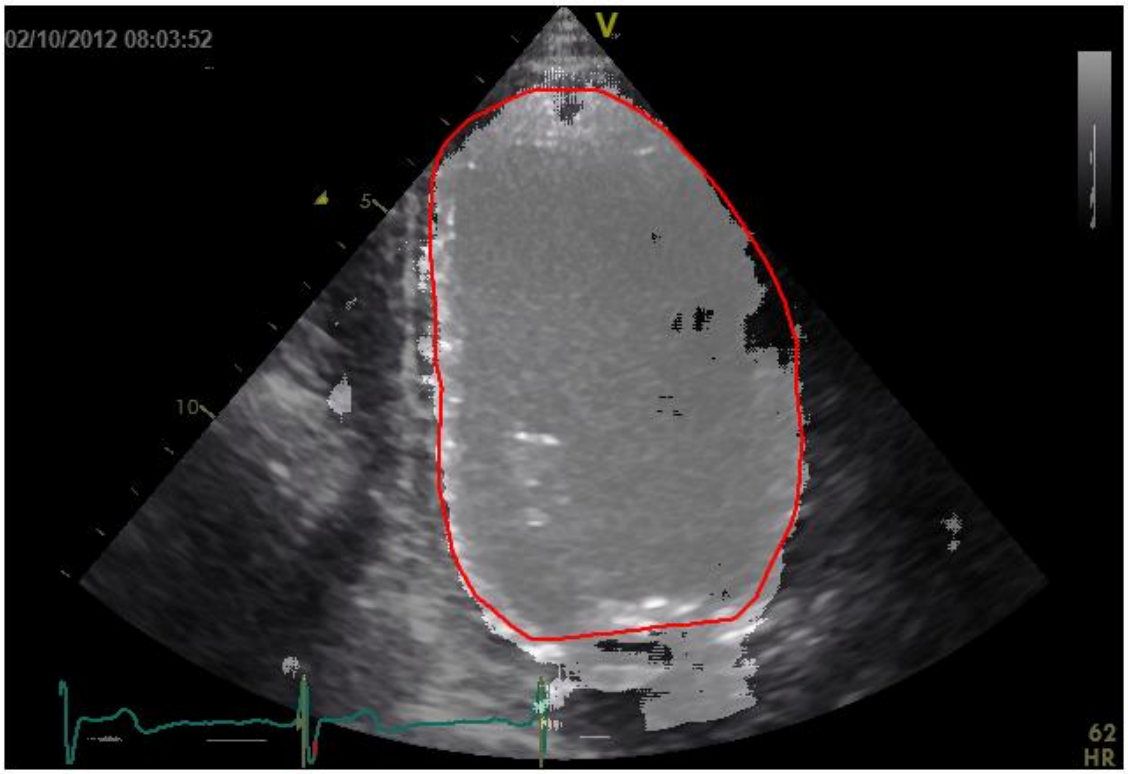
CHALMERS
UNIVERSITY OF TECHNOLOGY



Segmentation of the Left Ventricle of the Heart in 2D Ultrasound Images using Convolutional Neural Networks

Master's thesis in Biomedical Engineering

ELVIN ALCEVSKA

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Master's thesis 2016: EX093/2016

Segmentation of the Left Ventricle of the Heart in 2D Ultrasound Images using Convolutional Neural Networks

ELVIN ALCEVSKA

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden 2016

Segmentation of the Left Ventricle of the Heart in 2D Ultrasound Images using Convolutional Neural Networks
ELVIN ALCEVSKA

Supervisor: Jennifer Alvén, Department of Signals and Systems
Examiner: Fredrik Kahl, Department of Signals and Systems

Master's Thesis 2016: EX093/2016
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46-(0)31 772 1000

Cover: A 2D ultrasound image of the left ventricle and the left atrium of the heart. The red line represents the correct segmentation of the left ventricle given by an expert's annotation, while the grey areas in the image is the segmentation given by the convolutional neural network.

Typeset in LaTeX
Gothenburg, Sweden 2016

Segmentation of the Left Ventricle of the Heart in 2D Ultrasound Images
using Convolutional Neural Networks
ELVIN ALCEVSKA
Department of Signals and Systems
Chalmers University of Technology

Abstract

When an experienced cardiologist studies a 2D ultrasound image of the heart he or she can manually segment (i.e. outline) the correct border of the left ventricle and thereafter for example estimate the ventricle volume. The information provided from the segmentation (i.e. the delineation) is used in modern cardiovascular medicine for diagnosis, disease progression, schedule and choice of treatment etc. New guidelines on how to segment the left ventricle were published 2015, but the new guidelines are still not general knowledge among cardiologists. Therefore, an automatic segmentation method based on the new guidelines is needed. In this thesis, an automatic segmentation method following the new guidelines is implemented. The method includes pixel classification using a multilayer convolutional neural network, where supervised learning is used as the learning method. The network output is a probability map indicating the probability of each image pixel belonging to the left ventricle. Post-processing methods such as multi-atlas segmentation and graph cuts are used to obtain the final segmentation. The data consists of 2D ultrasound images with a 2-chamber view of the heart plus a corresponding manual delineation of the left ventricle. 30 images were used to train and validate the network, and 6 test images were used to evaluate the final segmentation framework. The segmentation results were evaluated by calculating the Dice coefficient for the test images, i.e. measuring the similarity between the automatically segmented area and the corresponding manual delineation. The average Dice coefficient for the test images was 0.82 when thresholding was used to obtain the final segmentation. The Dice coefficient increased to 0.92 when the network output was restricted to a region of interest defined by a multi-atlas approach and simple thresholding was replaced by graph cuts.

Keywords: image segmentation, left ventricle of the heart, 2D ultrasound image, machine learning, supervised learning, convolutional neural networks, multi-atlas segmentation, image registration, graph cuts, theano.

Acknowledgements

I would like to thank my supervisor Jennifer Alvén for her knowledge, patience and the great dedication and support I received throughout my thesis. I could not have wished for a better supervisor. Thanks to my examiner Fredrik Kahl, who introduced me to the world of neural networks. Thanks to Olof Enqvist and Måns Larsson, who shared their knowledge about neural networks. A great thanks goes to Dr. Odd Bech-Hanssen, who patiently took his time to provide me with excellent data, answered any questions I had about the human heart and who gave me the opportunity to take part in an ultrasound examination. Thanks to Rolf A. Heckemann and Leif Sandsjö for always being helpful with any concerns I had during my thesis. Also, special thanks to all my friends for their lovely encouragement.

Lastly, I would like to thank my dearest family, I would not have been where I am today if it was not for all your love and support.

Elvin Alcevska, Gothenburg, November 2016

Contents

List of Figures

List of Tables

1

Introduction

When an experienced cardiologist studies a 2D image of the heart he or she can manually segment the correct left ventricle area and thereafter estimate its volume. Echocardiography (i.e. ultrasound) is a non-invasive, low-cost, portable and time-efficient way to extract images of the heart [16]. However, the main issue with manual delineation of these ultrasound images, and existing automatic methods, based on these manual delineations, is that the left ventricle volume often is greatly underestimated.

There are specialists with the knowledge of delineating a different outline of the left ventricle, that generates a greater volume estimation, but this knowledge is not general. The new delineation guidelines published 2015 from the American Society of Echocardiography [9] include both the papillary muscles and trabeculae, which are types of tissues that appear on the inner surface of the ventricle, while the previous guidelines only include the papillary muscles. The new guidelines aim is to improve the volume estimation. The difference between using the different guidelines can be seen in Figure 1.1.

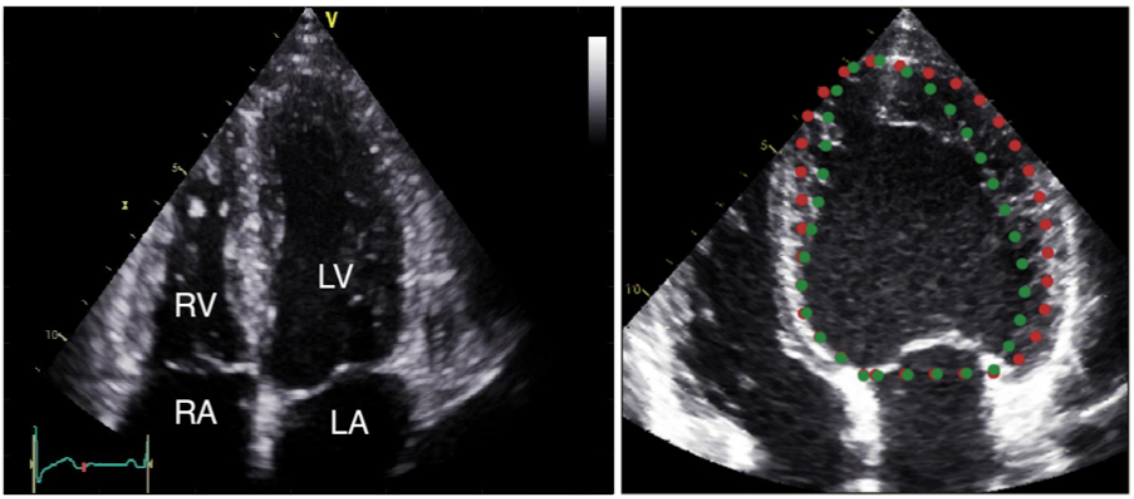


Figure 1.1: The left image shows a 4-chamber view ultrasound image where left ventricle (LV), right ventricle (RV), left atrium (LA) and right atrium (RA) are visible. A 2-chamber view only shows the LV and LA. The right image is a 4-chamber view image that shows how the area of the left ventricle differs when the new guidelines (red dots) and old guidelines (green dots) are used for manual delineation [15].

The information provided from the left ventricle is used in modern cardiovascular medicine for diagnosis, disease progression, choice and schedule of treatment [15]. Magnetic resonance imaging (MRI) is a technique which provides images that more easily provide the correct volume, but the method is expensive, time-consuming and can be an unpleasant experience for some patients [16]. This thesis will therefore focus on creating a program for automatically delineating ultrasound images based on manual labelings obtained with the new guideline.

1.1 Thesis aim

The aim of the thesis is to segment (i.e. delineate) the left ventricle in a 2-chamber view ultrasound image using convolutional neural networks (CNNs). Segmenting an image means that the image is divided in different categories, i.e. what area in the image belongs to the left ventricle and what does not. The proposed method could also be applied to 4-chamber view ultrasound images, but to fit the time frame for this thesis only 2-chamber view ultrasound images are used to train, validate and test the algorithm.

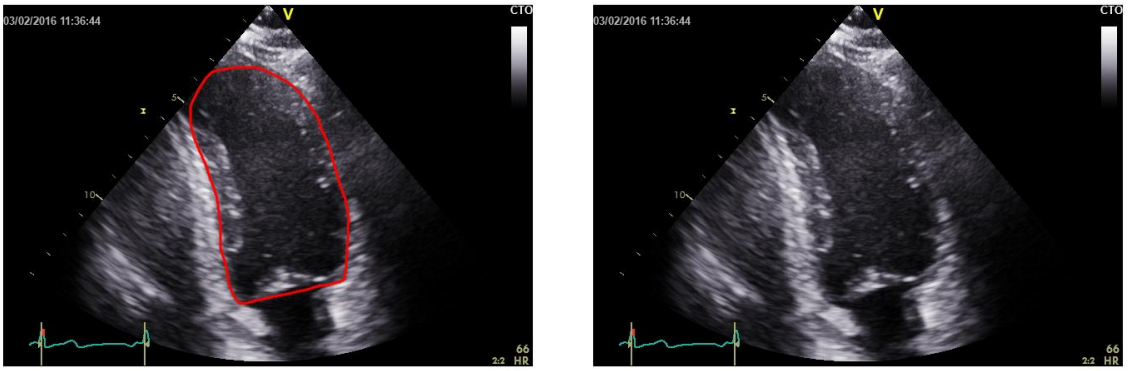
1.2 Proposed solution

The proposed solution for this project is to segment the left ventricle on a 2-chamber view ultrasound image with a machine learning algorithm called a convolutional neural network. These networks have shown promising results in segmenting medical images [8, 10]. The proposed method will segment the left ventricle based on the new guidelines where both papillary muscles and trabeculae are included in the segmentation. In particular, the convolutional neural network will be trained to classify each pixel in the image, either as a part of the left ventricle or background. Thus, each image pixel will be marked with a probability of belonging to the left ventricle according to what the convolutional neural network has learned. Thereafter, the final segmentation will be estimated by restricting the region of interest with a multi-atlas approach and finally applying graph cuts.

