Deep Learning for Coronary Artery Segmentation in CTA Images

Using fully convolutional neural networks with deep supervision for efficient volume-to-volume segmentation

Master's thesis in Biomedical Engineering

FREDRIK RING

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis 2018:NN

Deep Learning for Coronary Artery Segmentation in CTA Images

Using fully convolutional neural networks with deep supervision for efficient volume-to-volume segmentation

FREDRIK RING

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Signal processing and Biomedical engineering
Computer vision and medical image analysis group
Chalmers University of Technology
Gothenburg, Sweden 2018

Deep Learning for Coronary Artery Segmentation in CTA Images
Using fully convolutional neural networks with deep supervision for efficient volume-to-volume segmentation
FREDRIK RING

Supervisor: Jennifer Alvén, Department of Electrical Engineering
Examiner: Fredrik Kahl, Department of Electrical Engineering

Master's Thesis 2018:EX029
Department of Electrical Engineering
Division of Signal processing and Biomedical engineering
Computer vision and medical image analysis group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Segmentation of coronary arteries produced by the deep learning algorithm proposed in this thesis. Visualisation constructed in Matlab.

Typeset in LaTeX
Gothenburg, Sweden 2018

Deep Learning for Coronary Artery Segmentation in CTA Images
FREDRIK RING
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The mortality of cardiovascular diseases (CVD) calls for more knowledge of disease mechanisms and prevention strategies. The SCAPIS study, which is a collaboration between Swedish university hospitals and universities, aims to acquire this knowledge and is the worlds largest study focusing on CVD to date. Among other biomarkers, the SCAPIS study is interested in studying atherosclerotic plaque in coronary arteries which is a major risk factor of CVD. Atherosclerotic plaques can be studied from computed tomography angiography (CTA) images. The first step of the analysis is segmentation of the coronary artery tree, which is tedious and time consuming to do manually. Automatic segmentation techniques can speed up this procedure, but unfortunately, current automatic methods lack the accuracy required and have to be manually corrected. Thus, improving the performance of automatic software could be highly valuable to a large scale study such as SCAPIS.

This thesis investigates the viability of a deep learning based automatic segmentation approach and compares its performance to the commercial software Medis QAngioCT Re and to the interobserver variability in manual segmentations from two medical professionals. A version of the state-of-the-art network architecture for vessel wall boundary segmentation, I2I-3D, is implemented in PyTorch and applied to the task. I2I-3D is a fully convolutional network allowing for a variable input size and full volume-to-volume segmentations. It has an encoder-decoder architecture with multiple levels of deep supervision integrated into the decoder. Evaluation on manually corrected segmentations on three CTA images shows that the implemented method outperforms the commercial software for two different segmentation similarity indices; Dice index and modified Hausdorff Distance. However, visual inspection still implies some shortcomings. Although very precise in the segmentation border placement, the network segmentations include broken vessels and unconnected islands of labellings. In contrast, the Medis software's downfall is its inclusion of small, clinically irrelevant vessels. When comparing to the interobserver variability, both automatic approaches perform unfavourably. Nevertheless, I2I-3D shows promising performance despite the relatively limited data set of 25 manually corrected segmentations. The results encourage further development with clear room for improvement. Most importantly, in order to reach full potential the method requires a significantly larger data set. In conclusion, this thesis confirms the viability of using deep learning to segment coronary arteries in cardiac CTA and should be interpreted as a proof-of-concept, rather than the presentation of a finished product.

Keywords: coronary artery segmentation, volume-to-volume segmentation, deep learning, fully convolutional networks, deep supervision, I2I-3D, CNN, cardiac CTA, SCAPIS, medical image segmentation

Acknowledgements

Firstly, I would like to express gratitude to my supervisor Jennifer Alvén, PhD student in the Computer Vision and Medical Image Analysis research group at Chalmers, for her support and thoughtful insights throughout this project. Thank you for helping me break down problems that otherwise would have bested me. Without you, this thesis would not be what it is today. Secondly, I thank Ola Hjelmgren, physician at the Department of Clinical Physiology at Sahlgrenska university hospital, for his enthusiasm towards the project and for being my connection to the medical world. Thirdly, I thank my examiner Fredrik Kahl, professor and leader of the mentioned Computer Vision group, for the opportunity to work with this exciting project and for allowing me to use his office for the entirety of my thesis work. This project has been both educational and a seemingly never-ending source of fun challenges.

Fredrik Ring, Gothenburg, May 2018

Abbreviations

ANN	Artificial Neural Network
BCE	Binary Cross Entropy
CNN	Convolutional Neural Network
CTA	Computed Tomography Angiography
CVD	CardioVascular Diseases
DI	Dice Index
FCN	Fully Convolutional Network
GPU	Graphics Processing Unit
mHD	Modified Hausdorff Distance
ReLU	Rectified Linear Unit: $\mathrm{ReLU}(x) = \max(0, x)$
SCAPIS	Swedish CArdioPulmonary bioImage Study
SGD	Stochastic Gradient Descent

Data sets:

AS	Automatic Medis QAngioCT Re coronary artery segmentations
MS	Manually corrected segmentations from **AS**
PS	Manual pericardium (heart sac) segmentations
VS	Manually segmented sternums from CT images

Contents

1
Introduction

1.1 Background

Cardiovascular diseases accounted for 45% of all deaths from non-communicable diseases in 2015 [1]. Research is focusing on determining effective strategies for diagnosis and prevention, not least in Sweden. The Swedish CArdioPulmonary BioImage Study (SCAPIS) is the largest study of its kind to date and aims to find risk factors for diseases such as stroke, chronic obstructive pulmonary disease (COPD), sudden cardiac arrest and myocardial infarction, with hopes to ultimately prevent these diseases [2]. SCAPIS is performed in collaboration between six universities and six university hospitals, collecting loads of data[1], including cardiac CTA and anthropometric data, of 30 000 Swedish individuals in order to study the disease mechanisms and improve risk predictions.

One part of the data analysis is looking at coronary arteries in computed tomography angiography (CTA) images. CTA images are volumes built up of multiple X-ray measurements, where a liquid contrast agent has been injected into the vessels to make them visible. The coronary arteries can show buildup of vascular plaque causing *stenosis*; a narrowing of the vessel. Vascular plaque in coronary arteries is, among other things, a direct risk factor for myocardial infarction. Commonly known as *heart attacks*, myocardial infarctions are often the result of plaque coming loose from the vessel walls and clogging up an artery, thereby preventing blood flow and causing tissue damage. Segmentations of the lumen, i.e. the space available for blood flow, and segmentations of the full vessel, i.e. the full space inside the vessel wall, can provide information of stenosis, by comparing the diameters of the inner and outer segmentations. A figure showing lumen and vessel wall segmentations in a healthy vessel and a vessel with vascular plaque is shown in Figure 1.1. In the right image, the space available for the blood to flow is clearly limited due to the plaque buildup.

With segmentations of the coronary artery lumen and wall, problem areas are easier to detect than if manually parsing the entire image (volume). Although providing valuable information, the segmentation process is tedious and time consuming.

[1]The complete subject examination consists of cardiac CTA and anthropometric measurements, determination of ratio between abdominal fat and subcutaneous fat, fat content in the liver, ultrasound examinations and lung function tests as well as analysis of sleep quality, physical health and a survey of general health answered by patients.

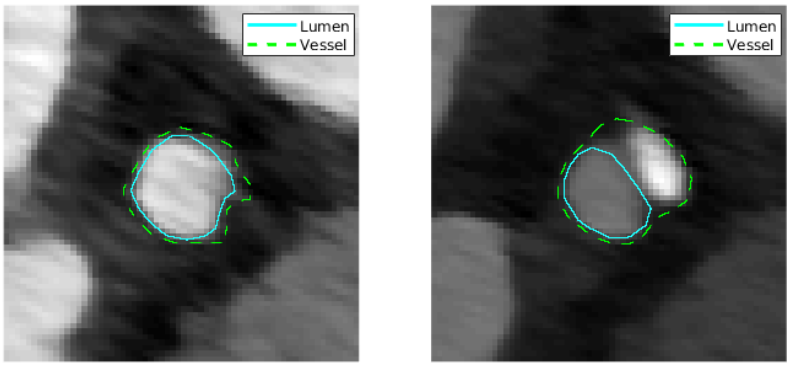

Figure 1.1: Cross section of coronary arteries in healthy vessel (left) and vessel with plaque (right). Rough outline for vessel segmentation showing full vessel (dotted green line), and lumen, i.e. the available space for blood flow (blue line).

With images being in 3D it is hard to view the data properly on a computer screen, adding another obstacle to the process. Even though manual segmentations are considered the gold standard, an automatic method, such as Medis QAngioCT Re [3], is typically used to create a proposal segmentation. That base segmentation is then manually corrected by medical professionals. This procedure drastically reduces the time needed to create the segmentations. Having fast and reliable segmentation software that performs in line with what medical professionals desire is thus highly valuable for large scale studies such as SCAPIS. Construction of an automatic coronary artery segmentation tool is a complex task, partly due to vessel segmentation being complex in its own right, and partly due to the volumes containing a large number of lung vessels which should not be included in the segmentation.

With the field of image analysis moving towards deep learning approaches for segmentation tasks, it is natural to wonder how a deep learning algorithm would compare to non-learning based software like QAngioCT Re at the task of coronary artery segmentation. If a deep learning tool performs well on the task, it could provide professionals with better tools in their complex task of combating terrible diseases. Deep learning algorithms have been successful on similar tasks in the past. In 2015, Ronneberger et al. introduced *U-Net*, a fully convolutional network for biomedical image segmentation which surpassed the performance of all competition on the ISBI challenge on neuronal segmentation [4]. Further, Merkow et al. built on the general structure of U-Net with integration of deep supervision in their network, *I2I-3D*, which was used in vessel wall boundary segmentation [5]. These networks, and their underlying building blocks, are reviewed in more detail in Chapter 2.

Investigating the possibility of using deep learning tools for coronary artery segmentation is the focus of this thesis. A state-of-the-art deep learning network for medical image vessel segmentation is applied to the task of coronary artery segmentation, with data provided by the SCAPIS study, to determine if such an approach could be viable.

1.2 Project description

The foundation for this thesis is a collaboration between researchers from Chalmers University of Technology and the SCAPIS study. The Computer Vision and Medical Image Analysis group at the Department of Electrical Engineering at Chalmers University of Technology have gotten access to manually segmented CTA data from the study. The vision is to create a deep learning based tool for segmentation of coronary arteries as well as detection and classification of stenosis and vascular plaque. This master thesis project comprises the first step of the creation of such a tool; deep learning based coronary artery segmentation. More specifically, the question that this thesis aims to answer is:

Is deep learning viable for coronary artery segmentation?

