Myocardial Segmentation in MR images using Convolutional Neural Networks

Mattias Nilsson

September 26, 2018

Master's Thesis
Centre for Mathematical Sciences at Lund University
Supervisors
Niels Christian Overgaard, Centre for Mathematical Sciences at Lund University
Einar Heiberg, Lund Cardiac MR Group

Abstract

A convolutional neural network for automatic myocardial segmentation of MR images is described and implemented, based on the readily available architecture SegNet. A general network trained on both end-systolic and end-diastolic images is determined to be superior to the networks trained on the separate data. The evaluation of myocardial segmentation is discussed, as well as the importance of manual, visual inspection.

Contents

Acknowledgments

This thesis was made possible by the Cardiac MR Group at Lund University Hospital. I would like to thank my supervisors Einar Heiberg and Niels Christian Overgaard, for their support and guidance. I would also like to thank all the wonderful people at the Cardiac MR Group and Klinisk fysiologi, for their hospitality and a rewarding semester of interesting discussions and reflections.

1 Introduction

Cardiovascular disease is the leading cause of death globally, according to a report published by World Health Organization 2011 [1]. It has been a great health care challenge for developed countries during the last decades. In medicine, imaging methods are important to make correct and timely diagnoses. The heart can be assessed by several imaging methods such as *ultrasound*, *electrocardiography*, *magnetic resonance imaging* (MRI), or *nuclear* imaging methods. These imaging methods requires effective, objective methods to assess heart function. An important task is to analyse the volume of the *left ventricle* (LV) of the heart to find if the heart is enlarged due to heart failure. The LV pumps blood to most of the human body, whereas the right ventricle (RV) pumps blood to the lungs, making the LV more important in heart function assessment. This thesis will focus on the analysis of the LV imaged by MRI.

1.1 Myocardial delineation

In cardiac MRI, a 3D image of the torso is captured by acquiring images of 'slices' of the body, showing cross sections of heart. The slices of the top part of the heart are called *basal*, slices in the middle of the heart are called *mid-ventricular*, and slices from the bottom of the heart are called *apical* slices. An example of a delineation of a mid-ventricular slice can be seen in Figure 1. The LV volume is typically evaluated at two time frames of the cardiac cycle; the end of a heartbeat (called *end-systolic*) and just prior to a beat (called *end-diastolic*). The appearance of the LV differs between the two time frames. The end-systolic LV is contracted, resulting in a thick muscle wall (called *myocardium*), whereas the end-diastolic is inflated due to the blood filling the LV, showing a thinner myocardium. The volumes are used to evaluate heart function, by calculating important heart function indicators such as *stroke volume* (the volume of blood pumped by the heart during a heart cycle), *cardiac output* and *ejection fraction*. To calculate the LV volume, one needs to delineate the *endocardium*, the inner wall of the myocardium. Other heart function indicators, such as the analysis of the myocardium strain, also require the delineation of the *epicardium*, the outer wall of the myocardium. One common method of delineation is for an expert to manually draw the contour of the myocardium for each slice, for both time frames. Manual delineation is time consuming and requires expertise, thus calling for automated methods.

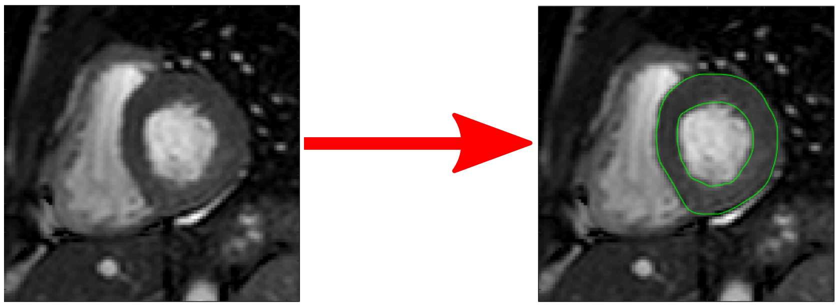


Figure 1: Illustration of myocardial delineation, being the output that we want the final system to produce.

1.2 Machine learning

There exists 'traditional' image analysis methods for image segmentation problems. However, the algorithms are far from perfect, and a lot of expertise and time is required to make them to perform well. Furthermore, the construction of these algorithms for segmenting a particular object is not necessarily translatable to another object, requiring further resources for each new segmentation problem. *Machine learning* is a highly adaptable tool which can be applied to a large array of problems, among them image analysis and image segmentation. By supplying data with already produced end results, e.g. images with an already delineated myocardium, a machine learning algorithm can *learn* to perform a certain task. The machine learning algorithm could be identical for two separate image segmentation problems, the only difference being the training data, producing an algorithm that could be very versatile and simplify the development of medical segmentation tasks. The requirement of data with correctly made (in this case) segmentation is the big issue in machine learning. However, recent development have shown methods that could perform well with lesser quantities of data.

The use of machine learning and later *deep learning* for understanding images and scenes has become very popular during the last decade, proving to be a powerful tool in image analysis. There are a number of reasons for this, the most important factors being computational power and large amounts of available data. The computational power comes from the use of *Graphical Processing Units* (GPU's), able to perform a large number of computations in parallel at a high speed. Deep learning is at the time of writing an a rapidly expanding field, showing very promising results in a large variety of applications. It has been a revolutionary step in the automatic segmentation and classification of objects in images, as well as being very versatile compared to traditional methods. An application that has received a lot of attention is the processing of traffic scene images, where objects need to be segmented and classified for a computer to successfully navigate in traffic.

1.3 Project aim

The aim of this thesis was to produce a deep learning model to automatically segment the myocardium of the left heart ventricle in magnetic resonance (*MR*) images. Thus, we would like a system that takes images and produces a delineation of the myocardium, shown as drawn contours over the image, illustrated in Figure 1. The project was enabled by the fact that a large amount of MR images of the left ventricle and the myocardium of the heart was available. Resources on the cloud service *Microsoft Azure* was available to fulfil the computational demands. The prospect was that available machine learning tools would be mature enough to enable an engineering student with basically no previous experience in machine learning to apply it. The project is among the first in line within the cardiac MR group to utilize machine learning, hopefully able to provide a base for the use of deep learning within the group.

The project goals are summarized in the following list:

- Produce a system to automatically segment the LV myocardium in MR images using deep learning.
- Investigate whether the model learns abstract concepts such as avoiding to classify dark spots in the LV as myocardium.
- Find a method of evaluation that is relatable in a medical perspective.
- Evaluate the importance of data set size.
- Determine which performs best; deep learning models trained on end-systolic and end-diastolic images separately or combined.

1.4 Previous Work

The use of CNN's for semantic segmentation, labelling pixels of an image into differerent classes, has been investigated by different research groups [2] [3] [4]. These publications have mostly been produced using publicly available data sets, such as the Kaggle Second Annual Data Science Bowl[5], with the CNN development in focus. These publications built upon the field of image classification, often using pre-trained networks trained on large data sets of labelled images.

Especially, medical image segmentation using deep learning has been researched [6] [7] [8] [9]. These have also been based on publicly available data sets, however not as large since well labelled medical data is harder to obtain. The focus is often on the technical advances in deep learning, with a certain lack in understanding of the networks, which could be necessary to be able to use the techniques clinically.

Some publications have made 3D segmentation, using the full 3D image information [10]. This is also used when doing manual segmentation, and is information that a CNN seem to be able to learn from. A publication by Tan et al [9] suggested a system of several deep neural networks and a polar representation of the images to extract the myocardial contour in MR images. This work used regression instead of pixel classification to obtain a set of contour points around the endo- and epicardium. This article also used the addition of a Fourier transformation of the image slices during a cardiac cycle, introducing information of the hearts movement during the cardiac cycle.

2 Theory

2.1 Image Processing

A basic principle in image processing is the detection of relevant patterns that says something about the contents of the image. These patterns are what we call *features*. Processing an image consists of extracting relevant features from the image and then using the features to gain information about the image, e.g. that it contains a dog, shows a landscape, if it is night or day etc. Feature extraction can be done by hand, where the programmer can decide on what features are relevant to the problem. The goal would be to extract all features which are relevant to the problem, and create a model that analyse an image based on those features. However, this would need us to have knowledge of all the relevant features of the data, and how important they are in respect to each other. Instead of basing the algorithms on the knowledge that a programmer must give it, the algorithm could extract it's own features, by analysing patterns in the data. This is what we call *machine learning*.

