UPTEC X 22020
Examensarbete 30 hp
Juni 2022

UPPSALA UNIVERSITET

Sample Image Segmentation of Microscope Slides

Maija Persson

Civilingenjörsprogrammet i molekylär bioteknik

UPPSALA
UNIVERSITET

Sample Image Segmentation of Microscope Slides

Maija Persson

Abstract

In tropical and subtropical countries with bad infrastructure there exists diseases which are often neglected and untreated. Some of these diseases are caused by parasitic intestinal worms which most often affect children severely. The worms spread through parasite eggs in human stool that end up in arable soil and drinking water. Over one billion people are infected with these worms, but medication is available. The problem is the ineffective diagnostic method hindering the medication to be distributed effectively. In the process of designing an automated microscope for increased effectiveness the solution for marking out the stool sample on the microscope slide is important for decreasing the time of diagnosis. This study examined the active contour model and four different semantic segmentation networks for the purpose of delineating the stool sample from the other parts of the microscope slide. The Intersection-over-Union (IoU) measurement was used to measure the performance of the models. Both active contour and the networks increased the IoU compared to the current implementation. The best model was the FCN-32 network which is a fully convolutional network created for semantic segmentation tasks. This network had an IoU of 95.2%, a large increase compared to the current method which received an IoU of 77%. The FCN-32 network showed great potential of decreasing the scanning time while still keeping precision of the diagnosis.

Teknisk-naturvetenskapliga fakulteten

Uppsala universitet, Utgivningsort Uppsala

Handledare: August Tynong Ämnesgranskare: Carolina Wählby

Examinator: Siv Andersson

Populärvetenskaplig sammanfattning

I tropiska och subtropiska länder med dålig infrastruktur är inälvsmaskar en vanligt förekommande sjukdom som främst skapar stort lidande för barn. Maskarna lägger ägg i människan som sedan sprider sin avföring vidare till jorden och det riskerar även att kontaminera vatten. Runt om i världen är ca. 1,7 miljarder människor smittade med inälvsmaskar. Men det finns väl fungerande mediciner mot denna sjukdom. Problemet är att det tar tid att diagnostisera sjukdomen och därmed minskar möjligheterna att få rätt behandling vid rätt tidpunkt. Den nuvarande metoden för diagnostisering görs av välutbildade människor genom att räkna ägg i avföringsprover med hjälp av ett mikroskop. Därefter bestäms infektionens intensitet baserat på hur många ägg det finns per gram avföring. Detta är en långsam metod som även är beroende av tillgången till utbildad personal. Effektivisering av diagnostiseringen skulle bidra till snabbare vård och minskat lidande.

För att lösa detta problem utvecklar Etteplan ett mikroskop som använder artificiell intelligens för att räkna äggen och specificera vilken art av inälvsmaskar som äggen kommer ifrån. Detta minskar det manuella arbetet som krävs och minskar därmed även tiden det tar att ställa en diagnos. I processen att gå från ett avföringsprov till diagnos så används en metod som heter Kato-Katz. Detta innebär att en viss mängd avföring överförs på en glasskiva som sedan används i mikroskopet för att titta på äggen. En människa kan tydligt se detta runda område med avföring och bestämma vart man vill titta närmare med mikroskopet. Men hur får man ett automatiserat mikroskop att göra samma sak?

Den nuvarande lösning som används av Etteplan baseras på traditionell bildanalys. I denna metod ritas en cirkel ut runt provet som är begränsningen för var mikroskopet ska scanna. Cirkeln kan justeras manuellt innan scanningen startas. Men om provet inte är helt runt och något utsmetat medför denna metod att mikroskopet scannar och sparar flertalet bilder på områden där ingen avföring finns. I detta projekt jämförs 2 olika typer av metoder, en äldre mer klassisk metod kallad aktiv kontur och fyra mer moderna djupinlärningsnätverk, med nuvarande metod i syfte att avgränsa provområdet. En metod som automatiskt kan ringa in området och avgränsa provområdet mer specifikt skulle medföra reducerad tid innan diagnos och därmed mer effektiv vård mot inälvsmaskar.

Utvärderingen av modellerna gjordes på bilder tagna på glasskivor med avföringsprov från flertalet olika studier utförda i olika länder för att samla data. De modeller som utvärderades visade god potential till att förbättra specificiteten för avgränsningen av provet jämfört med nuvarande metod. Allra bäst resultat visade ett av de fyra nätverken. Resultaten från detta nätverk var inte långt ifrån att kunna jämföras med en avgränsning markerad av en människa. Nätverket gjorde således ett bra jobb i att hitta de särdragen som utmärkte avföringsprovet från resterande delar på glasskivan. Det tränade nätverket visade stor potential för att hjälpa till att reducera tiden för scanningen utan att göra avkall på precisionen för diagnostiseringen av inälvsmaskar. Med andra ord skulle nätverket kunna se till att alla ägg räknas samtidigt som så lite bakgrundsbilder som möjligt sparas.

Contents

Abbreviations

Adam	Adaptive Moment Estimation
AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
FCN	Fully Convolutional Network
FNR	False Negative Rate
FPR	False Positive Rate
GPU	Graphical Processing Unit
IoU	Intersection-over-Union
mIoU	mean Intersection-over-Union
NTDs	Neglected Tropical Diseases
ReLU	Rectified Linear Unit
STHs	Soil-Transmitted Helminthiasis

1 Introduction

In tropical and sub-tropical countries there exists diseases grouped as Neglected Tropical Diseases (NTDs). Soil-transmitted Helminthiasis (STHs) and schistosomiasis are diseases which has been defined as NTDs. These infections are caused by parasitic intestinal worms and can be transmitted through contaminated soil or water. The parasites spread via human feces that contain parasite eggs and also through skin penetration by the larvae (Cheesbrough 2005). Worldwide, there are 1.7 billion people infected with either one of the most common STHs; roundworm (*Ascaris lumbricoides*), hookworm (*Necator americanus* and *Ancylostoma duodenale*) or whipworm (*Trichuris trichiura*). The infections are most common in developing countries and mainly affect children with symptoms such as abdominal pain, diarrhea and anemia (de Silva *et al.* 2003; Parija *et al.* 2017).

There are effective medications for treatment of STH infections, however, the diagnostics take time and require human expertise. An effectivisation of the diagnostics of STHs would increase the possibility of people receiving the right treatment in time. Therefore, Johnson & Johnson have now collaborated with Etteplan in creating an automated microscope for diagnostics of STHs. The microscope uses artificial intelligence (AI) to classify and count parasite eggs in human stool samples. This is a solution which decreases the manual work and speeding up the time for diagnosis.

1.1 Stool sample preparation and STHs diagnostics

For the diagnostics of STHs and schistosomiasis, studying feces samples is of interest since this is where the parasite eggs are found. One way of preparing feces samples and looking at the parasite eggs is by using a laboratory method called Kato-Katz (Cheesbrough 2005). This method is recommended by the World Health Organization (WHO) for STH diagnosis and consists of several preparation steps of the stool sample (Tankeshwar 2016).

The result of using the Kato-Katz method is ideally a round area of stool sample on a microscope glass slide shown in Figure 1. The slide is viewed under a microscope in order to see the parasite eggs. The eggs are classified i.e. determined which species they belong to based on shape, size and other characteristics and they are also counted. The counting is done to calculate the number of eggs per gram of feces, which determines the intensity of infection (Cheesbrough 2005; Tankeshwar 2016).

Figure 1: Image of microscope slide with the stool sample to the left and information about the sample to the right.

1.2 Project objective

In this project, a traditional image segmentation method called active contour and segmentation using deep learning networks were evaluated as methods for delineating the stool sample from the rest of the microscope slide. The performance of the methods were also compared to the current implementation used by Etteplan for marking out the sample. The active contour model was also used as an automated annotation tool for annotating a larger dataset used when training the deep learning networks.

The low resolution images of whole microscope slides, such as Figure 1, were used in this project for boundary detection. After marking out the sample, the microscope scans this area and collects high resolution images of the stool sample. An improved method which marks a more precise boundary of the sample would make sure that the microscope only collects high resolution image data of relevant parts of the microscope slide. This would then contribute to a reduced scanning time and more effective diagnosis of STHs.

2 Background

Finding and extracting information from the relevant parts in images is a common problem. One method used for this is image segmentation, where images are divided into different image objects. One example of this would be to find the boundary of a football placed on a lawn. When segmenting the image it would be split so that each pixel in that image which belongs to the ball would be assigned with the class ball and all other pixels would be assigned with the class lawn. This way of assigning pixels to classes is called semantic segmentation (Thoma 2016). Semantic segmentation does not separate similar objects from each other, e.g. multiple balls on a lawn. If objects are separated, it is instead called instance segmentation (Figure 2) (Hafiz & Bhat 2020).

Figure 2: A visualisation of the differences between semantic and instance segmentation.

There are several classical image segmentation techniques such as thresholding, watershed, active contours and mean shift (Szeliski 2010). But newer models based on deep learning have also been developed, resulting in an increase in performance of the segmentation. Some of the most commonly used deep learning methods for computer vision are convolutional neural networks (CNNs), fully convolutional networks (FCNs),

recurrent neural networks (RNNs), encoder-decoders and generative adversarial networks (GANs) (Hesamian *et al.* 2019; Minaee *et al.* 2020).

There are many networks created for image segmentation problems, such as U-net (Ronneberger *et al.* 2015), SegNet (Badrinarayanan *et al.* 2017) and DeepLab (Chen *et al.* 2017). These networks have been trained on large datasets, one of them being ImageNet (Russakovsky *et al.* 2015), and have achieved high performance results for segmentation problems. A popular way of using deep learning networks when the amount of labeled data is small is to use already trained networks and adapt it to your own data, this is called transfer learning (Minaee *et al.* 2020).

2.1 Active contour

Active contour is an image segmentation technique used for finding boundaries, lines and edges in images. The active contour model is also called snakes, where a snake is described as a spline influenced by forces in the image as well as external constraint forces. The snake tries to minimize its energy which consists of internal spline energy together with the external constraints and the image forces. The internal spline energy contain two terms controlled by the weights α and β (Kass *et al.* 1988). The active contour or snake consists of several points connected by lines, creating a curve which can be either closed or open (Bakoš 2007).

The active contour model is implemented as a function in *Scikit-image v.0.19.1* (van der Walt *et al.* 2014). The function is based on the model presented by Kass et al. (1988). An initial guess of a contour is required and this consists of a given number of points in a given sequence. The function also allows for setting the weights α (alpha) and β (beta) to different values. Alpha is described as the snake length parameter and regulates the contraction of the snake whereas beta is the snake smoothness shape parameter that regulates the smoothness of the snake.

Further, the function also allows for regulation of attraction to brightness and to edges with the parameters w_line and w_edge. Maximum number of iterations to optimize the snake can also be set as well as the maximum pixel distance to move each iteration. Lastly, there is an explicit time stepping parameter γ (gamma), boundary condition parameter and convergence criteria parameter implemented in the function (Python documentation 2022).

2.2 Convolutional neural network

In the area of image analysis and computer vision Convolutional Neural Networks (CNNs) are commonly used. A CNN is a neural network designed to work on image data for feature extraction and pattern recognition. As the classical artificial neural network (ANN), CNNs also consist of neurons that optimize weights by learning. However, the architecture of the networks allows features specific to images to be accounted for (O'Shea & Nash 2015). The great benefit with using CNNs on image data compared to ANNs is the reduced

amount of parameters in the model, which reduces the computational complexity (O'Shea & Nash 2015; Albawi *et al.* 2017).

A CNN is built with multiple layers such as convolutional layers, pooling layers and there is also an input layer which holds the values of the pixels in the input image. A CNN can also have fully-connected layers which connect all nodes from e.g. a pooling layer to the nodes in the fully-connected layer. The fully-connected layers are placed in the ending of the network due to its ability to be used for classification (O'Shea & Nash 2015).

