segmentation of microscopic images of H&E stained prostatic tissue using CNN". In: *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2017, pp. 1252–1256.

[8] Adam Karlsson. *Area-Based Active Contours with Applications in Medical Microscopy*. Licentiate Thesis. 2005.

[9] Golnaz Moallem et al. "Detecting and Segmenting White Blood Cells in Microscopy Images of Thin Blood Smears". In: *2017 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)* (2017), pp. 1–8.

[10] Grzegorz S. Nowakowski et al. "Percentage of smudge cells on routine blood smear predicts survival in chronic lymphocytic leukemia". In: *Journal of Clinical Oncology* (2009), pp. 1844–1849.

[11] Reina. *Stochastic Gradient Descent with Restarts*. URL: `https://thisgirlreina.wordpress.com/2018/07/11/stochastic-gradient-descent-with-restarts/`.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

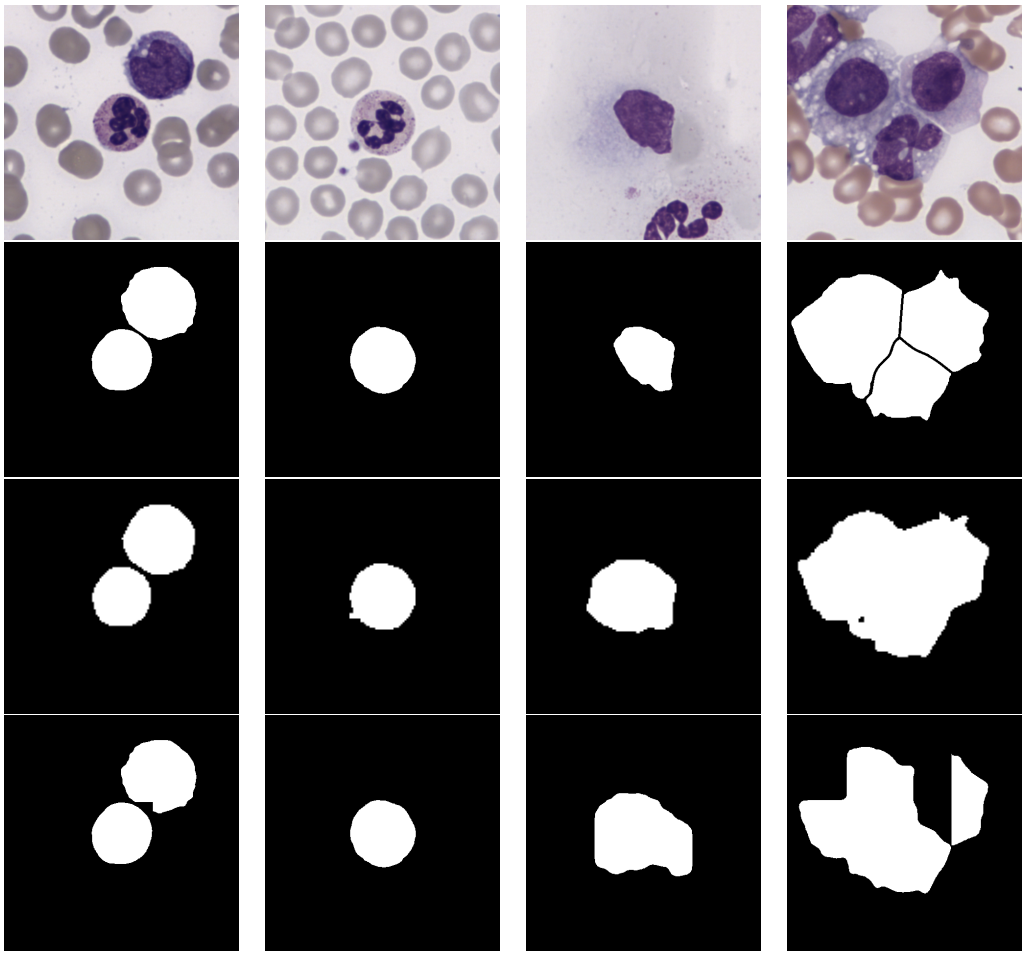
[13] Farnoosh Sadeghian et al. "A Framework for White Blood Cell Segmentation in Microscopic Blood Images Using Digital Image Processing". In: *Biological Procedures Online, volume 11* (2009).