1.3 Data

The data set used for this thesis consists of 36 2D ultrasound images of the left ventricle and the left atrium, referred to as 2-chamber ultrasound images. These images were obtained from 36 different patients. Each ultrasound image is paired with a corresponding image with an expert's annotation. These delineations were marked manually with the software called EchoPAC version 113 and done according to the new guidelines. Figure 1.2 shows the same given 2-chamber view ultrasound image with annotation and without. The ultrasound images are color images (i.e. RGB). The size of each image is 434×636 pixels and the pixel size is approximately 3×3 mm. The image were taken during the time in the heart cycle when the area of the left ventricle was largest. The 36 ultrasound images are divided in three data

sets: training data, validation data and test data, which consisted of 25 images, 5 images and 6 images respectively. The training and validation images are used during training of the network and tuning the algorithm parameters, while the test images are used to evaluate the proposed algorithm.



(a) **(b)**

Figure 1.2: The figure show the data given with (a) an expert's annotation and (b) the same image without an annotaion.

1.4 Related work

One example of previous work that has been done concerning the segmentation of the heart in ultrasound images is a database-guided segmentation by Georgescu et al. [5]. The segmentation is computed by using a structure detection to differentiate between the region of interest and background, and shape inference. The given method segments according to the previous guidelines, i.e. only the papillary muscle is included. Further, the manual delineations used for learning were not flawless since the ultrasound images contained a lot of noise.

Emad et al. [2] tried to localize the left ventricle in magnetic resonance images with a convolutional neural network. The training set consisted of 33 patients where a 6-layer network was used to train on these images. The input size for the network was a patch of the size 20 x 20 pixels and a batch gradient descent was used for learning. This method was able to classify the left ventricle with a result of 98.66 % in accuracy, 83.91 % in sensitivity and 99.07 % in specificity.

Carneiro et al. [1] also used deep learning architectures and derivative-based search methods to obtain a segmentation of the left ventricle in ultrasound images. Artificial neural networks were used to create a rigid and a nonrigid classifier. The rigid classifier gives the probability that the region contains the left ventricle, while the nonrigid classifier provides information on where the boundary is possibly located. The automatic segmentation was presented to a cardiologist, which in general approved the result and there were even cases where the automatic segmentation was

more prefered than the manual.

Nascimento et al. [12] used a shape tracking method with multiple models to find the boundary of left ventricle in ultrasound images. The proposed method uses a multiple model data association tracker which is based on nonlinear filters structured in a tree structure.

2

Theory

2.1 Deep learning

Deep learning is an area of study within the field of machine learning where algorithmic network structures are used to learn a model based on the given data. There are different types of network structures used in deep learning and one of them is called convolutional neural networks, which is the main building block in this thesis.

2.1.1 Neural networks

The human nervous system consists of neurons, which are the human brain's basic computational units. The main building blocks of the neuron are the axon and the dendrites. Dendrites provide the input signals for the neuron, see Figure 2.1, while the output signals from the neuron is sent through the single axon. This axon will thereafter connect with new dendrites of other neurons with the help of synapses. The dendrite will only receive a signal from the axon when the axon is activated i.e. when the incoming signal is sufficiently large. The synapses used for communicating between the axon and dendrites impact the neuron differently with different strengths of the synapses [7].

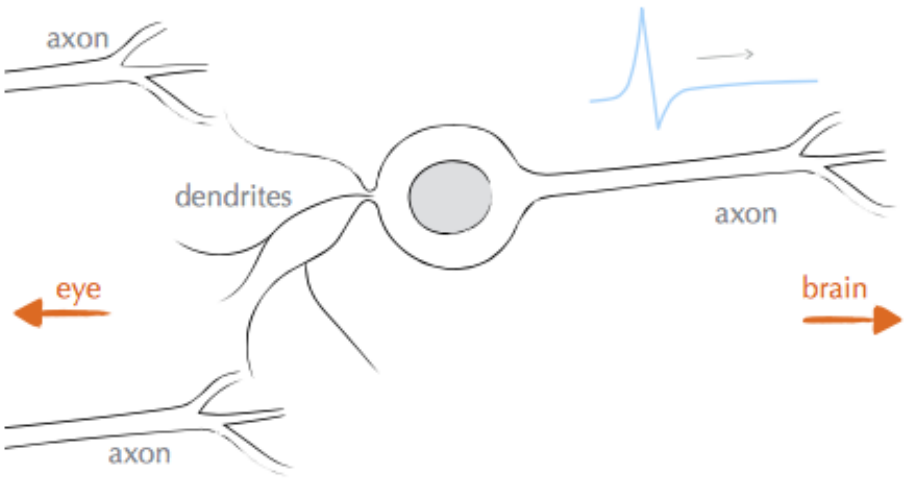


Figure 2.1: The figure shows a schematic neuron, where the dendrites receive the information for the neuron from activated axons from other neurons [3].

A simple artificial network neuron, commonly referred to as a perceptron, functions in a similar manner. In neural networks, the synapses strength can be seen as weights that are learnable, i.e. the weights are the network parameters. The cell body of the network neuron can contain a certain bias, seen as a threshold that decides whether the axon should be activated or not. The input signals from different dendrites will transmit to the cell body, where each input is multiplied with their respective weight and thereafter summarized. If this sum is above a certain threshold, then the neuron will activate the axon which will proceed to activate new neurons. The concept is visualized in Figure 2.2. The output for this type of neuron is either 1 when activated or 0 otherwise [7].

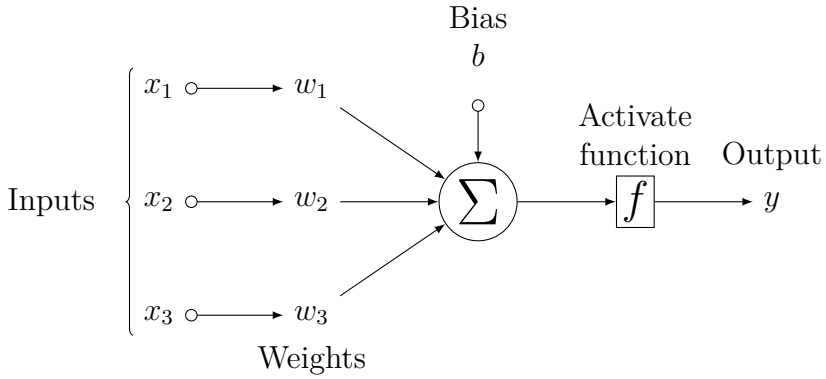


Figure 2.2: The neuron has a number of inputs x associated with a corresponding weight w and a threshold called bias b. The activation function is activated, which results in an output y, when the weighted sum is higher than the threshold b.

The neuron commonly used in artificial neural networks is called a sigmoid neuron and not the perceptron neuron that was earlier mentioned. The difference between these neurons is minor, where the principle of a sigmoid neuron is the same as for the perceptron, the difference is that the sigmoid neuron can output values ranging from 0 to 1, while the perceptron output equals either 0 or 1. If the weights for a network are adjusted somewhat then a corresponding change can be seen in the output of the network. In a network consisting of perceptrons, a small weight change can change the network's output from 0 to 1, which creates difficulties in the learning process [14].

2.1.2 Convolutional neural networks

An artificial neural network consists of three different types of layers; input layer, hidden layer and output layer, which can be seen in Figure 2.3. Each neuron from the input layer is connected to all or some of the neurons in the hidden layer, but the neurons within the same layer are not connected to each other. A network can have multiple hidden layers [14]. The type of artificial network used for this thesis is called a convolutional neural network, where the input neurons are the pixels in a small sample of an image, called a patch. The output has a value between 0 and 1 representing the probability of the patch belonging to the left ventricle. Applying

a convolutional neural network to a full image can be seen as filtering the image repeatedly at different scales.

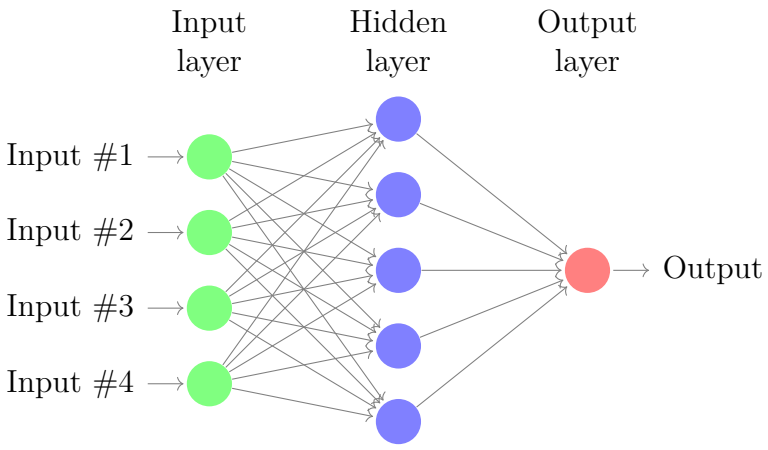


Figure 2.3: A basic artificial neural network with an input layer, a hidden layer and an output layer.

2.1.3 Loss function

A loss function is used to measure the performance of a network. The loss function represents the dissimilarity between the network output (i.e. the probability) and the true pixel labels, which was established with the help of the expert's annotation. A commonly used loss function for convolutional neural networks is called cross-entropy. The network's cross-entropy is computed when weights and biases has been set and is given by comparing the predicted value given from the convolutional neural network with the corresponding correct label. The cross-entropy for one patch is given by:

$$L = -tlog(p) - (1-t)log(1-p) \tag{2.1}$$

where L is the loss value, p is the predicted value from the network (i.e. the probability of belonging to the left ventricle) and t is the target (i.e. the true label, 0 for background and 1 for left ventricle). All the cross-entropies for the training data is summed up and divided with the size of the training data to give an average error. The loss function is close to 0 when the network predicts the correct value, i.e. the true label value, which is an indication that it does not need to train more and 1 when the output is incorrect [14].

2.1.4 Gradient descent

Gradient descent is what helps the neural network to learn weights and biases that make the network able to predict the correct output for a given input. The aim is to find a set of weights and biases that minimizes a loss function representing the dissimilarity between the output and the true pixel labels. The error surface can be seen as the shape of a multi-dimensional bowl (i.e. a nonlinear function), as seen in

Figure 2.4 with a view from above, where the smallest error is at the bottom of the bowl, which is the minimum. The goal is to reach the bottom of the bowl, which gives the smallest loss function value (i.e. when the error is the smallest), therefore the steps should be taken so that the minimum point for the bowl is approached. The error is given by calculating the difference between the expected value (i.e. the true labeling) and the predicted value according to Equation (2.1). Stochastic gradient descent only uses one patch at a time, where this single patch is used for computing the gradient and thus determines how the weights and biases are updated. The direction of the gradient is based on each patch and the step size of the gradient is determined by the learning rate (a large step size is given with a large learning rate), which is explained in Section 2.2.1.1. Learning can also be done with help of batches (subsets of patches), where the gradient direction is based on the average of multiple patches, which is called mini-batch gradient descent [3, 14]. When all the data has been used once to train on, then an epoch has finished. Training is usually run for several epochs.

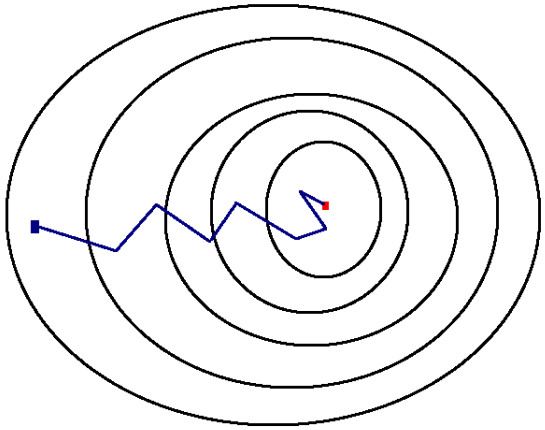


Figure 2.4: The black lines present level sets of the loss function. The gradient descent starts from a random point (blue square) representing the weights and biases initialization. The weights and biases are step-wise updated until the loss function reaches its minimum value, i.e. when the minimum point (red square) is reached.