The convolutional layers are where the feature extraction occurs. The features are extracted with linear operations called convolutions. A matrix of numbers, usually of size 3×3, 5×5 or 7×7, is called a kernel or filter and is applied to the input in a convolutional layer. The kernel traverses over the input and the elementwise product is calculated and then summed which results in a feature map (Figure 3). This operation can be done with multiple kernels in the layer where each kernel creates its own feature map. After performing a convolution the output is passed through a non-linear operation called an activation function (Yamashita *et al.* 2018). The most common activation function is the rectified linear unit (ReLU) (Krizhevsky *et al.* 2012).

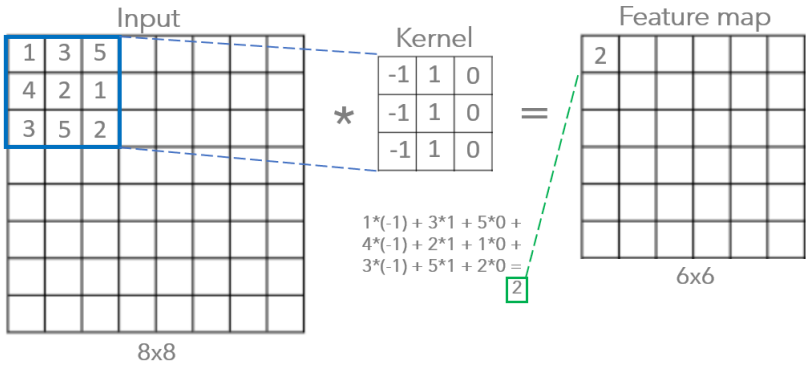


Figure 3: A visual representation of a convolution step.

When the kernel is sliding over the input it moves according to the stride. The stride determines how many pixels the kernel should move before calculating the next number in the feature map. The stride is often set to 1, but can be altered to a higher value to downsample the input (Albawi *et al.* 2017; Yamashita *et al.* 2018). In the convolution step, any information that exists at the borders of the image is lost. This is due to the inability of the convolution operation to overlap the kernel with the outermost element of the input. This problem can be solved by something called zero padding, where zeroes are added to the borders of the input, preventing the feature map from decreasing in size in the convolution step (Yamashita *et al.* 2018).

Other than using the stride to downsample the input, one can also use pooling layers. This is used to reduce the number of learnable parameters i.e. the complexity of the model. One of the most common pooling operations is max pooling. Max pooling layers

often have kernels of size 2×2 which move along the input with stride 2. At each position the highest value is extracted to create a feature map (Figure 4) (O'Shea & Nash 2015).

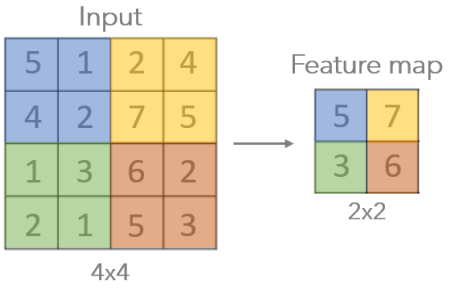


Figure 4: A visual representation of a max pooling operation with kernel 2x2 and stride 2.

2.2.1 VGG-16

One CNN that showed high performance results for localisation and classification in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014, a challenge with the purpose of enabling researchers to evaluate algorithms on a high object variety labeled dataset, is called VGG-16. The network was presented by Simonyan and Zisserman (2015) and it is made up of 16 trainable layers and 5 max-pooling layers. The kernel sizes in the convolutional layers are no larger than 3×3, the stride is 1 and padding is used to keep the spatial resolution after convolution. All hidden layers in the network are followed by the ReLU function. The detailed network architecture is shown in Figure 5.

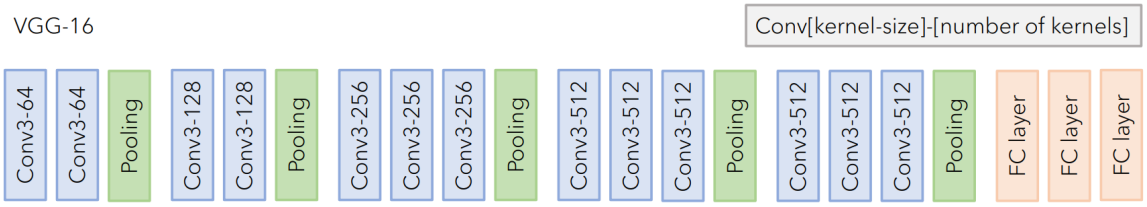


Figure 5: The network architecture of VGG-16.

This network is popular for image classification tasks and also for using it in transfer learning. However, some disadvantages with VGG-16 are the size of the trained network (500MB) and the training time. To train the network on the ImageNet dataset took 2-3 weeks (Rohini 2021).

2.3 Fully convolutional networks

A Fully Convolutional Network (FCN) has an encoder-decoder structure and is typically used for semantic segmentation. The encoder extracts features using a CNN architecture where the fully connected layers are replaced with convolutions. The decoder performs upsampling by transposed convolutions (deconvolution) to transform the low resolution feature map to full resolution. The last part of the FCN is a softmax activation function which is applied to get the pixel-wise classification i.e. a segmentation mask. An FCN can

take any input size and will produce a segmentation mask with corresponding size (Long *et al.* 2015; Xing *et al.* 2020).

2.3.1 FCN-32, FCN-16 and FCN-8

Long et al. (2015) describes the FCN and proposes three different architectures of the network, with and without additional connections to lower layers in a VGG-16 network. These are called FCN-32, FCN-16 and FCN-8, where FCN-32 has no additional connections to low level layers. In FCN-32 the output from the final layer in the CNN network is upsampled 32 times to match the size of the input image. FCN-16 has one additional connection in its' architecture where a 1×1 convolutional layer is added to the output from pooling layer 4 (pool4) and the result is summed with the result from upsampling the final layer of the CNN. For the FCN-8, the result from the first connection is upsampled and summed with the output from performing a 1×1 convolution on pooling layer 3 (pool3). The output is then upsampled back to the full image size, 16 times for FCN-16 and 8 times for FCN-8. This means that the FCN-8 architecture holds two connections while FCN-16 holds one (Figure 6). According to the authors, these added connections enable prediction of finer details in the upsampled output.

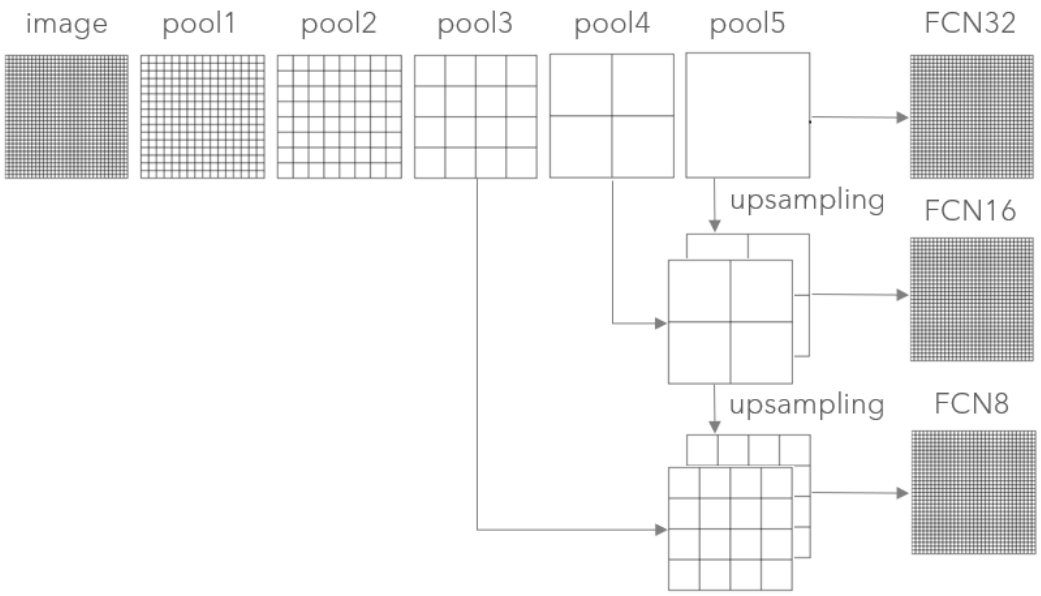


Figure 6: The differences in network architecture between the fully convolutional networks; FCN-32, FCN-16 and FCN-8. Pooling layers (pool1, pool2, pool3, pool4 and pool5) are representing the five pooling layers from the VGG-16 network.

2.3.2 SegNet

SegNet is also a fully convolutional network used for semantic segmentation. It is built with an encoder-decoder structure where the encoder has an architecture like the VGG-16 network. As the network is fully convolutional the fully connected layers have been removed in the encoder, resulting in 13 layers. The decoder also has 13 layers, corresponding to the layers in the encoder. In the max-pooling layers the positions of the maximum values i.e. pooling indices are stored inorder to be used by the decoder for non-linear upsampling. By

doing this, the network captures boundary information from the encoder layers in an efficient way. Apart from the upsampling layers, the decoder also has trainable convolution layers in an architecture corresponding to the encoder layers. The final layer in the decoder is fed to a softmax layer to perform pixel-wise classification (Badrinarayanan *et al.* 2017). SegNet is a smaller network in the sense of number of trainable parameters compared to other networks with competing architecture (Minaee *et al.* 2020).

2.3.3 U-Net

Another network which has shown high performance on segmentation tasks is the U-Net. Ronneberger *et al.* (2015) describes the network architecture as consisting of a contracting part and a symmetrical expanding part. The contracting part has an architecture of a typical CNN with convolutions followed by ReLU and pooling layers. The expanding part consists of up sampling, ”up-convolutions” and regular convolutions. A concatenation is done between the corresponding feature map from the contracting part and the step in the expanding part. This creates connections between the expanding and contracting parts. They also show that the network can be trained on a small amount of data using data augmentation to achieve high performance. This is why the network is suitable for medical image segmentation tasks where large annotated datasets can be hard to retrieve.

2.4 Training and testing networks

When training a network model with a CNN architecture it is the kernels which are optimized or trained. This is often done by supervised learning where there exists a ground truth to each element in the dataset and the model tries to minimize the difference in its output and the ground truth (Yamashita *et al.* 2018). When performing semantic segmentation the ground truth is a segmentation mask with the same size as the image and each pixel is labeled with the class it belongs to.

The learning procedure is often done with an algorithm called backpropagation (Yamashita *et al.* 2018). Since the values in the kernels are randomized at first, the model will probably perform badly i.e. have large errors between the output and the ground truth when predicting an output initially. The kernel values therefore have to be updated according to the errors during the training process. The evaluation of the difference between the output and the ground truth is done with a loss function (Ho & Wookey 2020). Cross-entropy is a loss function commonly used for classification problems. The values of the kernels are updated using an optimizer algorithm which adjusts the kernel values with regards to the loss function. Gradient descent is a common method for optimization (Yamashita *et al.* 2018). However, there have been developments made to this optimizer algorithm which has improved efficiency when working with larger datasets and models with many parameters. One such algorithm is called Adaptive Moment Estimation (Adam) (Kingma & Ba 2017).

Before training a model it is important to divide the available dataset. The dataset should be divided into two or three smaller datasets, either one for training and one for testing or one for training, one for testing and one for validation. The training set is of

course used for training the model. The validation set is used to tune hyperparameters and select a model. Hyperparameters are set before the training process starts e.g. kernel size, activation function, optimizer and number of epochs. The testing dataset is only used for evaluating the selected model and this set should consist of data unseen by the network inorder to perform a non-bias performance evaluation (Yamashita *et al.* 2018).

Other than tuning hyperparameters and selecting models, a reason for dividing datasets into smaller sets is to detect the problem with overfitting i.e. training a model to high accuracy and small loss on training data but receiving lower accuracy and higher loss on the validation set. Overfitting can be handled by regularization, data augmentation and batch normalization. It can also be mitigated by increasing the training data (Yamashita *et al.* 2018). One regularization method is called dropout. The dropout technique works by randomly setting input to zero with a certain frequency to reduce overfitting (Hinton *et al.* 2012). Batch normalization is a method which normalizes layer input, this makes the deep learning networks less sensitive to the initial values in the network and also reduces the number of epochs needed to achieve high performance (Ioffe & Szegedy 2015).