To answer this question, a deep learning architecture suitable for the characteristics of this particular problem has to be chosen. The architecture needs to be able to handle 3D image data and learn to segment tiny structures from a relatively limited data set. How viable the approach is can then be determined by comparison to other methods, both automatic and manual. The main research question can thus be divided into two sub-questions:

(1) What is state-of-the-art in the field of deep learning based medical image segmentation in general and, more specifically, with focus on vessels?

Answering this question requires knowledge of the field; of what is being used and researched in medical image segmentation, and what has led the field there. Through this investigation, knowledge is gained both of the key elements behind deep learning and the specific characteristics of deep learning applications on medical images. An exploration of the field also provides a scientific context in which to place the report as well as ensures that the choice of network architecture can be made based on previous research.

(2) How does a deep learning approach compare to other automatic approaches as well as manual segmentations?

With this question, a shift from the theoretical possibilities towards practical implementation is made. Having acquired both a solid foundation of knowledge on deep learning medical image segmentation and a network structure, the potential of the method needs to be tested. This requires reference points for performance. As the algorithm aims to produce segmentations as close to manual segmentations as possible, it is natural to compare the network segmentations to those from medical professionals. Further, a comparison with an automatic software that is currently in use for the task of coronary artery segmentation will be a clear test of its viability.

Combined, these questions lead to theoretical and practical knowledge of deep learning for medical image segmentation and, specifically, a conclusion of the viability of using deep learning to segment coronary arteries in cardiac CTA.

1.2.1 Limitations

This project does not deal with the construction of a full application. It does not include a graphical user interface or executable software for image segmentation. The end product is in the form of trained network parameters together with Python scripts to load the parameters into the network, to train it further, and to segment image volumes.

Further, the algorithm is not developed for stenosis detection. It is limited to vessel wall segmentation and does not separate between what is lumen and what is vessel wall. The aim of this thesis is to investigate coronary artery segmentation, meaning the full volume occupied by coronary arteries. Thus, additional support for lumen segmentation or stenosis detection are not covered in this project.

To study the potential of a deep learning based method for coronary artery segmentation, a network architecture is not developed from scratch. Rather, an existing architecture is used and adapted to the specifics of the problem. In the implementation of the deep learning network, hyperparameters and initialisation process will be chosen based on recommendations from scientific articles on the subject, and small scale tests. A thorough optimisation of the hyperparameters and investigation of different initialisations will not be part of the study.

The project is also limited in terms of data. The main data set of manually segmentated coronary artery images from SCAPIS to train and test performance on contains 25 CTA images. This is only a small subset of the complete SCAPIS data, as SCAPIS is still in an early phase and a protocol for manual segmentations has not yet been established. This means that the data available in this project might not represent the final data set for the SCAPIS study. The results should thus be interpreted as an isolated study. However, they can also serve as an indicator of the potential and viability of a deep learning approach using the full data set.

1.3 Outline of thesis

This chapter served as an introduction to the thesis, explaining the background and relevance of the project. Further, it specified the questions the thesis aims to answer as well as its delimitations.

Chapter 2 introduces the theoretical framework and key concepts of deep learning needed to understand the methods used in the project. This is followed by an overview of the field of medical image segmentation, starting broadly with automatic methods and ending with deep learning applications on vessel segmentations.

In **Chapter 3**, the methods used are described in detail. The chapter starts by describing the data sets that were used and how they were processed. It then moves on to describing network implementation in the deep learning framework PyTorch and ends by describing the training and evaluation processes.

In the following chapters, **Chapters 4 and 5**, the results of the implementation of the deep learning algorithm are presented and analysed, respectively. In these chapters, the focus lies on comparing the network segmentations to the Medis QAngioCT Re segmentations and the manual segmentations in order to establish the viability of the proposed method. Finally, a conclusion follows in **Chapter 6**.

2
Theory

This chapter aims to give the reader the foundation needed to understand the contents of this report, as well as to describe the research surrounding the topic. It is sectioned into two parts; a section on deep learning and a section highlighting related work.

2.1 Deep learning

The need for software able to make informed decisions from large amounts of data is ever increasing in our digital society. Such software is highly dependent on the interpretation of the data, which is a complex task to manage. Instead of manually programming the interpretation into the software, there might be benefits to gain from creating software that can take care of both the interpretation and the decision making. Such software, able to *learn* how to interpret data on their own are generally classified as *machine learning algorithms*.

More generally, machine learning refers to a family of data-driven algorithms that are able to automatically learn function mappings to desired output responses from given inputs. Machine learning can be divided into two categories; *supervised learning* where the desired output is explicitly stated, and *unsupervised learning* where the algorithm has to learn a suitable output on its own. The former will be the focus of this report.

The actual learning process is mathematically guided and allows the algorithm to find important *features* and disregard redundant information in the data without explicitly being programmed to. The algorithms find new ways of representing data to fit a desired response over many iterations of tuning their parameters, usually called training. This iterative optimisation process of tuning the parameters to the data can be computationally intensive. Machine learning techniques have been around for decades, but only with modern advances in computational power have their true potential fully been demonstrated.

This section will introduce *deep learning*; a machine learning sub-type based on *artificial neural networks*. The section will explore key concepts with a specific focus on *convolutional neural networks*. The reader is assumed to have basic knowledge of neural networks and machine learning. The following should serve as an overview, rather than a complete mathematical derivation of the underlying theory. For a

deeper review of deep learning, see *Deep learning* by LeCun et al. [6].

2.1.1 Artificial neural networks

Artificial neural networks (ANNs) are a branch within machine learning, inspired by the structure of the brain. ANNs are networks of inter-connected simple computing units called *neurons*. Just like in the brain, these neurons are responsible for passing along information in the huge network of neurons that constitute the brain. A typical ANN structure can be seen in Figure 2.1. This type of network is known as a *feed-forward* network as information only moves in one direction to produce an output. It is also a *fully connected* network due to every neuron being connected to all neurons in the previous and the next layer. The networks usually consists of multiple layers of neurons. The information flows through the network layer by layer and is manipulated in each pass between layers through weighted connections, represented by arrows in the figure, and *activation functions*. The more layers a network has, the *deeper* it is said to be and the more complex functions it can learn to model. In the case of the network shown in Figure 2.1 there is only one hidden layer, resulting in a quite shallow network.

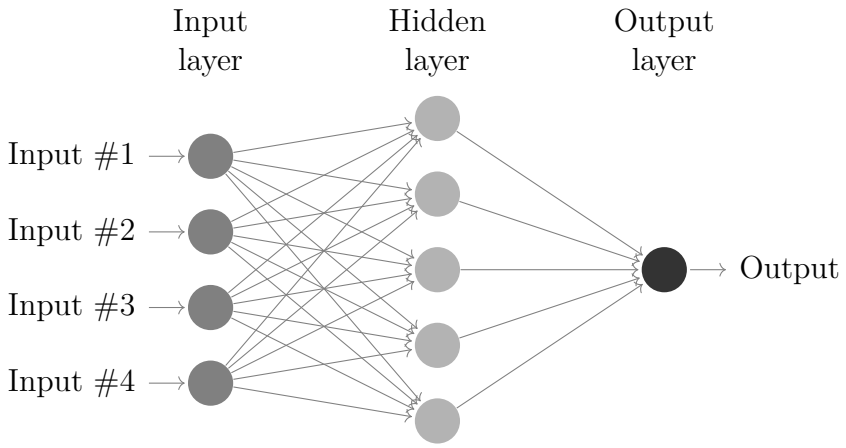

Figure 2.1: A simple feed-forward neural network structure with one hidden layer, mapping from $\mathbb{R}^4$ to $\mathbb{R}^1$. Neurons are represented by coloured circles.

The mathematics driving the neurons of the network is simple. Each of the neurons weighted inputs gets summed together, subtracted by a bias term and passed through an activation function before moving on to the next layer. For neuron i in the input layer of Figure 2.1 the computation at the neuron is simply

$$O_i^{(I)} = a(w_i^{(I)} I_i - B_i^{(I)}) \tag{2.1}$$

where I_i is input #i, $w_i^{(I)}$ is the weight from the weighted connection to the neuron, $B_i^{(I)}$ is the bias term, $a(\cdot)$ is the activation function, and $O_i^{(I)}$ is the output from the neuron, with superscript (I) referring to the *Input layer*. There is no summation in this expression due to there only being one input to this neuron. Similarly for the

jth neuron of the *Hidden layer*, which will be referred to with superscript (H), the expression for the computation at the neuron becomes

$$O_j^{(H)} = a(\sum_{i=1}^{4} w_{ji}^{(H)} O_i^{(I)} - B_j^{(H)}). \tag{2.2}$$

This expression shows the summation of the previous layers weighted outputs, but is otherwise the exact same calculation as in the previous layer. This calculation drives all of the neurons in the network and can be seen in the *Output layer*, with superscript (O), as well,

$$O^{(O)} = a(\sum_{j=1}^{5} w_j^{(O)} O_j^{(H)} - B^{(O)}). \tag{2.3}$$

The propagation of information through the network, from input to output, is completely described by these calculations. However, for the network to output a desired response from an input it needs to *learn* appropriate values for its parameters, the weights w and the biases B. This is achieved through iterative updates based on a process named *backpropagation*.

2.1.1.1 Backpropagation

A simple way of describing the mathematics behind learning is that it should update the parameters of the network in the way that most makes the output from the network resemble the desired output. To do this, one first needs to describe how different from the desired output the network output is. That is the role of the *loss function*. There are many different viable loss functions, one of them being the squared distance function, $L = (O^{(O)} - O^*)^2$ where $O^{(O)}$ is the network output and O^* is the desired output, which will be used in this demonstration. The task is now to minimise L. The ideal outcome is to reach a case of $L = 0$, meaning that there is no difference between the network output and the desired output.

Assume that for a specific time t_n, the weights of the network $\mathbf{w}(t_n)$ are such that $L \neq 0$, with bold $\mathbf{w}$ referring to all the weights stacked in a vector. One way to minimise L is to update the weights in the direction where L decreases fastest, or in other words, where the slope of $L(\mathbf{w})$ is largest. Mathematically, that can be computed by differentiation, or the *gradient* in the multivariate case. The update rule for each one of the weights, based on this approach, is shown in Equation 2.4,

$$w_j(t_{n+1}) = w_j(t_n) - \eta \frac{\partial L}{\partial w_j}\bigg|_{t=t_n}. \tag{2.4}$$

The factor η is called the *learning rate* and is a hyperparameter determining how large a single update step should be. The minus sign ensures that the update is down the slope, so that L decreases. This method of updating the weights is appropriately named *gradient descent*, and is one of multiple viable optimisation methods

to tune the weights of a neural network.