2.1.5 Backpropagation

The backpropagation algorithm provides the information on how the performance of the convolutional neural network loss function changes depending on the changed weights and bias. Imagine that one could rewind a happened event and improve the result based by having the knowledge of what choices was made to obtain the previous outcome. Then, one can start working backwards from the output to see where the largest mistakes where made that affected the output the most and alter them. That is what the backpropagation algorithm does, it starts by studying the error in the output layer in the network. Thereafter, the given error is propagated backwards in the network, because the error given in the output is a result from the errors in the previous layer, where the error for that layer is given from the layer

before etc. The errors for each layers are used in order to estimate the gradient [14]. Each of the layer's error is a guideline in how to change the weights and biases for that specific layer to minimize the loss function. See reference [14] for details.

2.1.6 Network layers

There are different types of hidden layers used for a convolutional neural network. The layers used in this thesis are: 2D convolutional layers, max-pooling layers, Rectified Linear Unit (ReLU) layers, softmax layers and lastly fully connected layers [7]. An example of a convolutional neural network with these types of layers is presented in Figure 2.5.

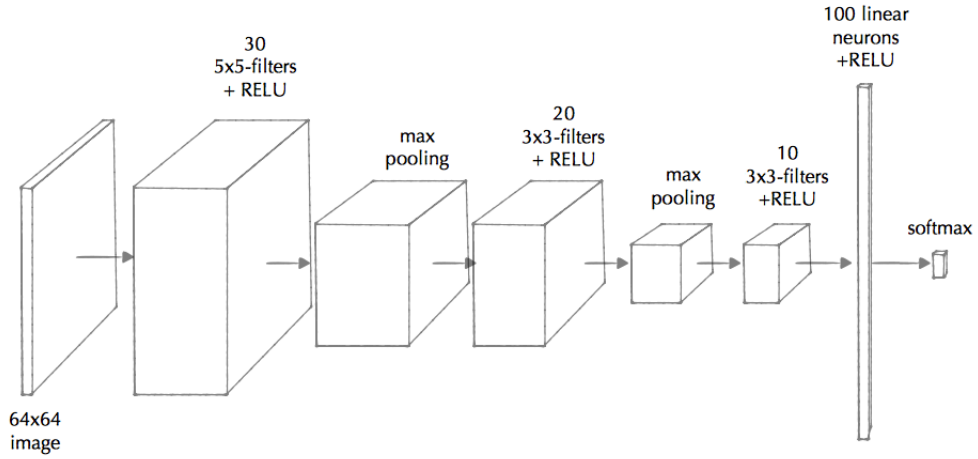


Figure 2.5: An example of a convolutional neural network with an input image patch of size 64×64 pixels, followed by a convolutional layer with 30 filters of size 5×5 pixels, which creates feature maps, and a ReLU as activation function. A max-pooling layer is then used to down-sample the feature maps. Next comes an additional convolutional layer with 20 filters of size 3×3 pixels with the a corresponding ReLU layer, a max pooling layer and another convolutional layer with 10 filters of size 3×3 pixels (also with ReLU activation). Lastly, a fully connected layer with 100 linear neurons (with ReLU activation) and a softmax converting the output to probabilities [3].

2.1.6.1 The convolutional layer

Each convolutional layer has k learnable filters. The filters are 2-dimensional of size $m \times m$ and can not be larger than the 2-dimensional input image patch $n \times n$ which is sent into the CNN. All these different filters extract different kinds of features of the input image patch and creates k feature maps of size n-m+1 [13]. Each neuron represents a filter and has the number of weights as the filter area. That is, if the filter size is 5×5 pixels, as in Figure 2.6, then each hidden neuron is associated with 25 input weights.

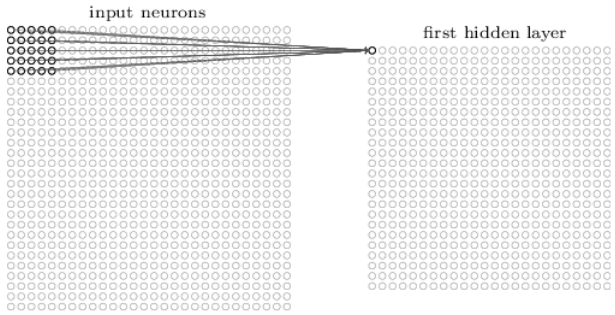
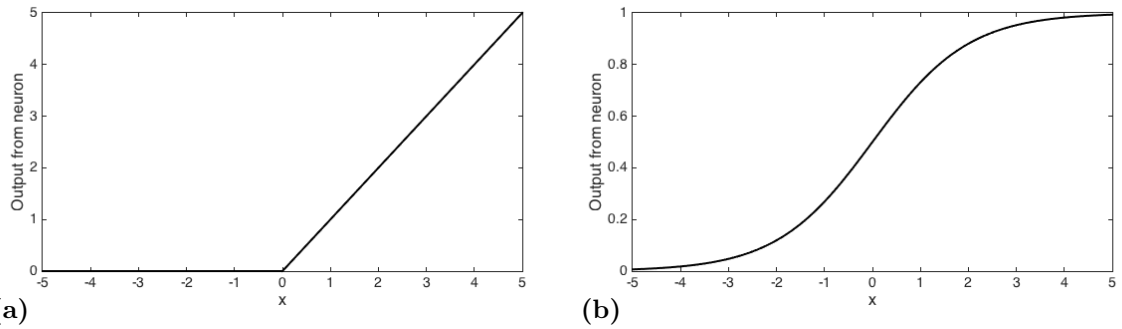


Figure 2.6: An example of a convolutional layer. The layer is created by sliding a filter of size 5×5 over the input neurons (i.e. the input patch) of size 28×28, filtering one neuron at a time. Each filtering of input neurons corresponds to a neuron in the next layer. The convolutional layer thus results in a feature map of size 24×24 [14].

2.1.6.2 Activation function

There are different activation functions, the ones used in this thesis are the rectified linear unit and the softmax function. The ReLU is the activation function for all convolutional layers, defined according to Figure 2.7a, while the softmax function is used for the output layer, defined in Figure 2.7b. These activation functions are commonly used in convolutional neural networks.



(a) **(b)**

Figure 2.7: The figure shows the different activation functions used for the network. a) The ReLU function, which was used for all convolutional layers as activation function. b) The softmax function, which was used as activation function for the output layer. X represents the input for a neuron.

2.1.6.3 Dense layer

The dense layer is a fully connected layer where all the neurons from the previous layer are connected to all the neurons in the next layer, this can be seen between the hidden layer and output layer in Figure 2.3. The dense layer can have a certain dropout rate to minimize the dependency between neurons which will be discussed further in Section 2.2.2.1.

2.1.6.4 Max pooling layer

Max pooling is used in convolutional neural networks to down-sample the feature maps that are created after the filtering in the 2D convolutional layer. It is done by dividing each map in regions of size $N \times N$ pixels, the most common matrix size when using max pooling is 2×2. Thereafter, the maximal value in each 2×2 matrix is chosen and thus, the spatial dimensions are reduced to half of the initial map size [14]. If the incoming feature map has a size of 4×4 pixels, after max pooling it is 2×2 pixels, which can be seen in the example in Figure 2.8. Max pooling will speed up the computational part for the convolutional neural network, because the number of weights that need to be optimize are decreased. A lower number of weights for a network means that the risk of overfitting is decreased, which is described more detailed in Section 2.2.2. The negative aspect of max pooling is that if the max pooling size is quite large compared to the patch size or the feature map, there is a risk of down-sampling too much and removing essential information for the CNN to be able to learn.

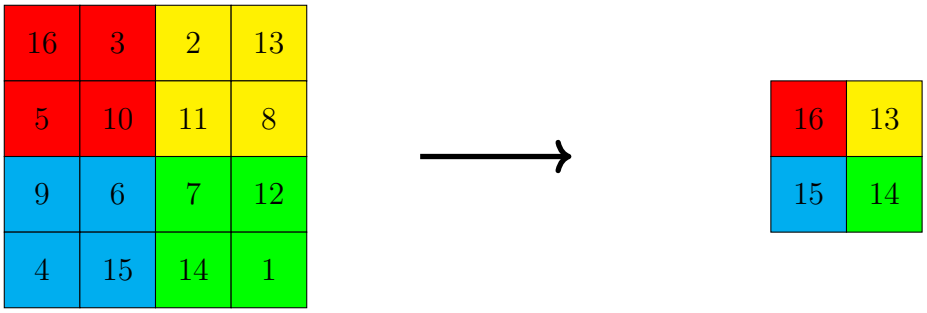


Figure 2.8: An example of max pooling. The left side shows a feature map before max pooling is applied where the regions are divided of size 2×2 and the result is presented on the right after max pooling has been used, where the maximum value of each region was taken.

2.2 Tuning a CNN

Tuning a convolutional neural network can be seen as building a house of cards, it can easily collapse. A minor change with the hyperparameters can improve the CNN or unfortunately make it worse. Training accuracy and validation accuracy is used to help one tune a CNN. The accuracy equals the percentage of the samples that were correctly classified for each data set. The training accuracy indicates whether the CNN is learning (and can be used to choose a good learning rate) and the validation accuracy provides information about whether the CNN is able to classify new unseen samples correctly, which it has not trained on. A common problem when training a CNN is overfitting, which can be seen as the validation accuracy being lower than the training accuracy. The CNN can be tuned by a dropout rate, increasing the number of epochs, batch size, patch size and momentum. The aim is to train a CNN with a high training accuracy that has a small difference between the training accuracy and validation accuracy.

2.2.1 Hyperparameters

2.2.1.1 Learning rate

The learning rate determines how well the training accuracy increases and how quickly the loss function is minimized during training of a CNN. Choosing a good learning rate leads to that the gradient descent, the learning algorithm, works properly and learns the samples presented to the CNN. The learning rate should not be chosen too small or too large. A learning rate set too large will decrease the loss function in the beginning, but then the loss function will stop decreasing or end up increasing after a certain amount of epochs, which means that it will stop learning. A too small learning rate will make the learning process time-consuming, because the weight updates in the CNN will be small, which results in a learning curve that decrease slowly [14]. The opposite happens for the training accuracy. A good learning rate will increase the accuracy while a too high learning rate will cause increasing training accuracy in the beginning and thereafter decreasing training accuracy, which can be seen in Figure 2.9.

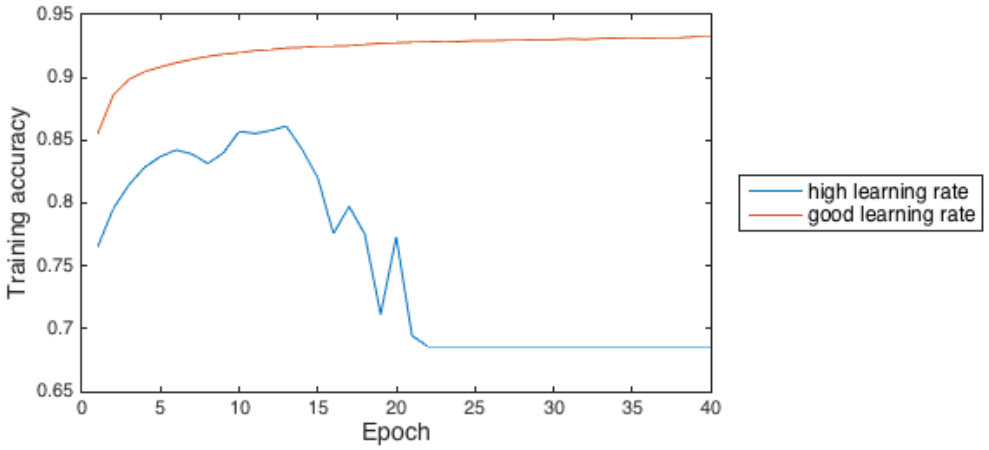


Figure 2.9: The figure shows how the learning rate can affect the training accuracy. The blue curve is the result when a too high learning rate is used and the red curve shows a good learning rate for the CNN.

The learning rate decides on how large each step is when updating the weights and biases. When the minimum of the loss function is starting to approach, the step size can be decreased by implementing a learning decay making sure that the optimal weights are not missed because of a too large step size. The learning decay can be implemented in different ways, the learning rate can be updated after each epoch, depending on the value of the training accuracy or decreased based on the validation accuracy.

2.2.1.2 Patch size

The size of the input patch is another factor. If the patch size is too small, the information will not be sufficient for the CNN to determine what to classify the

pixel as. If the patch is larger, more information will be given for the CNN to learn from. At the same time, more weights need to be regulated so the computational time is longer. If the size is too large, there is a possibility that the pixel within the same patch will have a low correlation with each other.