Classification of an image can sometimes be done to several objects in the image. This could be done by defining regions of interest or bounding boxes around the objects in an image, and labelling them with a class, such as dog, cat or fish. To find the contours around different objects in an image is called *image segmentation*. In our case, that means delineating the myocardium in an MR heart image from the rest of the image which is not within the myocardium.

2.2 Machine Learning

In this chapter, a short concise introduction to *machine learning*, *deep learning*, and *artificial neural networks*, and how they relate to each other. A multitude of textbooks, guides and articles exist that explain the different concepts in detail, which the interested reader is encouraged to read. In this section, the concepts will be presented in a simplified fashion in order to give an intuitive understanding about what the algorithms do and how they can be used. The concepts will be exemplified in the case of image segmentation, being the project subject. The theory is mostly drawn from *The Deep Learning textbook* by Ian Goodfellow, Yoshua Bengio and Aaron Courville from MIT [11], as well as standard practices from articles dealing with similar types of segmentation [2] [8].

Machine learning as a term was coined back in 1959 by IBM based Arthur Samuel, a pioneer in the field of *artificial intelligence* (AI). AI includes different fields in the pursuit of developing computers and algorithms that display intelligence, machine learning being one of the fields within AI. Figure 2 illustrates the hierarchy between AI, machine learning, deep learning. Machine learning is a tool within AI that involves the construction of algorithms that use data to make predictions and learn from that prediction, in order to improve the decision making. This can of course be related to *natural learning*, being the way humans and intelligent organism makes a prediction on how to react on a certain situation, and then learning how to react next time in order to yield a better outcome. Machine learning involves defining what a 'better' outcome is, and designing a way of changing the behaviour in order to achieve this outcome. In image segmentation, the machine learning tool allows us to make algorithms that extract features from the images and use them to label the pixels in the image with a class that it predicts that pixel belong to. The better outcome is based on a pre-made segmentation that the model then tries to learn to be able to perform with it's own features. This learning process is an *optimization* of the feature extraction and decision making, where the algorithm updates it's parameters to approach an optimum.

Within machine learning, there are several different approaches to design the algorithm's predictions and how it learns based on the data. One approach is the design of algorithms inspired by the neurons in the brain, dubbed *artificial neural networks*. They consist of a network of elements called neurons which receive input, activates based on that input, and produces an output based on the activation. The layout of this network allows some sort of input enter on one side, activating neurons which then feeds output to the connected neurons and eventually through the whole network, resulting in a final output on the other side.

Machine learning algorithm architectures are often illustrated using a graph of connected components, for example a set of neurons in an artificial neural network, which henceforth will be called a *layer*. The term *deep* learning is applied when we build these layers on top of each other. The deeper layers are fed by the activations of the layer above it, meaning that each layer has a new representation of the data. This enables the algorithm to learn more complex concepts, and separate simple and complex features. For example, a simple feature could be the edges in the image, which in the next layer could be used to determine the corners and contours of objects in the image. By going even deeper the algorithm could determine the shape, size and rotation of the object, and learn a representation of how this object should look like.

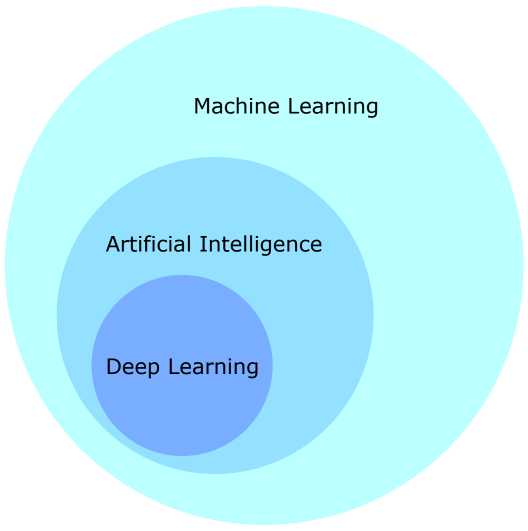


Figure 2: AI is a large research goal consisting of different fields, an important field being machine learning. Machine learning is a computer science field that enables computers to learn a task without being explicitly programmed. Deep learning is a class of machine learning algorithms, mostly based on some sort of artificial neural network.

2.3 Training

First, the term 'learning' should be made clear in the deep learning context. The deep learning textbook [11] refers to Tom M. Mitchell's book 'Machine Learning' [12] for a definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". In this project, T is delineating the myocardial contour, P is how close the predicted contour is to the true contour, and E is updating the model to improve the delineation.

The model learns by updating it's learnable layer parameters, the nature of which is described for every layer above. How the network performs the updates is controlled by a function of the parameters that we call a *loss function*. This is a key component in deep learning, engaging a lot of researchers to evaluate different loss functions as well as proposing specialized loss functions for different applications [13] [14] [15]. The training process consists of optimizing the loss function to a minimum by updating the parameters for each batch of images passed through the network. The operation of updating the network parameters is often referred to as *back propagation*. This can be done by different methods of optimization, a very popular method in deep learning is *Stochastic Gradient Descent*, used in this project. The gradient of the loss function (which can be interpreted as the slope of the function) in respect to the network parameters is calculated for each batch. In mathematical terms, an update of a parameter θ using a loss function L and *learning rate* ϵ is given by

$$\theta = \theta - \epsilon \nabla L(\theta),$$

where ∇ denotes the gradient. This means that a larger step is taken if the slope of the loss function is steep, and vice versa. The learning rate is a crucial parameter (parameters that is not learnable and often controls the training process is in this context called *hyperparameter*), which in practice needs to be controlled during the training progress. In order to successfully find a minimum in a reasonable training time, the learning rate is often initially high, and is then reduced as the training proceeds. How to choose and reduce the learning rate is hard, and there is no definite best way of doing it, but rather common practice. The training progress is often overseen by monitoring the loss function during training, and make decisions based on that. Choosing and tuning the hyperparameters of neural network training is a difficult and cumbersome process, often filled with trial and error. One technique, described by Snoek, Larochelle and Adams [16], is to use Bayesian optimization to find the most well suited set of hyperparameters for a particular training task. This involves training a network with different sets of hyperparameters to find the ones that produce the best model, and is therefore very time-consuming.

The nature of the learning progress suggests that only concepts contained within the training data that the model experiences can be learned from. The application of deep learning on visual recognition involves taking images and labelling the objects in them. This requires a huge amount of already labelled data, a well known example being *ImageNet*, containing at the time of writing over fourteen million images with labelled objects. ImageNet has been around since 2009 and has stimulated a lot of great deep learning research. However, for the myocardial segmentation application, the necessity for that amount of data is lesser, since the object is very alike in each image. The risk of using a small amount of data is instead a phenomenon we call *overfitting*, where the model heavily specializes on the training data. This does not necessarily produce a general model good at segmenting a general myocardium, but especially the training images. Thus, the training data is very important, and the less data you have the more important it is that the data is correct and representative for what you want to model.

Research groups have however shown techniques, including network architectures, the use of batch normalization and *data augmentation*, to produce accurate models using a small amount of data [8] [2]. Data augmentation is very important in deep learning, and is the process of increasing your effective data set by producing new, altered versions of your data. When using images, it often includes using translation, scaling, rotation and deformations in the images, which can increase your data set significantly. If the augmentation introduces something that could be present in 'real' images, the model can become more invariant to such things as object size and rotation, making it more general. Care should be taken not to show the model images that it would never encounter only for the sake of increasing your data set, as it will only produce a less specific model with no gain.

In practice, the training consist of passing images from a training set through the network and updating parameters a large number of times, hoping that a minimum producing a well-performing is reached. Each pass through the entire training set is referred to as an *epoch*.

2.4 Deep Convolutional Neural Networks

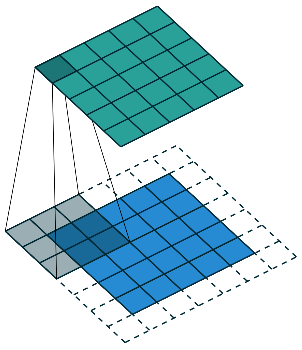
A deep *convolutional neural network* (CNN) is an artificial neural network which uses *convolutions*, which are a kind of linear mathematical operation, explained below. CNN's are designed to process data that has a known grid-like topology, such as images, that can be thought of as a 2-D grid of pixels. The network uses the 2-D grid of data as input, each operation produces an output that is fed into the next operation, which is fed into the next operation, and so on. In this chapter, the different operations used in the network architectures used in the project will be reviewed. The focus will be on the usage and importance of the operations, and once again the interested reader is encouraged to read further in the more comprehensive literature.