Calculating $\frac{\partial L}{\partial w_j}\Big|_{t=t_n}$ is a straight forward task through the use of the chain rule. The term can be split into its constituents as shown in Equation system 2.5, exemplified for the small neural network described in Figure 2.1 and Equations 2.1-2.3, where the gradient is calculated for a weight connecting the hidden and the output layer, $w_j^{(O)}$, a weight connecting the first layer and the hidden layer, $w_{ji}^{(H)}$, and a weight connecting the input to the first layer, $w_j^{(I)}$:

$$\begin{aligned}
\frac{\partial L}{\partial w_j^{(O)}} &= \frac{\partial L}{\partial O^{(O)}}\frac{\partial O^{(O)}}{\partial w_j^{(O)}} = \underbrace{2(O^{(O)} - O^*)\ a'}_{=\delta^{(O)}} O_j^{(H)} = \delta^{(O)} O_j^{(H)} \\
\frac{\partial L}{\partial w_{ji}^{(H)}} &= \frac{\partial L}{\partial O^{(O)}}\frac{\partial O^{(O)}}{\partial O_j^{(H)}}\frac{\partial O_j^{(H)}}{\partial w_{ji}^{(H)}} = \delta^{(O)} \underbrace{w_j^{(O)} a'}_{=\delta_j^{(H)}} O_i^{(I)} = \delta^{(O)}\delta_j^{(H)} O_i^{(I)} \\
\frac{\partial L}{\partial w_i^{(I)}} &= \frac{\partial L}{\partial O^{(O)}}\frac{\partial O^{(O)}}{\partial O_j^{(H)}}\frac{\partial O_j^{(H)}}{\partial O_i^{(I)}}\frac{\partial O_i^{(I)}}{\partial w_i^{(I)}} = \delta^{(O)}\delta_j^{(H)} w_{ji}^{(H)} a' I_i.
\end{aligned} \tag{2.5}$$

The same procedure is done for all of the weights and biases of the network. The right hand side expressions in Equation 2.5 are easy to evaluate at the specific time t_n once the values have been propagated through the network. As can be seen from the δs, the gradients all build on each other just like Equations 2.1-2.3 leading to the network output, but in the opposite direction. The input data flows through the network from input layer to output layer in order to calculate the output and the first gradient. The updates are then calculated from the output layer to the input layer, as gradients are propagated backwards in the network. This is the process of backpropagation, first introduced by Rumelhart et al. in 1986 [7].

2.1.1.2 Activation function

While present in all layers of the network in Figure 2.1, the role of the activation function has yet to be described. To understand why it is needed, assume a network without activation functions. In vector notation, the computation at every layer of the network can be described as follows: $f(\mathbf{x}) = a(A\mathbf{x} + B)$, where $a(\cdot)$ is the activation function, $\mathbf{x} \in \mathbb{R}^n$ is the input to the layer, $A \in \mathbb{R}^{m\times n}$ is the weight matrix, and $B \in \mathbb{R}^m$ is the bias vector. It is then clear that in a network without activation functions, every layer represents an affine mapping: $A\mathbf{x}+B$. Assume two such layers, with affine mappings $f(\mathbf{x}) = A\mathbf{x} + B$ and $g(\mathbf{x}) = C\mathbf{x} + D$. Propagating $\mathbf{x}$ through both layers, first through g and then through f, results in $f(g(\mathbf{x}))$. Expanding this expression gives as follows:

$$f(g(\mathbf{x})) = A(C\mathbf{x} + D) + B = (AC)\mathbf{x} + (AD + B),$$

with (AC) being a matrix and $(AD + B)$ being a vector, the result is a new affine mapping. Thus; combining affine mappings gives no new complexity to the model.

Multiple layers would not have any benefits in a neural network, since they could always be represented by a single layer. However, with the addition of non-linear activation functions between the affine mappings the linearity is broken and the network can model much more complex mappings.

While there are countless non-linear functions, many neural networks use one of $\text{ReLU}(x) = \max(0, x)$, $\tanh(x)$ or the sigmoid function; $\sigma(x) = \frac{1}{e^x+1}$. The reasoning behind this is that these functions have gradients that are easy to compute, which is crucial for learning as shown in Section 2.1.1.1.

2.1.1.3 Overfitting

The training of a neural network may seem straight forward but the process is quite delicate and complex. There are unwanted phenomenons that need to be considered, such as *overfitting*. Overfitting is the phenomenon where the neural network is overtrained on the training data set to the point that it fails to properly process new data. While producing correct results on the training set, it might have been tuned too well to the specific samples of the training set, rather than learning techniques to properly process it. That way, it becomes more of a dictionary of the training data, giving the correct output for the chosen input, but not being able to handle new data. This behaviour is very undesirable and there are ways to combat it such as data augmentation and drop-out layers. These methods are examples of *regularisation* methods which aim to help the network learn to generalise from the data it sees in order to handle new data well.

Data augmentation refers to different methods of extending the available data through software. By doing natural manipulations on the data, more variation can be represented which gives the network better conditions to learn important features. With more variation represented, the training process is less susceptible to overfitting. Which manipulations are viable for augmentation depends on the data. In the case of images, typical manipulations include flipping along axis, rotation, scaling and noise addition. For the coronary artery data, flipping along axis is not a viable manipulation due to the asymmetry in the vessel structure. The left and right - as well as top and bottom, and front and back - sides have characteristic shapes and flipping will not represent naturally occurring data. Small rotations, scaling, and additive noise are on the other hand feasible and were implemented in this project.

Drop-out is another technique to combat overfitting by randomly dropping units and their connections in each training iteration [8]. Dropping refers to removing the units from the learning process. This ensures that the network is not too dependent on any single unit or small group of units. The network constantly has to learn to process the data correctly from a new set of active neurons each training iteration, which has been shown to significantly reduce overfitting.

Another regularisation method is weight decay, which reduces the size of the weights by a factor every weight update. This has been shown to attenuate irrelevant components of the vector, improving the networks ability to generalise from data [9].

2.1.1.4 Optimiser

There are different ways of updating the network parameters, or *optimising* them, based on the gradients. Gradient descent, as mentioned in Section 2.1.1.1, is one such optimiser. In reality, *stochastic gradient descent* (SGD) is used in favour of regular gradient descent, due to it being faster and computationally cheaper. Gradient descent requires the gradients to be calculated for the entire training data set for each update. SGD allows for updates after every data sample, which is much more efficient.

While reliable, SGD requires careful choice of *hyperparameters* to tune the algorithm. An optimal learning rate with or without decay, weight decay factor and a *momentum* factor have to be set manually and greatly affect the outcome of the optimisation. Both learning rate and weight decay have been explained previously. Momentum acts as a sort of memory in the training process. Instead of calculating completely new update terms every iteration, the new update term is complemented by the previous update term, typically scaled down by a factor. The scaling factor then tunes the balance between previous update terms (the memory) and the current update term.

Another optimiser, Adam, builds on SGD and is able to adjust learning rate on its own and individually for each weight [10]. For specific details of how this is achieved the reader is referred to the paper in which the method is described, i.e. [10]. SGD still sets the bar for finding good minimas of the loss function, whereas Adam might get stuck in bad local mininas. However, Adam is faster to converge and does not require the same level of manual tinkering.

2.1.1.5 Loss function

The loss function solely determines what the network is optimised for, since it is what the learning algorithm minimises. It is a tricky task to choose the optimal loss function. Typically, there is a discrepancy between the qualitative characteristics wanted from the network and what can be mathematically described. An exampel of a simple loss function was demonstrated in Section 2.1.1.1; the squared distance loss $L = (O - O^*)^2$. There are many different loss functions used in deep learning, with each being suited for specific tasks. For image segmentation tasks with fully convolutional neural networks, *binary cross entropy* (BCE) is a commonly used loss function [4, 5]. It is formulated in Equation 2.6, with $\boldsymbol{O} = (o_1, ...o_N)$ being the network output and $\boldsymbol{O}^* = (o_1^*, ..., o_N^*)$ being the desired output, as described in Section 2.1.1, for an image with N voxels:

$$l_{BCE}(\boldsymbol{O}, \boldsymbol{O}^*) = -\sum_{n=1}^{N} o_n^* \cdot \log(o_n) + (1 - o_n^*) \cdot \log(1 - o_n). \tag{2.6}$$

This is a measure of negative log likelihood and quantifies the difference between two probability distributions, where the two terms in the sum represent the two classes, i.e. the binary nature. Intuitively, one can think of cross entropy as a measure of how easy it is to predict $\boldsymbol{O}^*$ from $\boldsymbol{O}$, where l_{BCE} closer to 0 means easier.

2.1.1.6 Deep supervision

In supervised learning applications, where the desired output is explicitly stated in the training process, the parameter updates are typically calculated based on the loss from comparing the network output and the desired output, as previously described. In theory, one could also add outputs from deeper layers in the network and calculate losses from them as well, in order to force the network to produce viable outputs in its earlier stages. This is the idea behind deep supervision, which was first introduced by Lee et al. [11]. Their article shows that the addition of deep supervision not only decreases classification error rate, but also speeds up convergence.

Assume a network with N total outputs, from N different layers in the network. Deep supervision works by calculating a loss for each of the outputs. Generally, to compare the network output to the desired output one might need an additional layer to produce an output shape corresponding to the desired output. All of the networks losses are then summed together to create the total loss, which is used for parameter optimisation. With N levels of supervision, and the individual losses denoted by l, the total loss, L, is simply as shown in Equation 2.7, with superscript denoting level,

$$L = \sum_{n=1}^{N} l^{(n)}(\boldsymbol{O}^{(n)}, \boldsymbol{O}^{*}). \tag{2.7}$$

2.1.1.7 Transfer learning

Transfer learning refers to the act of applying gained knowledge on one task to a new task. This is largely relevant in the field of deep learning. In many deep learning applications, there might not be a large enough data set to generalise from. Something that can help is then to initially train the network on different data sets with similar features to the main data set, before the main training. The data representations learnt from those data sets might improve the learning on the main data set, which gives the network a head start on learning to generalise [12]. This training on different data sets before training on the main task, generally known as *pretraining*, is a common practice in deep learning applications.

2.1.2 Convolutional neural networks

Convolutional neural networks (CNNs) are a special category of feed-forward networks whose core structure was first introduced under the name *Neocognitron* by Fukushima et al. in 1982 [13]. Since then, CNNs have become largely popular in computer vision applications. They are especially well suited for image analysis due to their shift invariance and non-sensitivity to natural image variations such as lighting and visual clutter [14]. As of 2017, CNN-based architectures are the top performers in ImageNet's *Large Scale Visual Recognition Challenge* in object detection, object classification and object localization [15, 16].

What separates CNNs from the network type introduced in Chapter 2.1.1 is that the operation driving the network is filtering, performed by *discrete convolutions*, rather than Equations 2.1-2.3. Each layer of a convolutional neural network applies trained filters on the input through convolutions. The filters are typically much smaller in size than the image, meaning that they are applied locally to the image. They are then shifted a set distance in the image, referred to as *stride length*, successively building up the output.[1] Each output voxel is the result of a dot product between a filter and a local neighbourhood of the corresponding voxel in the input. A visualisation of this is shown in Figure 2.2. This local application of the same filter all across the image is what gives CNNs their shift invariance.