When training a network one can decide to split the training dataset into batches. The batches all have the same size and the size decides how much data the network can see at each iteration. A higher batch size requires more memory when training the network. One epoch is when all the batches have been iterated.

2.4.1 Transfer learning

Small datasets can be challenging when trying to train a generalizing network. One way of handling this problem is by transfer learning. This is a method where you train the network on a small dataset but with weights and kernels which are already pre trained on a dataset. The main idea is that features learned on one task can be used on similar tasks with additional training on smaller amounts of data. Some CNNs, like VGG16 and ResNet, have been trained on large datasets such as ImageNet and the weights and kernels have been made available for public use.

One way to use the pretrained CNN is by fixed feature extraction, where the fully-connected layers in the network are removed and only the convolutional and pooling layers are left. This allows for creating a new classifier suitade for a task that might not be as similar to the task which the CNN was pretrained on (Yamashita *et al.* 2018).

2.5 Evaluation metrics

To evaluate the performance of a segmentation method, Intersection-over-Union (IoU) or mean Intersection-over-Union (mIoU) measurements can be used. These measurements are used when there is a ground truth image which the segmented image can be compared to. The mIoU is used when there are several classes and is defined as the average IoU over those classes. The IoU is defined as the area of intersection between the output segmentation map and the ground truth divided by the area of union between the same two elements (Long *et al.* 2015; Minaee *et al.* 2020).

$$IoU = \frac{A \cap B}{A \cup B} \tag{1}$$

The IoU value can take any value between zero and one, where values close to zero indicate poor segmentation and values close to one indicate good segmentation. In this project the IoU is calculated for images in a dataset, and the mean IoU within the dataset is then referred to as the IoU of the method. Not to confuse with the mIoU measurement.

To get an estimation of how much a model overestimates and underestimates a segmented area, the metrics false positive rate (FPR) and false negative rate (FNR) could be used. They are defined as follows, were FP is false positives, TN is true negatives, FN is false negatives and TP is true positives:

$$FPR = \frac{FP}{FP + TN} \tag{2}$$

$$FNR = \frac{FN}{FN + TP} \tag{3}$$

Another measurement which can give an indication of performance of a segmentation model is pixel accuracy. This metric is defined as the percentage of correctly classified pixels in an image.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4}$$

3 Material and method

3.1 Libraries and tools

Image data can be represented as *NumPy* arrays. This allows for easy processing and manipulation of image data. It is also required for usage of libraries such as *Pillow, OpenCV, Scikit-image* and *Matplotlib* for image processing and display. *Pillow* or *Python Image Library (PIL)* can be used to both read and write images e.g. PNG or JPEG files. Another library with similar functions as *Pillow* is *OpenCV*. Both of these libraries were used in this project for image processing. To display images and plot data, the library *Matplotlib.pyplot* was used. In the deep learning part of the project *keras* was used. *Keras* is an API built on *tensorflow* for enabling easy development of deep learning networks.

3.2 The data

The data in this project was made up of images of microscope glass slides with prepared stool samples on them as well as other information regarding the samples, such as a QR code and written text. These images were all of size $846 \times 2142 \times 3$. Throughout the development of the AI microscope, Etteplan has carried out studies in different countries to

collect data. In the available database at Etteplan there were images from several studies.

It was decided to use images which were representative of the current microscope imaging process and up-to-date. The images used were therefore extracted from study 26, 27, 29, 30 and 31. In total, 238 images were retrieved from those studies. Study 28 contained images where the sample was in a square shape. Square samples were no longer used within the microscope project and were therefore excluded from the dataset. The distribution of the number of extracted images in the dataset from each study was not even since there were different numbers of images in each study to begin with (Figure 7). Making the distribution even would have severely decreased the size of the dataset which was why an uneven distribution was accepted.

The testing data was the set which was first retrieved in this project. This was done to evaluate the active contour model and SampleFinder and their potential for being used for automatic annotation. After this, additional images were retrived from the database to create the larger dataset of 238 images. This resulted in an uneven distribution between the datasets within one study (Figure 7). This is not optimal as the distribution within one study would preferably be even with regards to the image percentage distribution between the datasets.

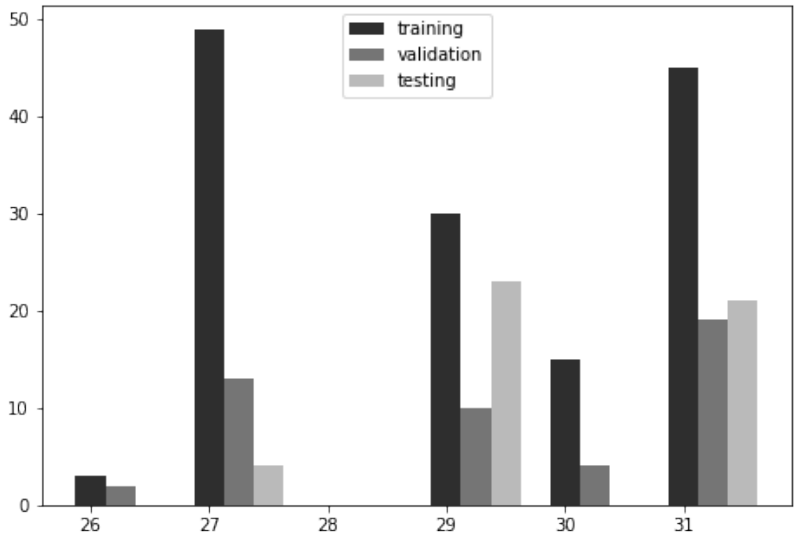


Figure 7: Barplot of number of images from each study in the three different datasets; training, validation and testing.

The dataset consisted of images where the stool sample had varying size and quality. With quality meaning how distinct the sample boundary was and how well the texture of the sample matched with the rest of the images. Images with samples which were smeared out, uneven in thickness or had deviating texture were regarded as lower quality. A more high quality image was characterized by a sample with a clear boundary and high contrast between sample and background. An example of a higher quality image and one of lower quality can be seen in Figure 8.

(a) Higher sample quality image.

(b) Lower sample quality image.

Figure 8: Two example images from the dataset, one with a higher quality sample and one with lower quality sample.

3.3 Manual annotations of images

48 images of varying quality from the dataset were annotated using *Gimp v.2.10.30*. The annotation was made using the intelligent scissor to mark the boundary of the sample. The area inside the boundary was filled with white color and the outside (background) was filled with black color. The resulting binary mask was saved as a 1-bit PNG image. This meant that the pixel value of the sample area was 1 and the background had pixel value 0.

This was done both by me (the author) and one other person. The reason for annotating the images twice was to make a comparison of how two humans can interpret a ground truth in different ways. The IoU between these ground truths was calculated as well as the mean percentage of sample pixels over the dataset, which would indicate the differences in generosity when marking out the sample in the images. The percentage of sample pixels was calculated by dividing the number of pixels belonging to the sample class with the total number of pixels in the image. The mean percentage of sample pixels was then taken over the whole dataset of 48 images.

Further in this project, my annotations were the ones used as the representation of the ground truth to test the performance of the models.

3.4 SampleFinder

The implementation currently used by Etteplan to delineate the sample from the background is called SampleFinder throughout this project. It uses dynamic thresholding along with dilation and erosion to separate the sample from the other parts of the microscope slide. The first step in this model is to convert the original BGR image to an YUV image. For values between 110 and 126 for the U-channel, with step size 2, thresholding is done to create a binary image were the sample area is separated from the background. Dilation and erosion is then used on the binary image to fill any holes in the sample area. For each iteration the thresholded binary image is used together with the *SimpleBlobDetector* algorithm in *OpenCV* to find the center point and radius of the sample. If a center point and radius is found this is then used to create a circle which marks the sample area where the microscope should scan.

The current implementation was evaluated on the 48 manually annotated images. IoU was calculated for both the circle as well as the thresholded binary image produced by SampleFinder. The total runtime was also measured on a Nvidia GeForce RTX 2070 GPU.

3.5 Active contour

3.5.1 Parameter optimization and evaluation

The active contour function in Scikit-image inputs 11 different parameters, two of them being the image and the initial spline which are required by the function. Before inputting the image to the function it was converted to single channel (grayscale) and a gaussian filter was applied. The effect that the optional parameters (alpha, beta, gamma, w_line, w_edge, max_px_move and max_iterations) had on the performance results were evaluated. Two parameters were not evaluated, these were boundary conditions and convergence. The evaluation was done by calculating the IoU for the sample class according to equation 1 on one image for different values of the parameter. This one image was of medium quality. It had contrast to the background, but a somewhat diffuse boundary. The results were plotted in order to find a threshold/optimal value which maximized the IoU value for each parameter.

Other than the inbuilt parameters in the active contour model, expansion/compression of the SampleFinder circle was also evaluated as well as the number of points in the initial snake. The difference in performance of the function when using either one of the three channels (red, green, blue) was evaluated as well as using OpenCV color conversion rgb2gray which weights each channel according to the following function: $Y \leftarrow 0.299 \cdot r + 0.587 \cdot g + 0.114 \cdot b$. For the evaluation of gray scale conversion, the SampleFinder circle was used as the initial spline for the optimized active contour model on the 48 manually annotated images.

3.5.2 Initial spline

As an initial spline, both the circle and sample boundary in the thresholded image from SampleFinder were evaluated with regards to both IoU and runtime on the GPU. For the usage of the SampleFinder circle as initial spline, it was necessary to initialize an average circle if SampleFinder did not output a circle. The average circle was calculated from the SampleFinder circles which were found for 45 of the 48 images. This resulted in a circle with center coordinate (920, 438) and radius 268 pixels.

A combination of using SampleFinder circle (when found) and SampleFinder threshold boundary when no circle was found was also tested. The boundary of the sample in the thresholded binary image was retrieved by using the find_boundary function in *OpenCV*. If the function gave multiple boundaries the longest retrieved boundary was used. For comparison reasons, the IoU for a circle placed in the center (1071, 423) of an image with radius of 300 pixels was calculated. This circle was also evaluated as the initial spline for the active contours function.

3.6 Annotation of images for deep learning

All 238 images were used for evaluating the deep learning models. Out of these 238 images, 48 were from the previous small dataset used for performance evaluation of SampleFinder and active contours. Since these 48 images were already annotated, they were held aside when using active contours as an automatic annotation tool.

The annotation of the remaining 190 images was done with the active contours model using either the SampleFinder circle as an initial spline or the threshold boundary from SampleFinder if no circle was found. Each annotation was manually inspected (visually) and regarded as either a good or bad boundary for the sample and thereafter the binary mask was saved if the result was regarded as good.

If the active contours annotation was regarded as bad, a visual inspection of the threshold boundary was also made and saved only if it was a good annotation. Same was made for using the threshold boundary as initial spline for active contours. The images in the dataset which could not retrieve a good and representative sample boundary from the active contours model or SampleFinder were instead manually annotated, see section 3.3 for method.

3.7 Deep learning

From studying literature about semantic segmentation using deep learning I learned that there are several networks which have been created for segmentation tasks. Many tasks and datasets presented involved multiclass segmentation, whereas this project only focuses on two classes, background and sample. However, the main issue was similar and therefore these models were regarded as appropriate to try for the project task of delineating the sample from the background. When studying literature of segmentation networks it seemed that some networks, e.g. FCN and SegNet had been created and evaluated with the VGG-16 network as encoder. Therefore, I found it relevant to test that architecture for this project.