Neural networks are often said to contain different kinds of *layers*. What this term means may vary, in this thesis it refers to several consecutive operations considered to be connected or that their usage is dependent on each other. This simplifies the understanding of the mechanisms behind the neural network and it's layers. Especially, the convolutional layers described in the network architecture (section 4.3) consist of a convolution, followed by a *batch normalization* and then a *rectified linear unit*, all explained below. Another common terminology is to name each operation a layer, which is more specific and suitable for deep learning platforms, where you would like to choose single operations at will.

Convolutional filters

The interpretation of a 2D convolution operator on an image will be considered, as it is type the convolution that will be used. The operation is more accurately a discrete convolutional filter, which consist of a *kernel* of some size, often square (in this thesis we always use 3x3 filter kernels for convolution). The convolution is a weighted average, calculating the sum of the element wise matrix multiplication between the kernel and an area of the image, called *perceptive field*, producing an output of one pixel. This calculation is done for every output pixel by sliding (the amount of pixel steps in one slide is called *stride*) the perceptive field over the entire input image, producing an output image called a *feature map*, as illustrated in Figure 3. The values making up the kernel is what we call *parameters* or *weights*, which are learnable, meaning that they can be changed during the training process.

When convolving the images, padding is used to make sure that the output feature map has the same size as the input image or feature map. The opposite of a convolution is a *transposed convolution*, basically reversing the operation. This can be used to upsample feature maps to higher dimensions, which is done in the proposed *encoder-decoder* architecture.


(a) A convolutional filter operating using padding, resulting in an equally sized output as input.

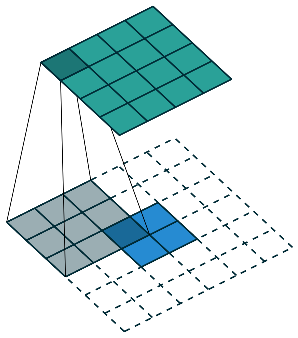

(b) A transposed convolution, resulting in an upsampled output.

Figure 3: An illustration of convolutional filters operating on an input, i.e. an image, (coloured blue) producing an output, i.e. a feature map (coloured green) [17].

Activation Function

If a network would only use linear operations, such as convolutions, all operations could be simplified into one single operation. This would erase the complexity of the deeper layers, reducing our network to the complexity of one single operation. However, we would like our model to be as close to linear as possible, since models are more easily optimized if their behaviour is closer to linear [11]. To introduce non-linearity into our network, a non-linear *activation function* is applied to the output of each convolution operation. The *Rectified linear unit* (ReLU) use the activation function $g(z) = max0, z$, which is a function that is close to linear.

Batch Normalization

Batch normalization is a concept introduced by Ioffe & Szegedy [18] in order to counter what they call *internal covariate shift* and allow for faster training. Internal covariate shift is an effect where the distribution of each layer's input changes during training as the parameters of the previous layers change. To put it in simpler terms we recognize that a layer learns based on the input that the previous layer fed into it, and updates it's parameters accordingly. However, the previous layer also updates it's parameters, causing it to change the output it produces. This causes a problem, which is in practice countered by the use of ReLU's, careful initialization and small learning rates, slowing down the training. Since the problem lies within that the input of a layer is affected by all precious layers, the problem becomes larger the deeper the network.

By using batches of data, in our case batches of several images, and normalizing each batch activation by both mean and variance, the internal covariate shift is remedied. This allows for the use of larger learning rates, as well as performing computations on several images at the same time. This makes use of the power of the parallel computational powers of GPU's, only limited by the necessity of enough GPU memory to process the whole batch. Batch normalization is performed after each convolution in the convolutional layer.

Momentum

Taking steps using the gradient of the loss function can result in undesirable effects. For example, the steps could oscillate along the steepest descent path to the optimum, avoiding convergence. A way to counter this is to introduce *momentum*, where we add a contribution of the previous gradient step the the current iteration. How much contribution the previous iteration should have on the current is decided by a momentum factor, defined before training.

Weight Decay

A method to reduce overfitting is to introduce a *weight decay*, where we introduce another criterion that should be minimized during training. A useful criterion is the preference for the weights of the network to have a smaller squared L^2 norm, this is called *L2 regularization*. This results in a choice of weights that make a tradeoff between fitting the training data and being small. How much this preference is enforced is determined by a *regularization factor*, defined before training.

Pooling

For images, the *pooling operation* produces an output that uses some sort of summary statistics of a neighbourhood of pixels. The most common pooling operation is *max pooling*, illustrated in Figure 4. Max pooling extracts the maximum value within a rectangular neighbourhood around each pixel and reports that value as output. As exemplified in Figure 4, the pooling operation can be used to downsample the feature map. This is a way of extracting a more complex representation of the features, e.g. being able to represent a contour from detected edges, at the cost of information of the location of the features. This also makes the representation approximately invariant to small translations of the input, which will prove beneficial in the countering of *overfitting*, see section 2.3. The pooling operation may vary in filter size and stride in the same way as the convolutional filter, the most common being a small filter size (2x2) and a stride of 1, to prevent the loss of too much location information.

The loss of locational information is very damaging in image segmentation, where the location of the boundaries is central. To be able to use pooling in order to obtain more complex feature maps, the feature maps may be saved and passed to the transposed convolutional layer performing the corresponding upsampling. This is used in the *U-net* architecture, where the feature maps from the max pooling are concatenated with the input of the transposed convolutional layers, thus retaining the locational information. This technique is often referred to as a *skip connections*. A more memory efficient solution is employed by *SegNet*, where only the pooling indices of the max pooling layer is saved, and passed to the transposed convolutional layer at the same depth (see section 4.3).

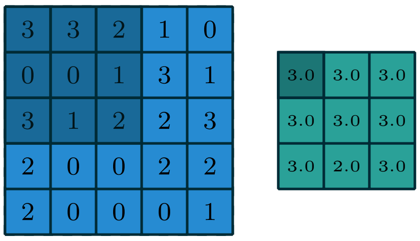


Figure 4: Max pooling illustrated numerically, where the filter size is 3x3 and the stride is 1. The matrix before the pooling is coloured blue and the matrix after is coloured green, with an example filter and output highlighted in darker colour.

Softmax and Classification

To generate a segmentation from feature maps, some sort of classification of each pixel is needed. This is done by putting a *pixel classification* layer at the end of the network. Based on the feature maps, this layer makes a prediction of what class the pixel is most probable to belong to. This is also based on the weight of the classes, e.g. how large amount of the total number of pixels belong to the different classes. Before the classification layer, it is common practice to put a so called *softmax* layer. The softmax function squashes the feature map values between 0 and 1, the probability of a pixel belonging to each class, the sum of which is 1 for every pixel.

2.5 Evaluation

There exist no single gold standard measurement to evaluate image segmentation, and different researchers and segmentation challenges use different measures. This calls for an understanding of what the different measures represent, as well as their strength and weaknesses. The general opinion seem to be that several measures should be used in conjunction to illustrate the performance of a segmentation model.

One important concept is the distribution of correctly and incorrectly classified pixels, where the correctly classified pixels are *true positives* (the pixel is correctly predicted to be myocardium) and *true negatives* (the pixel is correctly predicted to be background). The incorrectly classified pixels can either be *false positives* (the pixel is predicted to be myocardium but is actually background) or *false negatives* (the pixel is predicted to be background but is actually myocardium). In some applications, the effects of false positives and negatives may vary, calling for a stricter punishment on the model producing one or the other (this is generally made by a specialized loss function). The *Jaccard index* is a statistic used to compare the similarity of two sets, in this case the sets of pixels classified as myocardium. It is also known as the *intersection over union*, since it is defined as

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

Here, A is the pixels classified as myocardium by the network, and B is the ground truth myocardium pixels. The intersection (denoted $\cap$) of A and B is the true positives, and the union (denoted $\cup$) is the sum of the true positives, false positives and false negatives. We can consequently interpret the formula as

$$\text{Jaccard Index} = \frac{true\ positives}{true\ positives + false\ negatives + false\ positives}.$$

The Jaccard index describes how well the predicted and ground truth segmentation overlaps. However, it does not necessarily depict how similiar the produces contours are, which is of greater concern when a human would visually inspect the segmentation.