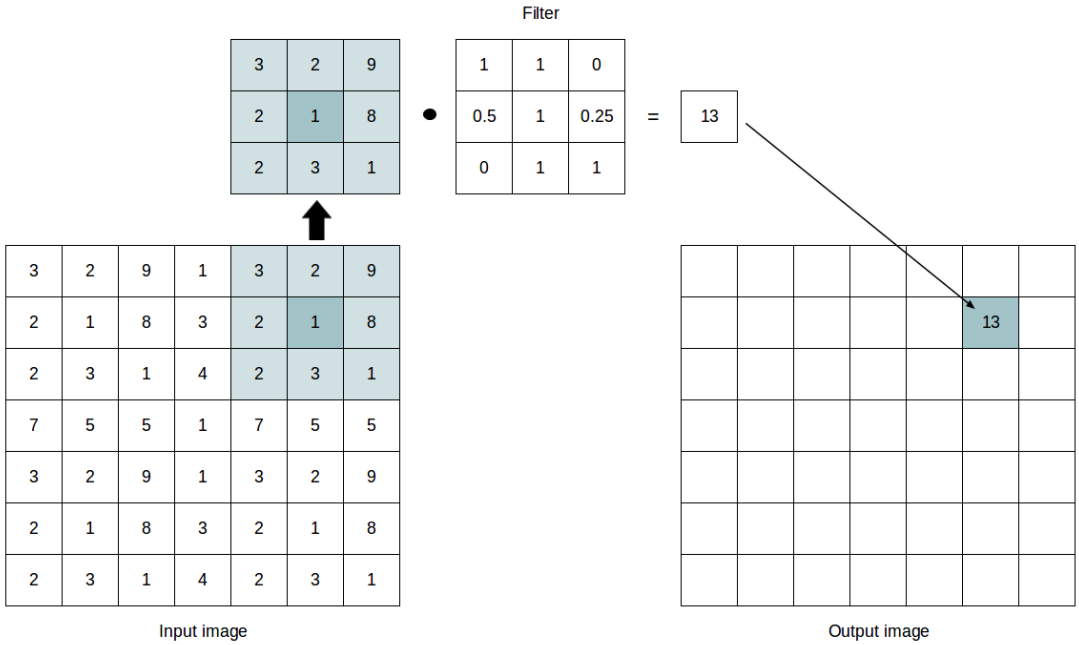

Figure 2.2: Visualisation of how the output is formed in a CNN. A filter is applied to a local neighbourhood in the image through dot product. The result is a single value at the centre of the corresponding position of the neighbourhood in the output image. The filter is applied to the entire input image in this way.

The filtering process creates a loss of pixels around the borders, which can be seen in Figure 2.3 as the filter can't be moved up or right in the input image from its current position, leaving the space above and to the right of the output pixel empty. This can be handled by *zero padding* the input, placing zeros on the outside of the input image, allowing the filter to be applied at the very borders of the image.

Intuitively, the filtering can be thought of as a means of feature extraction, where the network learns important features of the data, as well as the appropriate filters to extract them. It then applies these filters regularly along the image, "looking" for the specific features it has been trained for. The output becomes a *feature map*,

[1] In reality, filters are duplicated to match the size of the input image in order to perform the full filtering in one operation, rather than shifting and applying the filters iteratively.

containing information of if and where features are located. At shallow levels in the network these features may be distinctive gradients in images or characteristic shapes of signals, but deeper layer filters are often far too abstract to have interpretable conceptual counterparts.

In a CNN, convolutional filters are often interlaced with activation functions and *pooling layers*. Pooling layers reduce the size of the input, typically by half along each axis. There are different ways of pooling; max pooling converts a neighbourhood of voxels into one voxel containing the largest value from the neighbourhood, average pooling converts the neighbourhood into a voxel containing the average value, and so on. An example of max pooling on a 2D image with a pooling window of size $(2, 2)$ and a stride length of 2 is shown in Figure 2.3. Pooling enables the network to work with the data in multiple scales. By pooling the input the network effectively gets a larger *receptive field*. As in Figure 2.3, four pixels represent a total of 16 pixels in the original image. This allows CNNs to learn features in both large and small scales; using both global and local image information.

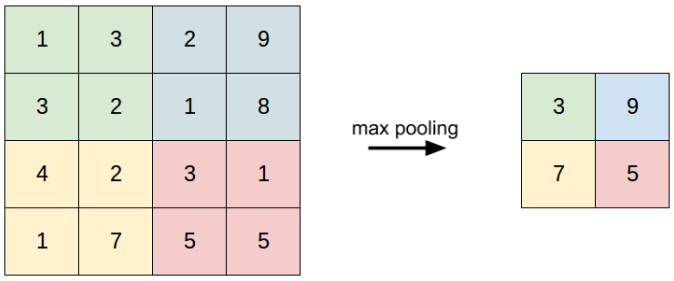

Figure 2.3: Maxpooling of the image matrix on the left with a pooling window size of (2,2) and a stride length of 2. The result is seen on the right, where each pixel is the maximum value of their respective neighbourhood in the image.

2.1.3 Terminology

To provide the reader with a quick view of terms that are useful to know in relation to deep learning and the specifics of this thesis, a small dictionary is shown below.

Activation function Non linear function to break linearity in networks.

Epoch One full pass over every data point in the training process.

Layer Every single function applied to the input can be thought of as a layer that the input propagates through and is changed by.

Level A block within the neural network where image resolution is constant.

Neuron A single computation unit in the network.

Tensor A multidimensional extension of vectors and matrices.

Training The process of having a neural network learn to produce desired outputs from inputs through iterative updates of the network parameters.

Volume / 3D image Used interchangeably to refer to the CTA images, which are 3D image volumes composed of stacks of 2D images.

Voxel The 3D equivalent of a pixel.

2.2 Previous work on medical image segmentation

Manual image segmentation is both hard and time consuming and many automatic approaches have been developed with the aim to reduce the amount of manual work. This section provides an overview of important automatic segmentation techniques, both non-learning and deep learning based, used in medicine and specifically for vessels.

2.2.1 Non-learning based medical image segmentation

Automatic image segmentation techniques have been used since before deep learning algorithms were applied to the task. A technique that has been around since before the 21st century is the use of *Watershed*. Watershed is an algorithm that interprets an image as a topological map where pixel intensities represent height [17]. Simplified, the segmentation is constructed from borders defined by "height" differences. In 2004, Yu-Len Huang et al. successfully implemented Watershed to segment breast tumors in 2D sonography [18]. The tool was used to assist medical professionals, aiming to save time and help inexperienced physicians reach the correct diagnosis.

Another automatic approach used specifically for vessel-like structures is *Frangi filtering* [19]. The filter enhances tube-like structures based on the eigenvalues of the Hessian matrix, locally applied to the image. The Hessian is a matrix containing the second order partial derivatives. From its eigenvalues one can determine the local principal directions in the data, making it a good measure for local tube-likeness in an image. Among others, a method based on Frangi filtering was applied by Martinez-Perez et al. to segment blood vessels in fluorescein retinal images and reached 0.94 in segmentation accuracy (in this case meaning the *number of correctly segmented pixels* divided by the *total number of pixels*) when compared to medical professionals [20].

A commercial software building on Frangi filters, exclusively used for coronary artery segmentation, is the Medis QAngioCT Re software [3]. The frangi filtered output in QAngioCT Re is post-processed using geometric information to separate coronary arteries from unwanted structures. To separate the aorta from the rest of the coronary artery tree the software uses a Hough transform to determine its location. The Hough transform is a feature extraction technique excelling at the detection of circular shapes in images [21]. The software has performed well on the Rotterdam Coronary Artery Evaluation Framework, currently placing 10th in coronary artery segmentation and 14th in stenoses detection [22, 23]. The SCAPIS study utilises

Medis QAngioCT Re for automatic coronary artery segmentation.

Even though automatic approaches like Watershed and Frangi filtering have proven useful, the field of image segmentation is shifting away from such non-learning based techniques towards deep learning. This is in part due to the impressive performance of CNNs in image analysis.

2.2.2 Deep learning based image segmentation

Due to CNNs impressive performance in image classification, as mentioned in Section 2.1.2, it was only natural to investigate their performance on segmentation tasks as well. The first approaches used pixel-by-pixel classifications, where a single pixel was classified based on the pixel values of a square neighbourhood centered on it. The filter is applied in a sliding-window manner, being locally applied to a neighbourhood at a time and building up the output as the filter moves across the image. With this approach, Ciresan et al. used four stages of convolutional filters together with fully connected layers to produce segmentations from pixel-by-pixel classification and outperformed the previous state of the art in the ISBI 2012 EM Segmentation Challenge [24]. Ning et al. used a similar architecture to segment images of C. elegans embryos into five classes with an error rate of 29.0% [25].

A breakthrough by Long et al. in 2015 with a network architecture named *fully convolutional networks* (FCNs) enabled the use of full image-to-image segmentations, rather than constructing segmentations from pixel-by-pixel classifications on image patches [26]. As the name suggests, FCNs have no fully connected layers. They are composed solely from convolutional filter layers, pooling and activation functions. This allows the networks to be size-agnostic, able to produce a full size segmentation regardless of the input image dimensions. In their paper, Long et al. show FCNs strong ability to produce both accurate classification and localisation while working on a full image, improving accuracy and reducing computational cost when compared to patch-by-patch pixel classification.

2.2.3 Medical image segmentation with U-Net

In 2015, Ronneberger et al. introduced a convolutional neural network structure specifically developed for biomedical image segmentation called *U-Net*, building on FCNs [4]. At its time of introduction, U-Net surpassed the performance of all other methods on the ISBI challenge for neuronal segmentation in electron microscopy images, as well as the ISBI cell tracking challenge. What makes it ideal for biomedical image segmentation is its ability to learn from a relatively small data set, something that can otherwise be a problem in medical applications.

The U-Net network structure is shown in Figure 2.4. It has the typical CNN components of filters, activation functions which are ReLU in this case, and pooling. What separates it from other architectures is that after the standard encoder structure (left side) of moving down in resolution and extracting features, it has a decoder structure

(right side), turning the features into a full resolution segmentation in the same level structure as in the encoder. This gives the network a U-shape, explaining the origin of its name. The upscaling in the decoder is performed by what Ronneberger et al. call up-convolutions, also known as transposed convolutions or deconvolutions. There are also interconnections between the levels of the encoder and the decoder. The outputs from the encoder levels are concatenated to the corresponding decoder level inputs, where they are mixed together by convolution. As the network pools the data in the encoder, the shallow fine detail information gets lost with the image size reduction. This is a problem, since small scale localisation is important in medical images. Having the encoder responses added back into the decoder combats this problem, allowing the final layers of the network to have information of both the shallow fine detail localisation and deep semantic information determining what class the individual pixels belong to.