Some models which have been used for semantic segmentation problems and have shown good performance results, such as FCN-8, SegNet, U-Net and PSPNet, were implemented in keras by Gupta (2022). The implementation is called *keras-segmentation* and also enables usage of common CNN architectures e.g. VGG-16 and MobileNet as encoders in the segmentation models. *Keras-segmentation* also implements a vanilla CNN which has 5 convolutional layers, each followed by a max-pooling layer with size 2×2 and stride 2 (Figure 9). The kernels in the convolutional layers are of size 3×3 and they move with stride 1. The number of kernels start with 64 followed by 128, and for the last 3 layers the number of kernels are 256. After each convolutional layer, batch-normalization is performed and the activation function ReLU is applied.

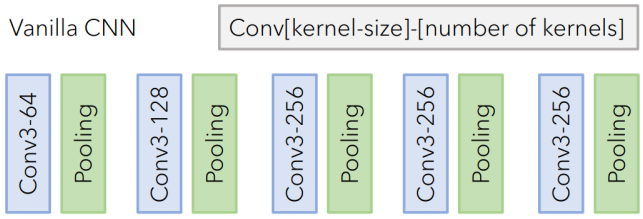


Figure 9: The network architecture of the vanilla CNN.

3.7.1 Dataset splitting and pre-processing

The dataset was split into a set for training the networks, one set for validation and one set to use for testing the performance of the networks. The distribution of the images were such that 60% (142) were put in the training set, 20% (48) were in the validation set and the 48 images which were used to evaluate SampleFinder and active contour were used as the testing set, also representing 20%. The reason for using that set as the testing set was to be able to make a relevant comparison of the performance of the three different types of models.

All images which were annotated with active contour or SampleFinder were put in the training and validation dataset. This was done so that 65% of the images in the sets were from the automatic annotation and 35% were manually annotated. This distribution made sure that both the validation and training set contained images with mixed sample quality, since the manually annotated images mostly consisted of images with lower quality samples and the automatically annotated images had higher sample quality.

Before training the networks using this dataset the dimensions of the images and the corresponding segmentation masks were altered. The dimensions were altered so that the width and height was evenly divisible by 32 (2^5). The initial width of the images were 846 and the height were 2142, which is not evenly divisible by 32. Therefore, a border of zeroes were added to the images and masks, 2 pixels to the left of the images and 18 pixels at the top of the images. This changed the dimensions of the images and masks to $864 \times 2144 \times 3$.

3.7.2 Training and evaluation

The vanilla CNN was used as the encoder for the first evaluation of FCN-32, FCN-8, SegNet and U-Net. Then, the VGG-16 network was also used as encoder for the FCN-8 and SegNet. This was done by loading the pretrained weights from the VGG-16 network which had been trained on the ImageNet dataset. The FCN-8 network was used to evaluate the effect of changing the number of epochs when training the network. The number of epochs used were 10, 20 and 25. Categorical cross entropy was used as the loss function together with the Adam optimizer. Most of the training processes had a batch size of two, however, some networks were trained with batch size one because of memory limitations. Larger batch sizes than two was not possible to use, also because of memory limitations.

The deep learning models were trained inside a docker container with a tensorflow installation as well as access to the servers' graphical processing units (GPUs). On the

server where the container was made there were two Nvidia GeForce RTX 2070 GPUs and one of them was used when training the models.

After training the networks, the model loss and accuracy for the training and validation sets were plotted and the IoU for the two classes, background and sample, were calculated on the validation set. The best performing networks out of the four architectures used, FCN-8, FCN-32, SegNet and U-Net, were then also evaluated with regards to IoU and runtime on the test set. The runtime was measured by taking the average time per image when predicting the segmentation masks for the test set on the GPU.

3.8 Displaying model boundaries and over- and underestimation

To get an estimation of how much each model underestimated and overestimated the sample area, FNR and FPR were used. Pixels with the label background (pixel value 0) were regarded as negatives and the pixels with the label sample (pixel value 1) were regarded as positives. Hence, the FNR represented underestimation and the FPR represented overestimation of the sample area. They were calculated for each segmentation model according to equation 2 and 3.

To display the results from the different models, the resulting boundaries from the models were plotted on the original images. For each model, the three images with the lowest IoU were extracted and the boundaries were plotted. Some of the models had images in common among the three worst and therefore only seven images were chosen. This was done to visualize the weaknesses of the different models and also to get an understanding of which types of images that the models performed worst on. Beyond those images, eight randomly chosen images from the testing dataset were also plotted to get an overall picture of how the models differed. Apart from the models; SampleFinder, active contour and the four deep learning networks, the manual annotations by me and person 2 were also plotted.

4 Results

The IoU between the two ground truth sets were 97.9% for the sample class and 99.6% for the background class. The annotation set made by me received a mean pixel percentage of the sample of 14.7% while the other person's annotations had a slightly higher percentage of 14.8%. This means that person 2 was generally a bit more generous when marking out the sample area. However, some images stood out a bit more in the difference between sample pixel percentage. These images were characterized by smeared out samples with a low contrast sample boundary. By looking at the IoU for the sample class of these images one can also see that these images had a lower IoU than the average of 97.9%.

4.1 Active contour and SampleFinder

4.1.1 Parameter optimization

The active contour model had predefined values of the parameters which were used by default if no other specification was made. The default parameter settings were as follows: alpha=0.01, beta=0.1, w_line=0, w_edge=1, gamma=0.01, bc=str, max_px_move=1, max_iterations=2500 and convergence=0.1. When using the default settings it took an average of 28 seconds to find a sample boundary in one image (Table 1). The runtime for the default settings was very long. This can be compared to the runtime after optimizing the parameter settings which was around 2 seconds per image (Table 2). The IoU of the sample class for the active contour model with default settings was 78.5% which was a small increase compared to the SampleFinder circle which had an IoU of 77% (Table 2).

The parameter optimization for the active contours model resulted in a model with the following settings: alpha=0.016, beta=0.15, gamma=0.001, max_iterations=100, w_line=1, w_edge=1, max_px_move=1. Other than that, the optimal number of points in the snake was determined to be 740 and the optimal expansion of the circle was 40 pixels. For the one image used for the optimization this resulted in an IoU of 94.7%.

The alpha and beta values were slightly increased because it gave higher IoU. However, even higher values of alpha and beta instead decreased the performance of the model. Negative values of w_line gave lower IoU values than positive values. This means that it was advantageous to make the snake attract to lighter regions. The max_iterations parameter had a large effect on the runtime of the model. Fewer iterations gave a faster algorithm. A decreased number of iterations did however not decrease the IoU, which was why fewer iterations, from 2500 to 100, was regarded as optimal.

A small number of points in the snake gave a lower IoU than a higher number of points. This was probably because the behavior of the snake became more edgy and less smooth when using fewer points. Expanding the SampleFinder circle before using it as an initial spline for the active contour model gave higher IoU than compressing the circle or just using the circle without any changes. The parameters which didn't have to be altered from the default were w_edge and max_px_move. All resulting plots can be seen in Appendix A.

The different single channel conversions gave different performance results. The green channel gave the highest sample IoU (Table 1). These results suggest that using the green channel when converting from the rgb image to grayscale would give the best sample boundary detection on the images.

Table 1: Performance results for the default settings of the active contour model and the results from using the different types of single channel conversions.

	IoU (sample)	std	IoU (background)	std	time (s/image)
Default	0.785	0.199	0.959	0.040	28
Red channel	0.830	0.154	0.968	0.031	-
Green channel	0.865	0.130	0.975	0.025	-
Blue channel	0.852	0.121	0.971	0.024	-
rgb2gray	0.863	0.137	0.974	0.027	-

4.1.2 Initial spline

Placing out a circle with a fixed radius in the middle of each image gave a very low IoU for the sample class of 39.6%. Using this mid circle as an initial spline for the active contour model increased the IoU but still it gave a very low IoU of 45.3% and a high standard deviation of 27.9%. This model is therefore not at all optimal to use for annotating images or as a boundary for microscope scanning. SampleFinder showed better performance results. The circle had an IoU of 77% with high standard deviation of 22.4% and the thresholded image gave an IoU of 84.4% (Table 2). One reason for the high standard deviation was that SampleFinder did not place out a circle in 3 out of 48 images, resulting in an IoU of 0 for those images.

The active contour model with SampleFinder circle as initial spline increased the IoU compared to just using the SampleFinder circle. This model gave an IoU of 86.5%, which is an increase of almost 10%. However, the active contour model with the threshold boundary as initial spline did not increase the IoU compared to just using the threshold boundary. Instead, it actually decreased the IoU from 84.4% to 83.2% (Table 2).

The best performance for the active contours model was reached when using the threshold boundary points as initial spline when no circle was given by SampleFinder and in all other cases using the SampleFinder circle (with expansion) as initial spline. This gave an IoU of 88.5% with a low standard deviation of 6.9% (Table 2).

The SampleFinder model had an average time of 0.07 seconds per image which is very fast. The active contour model gave an average of between 1.78 and 2.09 seconds per image (Table 2). The fastest active contour model was when using the circle as initial spline and an average circle when no circle was found, and the slowest was to use only the threshold boundary as initial spline. Active contour was a slower model than SampleFinder, however, the performance of the model was better. To run an annotation on 190 images with active contour would take around 6 minutes.

Table 2: Performance results for different segmentation models on 48 manually annotated images. The type of initial spline for the active contour model is defined in parentheses.

Model	IoU (sample)	std	IoU (back-ground)	std	time (s/image)
Mid circle	0.396	0.232	0.844	0.081	-
SampleFinder circle	0.770	0.224	0.963	0.028	0.07
SampleFinder threshold	0.844	0.103	0.972	0.023	0.07
Active contours (mid circle)	0.453	0.279	0.863	0.087	1.78
Active contours (sf circle)	0.865	0.130	0.975	0.025	1.86
Active contours (sf threshold)	0.832	0.090	0.968	0.021	2.09
Active contours (sf circle + threshold)	0.885	0.069	0.979	0.016	1.87

The performance results of the active contour model suggests it is appropriate to use for annotation of a larger dataset with similar images. The results also suggest the most efficient way to annotate would be to use the threshold boundary from SampleFinder as initial spline if no circle can be placed out since this would increase the probability of the annotation being good.

Out of 190 images, 124 were able to be annotated with either active contour or SampleFinder. The images which were not able to be annotated consisted mostly of images where the sample was smeared out, not round or uneven in thickness.

4.2 Segmentation networks (deep learning)

The validation accuracy of the FCN-8 network increased when training for 20 epochs instead of 10 (Appendix B). However, there was no increase when training for 25 epochs and therefore 20 epochs was used as a start when training the other networks. The difference in accuracy and loss between the training and validation was very low for the FCN-8 networks, hence, the networks overfitted to a low extent on the training data (Figure 10). Using the pretrained VGG-16 network as encoder for FCN-8 did not increase the performance of the model (Figure 11, Appendix B). For the following plots epoch 0 indicates the accuracy and loss after training the networks for one epoch.

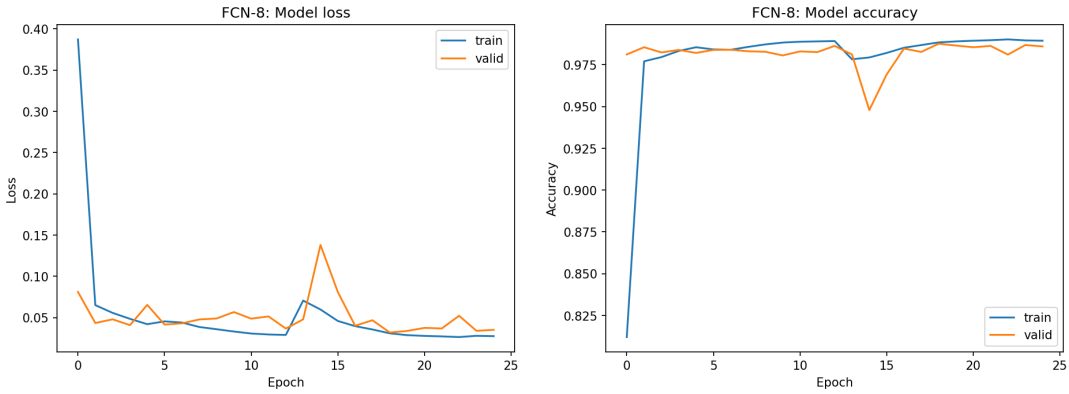


Figure 10: Loss and accuracy when training FCN-8 with vanilla CNN for 25 epochs.