The *BF score* was proposed by Perronin et al [19] as a measure of how similar two boundaries are to each other. The authors suggest that this measure should be used in conjunction with the Jaccard index, to provide a fuller understanding of segmentation performance. The BF score is a value between 0 and 1, and is based on if the segmentation boundary point belonging to the ground truth boundary. This is decided within a tolerance, which in the original article [19], as well as in the default MATLAB setting was set to 0.75 % of the image diagonal. This setting was also used for this thesis.

In the evaluation of the networks, the *accuracy* will also be posted. The accuracy is just simply the quotient of the amount of correctly classified myocardial and the total existing myocardial pixels. It is used since it is the accuracy that the training uses for optimization.

3 Project Process

Several publications on using deep learning for medical imaging segmentation problems, including LV segmentation, were studied in preparation for the project. Since there were quite a few publications on the subject, it was decided that the focus of this project would be to determine the value of applying pre-existing deep learning implementations to our data set. The intention was to make use of the cloud service *Microsoft Azure* and its machine learning platform *Azure Machine Learning Studio*. However, it turned out that this platform did not suit the projects needs, as it did not contain any pre-existing deep learning algorithms for images. Early in the project this solution was therefore dismissed and other options were investigated, hoping that the computational resources on Azure could be used with another deep learning framework. Since the amount of deep learning frameworks are many, the focus was to find a suitable network architecture that could easily be implemented and applied to LV segmentation.

The *U-net* segmentation architecture, suggested in 2015 by Ronneberger et al [8], was determined to be an compelling architecture for the project. U-net has since its publication been proven to be very powerful for different medical image segmentation tasks, and the publication is well cited[1]. The architecture has also been used as a base network and has been improved further and used in solutions of specific tasks, for instance a 3D segmentation scheme which would be very relevant for LV segmentation. This made it ideal to use as a starting point for LV segmentation by a student in the Cardiac MR group, neither carrying much deep learning experience. The U-net architecture was originally implemented using *Caffe*, a deep learning framework originating from *Berkeley AI Research*, making it the choice of framework to use.

Cloud instances provided by *Amazon Cloud Services*, equipped with the Caffe framework, was used as computational resources. Attempts to apply the U-net architecture to LV segmentation was made, but did not produce any promising results. The Caffe framework turned out to be difficult to use. Especially, it was very hard to determine in what step the models failed since the implementation of the pre-existing network architectures were unknown to us.

Finally, MATLAB's "Neural Network Toolbox" was used for the project. It proved to be easy to use, with plentiful documentation and support, as well as providing a readily available network architecture similar to U-net, *SegNet*. It's provided sample architecture was first utilized to get results as quickly as possible, this network was dubbed *SegIshNet*. The network was then improved by using the originally published *SegNet* architecture. MATLAB also contains readily available tools and guides to use Bayesian optimization for the tuning of training hyperparameters, which seemed promising. This was however not carried out due to its very time-consuming nature.

An alternative approach would be to utilize a polar image representation of the LV suggested by Tan et al [9], was studied early on in the project. The publication utilizing it suggested that the method imposed model constraints which yielded a more physiologically correct representation. The polar remapping required was implemented early in the project, but due to lack of time largely caused by the issues of finding a suitable deep learning platform, this method was not tested further.

[1] 784 citations according to Google scholar, *2017-11-27*

4 Methods

4.1 Data Set

The available images have been taken using a 1.5 Tesla and a 3 Tesla MRI system by Siemens, as well as a 1.5 Tesla system by Philips. The images have been acquired for several different studies performed by the Cardiac MR Group at the Skåne University Hospital in Lund. The data for each patient represents a 3D image volume of the torso in several time frames, capturing one full cardiac cycle. On the end-systolic and end-diastolic images, delineations have been made by experts from the Cardiac MR Group. In total, a data set of 6973 images was used.

The data set was acquired by writing a plug-in for the software segment to save the relevant images. The end-systolic and end-diastolic images depicting the heart was selected, consisting of around 12 slices for each time frame. The top slice and the bottom two slices were omitted in order to produce a data set of mid-ventricular slices. Patient images where the myocardial volume differed more than 2 % between the end-systolic and end-diastolic time frames was discarded. The data set was split into end-systolic and end-diastolic images, we call the two data sets *Systolic* and *Diastolic* data sets. A data set consisting of both these data sets was also used, we call this data set the *Merged* data set. This merged data set use the same training and testing data as the systolic and diastolic data sets, to make sure that the merged model can be compared to the single time frame models. This was done in order to be able to compare how the networks trained on the separate data sets performed versus the networks trained on the combined data set.

4.2 System Design

The proposed system design first uses pre-processing step, consisting of cropping and resampling, producing an image ready for the network. The system then passes the image through the network, producing a pixel-wise labelled image of the myocardium- and background (in this case, all non-myocardium pixels are background) pixels. The labelling is then post-processed, producing a contour of the segmented myocardium and remapping it to the original image. Thus, the potential user of the system will apply it to an image, producing a contour of the myocardium drawn on the image, as shown in Figure 5, and the details of the system steps are presented further below.

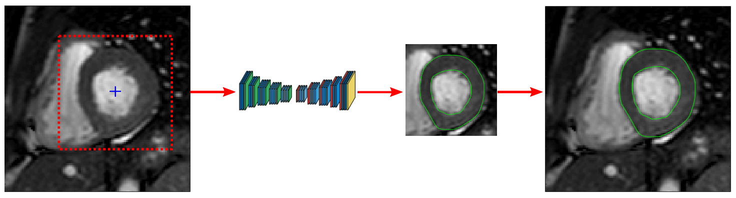


Figure 5: Illustration of the proposed system steps for myocardial segmentation. The image is first resampled and then a center-crop around the LV center is passed trough the trained network. The obtained segmentation is remapped to the original image size and resolution, and the contours are displayed on top of the original image.

The network takes images of uniform size as input. However, the images in the data set vary in both size and in spatial resolution. To counter this, we utilize the following pre-processing steps

- Resampling the images to the same spatial resolution, 1.5 mm/pixel.
- Obtaining the LV centerpoint.
- Extracting a LV centercrop image by cropping a 59 x 59 pixel (corresponding to 88,5 mm) image around the LV centerpoint.
- Resampling the image to 128 x 128 pixels, with a spatial resolution of about 0.69 mm/pixel .

This produces images of equal spatial range and resolution. The images also as a more even distribution between myocardium and background pixels than the original, which will hopefully improve the performance of the network. The LV centerpoint was obtained by calculating the center of mass of the delineated endocardium for the training and testing images. With no previously delineated myocardium, which is the target case, the extraction of the centerpoint will need to be implemented, for example using pre-existing code within Segment [20].

An inference pass of the network then produces a 128 x 128 pixel classification map, either classifying a pixel as myocardium or background. The contour of the myocardium is extracted from the map and then remapped to the original image size and resolution.

4.3 Network Architecture

SegNet

The network architecture used is called SegNet, proposed by Badrinarayanan et al [2] from the *Computer Vision and Robotics Group* at the University of Cambridge, UK. The SegNet architecture is a fully convolutional network consisting of an encoder and decoder network part, where each encoder has a corresponding decoder. In Figure 6 we see encoders, which consists of two or three convolution 'blocks' (coloured blue). These consist of a convolutional layer kernel size of 3 x 3, followed by a batch normalization layer and then a ReLU layer. After the convolution blocks a max-pooling layer (coloured green) with a 2 x 2 window and stride 2 is places. This downsamples the feature map, and at each downsampling we double the number of feature channels. The max-pooling indices from each max-pooling operation are saved and used in the corresponding decoder, as discussed in Section 2.4. The decoders are, in a sense, mirrored versions of the encoders, but with deconvolutional layers (coloured red) that upsamples the feature maps using the max-pooling indices from the encoders. At the upsampling steps we half the number of feature channels. At the end of the network there is a softmax layer (coloured yellow) followed by a pixel classification layer, producing a classifications for the full input image.