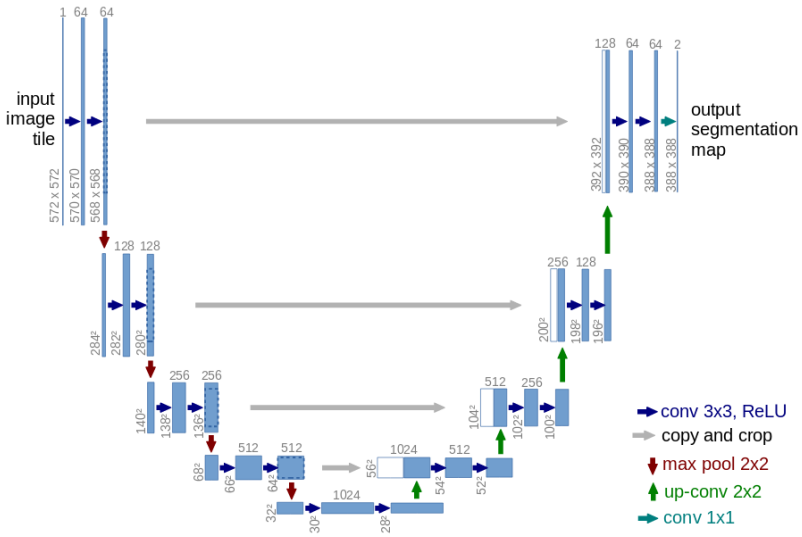

Figure 2.4: Image describing U-Net from the original paper by Ronneberger et al. [4]. The filter sizes are written on top of every convolutional layer and the filter input sizes are written on the sides.

The U-net structure has also been extended to 3D image segmentation by Çiçek et al. [27]. One might assume that 3D data could be sliced into a stack of 2D images and segmented slice by slice before restructuring the segmentations into a 3D volume, making the need for a 3D version of U-Net unnecessary. However, anatomical structures have features in all three spatial dimensions, meaning that leaving out one dimension would limit the information available to the network. Slicing 3D data is thus not good practice when working with medical images, unless other measures are taken to account for the three-dimensionality of anatomical features.

2.2.4 Vessel wall segmentation with I2I-3D

In 2016, Merkow et al. introduced a network structure originally designed for vessel wall segmentation named I2I-3D [5]. The architecture builds on a 3D U-Net struc-

ture with the addition of *deep supervision*. It has the general structure of a fine to coarse, or shallow to deep, encoder followed by a coarse to fine, or deep to shallow, decoder with four levels, as well as the interconnections between encoder and decoder that are present in U-Net. The complete structure, together with specifications of the number of filters used is shown in Figure 2.5. Upsampling is performed by deconvolutions. The filter levels contain no drop-out layers and are completely made up of the convolutional filters and ReLU functions shown in the image. The mixing layers in the decoder, coloured orange, combine the information from the previous layer with the corresponding encoder level output through $1 \times 1 \times 1$ convolution filters. Merkow et al. showed that this approach, with deep supervision placed at the end of every decoder level as illustrated in Figure 2.5, outperformed the previous state of art in 3D vascular boundary detection.

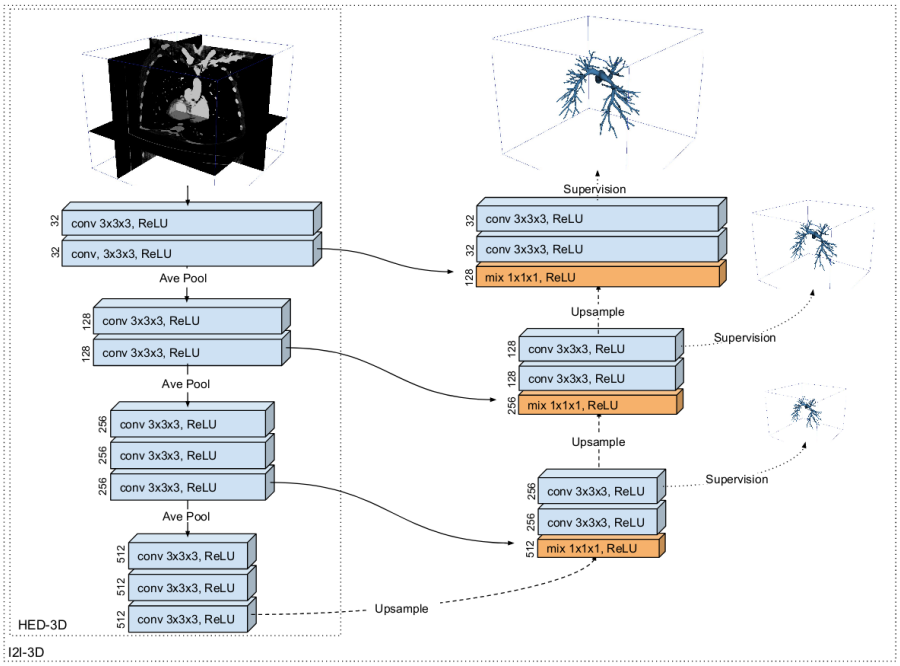

Figure 2.5: Image of I2I-3D network structure from *Dense Volume-to-Volume Vascular Boundary Detection* by Merkow et al. [5]. The filter size is written to the left of every filter layer. Deep supervision is applied at the end of every decoder level.

With I2I-3D having proved itself in vessel wall boundary detection, it was a natural choice to investigate its coronary artery segmentation potential. The development of a segmentation tool in this thesis was thus centered around this network architecture.

3
Data and Methods

This chapter will describe the data sets used in the project and how they were processed and used to train the model. The deep learning framework and the network implementation will also be described, as well as the training and evaluation processes.

3.1 Data

As described in Section 2.1, the data is what drives the training in deep learning and is thus extremely important. The network needs to generalise from the information it sees, meaning that the quality of the data can make or break the application. A common approach in deep learning is to preprocess the data before feeding it to the network, and to pretrain the network on different data sets before training on the main task. Even if the pretraining data sets are different to the main data set, they allow the network to start learning features, as described in Section 2.1.1.7. To virtually enlarge the data set it is also good practice to augment the data. This chapter serves to introduce the data sets, as well as the preprocessing and augmentation methods used.

Four data sets were used to train the network; three data sets for pretraining in addition to the main data set of manually segmented coronary arteries. All data sets were produced by CT scans, ensuring that the network does not encounter the extra difficulty of learning to handle images produced by different imaging modalities.

3.1.1 VISCERAL (VS) data set

A subset of the data from the study *Cloud-based evaluation of anatomical structure segmentation and landmark detection algorithms: VISCERAL anatomy benchmarks* was used for pretraining, here referred to as data set VS [28]. The study contains both MR and CT data with multiple segmented anatomical regions in the body, e.g. the sternum, the pancreas and the spleen. A subset of 20 CT volumes of the sternum were used in pretraining, due to their similarity in composition to the coronary artery images as they are images of the same anatomical region. These volumes had voxel dimensions of $0.82 \times 0.82 \times 1.5$ mm^3 An example image from the set is shown in Figure 3.1. The data set was split into two sets; a training set (17/20) and a validation set to evaluate the training (3/20).

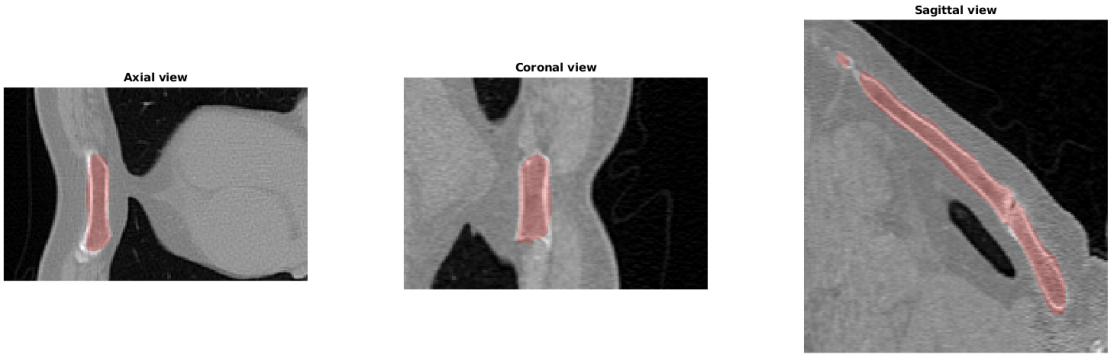

Figure 3.1: Example image volume from the VS data set, sliced along the three spatial axes. The corresponding segmentation of the sternum is marked in red.

3.1.2 SCAPIS pericardium (PS) data set

From the SCAPIS study, a data set of 30 manually delineated CTA volumes of the pericardium were used in pretraining [2, 29]. Voxel dimensions ranged between $0.32 \times 0.32 \times 0.30$ mm^3 and $0.43 \times 0.43 \times 0.30$ mm^3. The data was originally used to implement and evaluate a multi-atlas segmentation technique on pericardium segmentation. Due to being CTA volumes and heart segmentations from the SCAPIS study, they are the exact same type of images as the coronary artery images used in this project, but with a different corresponding segmentation; heart sac instead of coronary arteries. It should be noted that they are images of different patients than in the coronary artery segmentation data sets described below, meaning that there is no risk of the network ever having seen an image in the test set of the main data set during pretraining. This data set is referred to as PS and was split into a training set (25/30) and a validation set to evaluate the training (5/30). Slices from an image volume in the data set is shown in Figure 3.2.

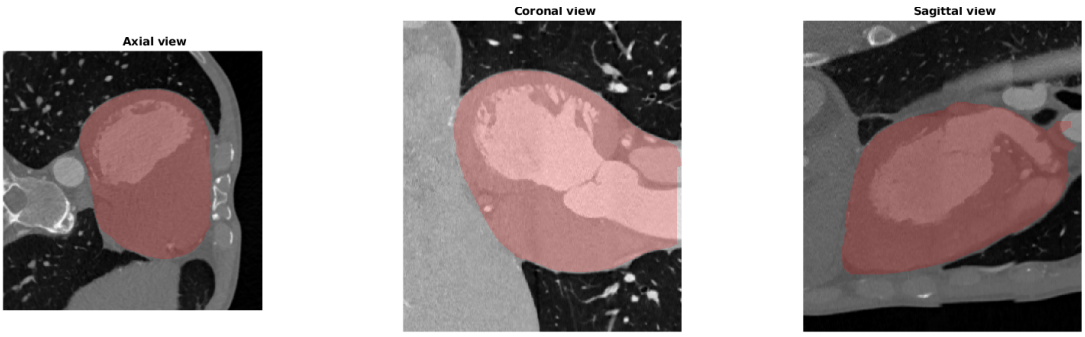

Figure 3.2: Example image volume from the PS data set, sliced along the three spatial axes. The corresponding segmentation is marked in red.