2.2.1.3 Batch size

The batch size is defined by how many patches are looked at before taking a step according to the gradient descent, which was mentioned in Section 2.1.4. A batch size greater than one can be beneficial while training a CNN with a large data set, because it converges quicker when the step taken is based on more patches than one. It is common to increase the learning rate when the batch size is large, which results in that optimized weights and biases are found quicker, because the step size of the gradient descent is larger when the learning rate is high.

2.2.1.4 Epochs

During one epoch, the batches in the training set will be randomized in order to make sure that the order of the samples do not contribute when the weights are optimized. The CNN should learn the samples and not their order. The training will go through the entire training set and optimize the weights. Thereafter, the validation set is classified with the help of the new weights and biases and the result of the validation depends on how well the CNN learned to recognize the samples it went through during the training phase. During validation the samples are not randomized, because no training is included, only classification with the current parameter settings.

2.2.1.5 Momentum

Implementing a momentum in the CNN can be seen as slowing down the speed when one knows that the goal (i.e. optimized parameters for the CNN) is starting to approach. Compared to the updates in Figure 2.4, the steps taken are more smooth and the updates are more visually straightforward to the minimum. This is because the previous step is considered when the next step is taken.

2.2.2 Overfitting

Convolutional neural networks have the ability to learn patterns between the input and the output with the help of few convolutional layers. What can happen though is that the CNN learns the data too well, which will lead to that the CNN rather recognize noise than actual patterns. This can be seen in Figure 2.10. This phenomena is called overfitting. Overfitting is a common problem when training a convolutional neural network, and which often occurs when the size of the training set is limited [17]. A sign of overfitting in the CNN is when the validation accuracy is lower than the training accuracy. The larger the gap is between the validation accuracy and the training accuracy the more the CNN is overfitting [14].

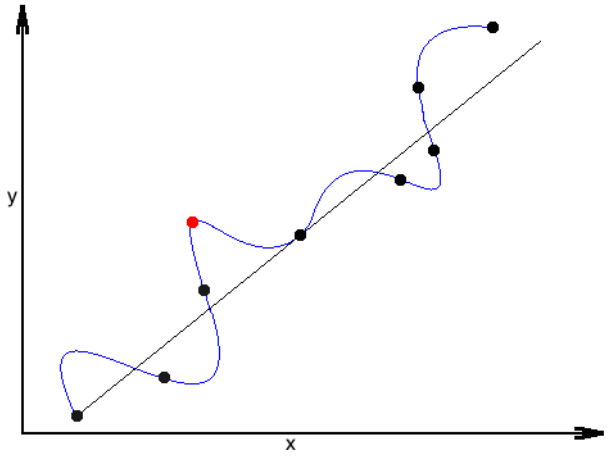


Figure 2.10: The model represented by the blue line in the figure has learnt from the data points so well that it fits to points which does not exist in the data, for example the red dot. The black line is an example where data is assumed to contain noise, the model has less overfitting.

2.2.2.1 Dropout

One way to decrease the amount of overfitting for a CNN is to apply a method called dropout. When using dropout, a hidden neuron in the CNN can be temporarily removed with a certain probability. The connections that goes to the neuron and from the neuron are also temporarily deactivated. By having a dropout rate of 50 % in a hidden layer means that the probability of each independent neuron being temporarily removed during the epoch is 50 %. Applying a dropout on a CNN can be seen as using only a fraction of the CNN to train on to avoid neurons depending too much on each other, an example of that can be seen in Figure 2.11. If all the neurons contain the same kind of information, a more robust relationship between the neurons is created. In the end, dropout can be seen as combining different convolutional neural networks with different information. All neurons that were deactivated are reset for each new epoch so other neurons can be deactivated.

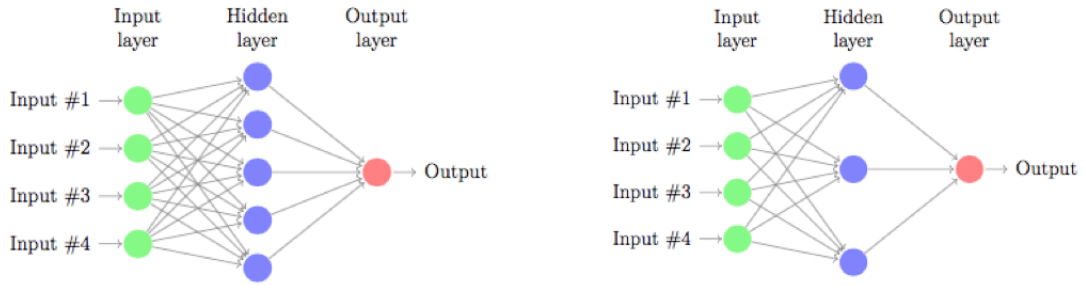


Figure 2.11: The left figure represents the complete network and the network on the right show the same network when dropout has been implemented, which has temporarily removed 2 neurons from the hidden layer.

2.3 Sliding window

A sliding window approach is used on a new full-size image in order to compute the probability being inside the left ventricle for each image pixel. The sliding window applies the learnt convolutional neural network on the image pixel by pixel and thus gives each pixel a probability from 0 to 1 of being a part of the segmentation.

3
Methods

The full framework consists of 3 parts: i) pre-processing the data, ii) pixel classification and iii) image segmentation, all parts are explained in detail in this chapter. The method used to classify whether the pixel is a part of the left ventricle is called convolutional neural networks, explained in detail in Chapter 2. The data set used for training a CNN is divided to two different types of data sets: training images and validation images. The CNN trains on the training images by optimizing the weight and biases such that the expected output for that given input is the predicted output, this method is called supervised learning. When the training is complete the validation images, which the CNN has not yet trained on, are introduced. The validation images are used to tune CNN parameters, for optimal CNN performance and in order to avoid overfitting, and also utilized when choosing post-processing parameters. The CNN output is processed into a final labelling by adding different post-processing methods such as multi-atlas segmentation and graph cuts. The segmentation on the test images is evaluated with the Dice coefficient.

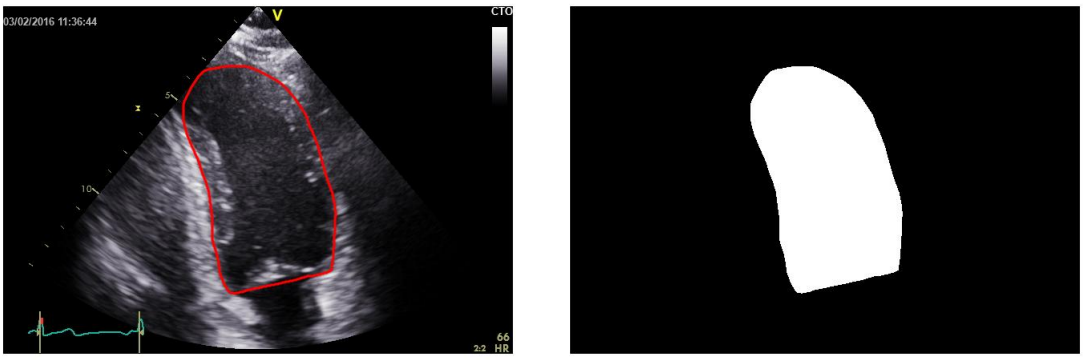
3.1 Pre-processing the data

The images used for training the CNN are 2-chamber view 2D ultrasound images, where the size of the image is 434 x 636, RGB, thought the area of interest in the image is in grayscale. Thus, the RGB images are converted to grayscale images of size 434 x 636.

3.1.1 Creating the gold standard

For each patient, an ultrasound image including an expert's annotations were used to create a gold standard, which represents the true labelling that is used for supervised learning and for evaluation of the algorithm. The expert's annotation is in the form of a line, where the area within the line is classified as the heart's left ventricle. The gold standard should be an image with only two types of pixel values, where every pixel classified as the left ventricle is set to 1 and the remaining pixels of the image (i.e. the background pixels) are set to 0. The ultrasound image is in RGB format where the information obtained from the ultrasound examination is grayscale and the annotation is in color. In order to extract the gold standard, all information in grayscale is removed from the image such that only pixels corresponding to the colored annotation remain. These pixels are connected such that a boundary is created and the pixels within the boundary are set as 1 and outside the boundary

are set as 0. The given ultrasound image with an expert's annotation and the resulting gold standard can be seen in Figure 3.1. The gold standard obtained was visually inspected and approved by the same expert that marked the annotations.



(a) (b)

Figure 3.1: An ultrasound image of (a) the heart's left ventricle with an expert's annotation, i.e. the red line, and (b) the corresponding gold standard where the left ventricle is marked as white (pixel value = 1) and background as black (pixel value = 0).

3.1.2 Mapping the gold standard to ultrasound images

Due to the software used for annotation, the ultrasound images with and without annotation could differ in size and scaling. In those cases, an affine transformation was used to align the ultrasound image with delineation and the corresponding without. This had to be done in order to create a gold standard with the correct size compared to the input ultrasound image. Let the image with annotation be refered to as the source image and the image without annotation be the target image. The coordinates of two different easily detected image points were taken from the source image x and the corresponding points were annotated in the target image y. These points are used to estimate an affine matrix A and translation vector t which are used to warp the gold standard, the expected output, into the coordinate system of the target image (i.e. image without annotation, thus the CNN input).

3.2 Classification using CNNs

The purpose of the learning algorithm is to be able to predict (i.e. classify) which pixels of the 2D ultrasound image belong to the heart's left ventricle. A CNN can learn in different ways, the method used in this thesis is supervised learning where the desired output, the gold standard, is given for each input. In other words, the desired output is the gold standard and the input is the corresponding ultrasound image without any annotation.

The purpose of the convolutional neural network is to assign each pixel a probability indicating whether they belong to the left ventricle or not. This is done by pixel-wise training using a patch surrounding the pixel as input and the pixel label given by the gold standard as desired output. When the training is completed and one would like to segment an unseen image, a sliding window approach is used. The sliding window parses each pixel in the image and classifies them according to what the CNN has learned. The output of the sliding window is a probability map with an interval ranging from 0 to 1, the value represents the probability that the pixel belongs to the left ventricle.

3.2.1 Pre-processing

Each image is normalized by subtracting the mean value from each pixel and then dividing by the standard deviation. The mean equals the mean pixel intensity value of all pixels in the image and the standard deviation equals the standard deviation of all pixels intensity value in the image.

In order to create training data for the CNN, coordinates for the ventricle and background pixels are extracted from the image. This is done by sampling each image pixel-wise, where each pixel labelled as background is sampled every third pixel (same for both dimensions) and the left ventricle is sampled every second pixel (same for both dimensions).

The patch is created by taking the surrounding pixels of a labeled centered pixel, where the label is given by the gold standard. The centered pixel determines where the patch is taken from, the left ventricle or the background (see Figure 3.2). This patch will provide the CNN with the necessary information for being able to classify pixels as left ventricle or background. The reason why all pixels in the image are not used for training is partly because the CNN trains faster with less training data and partly because a lot of the information in the patch would overlap with other patches with a more dense sampling of the image. The sampling method used in this thesis made the patches overlap each other somewhat. The background in the image is larger than the area of interest, which gives an unbalanced training set with 292000 patches labelled as left ventricle and 535000 patches as background.

3.2.2 Training the network

Different factors are looked for when a CNN is trained: 1) is it learning new data, 2) is it able to correctly classify the new data presented and 3) is it able to generalize to unseen data. These elements can be seen by studying the training accuracy and validation accuracy, which are given by how many patches of the total data are correctly classified, with an interval from 0 to 1, where 1 means that all patches are classified correctly.