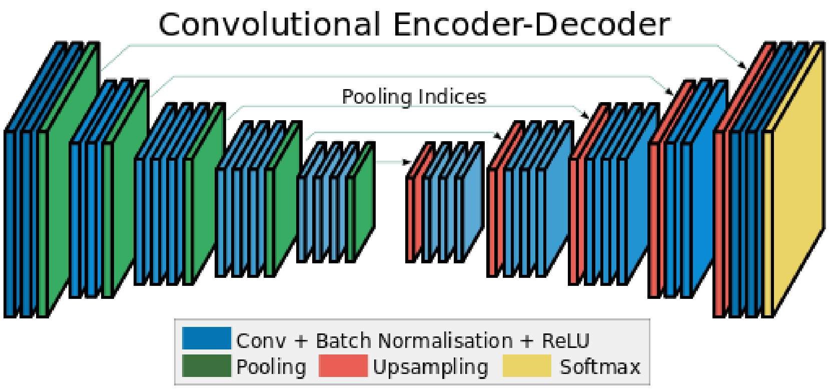

Convolutional Encoder-Decoder
Pooling Indices
Conv + Batch Normalisation + ReLU
Pooling
Upsampling
Softmax

Figure 6: An illustration of the *SegNet* architecture, taken from the original article [2].

SegIshNet

The *SegIshNet* architecture is a simplified version of the *SegNet* architecture, where the encoders always consist of two convolution blocks. The number of feature channels are kept the same, differing from *SegNet*, resulting in a lower memory requirement. This was the first network that we could use that could be found in MATLAB, and was in order to start training as quickly as possible.

4.4 Training scheme

The learning parameters was kept the same for each training, with the exception of batch size between *SegNet* and *SegIshNet*, as well as the reduction of number of training epochs for *SegNet* on the large data set. The base learning rate was set to 0.01, with a momentum of 0.9. The training was split evenly into 4 phases (50 epochs of each for all but *SegNet* on the large training set, where it was 15 epochs), and the learning rate was reduced by 10 for each new phase. The regularization factor was set to 0.0005. The batch size was 24 for *SegIshNet* and 12 for *SegNet*, which was opted to be as large as the GPU could handle without running out of memory. The training data was shuffled between each epoch.

The data was augmented by random translation in all directions, in the interval [-5 5] pixels. Augmentation that could be physiologically realistic was avoided, and it's investigation left to further research projects.

4.5 Software

The implementation and training of the networks was done using MATLAB 2017b, and especially utilizing it's *neural network toolbox*. The early stages of the project with no yield to the results of the thesis used Amazon Cloud Service instances and the Caffe deep learning framework. The training was performed on a GPU with 2 GB of internal memory.

5 Evaluation

5.1 SegIshNet

Using the *SegIshNet* architecture, models were trained according to the training scheme in section 4.4, on the three data sets. The *Systolic* and *Diastolic* data sets were randomly split into 80 % training and 20 % testing data, and the *Merged* data set used the same training and testing data as the two others, but combined.

The performance was evaluated by performing segmentation on the test set and comparing it to the ground truth. Three measures was calculated, the accuracy and the Jaccard index of the myocardial pixel classification, and the mean BF score of the myocardial contours. The merged data set was tested on it's complete test set as well as the systolic- and diastolic test set separately. Table 1a displays the results of the evaluation of the networks.

The *Merged* network performs better than the *Systolic* and the *Diastolic* network, even on the separate data sets, at least on average. This endorsed the later training using only the *Merged* data set when the *SegNet* architecture was adopted.

The same training was performed using a smaller training set, where 8 % of the data set was used for training and 92 % was used for testing. In all other aspects, the training and evaluation was performed identical to the above used training set. Table 1b shows the results of the evaluation using the small training set. Using a smaller training set results in a severe performance decrease, as expected. The results confirms the superiority of the *Merged* network versus the other two.

Table 1: Measured performance of the *SegIshNet* networks trained on the *Systolic*, *Diastolic* and *Merged* data sets. The networks are named after what data set they were trained on. The scores are the mean scores of the network classification of the Myocardium class on the entire test set.

(a) Performance of networks trained on the **large** training set.

Data Set	**Network**	**Accuracy**	**Jaccard**	**BF Score**
Systolic	Systolic	0.96	0.79	0.42
	Merged	0.96	0.82	0.48
Diastolic	Diastolic	0.95	0.76	0.49
	Merged	0.96	0.78	0.53
Merged	Merged	0.96	0.80	0.51

(b) Performance of networks trained on the **small** training set.

Data Set	**Network**	**Accuracy**	**Jaccard**	**BF Score**
Systolic	Systolic	0.90	0.60	0.19
	Merged	0.91	0.68	0.28
Diastolic	Diastolic	0.94	0.46	0.14
	Merged	0.94	0.61	0.25
Merged	Merged	0.92	0.64	0.26

5.2 SegNet

Using the *SegNet* architecture, a model was trained on the merged data set using the same training and testing data as used for *SegIshNet*. It was trained for 60 epochs with a decrease in base learning rate by 10 every 15 epochs (similar scheme as earlier but with a reduced number of epochs, due to a slower learning process). A model was also trained on the 8/92 data split set, using the exact same training scheme as for *SegIshNet*. The networks was then evaluated in the same way on the test sets. The evaluation of *SegNet* is shown alongside the corresponding *SegIshNet* performance in table 2a.

Table 2: Measured performance of the *SegNet* and *SegIshNet* networks trained on the *Merged* data set, for both the larger and smaller training/testing data split. The scores are the mean scores of the network classification of the Myocardium class on the entire test set.

(a) Performance of networks trained on the **large** training set.

Data Set	Network	Accuracy	Jaccard	BF Score
Systolic	SegIshNet	0.96	0.82	0.48
	SegNet	0.96	0.83	0.52
Diastolic	SegIshNet	0.96	0.78	0.53
	SegNet	0.96	0.81	0.59
Merged	SegIshNet	0.96	0.80	0.51
	SegNet	0.96	0.82	0.56

(b) Performance of networks trained on the **small** training set.

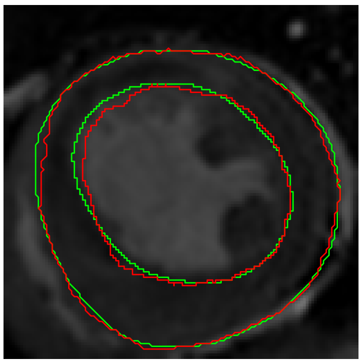
Data Set	Network	Accuracy	Jaccard	BF Score
Systolic	SegIshNet	0.91	0.68	0.28
	SegNet	0.91	0.78	0.43
Diastolic	SegIshNet	0.94	0.61	0.25
	SegNet	0.92	0.75	0.48
Merged	SegIshNet	0.92	0.64	0.26
	SegNet	0.92	0.76	0.46

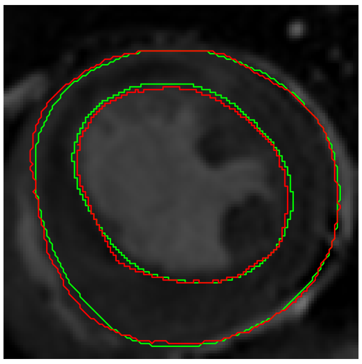
We observe a conclusive superiority of *SegNet* over *SegIshNet*. An important note is that *SegNet* trained on the large data set was trained for a reduced number of epochs than the corresponding *SegIshNet*.

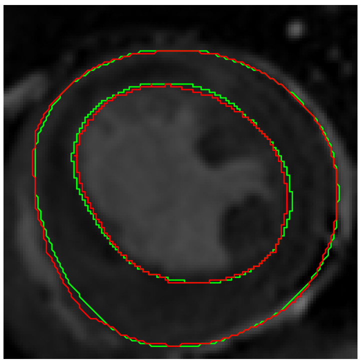
5.3 Visual Inspection

For the visual inspection, the contours were extracted using the system scheme proposed in Section 4.2. The segmented contour are shown alongside the ground truth contour on the center-cropped images.