3.1.3 Medis automatic coronary artery segmentations (AS) data set

Automatically produced coronary artery segmentations from the software Medis QAngioCT Re working on the SCAPIS data were also used in pretraining, referred to as the AS data set [2, 3]. The data was of two different voxel sizes; $0.332 \times 0.332 \times 0.5$ mm^3 and $0.32 \times 0.32 \times 0.30$ mm^3. Medis QAngioCT Re segments all coronary arteries it can find, regardless of size. This means that small, clinically irrelevant vessels will be part of this data. The data set contains 77 CTA volumes with segmentations which have not been reviewed by medical professionals. Althogh not manually corrected, AS contains acceptable segmentations of the coronary artery tree for the task of helping the network learn to segment vessel like structures in pretraining. The classes are highly imbalanced in this data set, with coronary arteries making up only an average of around 0.1% of the image voxels. This imbalance can be seen in the example image in Figure 3.3. The data was split into three sets: a training set used for training (59/77), a validation set used to evaluate the performance in the training process (11/77), and a test set which was only used once during the final evaluation (7/77).

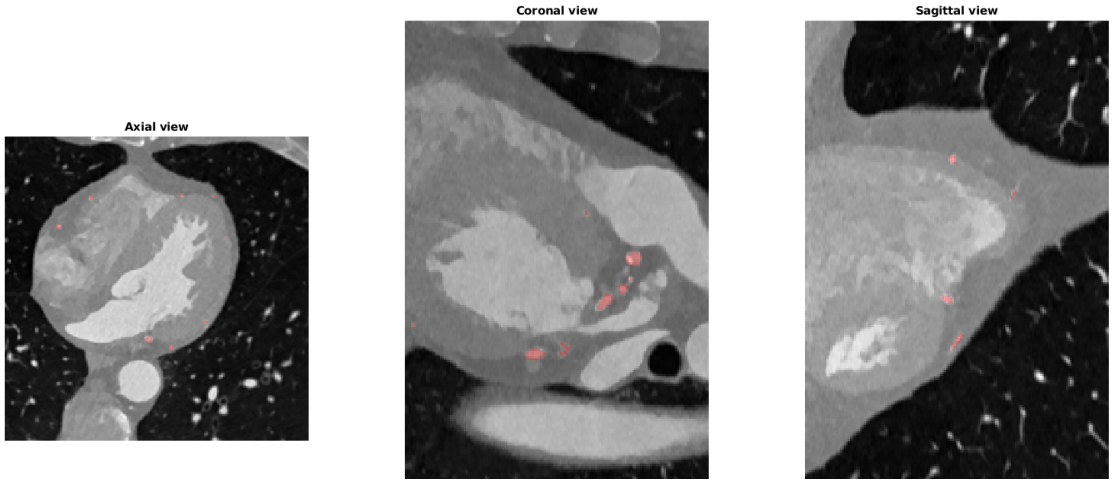

Figure 3.3: Example image volume from the AS data set, sliced along the three spatial axes. The corresponding segmentation is marked in red.

3.1.4 Manual coronary artery segmentations (MS) data set

The main data set for this study consists of 25 manually corrected segmentations from the Medis QAngioCT Re software performed on SCAPIS data. These manual corrections typically consist of small scale corrections of placing segmentation borders, as well as removal of small vessels far out in the coronary artery tree. Vessels with a diameter less than 2 mm are typically deemed clinically irrelevant and removed from the segmentation. Naturally, these images are similar to the AS data set, as can be seen in Figure 3.4. These volumes had the same voxel sizes as the AS data; $(0.332, 0.332, 0.5)$ mm and $(0.39, 0.39, 0.5)$ mm.

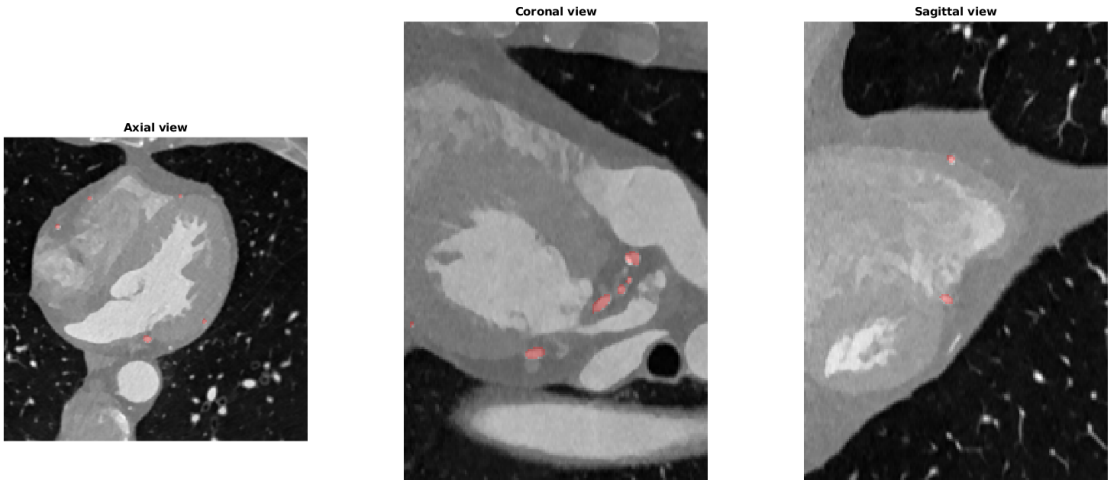

Figure 3.4: Example image volume from the AS data set, sliced along the three spatial axes. The corresponding segmentation is marked in red.

Each of the 25 CTA volumes were processed by two medical professionals, A and B, resulting in two sets of segmentations: MS-A and MS-B. The network trained on both MS-A and MS-B. The data was split into 3 sets: a training set (18/25), a validation set (4/25), and a test set for the final evaluation (3/25). Just like the AS data set, this data set also has a class imbalance with around 0.1% the image voxels being coronary arteries.

One should note that since medical professionals decide what vessels are interesting enough to be part of the segmentation, there is subjectivity involved in the manual corrections. As such, there is no guarantee that two medical professionals produce completely similar segmentations from the same image data. Over the 25 images, the mean Dice index similarity was 0.81 and the mean modified Hausdorff distance similarity was 2.34 between MS-A and MS-B.

3.1.5 Preprocessing

All data samples were normalised to have zero mean and unit variance. A large range of values in data will create a large range of values in the features extracted from the filtering processes in the network. Normalising all data to zero mean and unit variance is a way to keep the data consistent. In *Efficient Backprop*, LeCun et al. describe that this process facilitates faster convergence [30].

Due to CT images having different sample spacings in different spatial directions, the images are scaled to cube voxels. Having cube voxels means that moving one voxel in any direction represents the same real world distance travelled, no matter which direction is chosen, so that the image matrix represents the real world more closely. The information needed for this, the slice thickness (z-dimension) and the pixel spacing (x-y-dimensions) from the CT scan, is readily available in the DICOM header incorporated into every DICOM image file [31].

The VS data set images were small enough to be processed on the available GPU memory, being of size $176, 176, 128$. The PS data set was however too large and was therefore downscaled to a factor 0.39, to a size of $(200, 200, 176)$. For the AS and MS data sets, the original image size of $(512, 512, Z)$, where Z represents a variable number of slices, also proved too large to process. This meant that all images were shrunk to a maximum size of $(248, 248, \hat{Z})$, where $\hat{Z}$ had a maximum value of 184 due to the GPU memory restriction. Any value larger than 184 in the z-dimension resulted in a cut down to 184, equally from the top and bottom of the image. The image shrinking was achieved through downscaling to a factor of 0.5 followed by the removal of a 4 voxel wide frame in the x and y dimensions. No coronary artery vessels are present in this 4 voxel wide frame, so this removal should not affect the performance of the network.

3.1.6 Overview of data fed to network

To summarise; the network was fed data from four different data sets. Two of which are of the anatomical region surrounding the heart, but not of coronary artery segmentations, VS and PS, and two coronary artery segmentation data sets, AS and MS, created by an automatic software and manually by medical experts, respectively. All of the data was normalised to zero mean and unit variance and, if needed, shrunken down to a smaller size due to GPU memory limitations.

During training, full images were used for the VS and PS data sets, and both full images and patches from the images were used for the AS and MS data sets, with roughly 1/6 of the images used in full size and 5/6 used to create patches. Data augmentation was performed both on the full images and on the patches for the two latter data sets in order to prevent overfitting.

3.1.7 Data Augmentation

Data augmentation was performed *online*, meaning that it was done in the training loop. *Offline* data augmentation refers to augmentation done separately from training where the manipulated images are stored as an extra data set. With 3D images being quite large to begin with, this can become a problem. Another drawback is that the network will see the same images every epoch. The network may still be susceptible to overfitting because of this. With online augmentation, the data set is instead loaded into the training loop where it is manipulated with the methods described in Section 2.1.1.3 every iteration. Since angles of rotation, scales, and noise are chosen randomly, new transformations of the images will be used every epoch, ensuring that the network never sees the same image twice.

The transformations used on the data were small scalings, small rotations and additive noise. Implementation of noise and rotations are straight forward with Pythons SciPy library [32]. `numpy.rand.normal()` was used to generate both noise (mean 0, variance $3 \cdot 10^{-2}$) and small rotation angles (mean 0, variance 1), and the rotations were performed using `scipy.ndimage.interpolation.rotate()`. On the patches,

scaling was implemented by selecting different sized patches from the image. With $32 \times 32 \times 32$ acting as the standard patch size, a side length was chosen from a normal distribution with a mean of 32 and a variance of 3.2, with lower and upper limits set to 16 and 64 respectively. On the recommendation of Merkow et. al, only patches with at least 0.25% coronary artery voxels were used in training [5].

3.2 Implementation of I2I-3D

The focus of this thesis lies on the I2I-3D network described in Section 2.2.4. This section describes the implementation of I2I-3D in the deep learning framework PyTorch. A detailed description of the training process is included, with choices for initialisation, loss function, and optimiser.

3.2.1 PyTorch

PyTorch is a deep learning framework released on Jan 19 2017 [33]. At the time of writing this report, PyTorch is still in early-release beta but has already been adopted by the deep learning research community. A year after its beta release on Jan 12 2018, monthly arxiv.org framework mentions where 72 for PyTorch, compared to 273 for the widely popular TensorFlow, 100 for Keras, 94 for Caffe, and 53 for Theano, showing its quick rise in popularity [33].