The training of the CNN can be stopped by using early stopping, where the training is stopped either when the validation accuracy is not increasing or when the train-

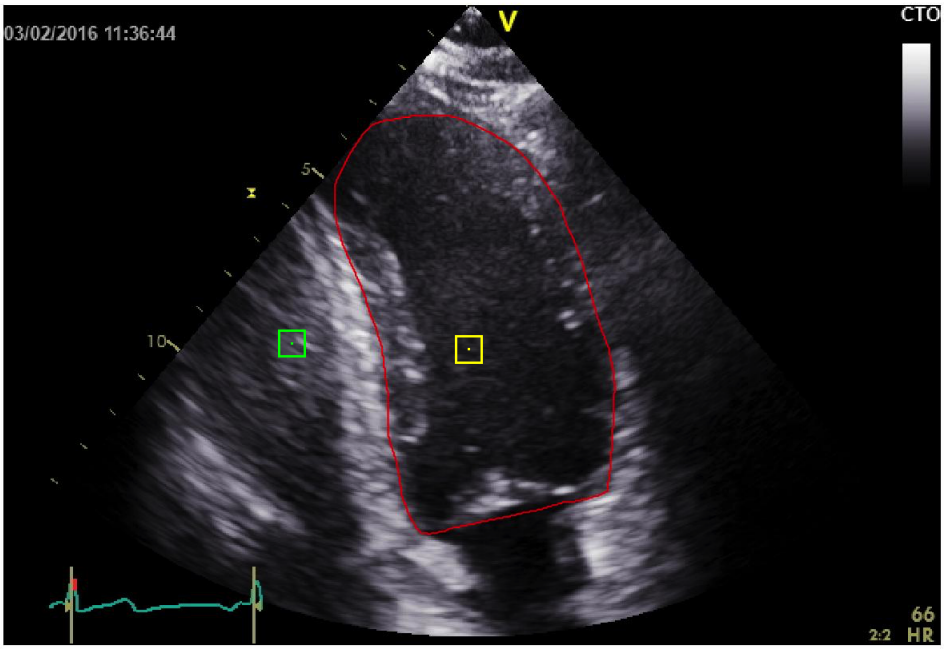


Figure 3.2: The patches are extracted from the input ultrasound image. The sampled pixel is the center of the patch. The red line represents the gold standard, which is not included in the sampled grayscale ultrasound image, and all patches extracted within and on the line are labeled as left ventricle, i.e. the yellow square. Every patch outside the red line is labeled as background, i.e. the green square.

ing accuracy has remained the same for a number of epochs, which means that the CNN is not learning anything new. This method was not used during online training, because the validation accuracy normally fluctuated between epochs, i.e. the validation accuracy did not increase steadily for each epoch. Therefore, the CNN was saved after each epoch, instead of applying early stopping. The final CNN was chosen such that it had the highest possible validation accuracy and also such that the overfitting was not too large. A measure of the overfitting was given by taking the difference between the training accuracy and the validation accuracy.

The convolutional neural network was implemented with Python 2.7, where the publicly available Theano library was used for the different mathematical calculations involving the convolutional neural network. The structure of the CNN itself was written with help of the Lasagne library, also publicly available. Input patches with the size of 33×33 pixels was used with the following CNN structure, which was inspired by Emad et al. [2] where a similar CNN structure was used.

Convolutional neural network structure:

- Input: Image patch of size 33×33 pixels.
- Layer 1: Convolutional layer with 6 filters of size 6×6 (gives a feature map of size $28 \times 28 \times 6$).
- Layer 2: ReLU.
- Layer 3: Max-pooling layer of size 2×2 (gives a feature map of size $14 \times 14 \times 6$).
- Layer 4: Convolutional layer with 12 filters of size 3×3 (gives a feature map of size $12 \times 12 \times 12$).
- Layer 5: ReLU.
- Layer 6: Max-pooling layer of size 2×2 (gives a feature map of size $6 \times 6 \times 12$).
- Layer 7: Fully connected layer with 10 neurons with a dropout rate of 0.3.
- Layer 8: ReLU.
- Layer 9: Softmax output with 2 neurons.
- Output: The softmax consists of two neurons but the probability map is given from the neuron, where the value represents the probability of the central pixel belonging to the left ventricle, with values ranging from 0 to 1.

3.3 Post-processing

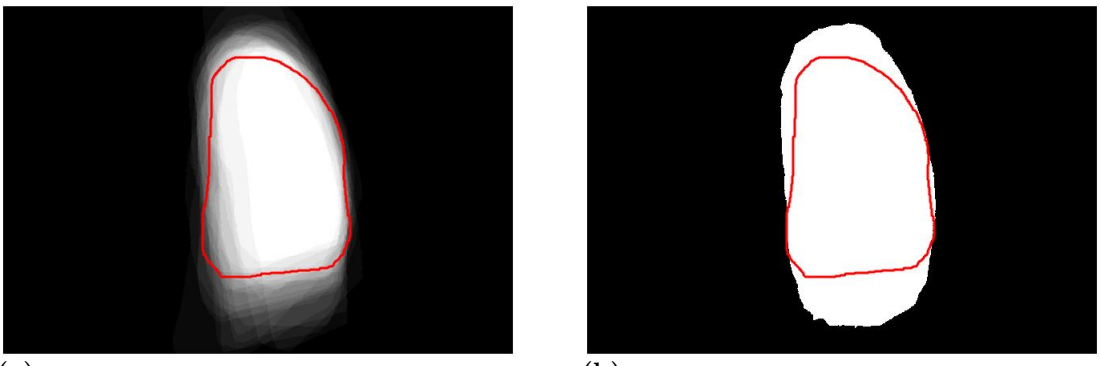
3.3.1 Region of interest

Multi-atlas segmentation is used as a part of the post-processing to restrict the region of interest for the sliding window computation. An atlas is an image with a corresponding gold standard. The region of interest is obtained by aligning all the training images to one test image by using feature-based registration, which is explained below. The registration results are used to warp the gold standard of the training images to a voting map, where each pixel gets votes indicating whether it is a part of the left ventricle or not. The region of interest is chosen by warping a certain number of gold standards chosen with help of the amount of inliers (see below for an explanation). The voting map is normalized and thereafter, a certain

threshold is applied to determine whether the pixel should be included in the region of interest.

There are different methods used for multi-atlas segmentation, the method used in this thesis is called feature-based registration. Feature-based registration relies on finding similar features, such as points, contours and lines, between the target image (i.e. the validation image or the test image) and the source image (i.e. the training image). These features are used to find matches between the images. The information obtained after the feature matching are the points that correspond to each other in each image. These points are used to estimate an affine transformation, where multiple sets of points are matched with an iterative method called random sample consensus (RANSAC), between the target and the source with a given threshold. The threshold equals the maximal allowed distance between two correctly mapped points. A correctly mapped point is called an inlier, an erroneous correspondence is called an outlier.

The implementation of the feature extraction, description and matching was done according to Svärm et al. [18] where a Lowe ratio [11] of 0.95 was used for matching features. A threshold of 80 pixels was used for sorting out outliers in a RANSAC [4] algorithm run with 500000 iterations. The multi-atlas voting map was then created by only using the top 5, 10, 15, 20, 25 (respectively) warped gold standards from the training set. The top gold standards were ranked according to the amount of inliers, i.e. only the gold standards with the most inliers were used for creating a voting map. The pixels in the voting map were normalized to an interval from 0 to 1, where all pixels with a probability higher than 0.2 were kept to create a final region of interest for both the validation and test images. An example of such region if interest can be seen in Figure 3.3.



(a) **(b)**

Figure 3.3: The figure shows how the region of interest was created, where a) is the normalized voting map of 25 warped gold standards from the training data, b) a threshold of 0.2 is used where every pixel greater than 0.2 is a part of the final region of interest. The final region of interest is marked as white and the background is marked as black. The red line in all images is the correct gold standard for the image.

Multi-atlas was also utilized in the network training to see if the network learns differently if the network only trains on the region of interest defined by the multi-atlas. This multi-atlas region of interest was created by using all the training images except for the one currently training on. The resulting image-specific region of interest was used as a guideline that showed what pixels the network should train on for that specific training image. The same sample density was used for the region of interest according to the multi-atlas, which was mentioned earlier in Section 3.2.1. This method also creates an unbalanced training set with 292000 patches labelled as left ventricle and 133000 patches labelled as background.

3.3.2 Segmentation

Now, a network is trained to classify pixels as ventricle or background. The network has been applied to a region of interest (computed with multi-atlas techniques) of a new ultrasound image with a sliding window approach. What is left is to derive a segmentation from the resulting probability map. An example of a probability map from the network without any post-processing can be seen in Figure 3.4. The figure shows a noisy probability map where some larger areas are misclassified, such as parts within the gold standard and the area below the gold standard.

The final segmentation can be obtained by using so called graph cuts, which is a method used to find the global optimal segmentation based on both the probability map $P(i)$ and a smoothness term. Graph cuts finds the minimum for a loss function based on the Potts model [19] where neighboring pixels are punished if the labels are different. If two neighboring pixels x_i and x_j are labelled equally, then the smoothness cost, i.e. $RW \cdot x_i(1 - x_j)$, is set to zero and RW otherwise, which is an regularization weight (RW) with a positive value. The data cost of pixel i (i.e. the data term) has a value of $0.5 - P(i)$ if $x_i = 1$ and 0 otherwise. This setting benefits labelling of pixels with probability between $[0.5, 1]$ as left ventricle and pixels with a probability between $[0, 0.5]$ as background. To sum up, the resulting segmentation, x^* is found by finding the solution to the following optimization problem:

$$\boldsymbol{x}^* = _{\boldsymbol{x}\in\{0,1\}^n} \left(\sum_{i=1}^{n} x_i(\frac{1}{2} - P(i)) + RW \sum_{i=1}^{n} \sum_{j\in N(i)} x_i(1 - x_j) \right), \tag{3.1}$$

where an 8-connected neighborhood N was used for all the probability maps [6].

The reason why a simple threshold is not used as post-processing step, where all pixels of value $[0.5, 1]$ are classified as left ventricle can be seen in Figure 3.4b, since $RW = 0$ equals thresholding the probability map at 0.5. This can result in holes within the segmentation, which should be classified as part of the left ventricle, or a disjoint segmentation. Using $RW = 5$ or $RW = 10$ solves this problem. When using a sufficiently large RW graph cuts can successfully remove misclassified areas, which can be seen in Figure 3.4c and 3.4d.

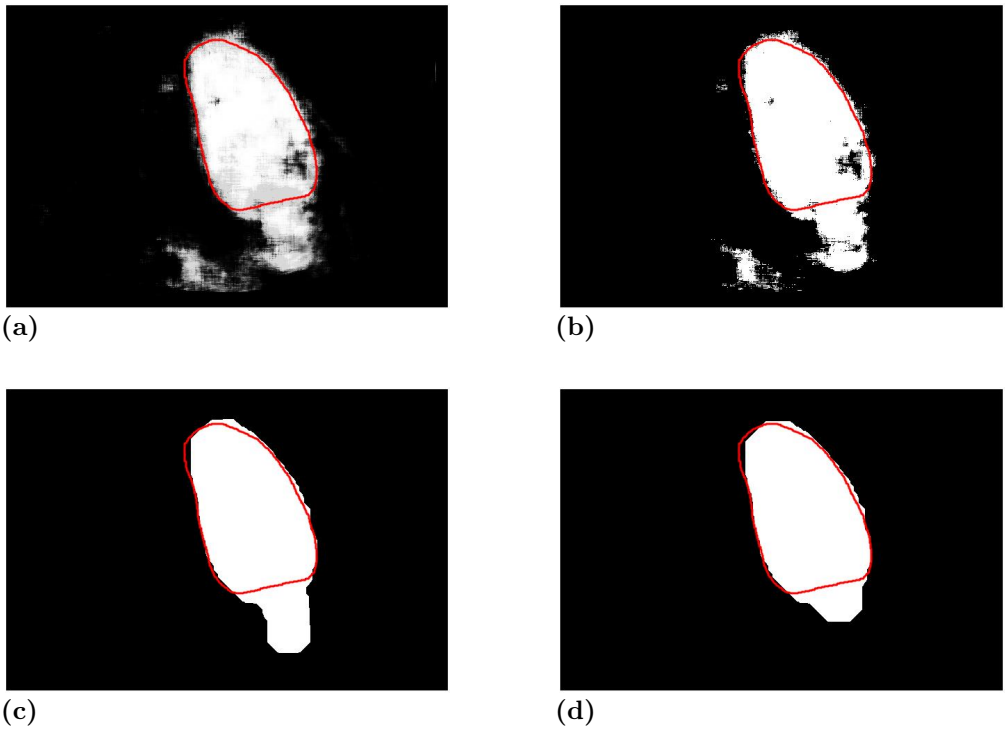


Figure 3.4: (a) The probability map, with a interval between 0-1, given from the network. (b-d) Graph cuts is used on the probability map with a regularization weight of 0, 5 and 10. The result is a binary image, where pixels classified as left ventricle are set to 1 and background to 0. The red line in all images is the gold standard.