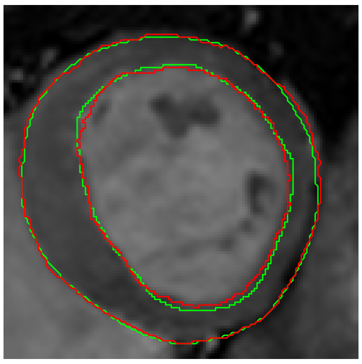
Separate *SegIshNet* **Merged *SegIshNet*** ***SegNet***

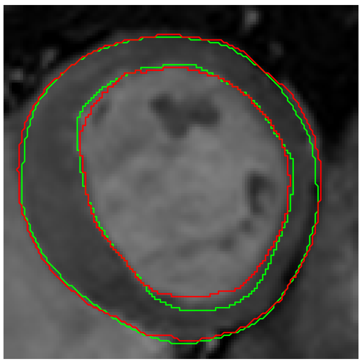

Jaccard: 0.87 BF: 0.69

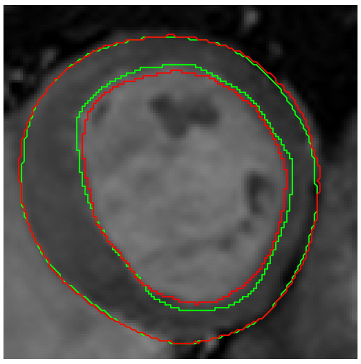

Jaccard: 0.89, BF: 0.69


Jaccard: 0.92, BF: 0.82

Good delineation of a systolic image with papillary muscles in the LV, illustrating the superiority of *SegNet*.

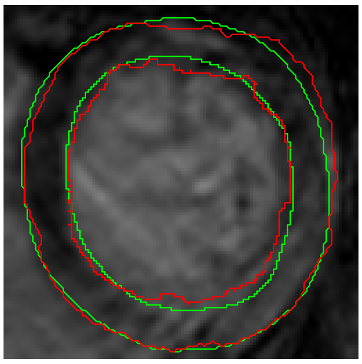

Jaccard: 0.89, BF: 0.80

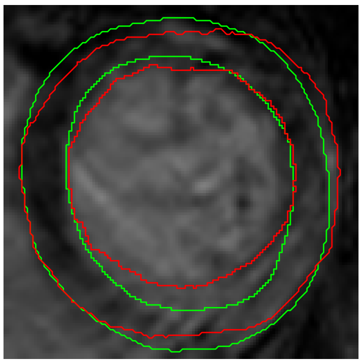

Jaccard: 0.85, BF: 0.72

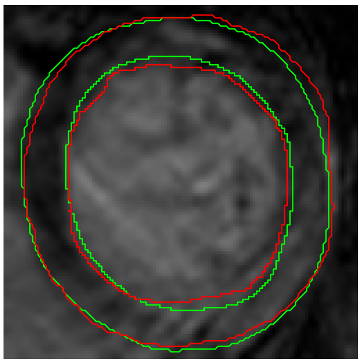

Jaccard: 0.86, BF: 0.65

Good delineation of a diastolic image with a varying myocardial width. In this case, the specialized *SegIshNet* performs best.

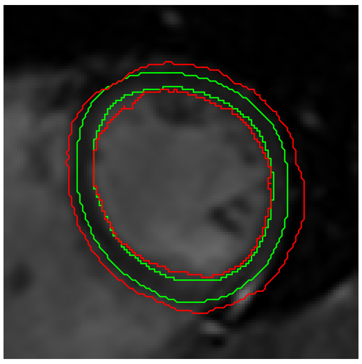
Separate *SegIshNet* **Merged *SegIshNet*** ***SegNet***

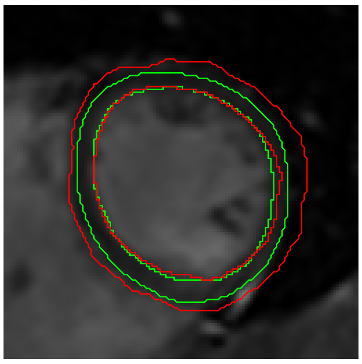

Jaccard: 0.78, BF: 0.46

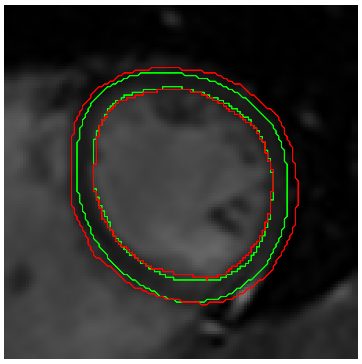

Jaccard: 0.65, BF: 0.32


Jaccard: 0.78, BF: 0.39

Good delineation of a grainy diastolic image, where the merged *SegIshNet* is worse than the diastolic.

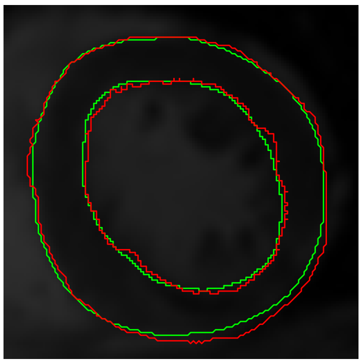

Jaccard: 0.53, BF: 0.28

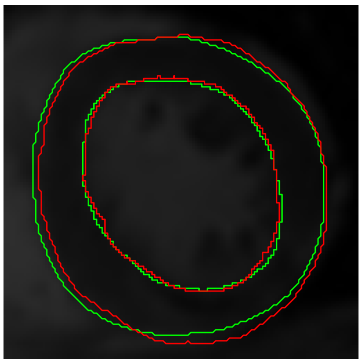

Jaccard: 0.50, BF: 0.37

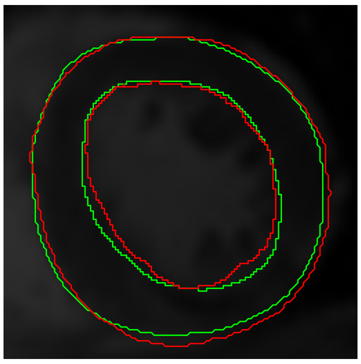

Jaccard: 0.64, BF: 0.47

Diastolic image slice with a good delineation of the epicardium on all sides, even though the intensity outside the myocardium is different on the different sides.

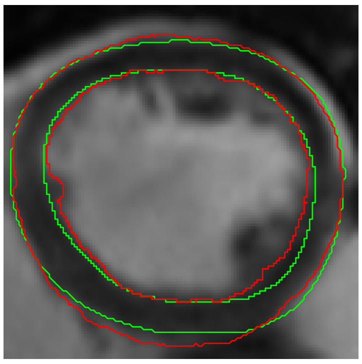
Separate *SegIshNet* **Merged *SegIshNet*** ***SegNet***

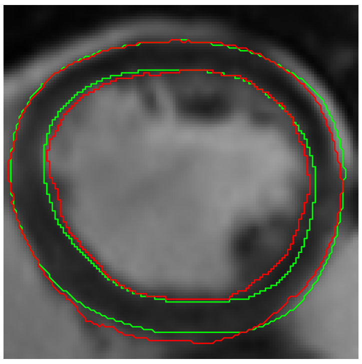

Jaccard: 0.86, BF: 0.63

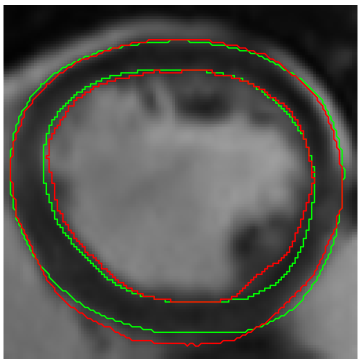

Jaccard: 0.83, BF: 0.52


Jaccard: 0.83, BF: 0.43

Diastolic image which is successfully delineated despite the low contrast.