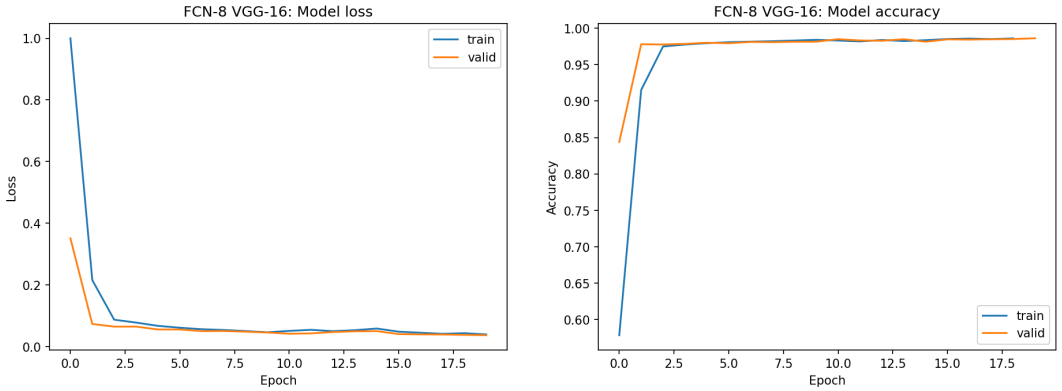


Figure 11: Loss and accuracy for FCN-8 trained with VGG-16 for 20 epochs.

The FCN-32 network was trained to the highest validation accuracy among the networks. The model loss was also very low (Figure 12). FCN-32 also had the highest IoU for the sample class on the validation set (Appendix B). Because of the good results this network was evaluated on the independent testing set.

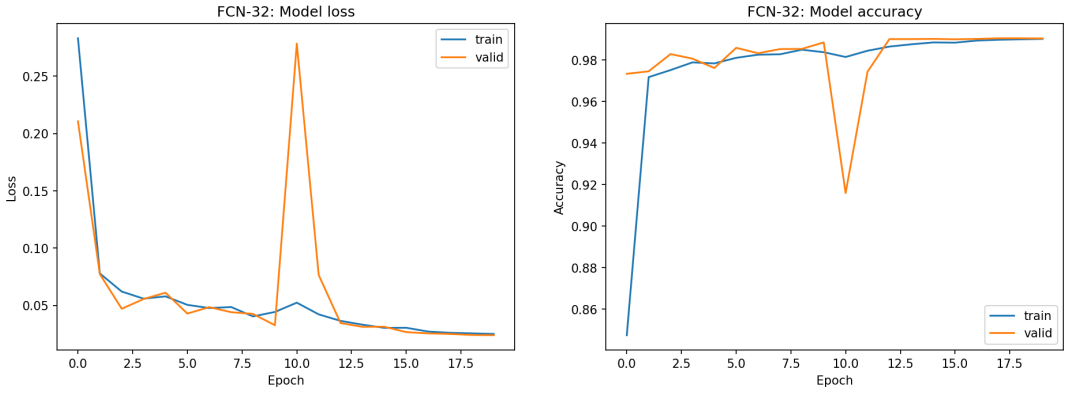


Figure 12: Loss and accuracy when training FCN-32 with vanilla CNN for 20 epochs.

SegNet showed the worst results on both the validation accuracy and sample IoU. However, when changing the encoder from the vanilla CNN to the pretrained VGG-16 network there was an increase in accuracy and IoU (Figure 13). The U-Net had higher accuracy and IoU compared to SegNet, but lower than both FCN networks (Figure 14, Appendix B).

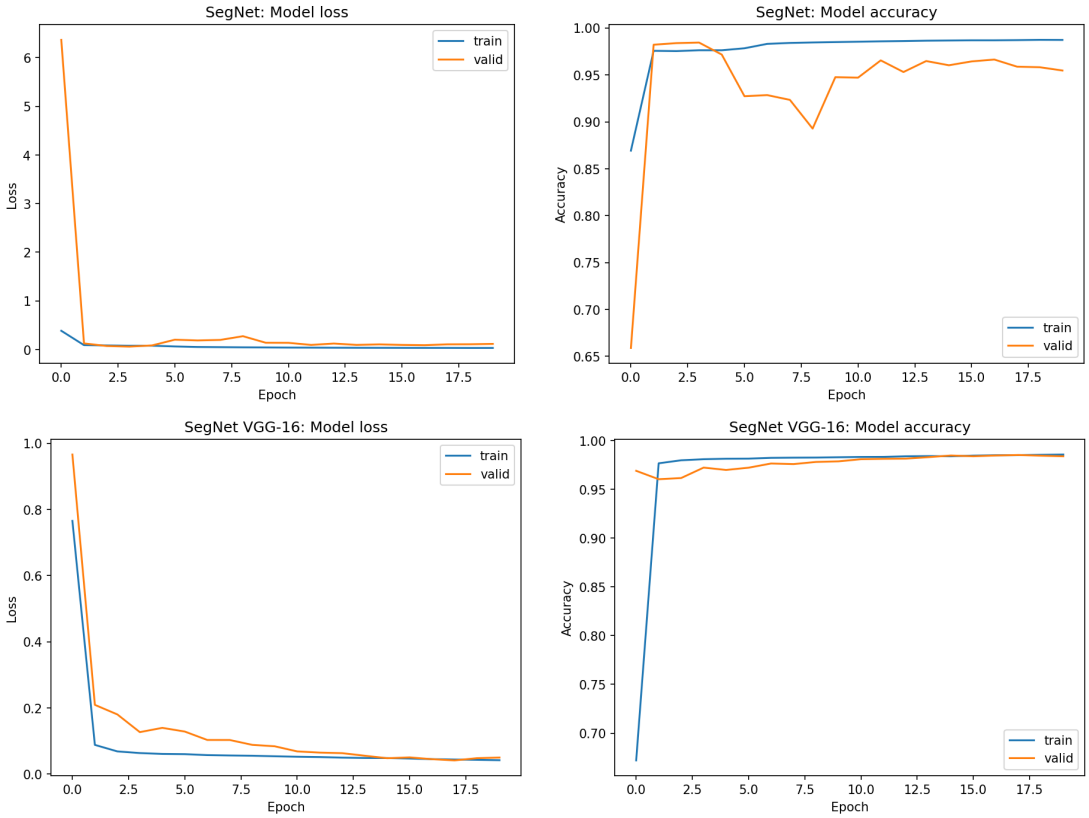


Figure 13: Loss and accuracy when training SegNet with vanilla CNN and VGG-16 for 20 epochs.

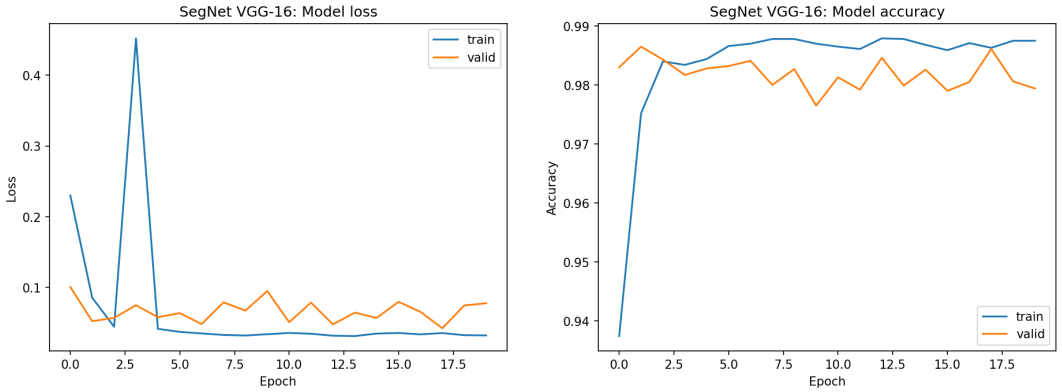


Figure 14: Loss and accuracy when training U-Net with vanilla CNN for 20 epochs.

The evaluation on the test set showed that the FCN-32 network performed the best out of the four networks with an IoU for the sample class of 95.2% and a low standard deviation

of 2.1%. U-Net had the second highest sample IoU, however, it had the highest standard deviation of 7.2%. SegNet showed the lowest IoU of 88.8% with a standard deviation of 4.6%. The FCN-8 network had a sample IoU of 91.5% with a standard deviation of 5.2% which makes it the third best. There were no large differences in runtime between the networks (Table 3).

Table 3: Performance evaluation of the networks on the test set.

Model	IoU (sample)	std	IoU (background)	std	time (s/image)
FCN-8	0.915	0.052	0.985	0.014	0.37
FCN-32	0.952	0.021	0.992	0.006	0.38
SegNet (VGG-16)	0.888	0.046	0.979	0.014	0.44
U-Net	0.922	0.072	0.987	0.014	0.30

The best performing network, FCN-32, consisted of 69,745,026 parameters and was of size 837MB. The FCN-8 also had a large number of parameters 69,730,822 and were of size 837MB. This can be compared to the second best network, U-Net, which only had 4,471,746 parameters and was of size 54MB. SegNet was also small in comparison to FCN-32 and FCN-8. The network only had 3,697,602 parameters and a size of 139MB. The training time was similar for all the networks, in total it took around 15-20 minutes to train a network for 20 epochs.

4.3 Boundaries, over- and underestimation

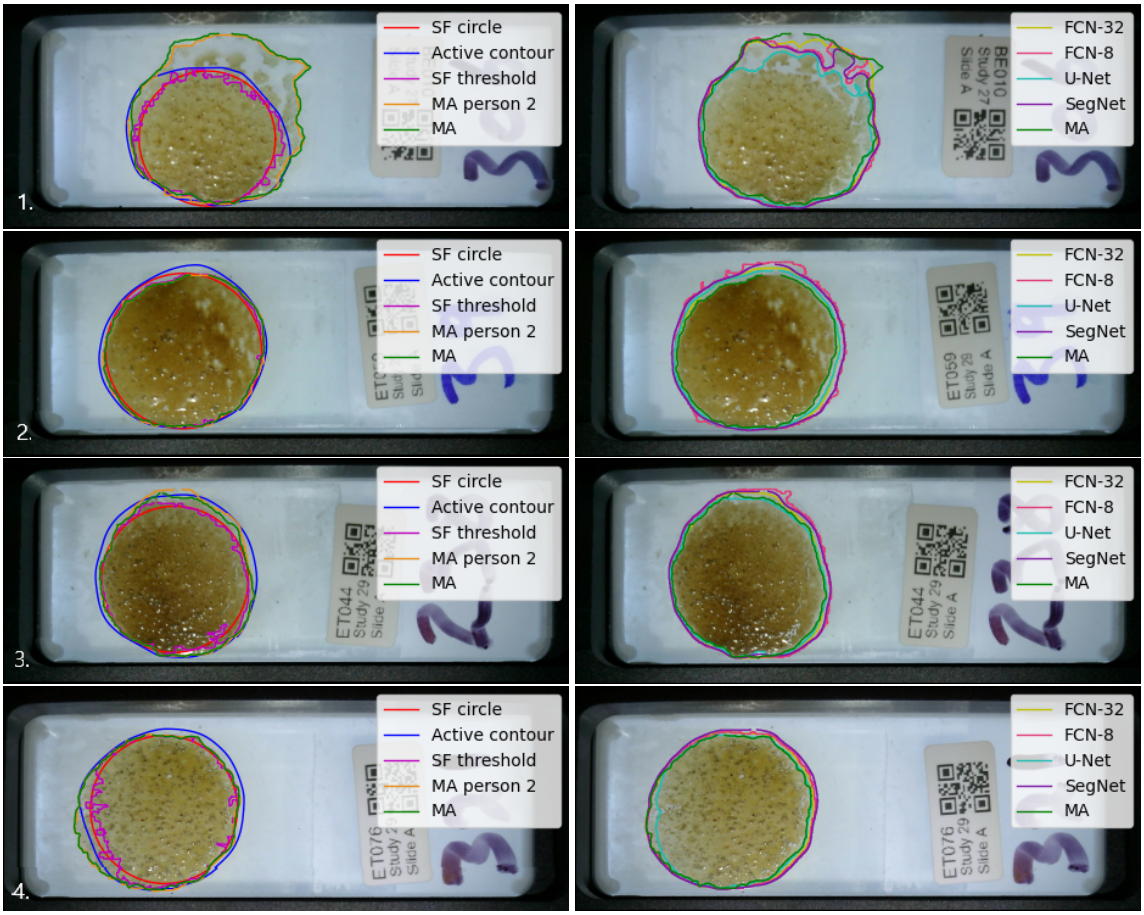
SampleFinder circle and threshold were the models which had the highest FNR. This means that these models often mark out too little of the sample on the microscope slide. The active contour model had a lower FNR, however, all four networks show even lower FNR. The lowest FNR belongs to the FCN-8 network, only 1%. SampleFinder circle and threshold had the lowest FPR which then means these models do not mark out a too large boundary of the sample. FCN-8 and SegNet are the two models which most often mark out a too large boundary since these models had the highest FPR (Table 4).