3.3.3 Evaluation of network

The full segmentation framework, including pre-processing, classification and post-processing, is evaluated by using the Dice coefficient [20]. The Dice coefficient is evaluated on the test data, where set A is the predicted segmentation and set B is the corresponding gold standard, given by the following formula:

$$\text{Dice} = \frac{2\,|A \cap B|}{|A| + |B|}, \tag{3.2}$$

where the Dice coefficient has a value between 0 and 1. A Dice coefficient equal to 1 means that the predicted segmentation and the gold standard is perfectly overlapping, while 0 means no overlap. A and B are images with binary pixel values. The Dice coefficient of the test data is evaluated for three different types of post-processing settings. The first method obtains the predicted segmentation from the probability map evaluated on the whole test image by using a simple threshold. The second method obtains the segmentation by thresholding the probability map restricted to the region of interest given by a multi-atlas, the third method establishes the segmentation with help of graph cuts with the probability map restricted to the multi-atlas region of interest.

4
Experimental results

In the first subsection, different CNN parameters are evaluated and chosen with respect to the training and validation accuracy. In the second subsection, parameters associated with post-processing are evaluated and chosen with respect to the Dice coefficient of the validation data. In the third subsection, the final results are evaluated for the test images with the Dice coefficient. In the final section, CNN training based on either the full image or the region of interest defined by the multi-atlas is compared.

4.1 Tuning CNN parameters

The CNN parameters that were chosen for the final algorithm were the ones that had the highest validation accuracy with the least amount of overfitting. That is, the training was run for a maximal amount of epochs where the weights and biases where saved for every epoch and thereafter the optimal CNN was chosen among all saved CNNs.

The learning rate and the different thresholds were chosen based on the training accuracy, a high learning rate would cause the CNN to learn in the beginning but after a few epochs decrease in training accuracy. A very small learning rate is time-consuming because the training accuracy increases very slowly for each epoch. The learning decay is set by trying to maximize the validation accuracy so that an optimal CNN is not missed, which can happen when the learning rate is set to high. The maximal number of epochs that the training could run for were chosen based on the training accuracy, because we want to obtain the highest possible training accuracy during training. The training accuracy also provided information on what patch size was most suitable for the CNN. The patches should be of a size such that they provide enough information for the CNN to learn.

Different batch sizes were evaluated for the given CNN structure presented in the previous chapter. The training accuracy, validation accuracy and the amount of overfitting (defined as the difference between training and validation accuracy) for different batch size settings are presented in Table 4.1. Table 4.2 presents the hyperparameters setting for each batch size setting. Two different learning decay settings were used, a higher value when training accuracy exceeded a lower threshold (0.90 for batch size = 1 and 0.85 for batch size > 1) and a lower value when training accuracy exceeded a higher threshold (0.95 for batch size = 1 and 0.90 for batch size

> 1). All different settings had an input patch size of 33×33 pixels. Two different numbers of maximal amount of epochs were used during training, the maximal amount of epochs = 100 with a batch size = 1 and the maximal amount of epochs = 60 when batch size > 1.

Table 4.1: Training accuracy, validation accuracy and overfitting (defined as the accuracy gap) for different batch size settings.

Batch size	1	8	16	64
Training accuracy	0.9498	0.9424	0.9479	0.9442
Validation accuracy	0.9282	0.9354	0.9287	0.9342
Overfitting	0.0216	0.0070	0.0192	0.0100

Table 4.2: The hyperparameters for different batch sizes, where LR = learning rate and LD = learning decay, where LD1 is used when training accuracy has reached the lower threshold and LD2 is used when the training accuracy has reached the higher threshold.

Hyperparameter	LR	LD1	LD2	Epoch
Batch size = 1	0.0005	0.00005	0.00001	59
Batch size = 8	0.0001	0.00005	0.00001	56
Batch size = 16	0.0010	0.00010	0.00005	35
Batch size = 64	0.0050	0.00050	0.00010	3

We can see that the batch size that gave the least overfitting, batch size = 8, also gave the highest validation accuracy of 0.9354. Based on this result a CNN trained with a batch size of 8 is used. The other CNN parameters are given by Table 4.2.

The highest value that the training accuracy was able to reach was 0.95. A higher training accuracy could be possible (in theory, a sufficiently large CNN should be able to learn to recognize all the training data, of course with the risk of rather learning noise than actual patterns). Though, reaching a higher training accuracy was not possible, even when the learning rate was very small. Figure 4.1 shows the training accuracy and the validation accuracy when a local minimum is assumed to be found around epoch = 54, where both accuracies decrease drastically.

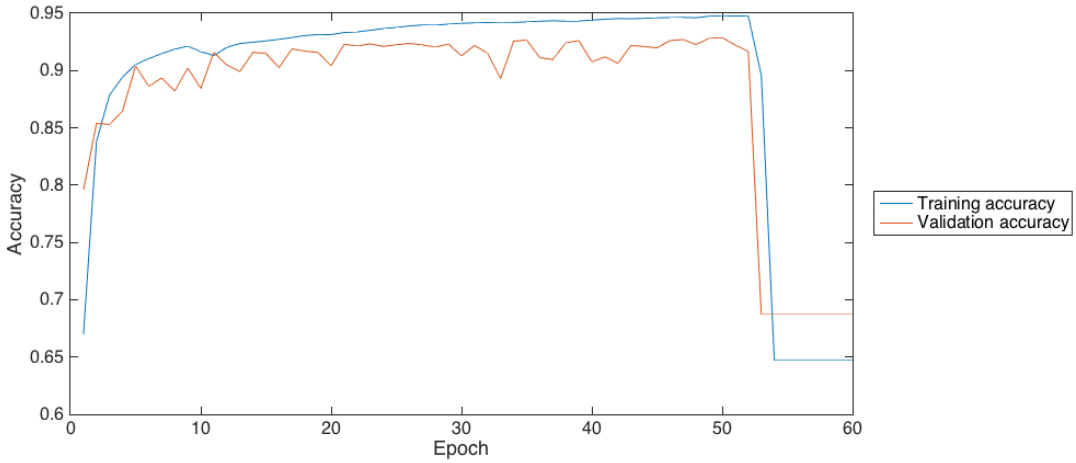


Figure 4.1: The figure shows how the training accuracy and validation accuracy behave during training when a local minimum is assumed to be found around epoch = 54, which result to a drastic decrease in both accuracies.

4.1.1 Dropout

Dropout was used in the fully connected layer, which down-samples the CNN structure, see Figure 2.11. As mentioned before, dropout is used to minimize the gap between the training accuracy and validation accuracy, also called overfitting. The different dropout rates 0, 0.1, 0.2, 0.3, 0.4, 0.5 were tested to see what kind of effect it had on the training accuracy and the validation accuracy.

Figure 4.2 shows how the training accuracy and validation accuracy (for the optimal CNN) changes when different dropout rates were used during training. The highest validation accuracy is given when a dropout rate of 0.3 is chosen. Keep in mind that each optimal CNN is chosen at the epoch where the validation accuracy is as high as possible with the least amount of overfitting (see description above). Even though the overfitting is larger when the dropout rate is 0.3 compared to 0.1, we chose the dropout setting which showed the highest validation accuracy. The other hyperparameters were fixed when tuning the dropout rate, and can be found in Table 4.2 when batch size = 1, but the maximal amount of epochs was changed from 60 to 40.

4.2 Post-processing

For the post-processing a parameter study is done in order to investigate how many atlases that should be used when computing the region of interest and the regularization weight for the graph cuts that should be used. The parameter setting that gave the highest possible Dice coefficient for the validation images was chosen.

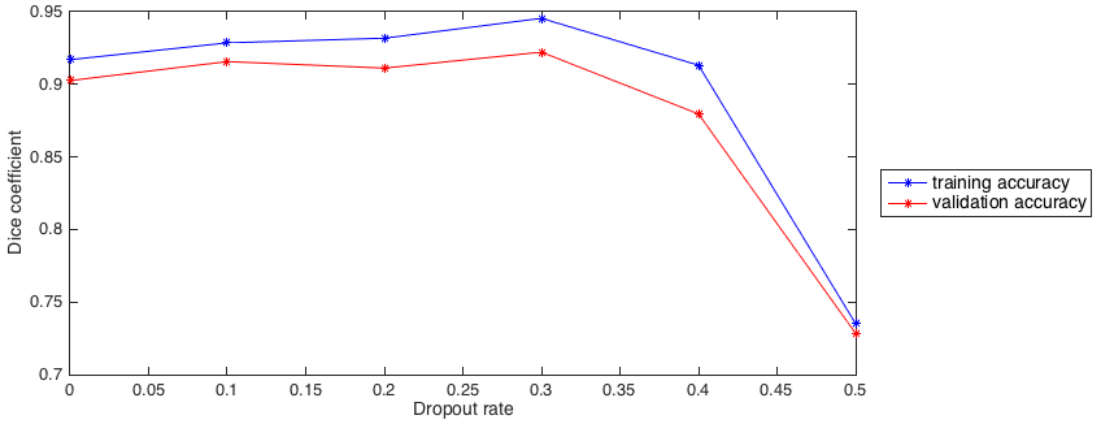


Figure 4.2: The plot displays the training accuracy and the validation accuracy for the optimal CNNs with different dropout rates (marked as an asterisk in the graph). The highest validation accuracy is obtained when a dropout rate of 0.3 is used.

4.2.1 Regularization weight

A parameter study is made to decide the regularization weight, when using the graph cuts, that gives the highest Dice mean coefficient. This is done by using different regularization weights for a fixed number of atlases, which can be seen in Figure 4.3. This figure shows how the Dice coefficient for the validation images changes depending on what regularization weight is used. The regularization weights that was tested were 0 (simple thresholding at 0.5), 0.01, 0.1, 1, 5, 10, 15, 20. The figure shows that the highest Dice coefficient for the validation images is obtained when a regularization weight of 15 is used. The Dice coefficient decreases when set higher and this is because some parts of the segmentation is over-regularized when using graph cuts with a regularization weight of 20 or higher. This causes less overlapping between the predicted segmentation and the gold standard. This can often be the result when a too high regularization weight is used to find the global segmentation in an image.

4.2.2 Number of atlas for computing the region of interest

The same evaluation method used for investigating regularization weights was used to determine how many atlases should be used when computing the region of interest in order to obtain the highest possible mean Dice coefficient. This can be seen in Figure 4.4. The atlases with the most number of inliers was chosen to create a multi-atlas, explained in more detail in Section 3.3.1. Each figure has a fixed regularization weight. Overall the Dice coefficient does not change that much depending on how many atlases that were used to define the region of interest, but we can see that the largest Dice coefficient is given when the top 5 atlases are used. This is the number of atlases that are used for evaluating the full framework on the test images.

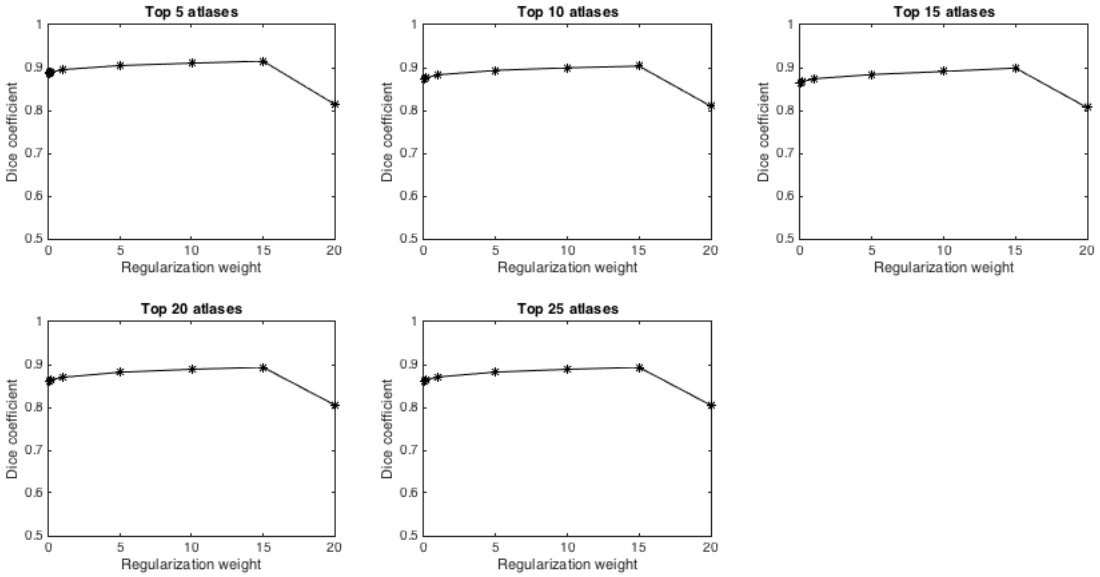


Figure 4.3: The figures show how the average Dice coefficient on the validation data changes depending on the regularization weight, when the number of atlases is fixed. The evaluated regularization weights are marked with an asterisk in the plot. The highest Dice coefficient was obtained when RW = 15.