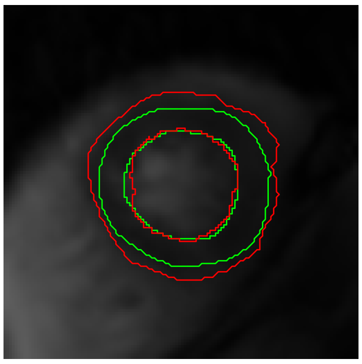

Jaccard: 0.77, BF: 0.57

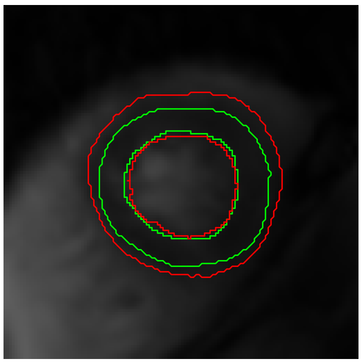

Jaccard: 0.78, BF: 0.54

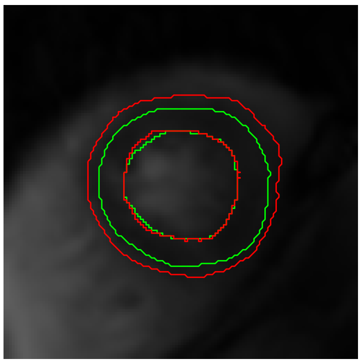

Jaccard: 0.78, BF: 0.58

Diastolic image where we observe an issue with delineation in the presence of dark spots within the LV.

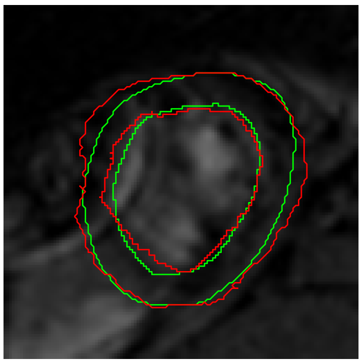
Separate *SegIshNet* **Merged *SegIshNet*** ***SegNet***

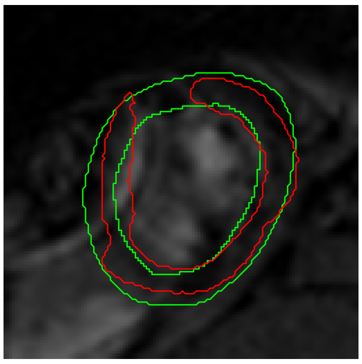

Jaccard: 0.62, BF: 0.32

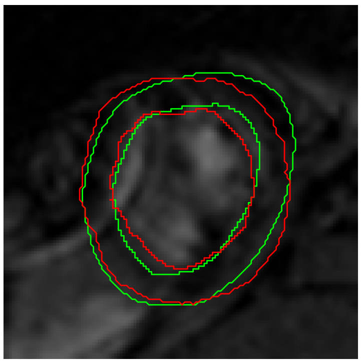

Jaccard: 0.60, BF: 0.23


Jaccard: 0.63, BF: 0.35

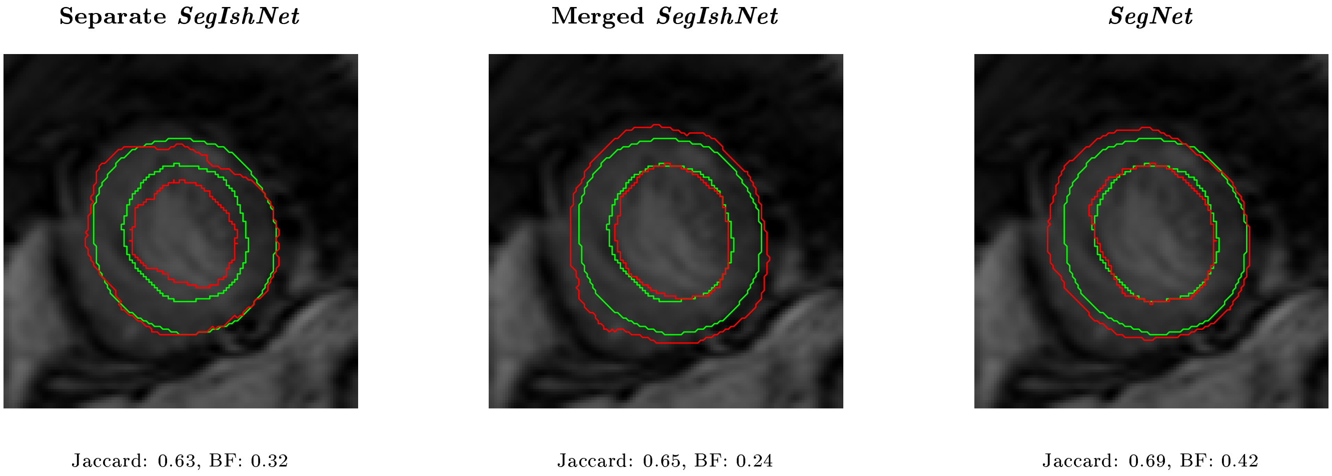
A diastolic image slice close to the base of the heart with a small myocardium area, where the delineation of the epicardium is put outside the true contour. In this case, the separate *SegIshNet* and *SegNet* yields similiar Jaccard Index, but the BF score is different. This can also be seen in the images, where the contour produced by *SegNet* is more in line with the ground truth.


Jaccard: 0.72, BF: 0.44


Jaccard: 0.53, BF: 0.19


Jaccard: 0.65, BF: 0.31

Diastolic, grainy image slice close to the base of the heart, where the delineation is unsuccessful by the merged *SegIshNet*, and *SegNet* performs worse than the specialized *SegIshNet*.

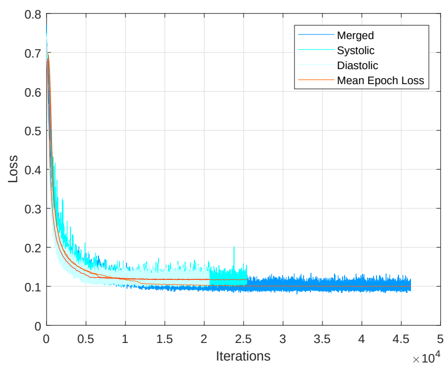


Systolic image slice close to the base of the heart, further showing the issue of delineating the epicardium and the tendency of drawing the contour too far out.

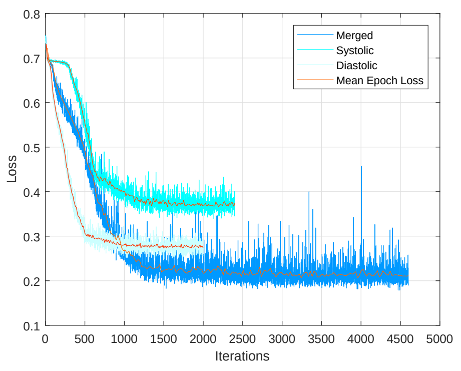
Figure 11: Visual evaluation of the network performances. The ground truth contour is displayed in green and the contour produced by the network is displayed in red. The images have been picked out if they show a certain behaviour of the models, or if they display a certain difference between the performance of the different networks. The delineations have been made by the separate (systolic and diastolic depending on the image) and merged *SegIshNet*, and *SegNet*. The individual BF and Jaccard scores are displayed under each image, for comparison and further showcasing the nature of the scores.

5.4 Training characteristics

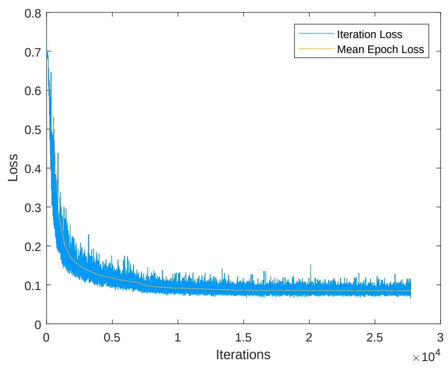
The minimisation of the loss function is an indicator of the training progress. It is, however, not necessarily a measure of the performance of the produced model, which we ideally would like to monitor during the training progress. The loss function should however enable conclusions to be drawn about the nature of the different networks and their performance. Figure 12 shows the loss function during the training progress of the *SegIshNet* and *SegNet*.



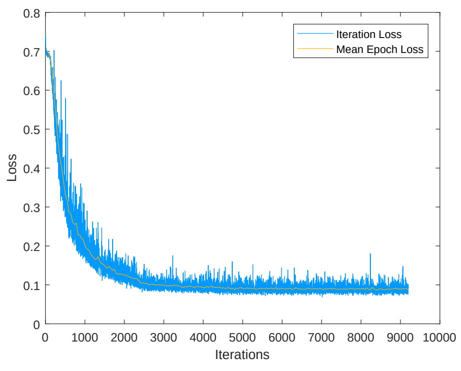
(a) *SegIshNet* trained on the **large** *Systolic*, *Diastolic* and *Merged* data sets.