[14] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overtting". In: *Journal of Machine Learning Research* (2014).

[15] Fuyong Xing et al. "Towards pixel-to-pixel deep nucleus detection in microscopy images". In: *BMC Bioinformatics volume 20* (2019).

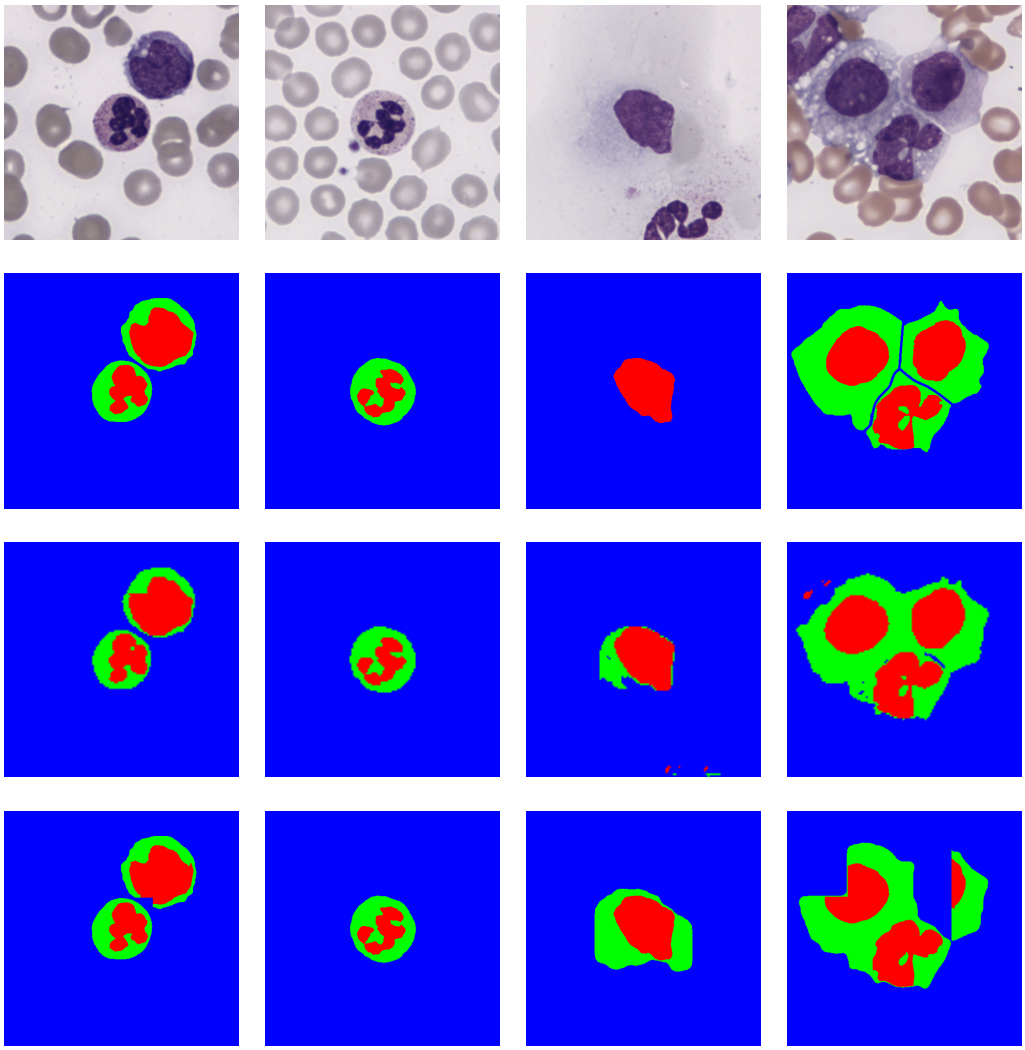
[16] Zongwei Zhou et al. "Unet++: A nested u-net architecture for medical image segmentation". In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Springer, 2018, pp. 3–11.