Table 4: FNR and FPR for SampleFinder circle and threshold boundary, the active contour model with SampleFinder circle and threshold boundary as initial spline as well as the deep learning networks.

Model	FNR (under)	FPR (over)
SampleFinder circle	0.215	0.002
SampleFinder threshold	0.161	0.002
Active contours	0.073	0.009
FCN-8	0.010	0.014
FCN-32	0.016	0.006
SegNet (VGG-16)	0.012	0.019
U-Net	0.047	0.006

By looking at the randomly chosen images in Figure 15 one can see the differences in the resulting boundaries between the traditional models (SampleFinder and active contour) and the deep learning networks. Image 7 clearly shows how the networks adapt their boundary more accurately to a sample which does not have a round character compared to the traditional models. The traditional models show high accuracy on image 2 which has a more roundly shaped sample. When compared to the results from the networks one can even see that two of the networks, SegNet and FCN-8, overestimates the area of the sample in image 2.

Image 1 in Figure 15 shows how the networks are more generous when marking out the sample compared to the traditional models. Even if the sample is smeared out and uneven in thickness, the networks can still detect the smeared out area and include it in their binary mask. In general, one can see that the networks follow the boundary of the sample more accurately in comparison to the SampleFinder and active contour model which agrees to the IoU results for the models.



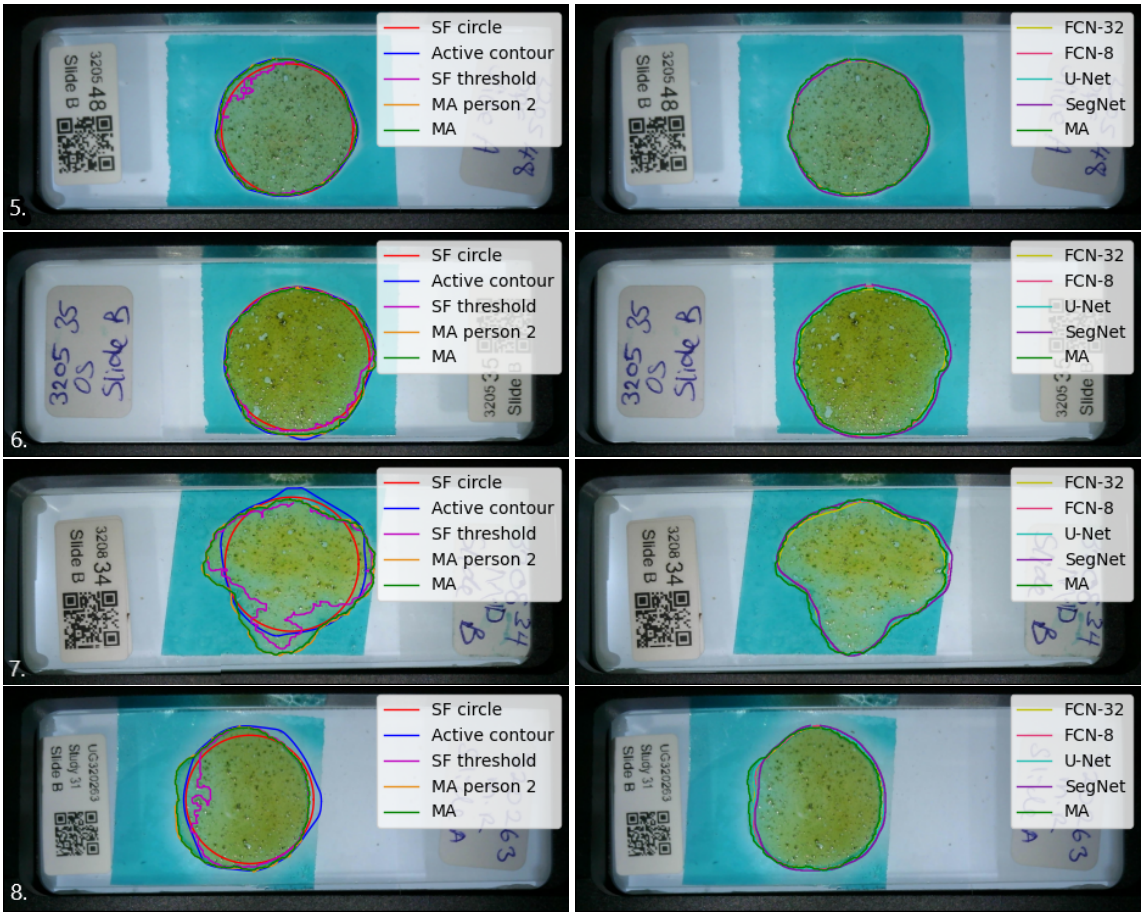


Figure 15: A collection of eight randomly chosen images from the testing set. The images to the left show the results from SampleFinder (SF) circle and threshold as well as the active contours and manual annotations (MA). The images to the right are the same as to the left but with the contours from the different deep learning networks.

Images 1, 4 and 5 in Figure 16 shows how FCN-8 and SegNet overestimates the sample area to a large extent. These results also align with the FPR values and somewhat explain why the FPR were the highest for these networks. Out of these three images, the traditional models show much better results on image 5 compared to FCN-8 and SegNet. Images 2, 6 and 7 all show that U-Net tends to underestimate the sample area more than the other networks. For images 6 and 7, the SampleFinder and active contour also underestimated the area. This is probably because the samples are uneven in thickness and not round.

In Figure 16 one can also see some differences between the two manual annotations. Two clear examples of different interpretations of the ground truth are shown in image 1 and 3. There is an area in the bottom parts of both of the images which have been marked by person 2 but not by me. These are also areas which some of the deep learning networks, such as FCN-32, have marked out as part of the sample.

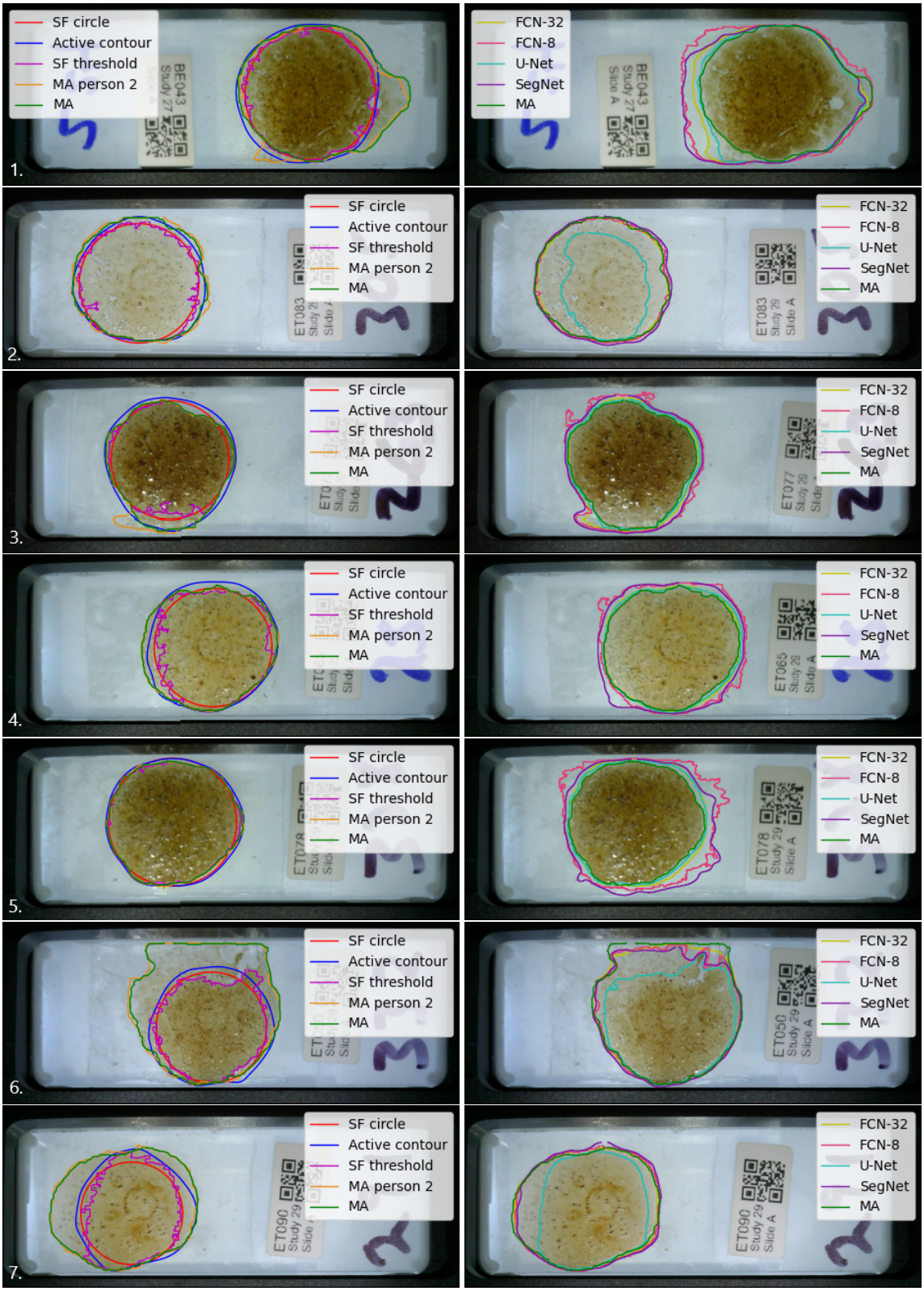


Figure 16: A collection of the images which were among the three images with the lowest IoU for the models, in total seven images. To the left the SampleFinder (SF) circle and threshold boundaries are displayed together with the active contour result and the two manual annotations (MA). To the right all network boundaries are drawn on the images.

5 Discussion

The active contour model gave an increase in performance for the task of delineating the stool sample from the microscope slide background compared to the current method (SampleFinder). However, the model was much slower than the current one. A general disadvantage with the active contour method is its sensitivity to a good initial guess. Since the application of the model in this project used the SampleFinder circle as initial spline, it became highly dependant on SampleFinder doing a good job of finding the sample.

All four segmentation networks, FCN-32, FCN-8, SegNet and U-Net, also showed an increase in performance compared to the current method. A comparison between SampleFinder circle and best performing network (FCN-32) show an increase of 18,2% in IoU for the sample class. The networks also show higher IoU compared to the active contour model. The networks show a large decrease in underestimation compared to SampleFinder and active contour. The reason for wanting a low underestimation is due to the method of STHs diagnosis. As described, the degree of infection is measured in eggs per gram and therefore it is very important to count all eggs in the stool sample. Hence, all of the sample must be scanned by the microscope.