(b) *SegIshNet* trained on the **small** *Systolic*, *Diastolic* and *Merged* data sets.



(c) *SegNet* trained on the **large** *Merged* data set.



(d) *SegNet* trained on the **small** *Merged* data set.

Figure 12: The loss function for each iteration during training, coloured in blue, as well as the mean loss over the epoch, coloured in orange. 12a and 12b displays the loss for the networks trained on the *Systolic*, *Diastolic* and *Merged* data sets in different shades of blue. An important note is that the number of iteration depends on the number of training epochs and the batch size. *SegNet* is trained using half the batch size of *SegIshNet*, and only for 60 epochs for the large data set, resulting in different number of iterations.

6 Conclusion and Discussion

6.1 Network Performance

A segmentation model for the left ventricle of the heart using deep learning was produced, fulfilling the main project goal. The models shows promising results in some cases, successfully segmenting the myocardium while handling difficulties like papillary muscles. The networks have learned not to just take intensity edges into account, but also that dark spots may appear in the LV, and successfully includes the dark spots in the LV. This really showcases the strength of the learning process, showing that a concept that can be hard to manually implement into a model is easily trained by showing the model examples. The network successfully delineates the endocardium when papillary muscles are present and when they are not, showing the generality of the model. The model is also invariant to scaling, being able to segment smaller and larger than average myocardium cases. However, the outer delineation generally is segmented further out for the smaller cases. This scale invariance could possible be corrected by augmenting the data set to further increase the variance of myocardial wall thickness. This could make the model more general and improve it's performance on varying sizes.

For the myocardial segmentation problem, the mixing of images from the end-systolic and end-diastolic time phases in the training set proved to produce a more powerful model. The more general model proved to be better than the specific models even on the specific data sets. This could be due to that the specific model is more likely to overfit on the training data, producing a model that learns more about the training data itself than about the properties of the myocardium in general. Another cause could be that images in the data set that result in "bad" training is less of an issue in a larger data set. However, this should be countered by the use of batches during training. Intuitively, the specific data sets should produce a better performing model on that specific data, however countermeasures to overfitting and other possible causes would be more crucial. As a first step toward a well functioning model, it is concluded that it is favourable to use the full data set in this case.

For the small data set, the loss function plots clearly show why the training of *SegIshNet* is unsuccessful. They converge toward a greater magnitude of loss function than for the large data set, meaning that we do not reach the same level of optimization of the model. The greater success of the *Merged* network in this case is more probably the result of a the greater number of training data, since the separate data sets yields a even worse level of optimization. In contrast, the achieved loss does not vary much between the different data sets when using a large training data set. *SegNet*, however, does reach a low minimum, as well as decent segmentation performance on the small data set. This indicates that we could possibly train *SegNet* on a smaller data set, which makes it easier to study the data set and ensure that it is representative for how a myocardium could look like. It also allows for the use of specialized networks for different parts of the heart, e.g. basal or apical slices, which would produce smaller data sets if they were divided. Here, heavy data augmentation could prove very useful.

When evaluating segmentation, the use of complementing measures is encouraged, and the understanding of what the measures represent is important. The visual inspection of the segmentation is very important to understand the performance, strengths and limitations of the network.

This architecture and approach should be able to be applied to similar segmentation problems, e.g. the segmentation of other organs in the body as well as being used for different imaging techniques. This could be done very quickly, only requiring the work of obtaining a data set. This will hopefully serve as a basis for the continued application of deep learning in the Cardiac MR Group, and a fast way of testing feasibility of using deep learning for future segmentation problems. The experienced difficulties in the usage of deep learning platforms will hopefully be valuable lessons for future work, and reduce the effort required.

6.2 Limitations

There exist some limitations and problems with the system and networks. We list those which have not been corrected due to lack of time, and refer to Section 7 for the ones that is outside the scope of the thesis and pose challenges for further work.

Crop size The crop size is at the moment too small for some cases. We would like to investigate and find a minimum size so that we always crop the whole myocardium, and crop all images to that size.

Data set The data sets contain some image slices which are borderline cases if they actually are midventricular slices. However, we produce decent delineations on these images. We would like to thoroughly separate data sets between basal, apical and midventricular slices. This would allow for an investigation similar to the combination of end-diastolic and end-systolic slices performed in this thesis.

Pixel classification The pixel classification approach is strong for classifying several different classes, such as in a traffic scene. For the myocardial segmentation application, the discrete way of classifying a pixel as either myocardium or background is not necessarily the best approach. If the certainty of the classification of a pixel is low, we would rather draw the contour through the pixel than around the pixel based on the discrete classification. This is especially true when the image resolution is low. Therefore, we would like to trace a contour by directly using the output from the softmax layer to avoid the uncertainty of the discrete pixel classification. One example of this could be to use an implementation of *Dijkstra's algorithm.*

7 Future Work

The goal of a functioning system that could compete with existing automatic LV segmentation algorithms is still far away. Experimentation with different network architectures, training schemes and data augmentation is still unexplored. Another attempt to improve the model would be to label the images with more specific labels, for example labelling pixels belonging to the left and right ventricle.

The investigation of whether to use general networks trained on all sort of image slices or divided into basal, apical and mid-ventricular slices would give insight in how to tackle such a diverse data set. Further work could also be to train a general model on a diverse data set, and further train it on specific data sets. The general approach is clearly superior, and could then be split into several networks fine-tuned for each specific case.

The nature of the network training process is not investigated in this thesis, more than the monitoring of the loss function during training. An approach is to test the network on a so called *validation set*, separate from the training and test set, during training. This gives an indication of the network's performance during training. This was not available for pixel segmentation in MATLAB during the project. However, the latest version of MATLAB, 2018a, is said to to include this feature. This would be a great asset in the study of when the important aspects of the model is trained and how to design the training scheme. The training parameters could also be optimized by using Bayesian optimization, readily available in MATLAB. This involves training several networks with different parameters in an attempt to find an optimal set of parameters.

Especially, replacing the transfer of pooling indices with skip connections, discussed in Section 2.4, utilized in the U-net architecture is promising to produce a more accurate contour. The cost of this would be a higher memory requirement, which would not pose a problem considering the application does not pose any strict demands on processing speed.

8 References

[1] S. Mendis, P. Puska, B. Norrving, World Health Organization, World Heart Federation, and World Stroke Organization. *Global Atlas on Cardiovascular Disease Prevention and Control*. Nonserial Publications Series. World Health Organization in collaboration with the World Heart Federation and the World Stroke Organization, 2011.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[3] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1605.06211, 2016.

[4] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

[5] Data science bowl cardiac challenge data. `https://www.kaggle.com/c/second-annual-data-science-bowl`. Accessed: 2018-02-08.

[6] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42(December 2012):60–88, 2017.

[7] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep Learning in Medical Image Analysis. *Annual Review of Biomedical Engineering*, 19(1):221–248, 2017.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[9] Li Kuo Tan, Yih Miin Liew, Einly Lim, and Robert A. McLaughlin. Convolutional neural network regression for short-axis left ventricle segmentation in cardiac cine MR sequences. *Medical Image Analysis*, 39:78–86, 2017.

[10] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. *CoRR*, abs/1606.06650, 2016.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[12] T.M. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997.

[13] Seyed Sadegh Mohseni Salehi, Deniz Erdogmus, and Ali Gholipour. Tversky loss function for image segmentation using 3d fully convolutional deep networks. *CoRR*, abs/1706.05721, 2017.

[14] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sébastien Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. *CoRR*, abs/1707.03237, 2017.

[15] Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *CoRR*, abs/1702.05659, 2017.

[16] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. *ArXiv e-prints*, June 2012.

[17] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *ArXiv e-prints*, mar 2016.

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[19] Florent Perronnin (Xerox (XRCE) Grenoble) Gabriela Csurka (Xerox Research Centre Europe), Diane Larlus. What is a good evaluation measure for semantic segmentation? In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2013.

[20] Einar Heiberg, Jane Sjögren, Martin Ugander, Marcus Carlsson, Henrik Engblom, and Håkan Arheden. Design and validation of segment - freely available software for cardiovascular image analysis. *BMC Medical Imaging*, 10, Jan 2010.