4.3 Evaluation of the full framework

The Dice coefficient was used to evaluate the CNN on the full segmentation framework on the test images. When thresholding the probability map on the full image (without a region of interest) at 0.5, the Dice coefficient was on average equal to 0.82, which can be seen in Table 4.3. Take notice from the table that the largest Dice coefficient is equal to 0.92 and the lowest is equal to 0.72, these test images with their corresponding probability map may be studied in Figure 4.5 and 4.6. The average Dice coefficient increases to 0.87, according to Table 4.4, when a multi-atlas approach is used to define a region of interest to apply the sliding window on. The same threshold as before was used. The Dice coefficient improved greatly for all test images, where the biggest improvement is for test image 4, but it still has the overall lowest Dice coefficient. Table 4.5 has an average Dice coefficient of 0.92, this is when the thresholding is replaced by using the graph cuts instead, which shows a great improvement for the different segmentations. We can see that this type of post-processing improves the segmentation for all test images.

Table 4.3: The Dice coefficient for the test images without using a region of interest and with a threshold of 0.5 for the probability map.

Test image	1	2	3	4	5	6	Average
Dice coefficient	0.7853	0.8288	0.9196	0.7163	0.8887	0.7983	0.8228

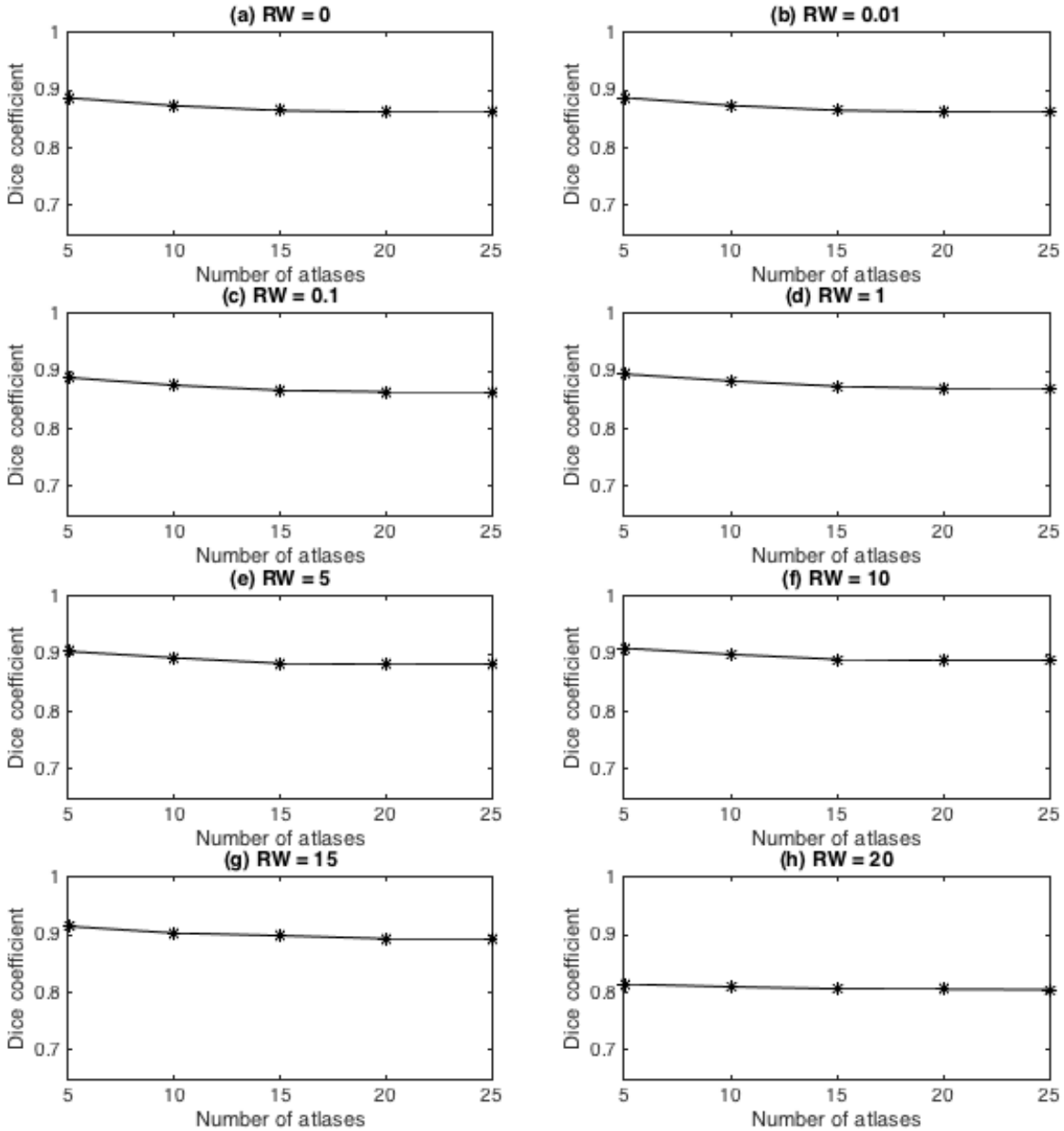


Figure 4.4: The figures (a) to (h) show how the average Dice coefficient for the validation images changes based on the number of atlases used when defining a region of interest. The evaluated settings are marked with (*) on the graph and the RW is kept fixed for each figure.

Table 4.4: The Dice coefficient for the test images when the region of interest is defined according to the multi-atlas and with a threshold of 0.5 for the probability map.

Test image	1	2	3	4	5	6	Average
Dice coefficient	0.8470	0.8763	0.9525	0.7897	0.9344	0.8233	0.8705

Table 4.5: The Dice coefficient for the test images when the region of interest is defined according to the top multi-atlas and with graph cuts, with a regularization weight of 15.

Test image	1	2	3	4	5	6	Average
Dice coefficient	0.9033	0.9210	0.9600	0.8786	0.9441	0.9354	0.9237

The highest Dice coefficient was obtained for test image 3 in all cases of different post-processing methods. In Figure 4.5, we can see how the segmentation changes based on what post-processing was used. Figure 4.5a shows the input image with the given gold standard, i.e. the red line. The left ventricle socket is noticeable in the image, where a clear boundary can be seen, more clear on the left side than the right side. The thresholded probability map with no region of interest shows a clear segmentation without any heavy post-processing applied. The CNN has difficulties classifying the left atrium correctly, which is similar visually as the left ventricle. We can see small local segmentations all over the image. These are removed by using a multi-atlas defining the region of interest with the same simple thresholding method. The multi-atlas successfully removes the misclassified area located below the outline, i.e. the left atrium, but unfortunately also removes a part of the left boundary, which can be seen if Figure 4.5b is compared with Figure 4.5c. The change is noticeable, which shows that the multi-atlas approach used is not optimal looking for the region of interest in the image. Lastly, if graph cuts is applied instead of a simple thresholding, see Figure 4.5d, the boundary of the segmentation appears less noisy and it succeeds in filling the small holes that appeared within the segmentation when using thresholding. The last post-processing method gives a final Dice coefficient of 0.96 for the test image 3, which is an improvement from 0.92 when a simple threshold is used, which can be seen in the Table 4.3.

Test image 4, which can be seen in Figure 4.6, had the lowest Dice coefficient for all different cases of post-processing. It becomes clear why this is the case when Figure 4.6a is studied, where the image is very dark and we can no longer see a pocket with clear boundaries as in test image 3. The problem of the quality of the image becomes more apparent when the thresholded segmentation with no region of interest is seen in Figure 4.6b. A clear segmentation does not exist as in Figure 4.5, the CNN has difficulties finding the boundary of the left ventricle, especially on the left side of the gold standard and near the left atrium. The same problem with the atrium area also appears in this, as in Figure 4.5. An improvement can be seen when a multi-atlas is used to define a region of interest, but it also removes regions within the gold standard, see Figure 4.6c. The graph cuts increases the Dice coefficient

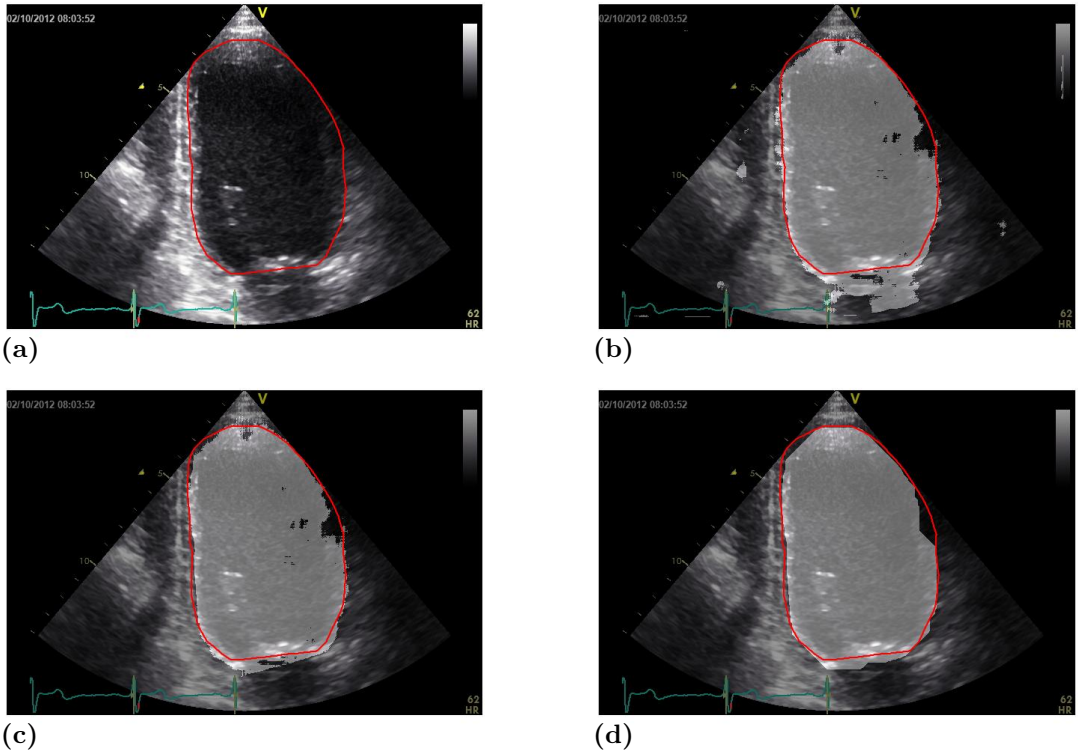


Figure 4.5: The images show the segmentation of the left ventricle for test image 3 when different post-processing method are applied. (a) The test image with the gold standard, i.e. the red line. (b) The segmentation when using a probability map with thresholding of 0.5 and no region of interest. (c) The segmentation when using a probability map with a multi-atlas defining the region of interest and thresholding of 0.5. (d) The segmentation when using a probability map with a multi-atlas defining the region of interest and graph cuts.

by classifying the hole that exists inside the segmentation correctly, which was not possible using a simple thresholding. The graph cuts did also improve the boundary on the right side of the segmentation, see Figure 4.6d. We can see that the CNN has not trained enough on cases like these where the quality of the image is poor. The post-processing improved the segmentation with a Dice coefficient from 0.72 to 0.88, but there are still improvements that can be made when it comes to detecting the boundary correctly.