Further, one can conclude that the diagnosis method is not sensitive to an overestimation of the sample area. But overestimation is not optimal fore the sake of decreasing the amount of background images taken by the microscope. The networks which had the highest overestimation, FCN-8 and SegNet, also showed the lowest underestimation. With respect to the importance of scanning all of the sample, these networks would then be preferable over the other models. However, FCN-32 showed an underestimation of 1.6% which is not much higher than SegNet which had 1.2%, but still showed higher IoU compared to SegNet and FCN-8.

The results show that the networks which previously have been used for semantic segmentation problems also work for this project's problem. As a whole, viewing the problem as a semantic segmentation task is highly suitable. The networks do a good job of learning the general features of the sample on the microscope slide images, such as color and texture, which leads to high performance. In the majority of the images the sample sticker and handwritten numbers are ignored even though the whole image is inputted to the network models. Most often, the sample area is also cohesive but with some exceptions. For example, SegNet marked parts of the sticker and other irrelevant areas in some images, areas which are separated from the sample area. One way of handling this could be to only scan the largest cohesive area, but this could be a problem if the actual sample would be distinctly separated on the slide or the largest cohesive area is not the sample.

One difference between the networks is the smoothness of the sample boundary. In general, FCN-32 and SegNet gives smoother boundaries than FCN-8 and U-Net. FCN-8 generates the least smooth boundaries. One reason for this could be that FCN-8 picks up finer details than the FCN-32 because of the skip connections in its architecture. Another difference is the size of the networks. The smallest network, U-Net, also had the second

best IoU results. With regards to size, this network would probably be preferred over the large FCN-32 network.

Between the networks there was no large runtime difference, but all four networks were faster than the active contour model. A comparison between the best active contour model and FCN-32 show that the network is almost 5 times faster than the active contour model. The fastest model was however the current one and the time difference was large between both the networks and active contour. SampleFinder is almost 27 times faster than active contour and 5 times faster than the networks on the GPU.

One difficulty in the project were the annotations, both the manual annotations and the annotations made with active contour and SampleFinder threshold. The annotations were done using only one cohesive area to bound all of the sample which in some cases could have been regarded as less accurate than having multiple separate areas. A motivation for doing this is to maintain the accuracy of the diagnosis method, i.e making sure all eggs are found on the slide to ensure that the right degree of infection can be determined. On some of the images there were not a clear boundary of the sample and this left more room for interpretations of what should be regarded as sample. This was one reason for why two different annotations were made by two different persons.

Further, the results from the models can be compared to the IoU between the two manual annotations. If the difference between two humans and a human and a computer is similar the computer could be said to have similar correctness to the work of a human. In this case, the IoU was 97.9% between the two persons who annotated the dataset and 95.2% between the manual annotation and FCN-32, the best performing model tested. There is a difference, but it is not large. In conclusion, FCN-32 came closest to perform the same as a human.

Using pretrained weights from the VGG-16 network trained on ImageNet increased the performance of SegNet. However, this was not the case when training the FCN-8 network. One thing that did increase the performance of FCN-8 were the number of epochs, but only to 20 epochs, higher numbers were not advantageous. Since the number of epochs were only evaluated on one network this may not have been optimal for all networks. To improve the individual optimization of number of epochs for each network, early stopping could be used (Keras documentation 2022).

Something which could also improve the performance of the deep learning networks is more data. More data from newer studies could be used to train and evaluate the networks and improve the generalization ability of the networks. Another aspect which could potentially improve the performance of the networks would be to use IoU as loss function instead of categorical cross entropy. This was suggested by Nagendar *et al.* (2018) where they show that training a segmentation model on IoU can in some cases improve the model. Data augmentation could also be interesting to try for improving the performance of the networks. Especially for the U-Net since the article which describes this network, Ronneberger *et al.* (2015), emphasises its suitability for segmentation tasks

where only small datasets are available by partially using data augmentation.

The parameters in the active contour model were evaluated on only one image because of time limitations. This is not an optimal way of finding the best settings for the whole dataset, but it was done due to time limitations. This could have limited the model's potential of showing higher performance. One potential improvement could have been to also perform the parameter optimization on a higher quality image and compare the results from using the settings on the small dataset (48 images). The parameters which gave the highest average IoU over the dataset would then be preferred.

6 Conclusion

The objective of this project was to find an improved method for delineating the stool sample on the microscope slide. To conclude, the project was successful with regards to the goal. Both the active contour model and the segmentation networks performed better than both SampleFinder circle and threshold when using IoU as a performance measurement. The runtime of the models did however not outperform the runtime of SampleFinder. Out of all the models tested, FCN-32 performed best and would be recommended if model size is not an issue. The FCN-32 network show potential for decreasing the amount of background images collected by the microscope while retaining the accuracy in STHs diagnosis. An implementation of this network could contribute to reducing the scanning time as well as reducing the dependency of a human to accept and possibly adjust the scanning area.

7 Acknowledgement

I would like to acknowledge the AI4NTD consortium and it's partners (Janssen Pharmaceutical Companies of Johnson & Johnson, Merck, Etteplan, Bridges to Development, Ethiopian Public Health Institute, Jimma University, Vector Control Division at the Ministry of Health of Uganda and Ghent University) for access to their image and data collection on Soil-transmitted Helminthiasis (STHs) and schistosomiasis, and the opportunity to conduct this research. A big thanks to Etteplan and my supervisor August Tynong for all the support and help I received. I would also like to thank my subject reader Carolina Wählby for great guidance and important tips throughout the project. At last, I would like to thank my examiner Siv Andersson and coordinator Lena Henriksson for their support and good advice.

References

Albawi S, Mohammed TA, Al-Zawi S. 2017. Understanding of a convolutional neural network. 2017 International Conference on Engineering and Technology (ICET). 1–6.

Badrinarayanan V, Kendall A, Cipolla R. 2017. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 39: 2481–2495. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

Bakoš M. 2007. Active Contours and their Utilization at Image Segmentation 5.

Cheesbrough M. 2005. District laboratory practice in tropical countries. Part 1 Part 1. Cambridge University Press, Cambridge; New York. OCLC: 432661238.

Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL. 2017. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. arXiv:1606.00915 [cs] ArXiv: 1606.00915.

Hafiz AM, Bhat GM. 2020. A survey on instance segmentation: state of the art. International Journal of Multimedia Information Retrieval 9: 171–189.

Hesamian MH, Jia W, He X, Kennedy P. 2019. Deep Learning Techniques for Medical Image Segmentation: Achievements and Challenges. Journal of Digital Imaging 32: 582–596.

Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR. 2012. Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580 [cs] ArXiv: 1207.0580.

Ho Y, Wookey S. 2020. The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling. IEEE Access 8: 4806–4813. Conference Name: IEEE Access.

Ioffe S, Szegedy C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167 [cs] ArXiv: 1502.03167 version: 3.

Kass M, Witkin A, Terzopoulos D. 1988. Snakes: Active contour models. International Journal of Computer Vision 1: 321–331.

Keras documentation K. 2022. Keras documentation: EarlyStopping.

Kingma DP, Ba J. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs] ArXiv: 1412.6980.

Krizhevsky A, Sutskever I, Hinton GE. 2012. ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems. Curran Associates, Inc., volume 25.

Long J, Shelhamer E, Darrell T. 2015. Fully Convolutional Networks for Semantic Segmentation. arXiv:1411.4038 [cs] ArXiv: 1411.4038.

Minaee S, Boykov Y, Porikli F, Plaza A, Kehtarnavaz N, Terzopoulos D. 2020. Image Segmentation Using Deep Learning: A Survey. arXiv:2001.05566 [cs] ArXiv: 2001.05566.

Nagendar G, Singh D, Balasubramanian V, Jawahar C. 2018. NeuroIoU: Learning a Surrogate Loss for Semantic Segmentation 12.

O'Shea K, Nash R. 2015. An Introduction to Convolutional Neural Networks. arXiv:1511.08458 [cs] ArXiv: 1511.08458.

Parija SC, Chidambaram M, Mandal J. 2017. Epidemiology and clinical features of soil-transmitted helminths. Tropical Parasitology 7: 81–85.

Python documentation d. 2022. active contour - skimage - Python documentation.

Rohini G. 2021. Everything you need to know about VGG16.

Ronneberger O, Fischer P, Brox T. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. Navab N, Hornegger J, Wells WM, Frangi AF, editors, Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. Springer International Publishing, Cham, Lecture Notes in Computer Science, 234–241.

Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L. 2015. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision 115: 211–252.

de Silva NR, Brooker S, Hotez PJ, Montresor A, Engels D, Savioli L. 2003. Soil-transmitted helminth infections: updating the global picture. Trends in Parasitology 19: 547–551.

Simonyan K, Zisserman A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556 [cs] ArXiv: 1409.1556.

Szeliski R. 2010. Computer Vision: Algorithms and Applications 979.

Tankeshwar A. 2016. Kato Katz Technique: Principle, Procedure, Results • Microbe Online.

Thoma M. 2016. A Survey of Semantic Segmentation. arXiv:1602.06541 [cs] ArXiv: 1602.06541.

van der Walt S, Schönberger JL, Nunez-Iglesias J, Boulogne F, Warner JD, Yager N, Gouillart E, Yu T. 2014. scikit-image: image processing in Python. PeerJ 2: e453.

Xing Y, Zhong L, Zhong X. 2020. An Encoder-Decoder Network Based FCN Architecture for Semantic Segmentation. Wireless Communications and Mobile Computing 2020: e8861886. Publisher: Hindawi.

Yamashita R, Nishio M, Do RKG, Togashi K. 2018. Convolutional neural networks: an overview and application in radiology. Insights into Imaging 9: 611–629. Number: 4 Publisher: SpringerOpen.

Appendix A

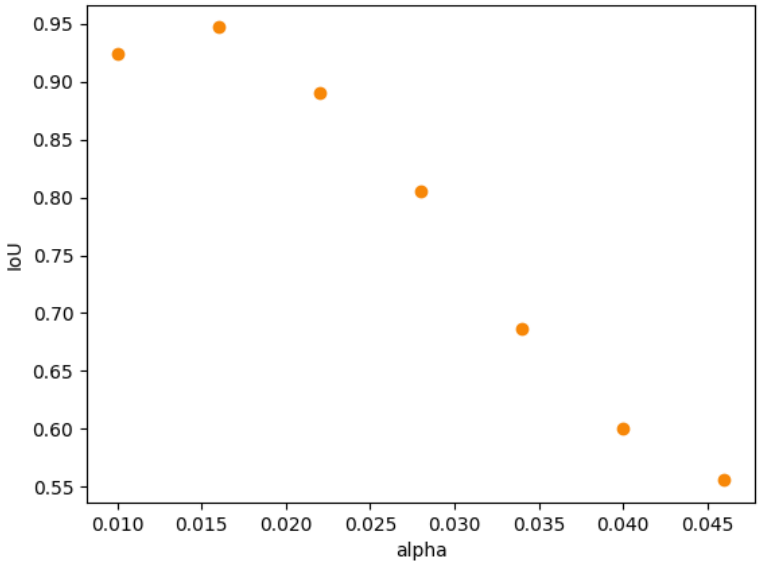


Figure 17: Plots of IoU for different values of alpha.

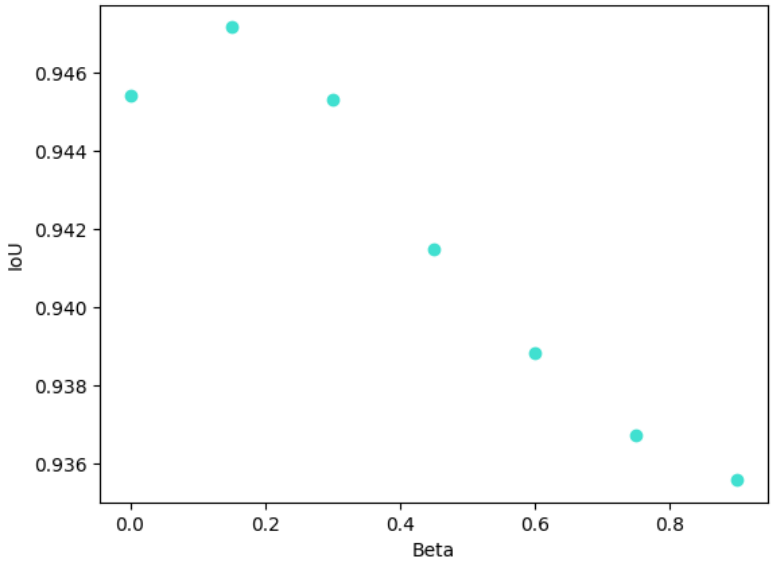


Figure 18: Plots of IoU for different values of beta.