PyTorch was created for deep Python integration and builds upon the scientific computation package Torch for Lua [34]. Unlike frameworks like Tensorflow which require the user to create a static graph of the network which is differentiated symbolically before any information is passed to it, PyTorch simply runs computations function by function, line by line, and dynamically constructs the network graph with each new computation. Although requiring that differentiation is performed every iteration, it offers the possibility to easily integrate new code into the graph and change the network structure dynamically and without compilation [35]. An additional benefit when debugging is that one only has to break through the Python abstractions, rather than both the Python abstractions and the abstractions of a static graph to find errors.

PyTorch supports NVIDIAs parallel computing platform, CUDA, for GPU processing. Loading networks and PyTorch tensors containing the inputs and outputs to a GPU is handled by the `cuda()` function available in all tensor and neural network objects. Using multiple GPUs for data parallelisation is initiated by passing the neural network model through the `torch.nn.DataParallell()` function. Both of these computation accelerators were used in training and evaluation.

3.2.2 Implementation of I2I-3D architecture

The I2I-3D network was implemented from scratch in PyTorch based on the article by Merkow et al. and their Caffe implementation available at GitHub [5, 36]. In

PyTorch, all neural networks are subclasses of the `torch.nn.Module` class. This class contains all the functions needed to build a network; fully connected layers, convolutional layers, activation functions, pooling layers, and so on. PyTorch also has a sequential container, `torch.nn.Sequential`, which builds a block of the neural network from multiple functions connected in the order they are added to the sequential container. For example, the line of code below will create a neural network block consisting of two layers of $(5, 5)$ convolution filters acting on a single layer input, followed by a ReLU activation function.

```
torch.nn.Sequential(torch.nn.Conv2d(1, 2, (5,5)), torch.nn.ReLU())
```

The sequential block was used to build each level in the encoder and the decoder of the I2I-3D network. A snippet of the code, containing the class construction and the first level of the encoder is shown below.

```
import torch
import torch.nn as nn

class I2I3D(nn.Module):
  def __init__(self):
    super(I2I3D, self).__init__()

    self.encoder_lvl_1 = nn.Sequential(
      nn.Conv3d(1, 4, (3,3,3), padding=1),
      nn.ReLU(),
      nn.Conv3d(4, 4, (3,3,3), padding=1),
      nn.ReLU()
      )
```

The code shows the creation of the `I2I3D` class, as well as the construction of the first encoder level, with two sets of four filter layers using filters of size $(3, 3, 3)$ with zero padding to keep the dimensions unaltered, each followed by ReLU activation functions. The structure of the network is specified in the `__init__` function, meaning that it is created whenever an instance of the `I2I3D` class is created. The flow of information has to be specified in a function named `forward` within the class, showing how the network output is formed from the input, using the created blocks from `__init__`. For example, if the `I2I3D` class only had the one layer shown above, the `forward` function might look like the following:

```
  def forward(self, x):
    output = self.encoder_lvl_1(x)
    return output
```

The encoder and decoder blocks, as well as the mixing layers for interconnections and upscaling layers for deep supervision outputs were created in the `__init__()` function and connected in the `forward()` function. To create probabilistic outputs, where each voxel has a probability of being a coronary artery, representing the network's certainty in its classification, a sigmoid was added to the outputs. This normalises the voxel values to the interval $(0, 1)$ which is easily interpreted. A voxel

value larger than 0.5 was determined as coronary artery in the binarisation process to create a segmentation.

One change to the original network architecture is the filter sizes used in the network. Merkow et al. suggest using 32, 128, 256 and 512 filter layers in the four levels. Due to memory limitations of the GPUs, these were changed to 4, 8, 16, and 32 layers, which represents a significantly lower amount of network parameters. One attempt to tackle memory limitations was done by converting parameters and data to floating point 16 instead of floating point 32 during training through a method known as *mixed precision training*. Unfortunately, this attempt was ultimately unsuccessful. For more information on mixed precision training the reader is referred to an article by NVIDIA [37].

In total, the implemented version of I2I-3D network consists of 49 layers, counting convolution and ReLU layers separately and including the 3 upscaling and/or flattening layers, and a total of 113,185 trainable parameters. A schematic illustration of the network structure is shown in Figure 3.5, where filter sizes are written on the side of every filter layer. The flattening layers are used for the deep supervision outputs to create single-layered outputs from the 8 and 16 layered decoder outputs. The full code for the network can be found on GitHub[1].

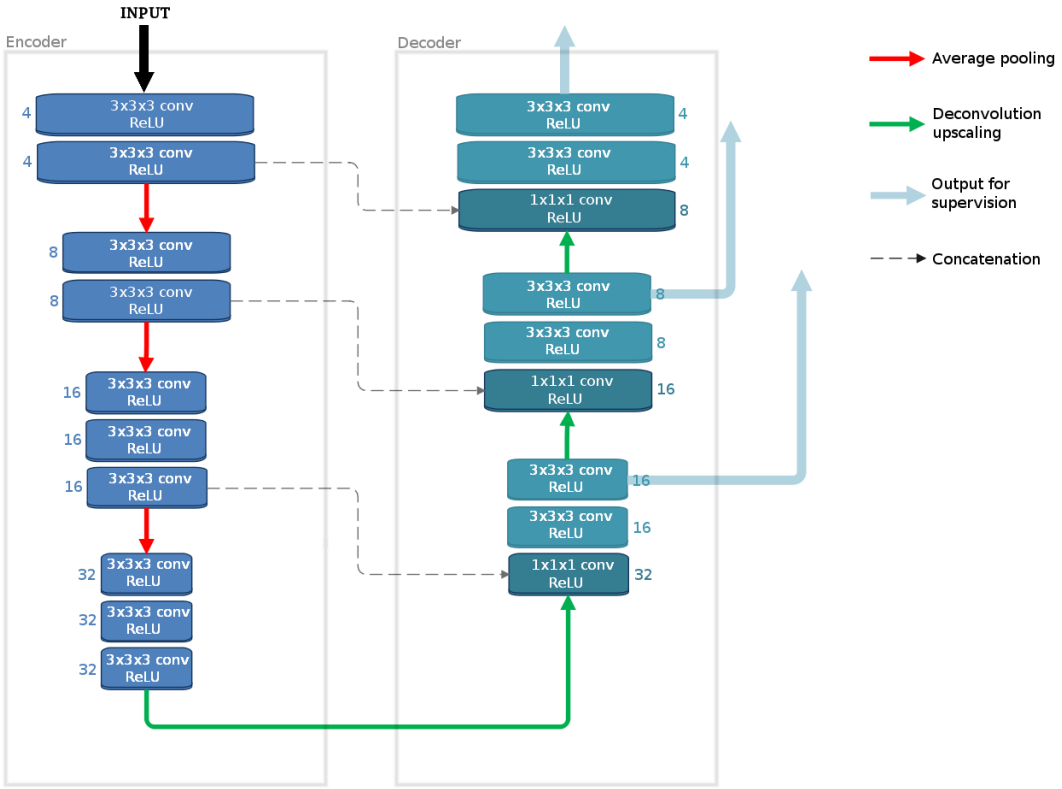

Figure 3.5: The I2I-3D network archtitecture implemented in PyTorch for coronary artery segmentation.

[1] https://github.com/frefdrk/i2i3d-pytorch

3.2.3 Training I2I-3D

Moving from just a network structure to a useful model requires training. A training loop in PyTorch requires the specification of a loss function and an optimiser. Choice of loss function, optimiser as well as the optional initialisation process are covered in upcoming sections. Assuming these choices have been made, example code showing the simple fashion behind a PyTorch training loop is shown below.

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.autograd import Variable

network = I2I3D()                    # Construct a network from the class created above
network = nn.DataParallell(network) # These two lines enable
network.cuda()                       # parallell GPU computing

lossFunction = nn.BCELoss()          # Binary cross entropy selected as loss
optimiser = optim.Adam(network.parameters())    # Adam chosen as optimser, gets fed with
                                                # the network parameters
for data in dataset:
    image, seg = data                          # Extract image and segmentation from data
    output = network(Variable(image).cuda()) # Load to GPU, produce output