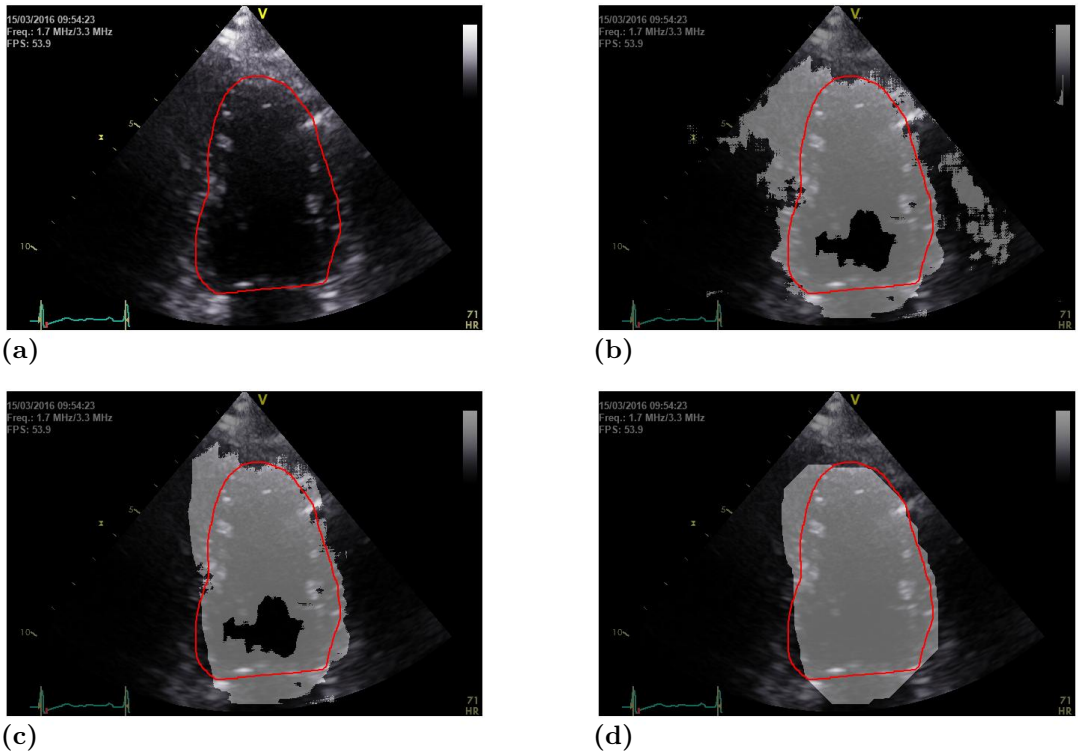


Figure 4.6: The images show how the segmentation changed when different post-processing method were applied. (a) The test image number 4 with the gold standard, i.e. the red line. (b) The segmentation when using a probability map evaluated with no region of interest and thresholded at 0.5. (c) The segmentation when using a probability map evaluated with a multi-atlas defining the region of interest and thresholded at 0.5. (d) The segmentation when using a probability map evaluated on a region of interest computed with a multi-atlas and graph cuts.

4.3.1 CNN training based on the region of interest

The data used for training the CNN had a lot of redundant information, consisting of all the pixels in the black background area in the images. Therefore, we wanted to investigate if the CNN training changes when the black area is included less often in training patches. This was done by only training the CNN on the region of interest defined according to the multi-atlas, which was mentioned in Section 3.3.1. The training had the same hyperparameters in both cases, i.e. a maximal number of epochs = 60, learning rate = 0.0001, learning rate when training accuracy greater

than 0.85 and 0.90 was 0.00005 and 0.00001 respectively and a batch size of 8. As a recap, this parameter setting gave a training accuracy and validation accuracy of 0.9424 and 0.9354 when the training was performed on the whole image. When only training on the region of interest defined by the multi-atlas, the training and validation accuracy was 0.9122 and 0.8660. Each CNN was chosen at the epoch where the validation accuracy was the highest with the least amount of overfitting.

Table 4.6 shows that the average Dice coefficient does not show a large difference for the validation images. The region of interest was computed with multi-atlas using the top 5 aligned atlases with a regularization weight of 15 for the graph cuts. Though, only training on the region of interest means fewer patches to train on. This affects the training time, because one epoch took 13 minutes on average when the CNN trained without a region of interest and 7 minutes when the CNN trained with a region of interest defined by the multi-atlas. This shows that similar results can be obtained with fewer patches by choosing the training area wisely.

Table 4.6: The Dice coefficient on the validation data when the CNN trains on the region of interest in the image according to the multiatlas (MA) and without, which means that the CNN trains over the whole image.

Validation image	1	2	3	4	5	Average
Training without MA	0.9088	0.9554	0.8725	0.9239	0.9125	0.9146
Training with MA	0.9118	0.9574	0.8918	0.9091	0.8987	0.9138

5

Discussion

5.1 The CNN

The CNN was only trained on 2-chamber view ultrasound images, but it would be possible to try the pre-trained framework on 4-chamber view ultrasound images directly. Though, the CNN had difficulties in classifying the left atrium as background in the images, so if the CNN would have been applied on 4-chamber ultrasound images the misclassified background areas could possibly increase, because there would be more areas in the image with similar features as the left ventricle. Since no parts of the algorithm were designed for purely 2-chamber view images, the same CNN structure could be tested on 4-chamber view images after retraining.

To improve the results in general, two convolutional neural networks could be merged, one training on global features in the image and one training on local features in the image.

The method, convolutional neural networks with a sliding window approach, shows that it is possible to create a decent segmentation of the left ventricle based on pixel classification, when the CNN has merely trained on 25 ultrasound images. The method is easy to implement, but the tuning of the CNN can be difficult, because there are a lot of different parameters that can be changed without guaranteeing any improvements for the CNN. The performance of the CNN in this thesis is not consistent for the test images, because a higher Dice coefficient is given when the ultrasound image is of better quality and a lower dice coefficient is obtained with a noisy ultrasound image. Training on more images could solve the inconsistency on the test images, but it is time-consuming for experts to provide annotated images to train on, since the knowledge on how to annotate the images is not general. Therefore, data augmentation could be a possible solution to increase the amount of images to train on.

The highest training accuracy obtained with the convolutional neural network structure used was around 0.95. This can be an indication that more parameters are needed, such as more learnable filters or layers, for the CNN to be able to correctly identify all patches used during training. But more parameters in the CNN can lead to overfitting, which we want to avoid when training a CNN. It will also take a longer time to train and a larger CNN can be more difficult to tune.

5.2 The software

The package that was used for implementing the convolutional neural network is called Theano. The training accuracy converged very quickly when a good learning rate was used, but it was not able to exceed a training accuracy of 0.95, where we assumed that the gradient descent ended up in a local minimum which it could not get out from but software issues could be another explanation. Another package, Torch 7, was used once to see if the training accuracy and validation accuracy were similar using the exact same settings and they were not. The CNN converged much slower than it did for Theano, resulting in a lower training accuracy and validation accuracy for the same number of epochs. It is unknown if the CNN would end up in a local minimum around the same training accuracy with Torch 7 or experience similar software issues as we did with Python. The advantage of the Theano package is that the CNN converges faster compared to Torch 7, but it would be interesting to see if it is possible to obtain a higher training accuracy using Torch 7.

5.3 Post-processing methods

The post-processing method used in this thesis was partly a multi-atlas segmentation, which was computed with a feature-based registration method, that provided us with a region of interest in the 2-chamber ultrasound image. The method was able to estimate a region of interest, but it was not optimal in the sense of finding the region of interest in the image, because it removed areas in the segmentation, which were correctly classified, see Figure 4.5 and Figure 4.6. Therefore, it can be interesting to study if better registrations could be obtained with an intensity-based image registration method or a combination of these two registration methods.

Graph cuts can successfully improve the segmentation by removing misclassified areas close to the boundary, smoothing out the appearance of the boundary and filling in holes that can appear within the segmentation. These types of improvements are not possible when thresholding is used, see Figure 4.5 and Figure 4.6.

5.4 Network type

The probability map over the ultrasound image was given by using a sliding window approach. This method can be time-consuming, which is not always ideal in a hospital environment where one wants to have results in real time. A fully convolutional network gives a segmentation directly without using a sliding window, see for example by Long et al. [10] for more details about fully convolutional networks.

5.5 Future work

The results show that it is possible to find a segmentation of the left ventricle when using a convolutional neural network, but there is room for improvements for the fu-

ture that can be made. It is important to have a large dataset while training a CNN, so it would have been interesting to see if both overfitting could be minimized and if the segmentation could be improved with a larger dataset. To increase the training set even further one could use data augmentation or the validation set can be eliminated by using cross-validation. Data augmentation could extend the training set by simply adding noise or by using plausible transformations on the training images.

Different structures for the CNN can be tested to see if that can make any improvement. Example, combining two different CNNs, where one trains globally (i.e. with larger down-sampled patches) on the image while the other CNN trains locally (with smaller patches), could be an option (see the first section in this chapter). It could also be interesting to study if the CNN trains differently if the image would be kept as RGB instead of grayscale.

6
Conclusion

A segmentation of the left ventricle of the heart in 2-chamber 2D ultrasound images was obtained using convolutional neural networks. The CNN consisted of convolutional layers, max-pooling layers, activation layers and a fully connected layer. The CNN in this thesis had 2 convolutional layers with an input size of 33×33 pixels. The method used for training the CNN is supervised learning, where each training image has its corresponding labelled image (i.e. gold standard), which represents the expected labelling for the given input grayscale image. The labelled image consists of two types of labels, the left ventricle and the background, which is given by an expert's annotation. The output of the CNN is a probability map with pixel values ranging from 0 to 1 representing the probability of being a part of the left ventricle. The segmentation is created from the probability map by using three different post-processing methods: a simple threshold of the probability map, thresholding a probability map computed over a region of interest and lastly a probability map computed on a region of interest plus using graph cuts. A region of interest is given by a feature-based registration that creates a multi-atlas segmentation constructed with the top 5 atlases with the most inliers. The region of interest restricts the image area where the sliding window needs to be applied, which decreases the size of the misclassified areas. The final segmentation is evaluated using the Dice coefficient on the 6 test images, where the average Dice coefficient was equal to 0.82, 0.87 and 0.92 respectively, using a simple threshold, a simple threshold with a region of interest and a region of interest with graph cuts.

Bibliography

[1] G. Carneiro, J. C. Nascimento, and A. Freitas. The segmentation of the left ventricle of the heart from ultrasound data using deep learning architectures and derivative-based search methods. *IEEE Transactions on Image Processing*, 21(3):968–982, 2012.

[2] O. Emad, I. A. Yassine, and A. S. Fahmy. Automatic localization of the left ventricle in cardiac MRI images using deep learning. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 683–686. IEEE, 2015.

[3] O. Enqvist. Lecture Notes in Image Analysis. 2016.

[4] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[5] B. Georgescu, X. S. Zhou, D. Comaniciu, and A. Gupta. Database-guided segmentation of anatomical structures with complex appearance. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 429–436. IEEE, 2005.

[6] F. Kahl, J. Alvén, O. Enqvist, F. Fejne, J. Ulén, J. Fredriksson, M. Landgren, and V. Larsson. Good features for reliable registration in multi-atlas segmentation. *Proceedings of the VISCERAL Challenge at ISBI*, 1390:12–17, 2015.

[7] A. Karpathy, F. Li, and J. Johnson. CS231n Convolutional Neural Network for Visual Recognition. *Online Course*, 2016.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] R. M. Lang, L. P. Badano, V. Mor-Avi, J. Afilalo, A. Armstrong, L. Ernande, F. A. Flachskampf, E. Foster, S. A. Goldstein, T. Kuznetsova, et al. Recommendations for cardiac chamber quantification by echocardiography in adults: an update from the American Society of Echocardiography and the European Association of Cardiovascular Imaging. *Journal of the American Society of Echocardiography*, 28(1):1–39, 2015.

[10] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[11] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[12] J. C. Nascimento and J. S. Marques. Robust shape tracking with multiple models in ultrasound images. *Image Processing, IEEE Transactions on*, 17(3):392–406, 2008.

[13] A. Ng, J. Ngiam, C. Y. Foo, Y. Mai, and C. Suen. UFLDL tutorial, 2012.

[14] M. A. Nielsen. Neural network and deep learning. *Determination Press*, 2016.

[15] C. L. Polte, K. M. Lagerstrand, S. A. Gao, C. R. Lamm, and O. Bech-Hanssen. Quantification of Left Ventricular Linear, Areal and Volumetric Dimensions: A Phantom and in Vivo Comparison of 2-D and Real-Time 3-D Echocardiography with Cardiovascular Magnetic Resonance. *Ultrasound in medicine & biology*, 41(7):1981–1990, 2015.

[16] J. L. Prince and J. M. Links. *Medical imaging signals and systems*. Pearson Prentice Hall Upper Saddle River, NJ, 2006.

[17] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[18] L. Svärm, O. Enqvist, F. Kahl, and M. Oskarsson. Improving robustness for inter-subject medical image registration using a feature-based approach. In *International Symposium on Biomedical Imaging*, 2015.

[19] F.-Y. Wu. The Potts model. *Reviews of modern physics*, 54(1):235, 1982.

[20] K. H. Zou, S. K. Warfield, A. Bharatha, C. M. Tempany, M. R. Kaus, S. J. Haker, W. M. Wells, F. A. Jolesz, and R. Kikinis. Statistical validation of image segmentation quality based on a spatial overlap index 1: Scientific reports. *Academic radiology*, 11(2):178–189, 2004.