Appendix A



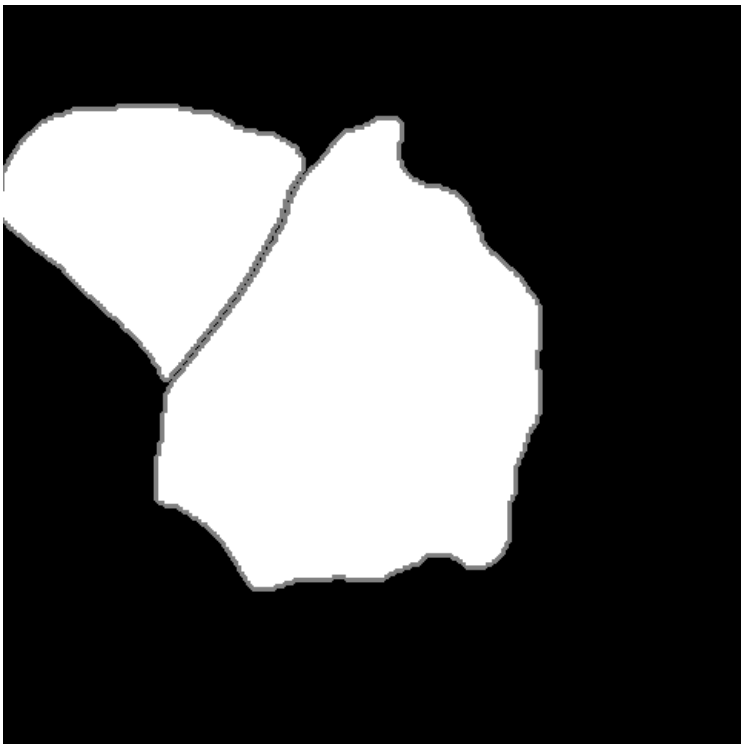
(a) Class 2. (b) Class 5. (c) Class 6. (d) Class 8.

Figure A.1 – Examples of original images (first row), ground truth segmentations (second row), binary segmentations produced with the U-net model (third row), and segmentations produced with the active contour method (last row) for class 2, 5, 6, and 8.

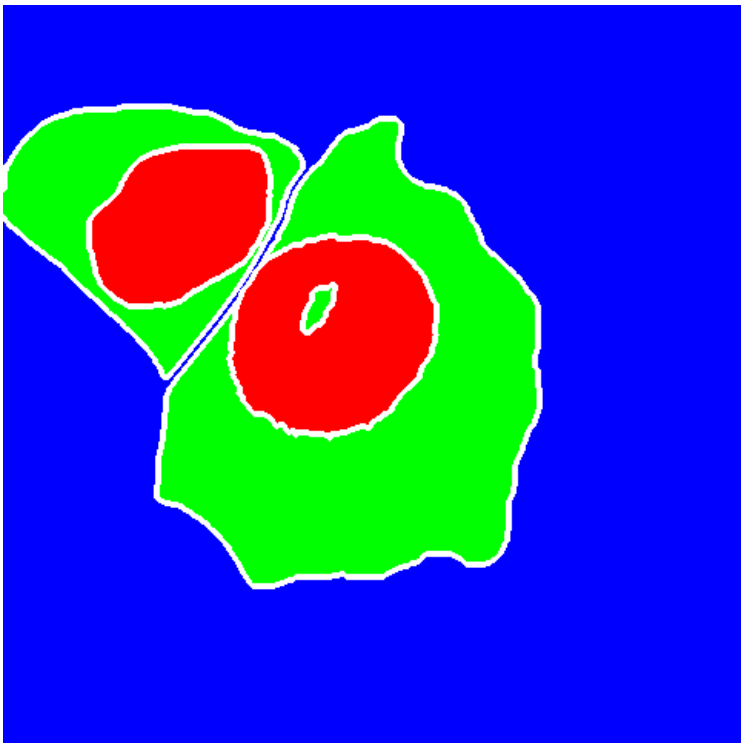


(a) Class 2. (b) Class 5. (c) Class 6. (d) Class 8.

Figure A.2 – Examples of original images (first row), ground truth segmentations (second row), multi-class segmentations produced with the U-net model (third row), and segmentations produced with the active contour method (last row) for class 2, 5, 6, and 8.



(a) Ground truth image for binary segmentation.



(b) Ground truth image for multi-class segmentation.

Figure A.3 – Example of a ground truth images with pixels classified as border. Images of these types were used to calculate IoU_b.