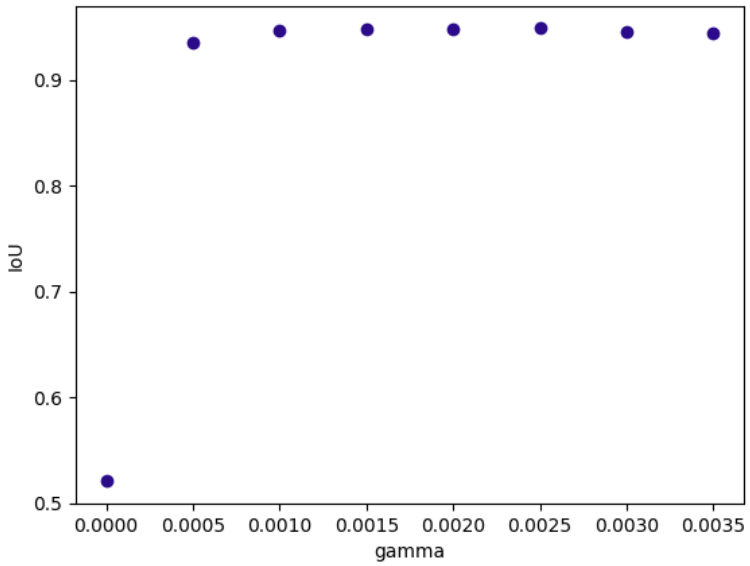


Figure 19: Plots of IoU for different values of gamma.

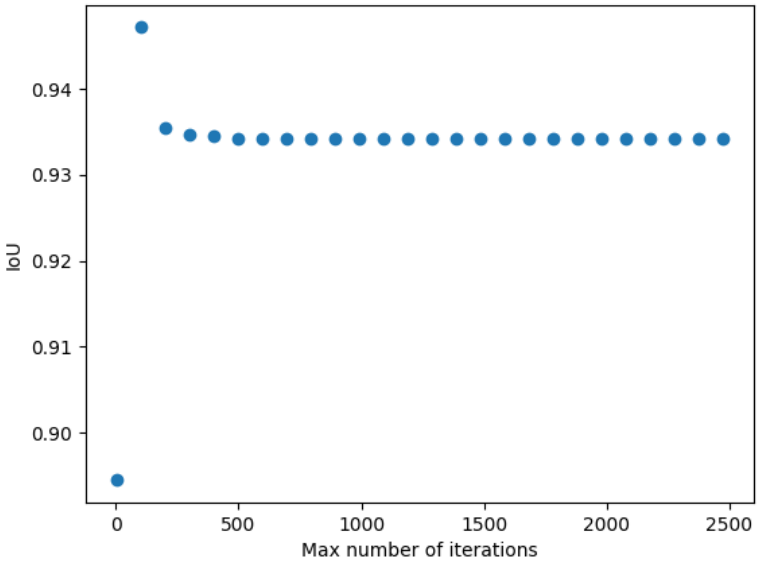


Figure 20: Plots of IoU for different number of iterations.

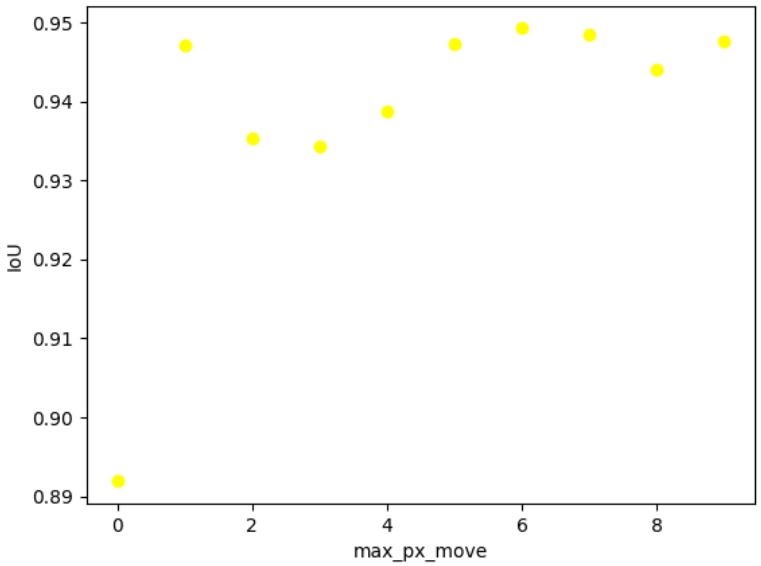


Figure 21: Plots of IoU for different values of max_px_move.

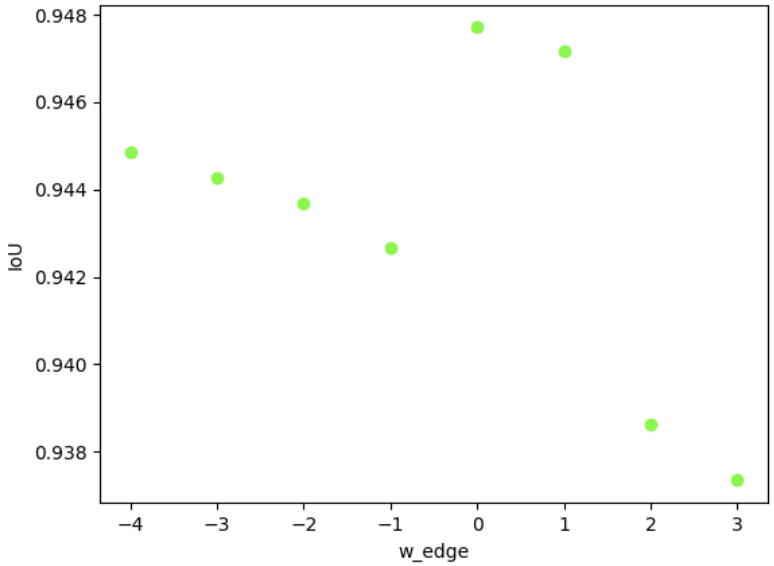


Figure 22: Plots of IoU for different values of w_edge.

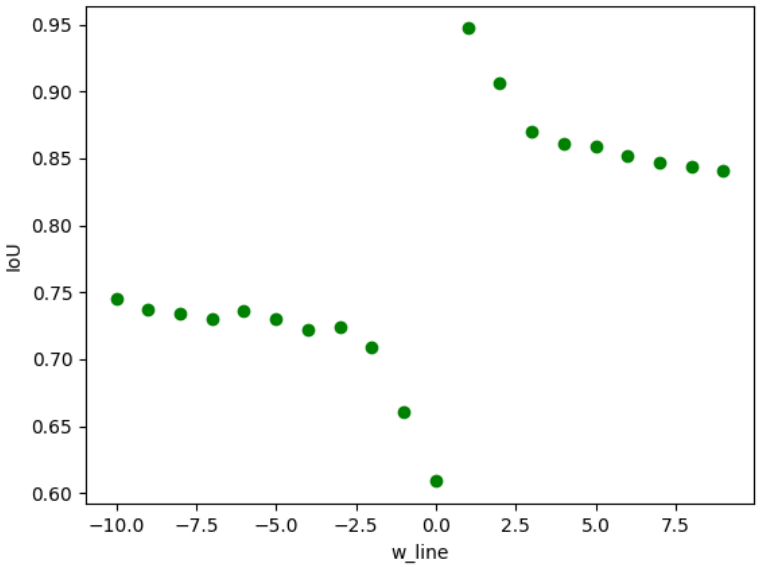


Figure 23: Plots of IoU for different values of w_line.

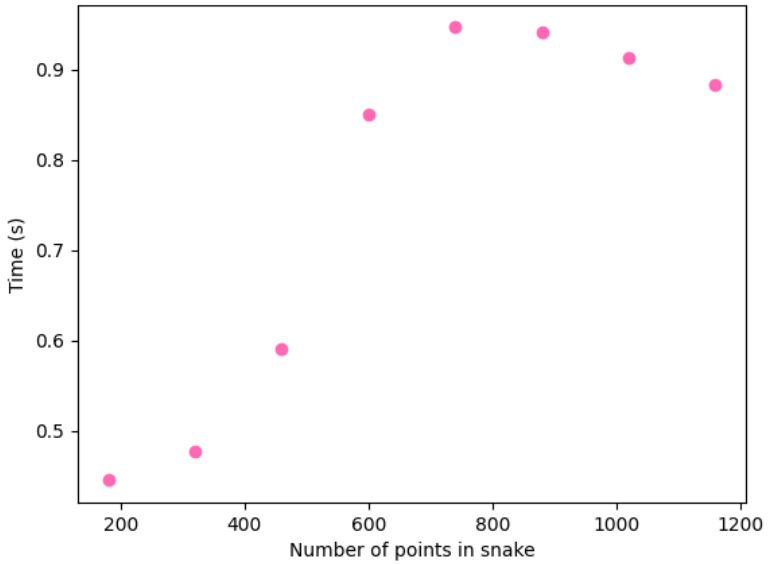


Figure 24: Plots of IoU for different number of points in the snake.

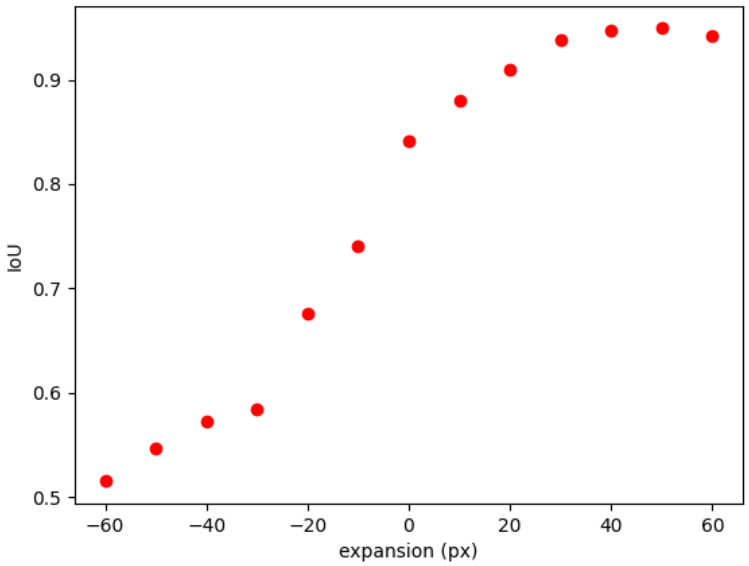


Figure 25: Plots of IoU for different circle expansion/compression values.

Appendix B

Table 5: Epochs, validation accuracy for the last epoch and IoU values on validation set. The highlighted networks are the ones which performed best for each network architecture type.

Model	epochs	CNN type (base model)	val acc	IoU (sample)	IoU (back-ground)
FCN-8	10	Vanilla CNN	0.981	0.884	0.978
FCN-8	20	Vanilla CNN	0.986	0.912	0.984
FCN-8	25	Vanilla CNN	0.986	0.910	0.984
FCN-8	20	VGG16 (batch_size=1)	0.986	0.912	0.984
FCN-32	20	Vanilla CNN	0.990	0.935	0.989
SegNet	20	Vanilla CNN	0.955	0.761	0.947
SegNet	20	VGG16	0.984	0.897	0.981
U-Net	20	Vanilla CNN (batch_size=1)	0.979	0.861	0.976