    loss = lossFunction(output, Variable(seg))      # Calculate loss
    optimiser.zero_grad()                           # Reset optimiser to not keep
                                                    # gradients from previous iteration
    loss.backward()                  # Backpropagate from loss
    optimiser.step()                 # Perform the update
```

After the network has been initialised, one can simply pass the data through it, calculate the loss and update the parameters with a few lines of code. Naturally, with the addition of data augmentation in the loop and preprocessing before it, the code can become more complex. Still, the core of training a neural network in PyTorch is encapsulated in the code above.

3.2.3.1 Initialisation of network parameters

Initialisation of network parameters can be performed in a multitude of ways. For this thesis, initialisation follows the method proposed by Xavier Giorot et al., suitably named *Xavier initialisation* [38]. This initialisation is chosen based on the fact that Merkow et al. used it in their original implementation of the I2I-3D network [36]. With Xavier initialisation, biases are initialised to be 0 and weights, w, are initialised with values sampled from a distribution according to Equation 3.1

$$w_l \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_{l-1}+n_{l+1}}}, \frac{\sqrt{6}}{\sqrt{n_{l-1}+n_{l+1}}}\right] \tag{3.1}$$

where $U[a, b]$ is the uniform distribution in the interval (a, b), l represents the current layer and n_l is the number of parameters in layer l. Clearly, the distribution has mean 0. The variance of a uniform distribution X, $\text{Var}[X]$, in the interval (a, b) is

$$\text{Var}[X] = \frac{(b-a)^2}{12}$$

which gives that

$$\text{Var}[U] = \frac{1}{\frac{n_{l-1}+n_{l+1}}{2}}$$

meaning that the variance of the weights is not dependent on the size of the layer they belong to, n_l, but rather on the average size of the previous and next layer, (n_{l-1} and n_{l+1}).

3.2.3.2 Loss function and deep supervision

Binary cross entropy (BCE), described in Section 2.1.1.5, was chosen as loss function due to its popularity in binary segmentation tasks. Due to deep supervision integration, the I2I-3D network has three outputs from which to calculate a loss as visualised in Figure 3.5, with two deep outputs that get flattened and upscaled in addition to the standard network output. The BCE loss is calculated for each output and summed together to create the total loss for the network. Using the definition of BCE from Equation 2.6, the total loss L becomes

$$L = -\sum_{k=1}^{3}\sum_{n=1}^{N} o_n^* \cdot \log(o_n^{(k)}) + (1 - o_n^*) \cdot \log(1 - o_n^{(k)}) \tag{3.2}$$

where superscript k denotes the outputs from the network.

3.2.3.3 Optimiser

The role of the optimiser is to update the parameters of the network based on the gradients, as described in Section 2.1.1.1 with gradient descent. In reality, *stochastic gradient descent* (SGD) is used in favour of regular gradient descent, due to it being faster and computationally cheaper. Gradient descent requires the gradients to be calculated for the entire training data set for each update. SGD allows for updates after every data sample, which is much more efficient.

While reliable, SGD requires careful choice of hyperparameters to tune the algorithm. An optimal learning rate with or without decay, momentum which acts as a memory that keeps part of the previous update term in the next iteration, and weight decay which reduces the size of the weights by a chosen factor each iteration have to be set manually and greatly affect the outcome of the optimisation.

3.2.3.4 Training algorithm

We can now combine the previously described topics into an actual algorithm for the training process of a neural network. A step by step description follows below.

1. Preprocess the data to be used
2. Initialise the network
3. Select and augment a sample or subset of the data
4. Feed data to network and produce output
5. Calculate loss between output and ground truth data
6. Backpropagate gradients from the loss
7. Update parameters based on gradients with optimiser
8. Repeat from step 3 until stopping criteria is met

This process continues for a chosen number of epochs or until a certain criteria is met, for example reaching a plateau of the loss or some other measure on the validation data set. The latter criteria was used in this project, where a plateau of Dice index decided when to stop training.

The training of the I2I-3D network was sectioned into two parts, both following the steps above. The first part implemented the loss function for deep supervision from Equation 2.7 and continued until the Dice index plateaued. The second part continued from the first but with the deep supervision removed, only calculating the loss from the top-most output of the network, once again until Dice index plateaued.

3.3 Evaluation

To judge whether a deep learning approach is viable for coronary artery segmentation, the I2I-3D network was tested against the Medis QAngioCT Re software and the medical professionals. On each of the two medical professionals' segmentations, similarity is calculated between it and (1) the deep learning segmentation, (2) the Medis QAngioCT Re segmentation and (3) the other professional manual segmentation. This gives an assessment of whether or not the deep learning segmentations exhibit better conformity to the manual segmentations than the automatic software or the medical professionals themselves. Similarity is measured by two metrics: Dice index and modified Hausdorff distance, which are both good similarity measures when working with unbalanced classes, and the latter being especially suited for cases where the shape of the segmentation is of importance, as in this thesis [39].

Dice index (DI) is a commonly used similarity measure in medical image segmentation [40, 41, 42]. It is also known as the *overlap index*, with formulation shown below [39]:

$$DI = \frac{2TP}{2TP + FP + FN}.$$

Here, TP stands for *True Positives* which is the number of voxels correctly classified as coronary arteries, FP stands for *False Positives* which is the number of voxels in-

correctly classified as coronary arteries, and FN stands for *False Negatives* which is the number of voxels incorrectly classified as non-coronary arteries. With DI taking all of this into account, it penalises segmentations that are either too inclusive or too exclusive. It gives a much better measure of performance than simply counting the number of correct voxel segmenations. Due to the imbalance of classes, with only 0.1% coronary arteries, a network that would produce an output of only zeros (representing the non-coronary artery class) would have 99.9% of voxels correctly classified, seeming like good performance while actually failing completely to segment the arteries. On the other hand, that same network's DI would be 0, which is completely in line with the performance. However, one shortcoming of DI is the lack of consideration to spatial position. If a segmentation result is wrong by consistently being one voxel distance off the true segmentation, it would not improve the score in any way compared to a segmentation result that is wrong by being consistently off by a hundred voxels. There is no relativeness in DI, only whether each voxel is correct or not is considered. To complement this measure, a *distance metric* called the modified Hausdorff distance is also used.

Instead of counting voxels, the modified Hausdorff distance gives a measure of how close in space the edge points of the respective segmentations lie. Given two sets of points, $\mathbf{X} = \{x_1, ..., x_{N_x}\}$ and $\mathbf{Y} = \{y_1, ..., y_{N_y}\}$, the modified Hausdorff distance (mHD) is given by:

$$mHD = \max\Big(d(\mathbf{X}, \mathbf{Y}), d(\mathbf{Y}, \mathbf{X})\Big)$$
$$d(\mathbf{X}, \mathbf{Y}) = \frac{1}{N_x} \sum_{x \in \mathbf{X}} \min_{y \in \mathbf{Y}} ||x - y||$$

where $||x - y||$ refers to the Euclidean distance [43]. It measures the average distance from each point in set $\mathbf{X}$ to its closest point in set $\mathbf{Y}$, as well as the other way around, and returns the largest of the two averages as the similarity measure. A low mHD value thus represents high similarity.

For the evaluation, the network output was produced as described in Section 3.2.2. After binarising the network output by thresholding at 0.5 to construct a segmentation, the output was post-processed by manually editing the segmentation in a 20 voxel wide outer frame in the x and y dimensions to the non-coronary artery class. This was done due to the training data not having any coronary arteries in this region. The reason for this was twofold; not including this region frees up memory in the GPU during training, and prevents misclassification in this region as long as there are no coronary arteries present.

4
Results

This chapter presents the results of this study. It covers the speed of the network, the outcome of the training and, most importantly, the segmentation performance on coronary arteries in cardiac CTA compared to Medis QAngioCT Re and manual segmentations by medical professionals.

4.1 Run times

The results were produced on a computer with a NVIDIA GeForce GTX TITAN X GPU with 12 GB of memory and a NVIDIA GeForce GTX 1070 GPU with 8 GB of memory running on Ubuntu 16.04. Segmentation of an image of size $(256, 256, 192)$ took 0.4 seconds on the TITAN X and 1.7 seconds on the 1070. Training for an epoch took 28.6 seconds on the VS data set, 127.6 seconds on the PS data set, 339.7 seconds on the AS data set, and 182.0 seconds on the MS data set, using both GPUs in parallel.

4.2 Network training

During all stages of training, the learning rate was set to 10^{-4} as this proved to be large enough to provide a relatively fast learning process, after tests starting from a learning rate of 10^{-7}. Decreasing the learning rate during later epochs was handled automatically by the Adam optimiser. No weight decay was used, due to early testing on the AS data set showing better performance without it. The training on the AS and MS data sets was performed with both image patches and full size images, after early testing showed that this was advantageous. The data serving as foundation for these decisions is shown in Figure 4.1.

The I2I-3D network was trained for a thousand epochs on each of the pretraining data sets; VS, PS and AS. The segmentation performance on each of the data sets before and after training for the training and validation set respectively is listed in Tables 4.1 and 4.2. The tables show the performance measured in Dice index, precision, and recall.

These tables show large increases in all performance metrics after training for the VS and PS data sets, and significantly smaller increases for the two data sets of coronary arteries, AS and MS, indicating that learning to segment coronary arteries

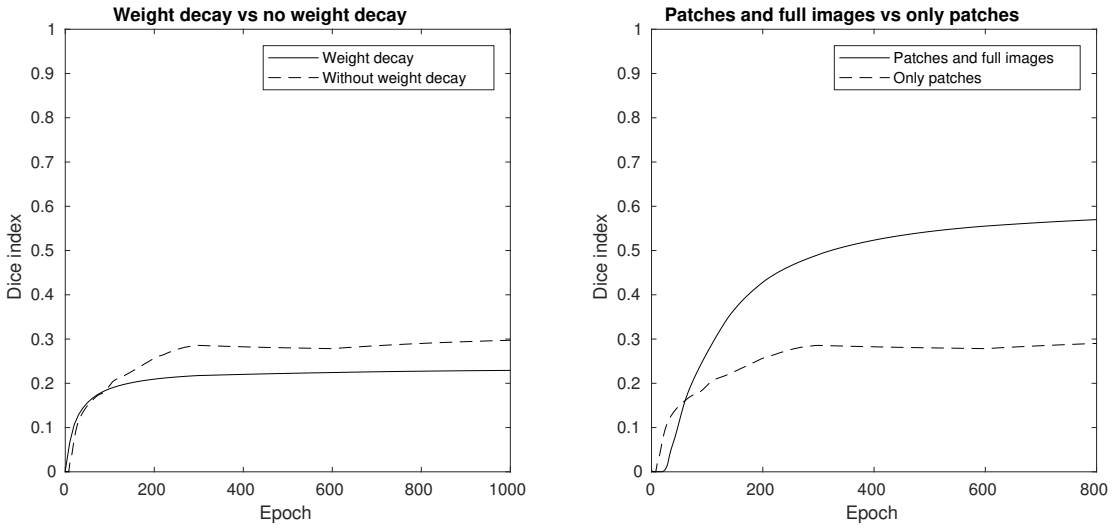

Figure 4.1: Plots of segmentation performance for training with and without weight decay (left) as well as training on only patches and on both patches and full images (right) on the AS data set. Using weight decay resulted in decreased performance, whereas training on both patches and full sized images resulted in a large improvement.

Table 4.1: Training performance on training data. Table shows how the training affected performance on each of the datasets, where Dice index (DI) before and after training are listed together with precision and recall. The network column shows the training in chronological order, where "post XX" refers to a completed training on dataset XX.

Network	Dataset	DI	Precision	Recall
Initialised	VS	0.03	0.01	0.41
post VS	VS	0.92	0.94	0.90
post VS	PS	0.00	0.22	0.00
post PS	PS	0.97	0.98	0.97
post PS	AS	0.01	0.00	0.71
post AS	AS	0.70	0.66	0.75
post AS	MS	0.50	0.46	0.59
post MS	MS	0.64	0.59	0.72

is a tough problem. Note that after training on VS, the performance on the VS validation data set is lower than on the VS training data set. This is a sign of overfitting, where the network is tuned too well to the training data. On the PS data set, the metrics are consistantly high both for the training and validation set, indicating that the I2I-3D structure is apt at heart sac segmentation. Another interesting fact is that performance on the MS data set is significantly better on the validation data than on the training data. With only four images in the MS validation data set, this effect may be due to those images being "easy" cases that do not include hard to learn features that may be present in the training set. This behaviour is also visible for the AS data set, but to a lesser degree.

Table 4.2: Training performance on validation data. Table shows how the training affected performance on each of the datasets, where where Dice index (DI) before and after training are listed together with precision and recall. The network column shows the training in chronological order, where "post XX" refers to a completed training on dataset XX.

Network	Dataset	DI	Precision	Recall
Initialised	VS	0.03	0.01	0.99
post VS	VS	0.81	0.84	0.80
post VS	PS	0.01	0.4	0.01
post PS	PS	0.97	0.98	0.96
post PS	AS	0.01	0.00	0.74
post AS	AS	0.78	0.78	0.78
post AS	MS	0.61	0.58	0.64
post MS	MS	0.74	0.71	0.79

The AS and MS data sets, which are both coronary artery segmentations, are of special interest to the study, and were thus investigated in greater detail. The mean DI on the validation data set during pretraining on AS is shown in the left plot of Figure 4.2. The plot indicates strong performance in vessel segmentation, as the DI reaches a value of 0.78. The jumps visible in the left plot at epoch 350 and 750 correspond to reinitialisations of the optimiser after continuation from a previously completed training session. Adam changes learning rates individually for each parameter during training, and the reinitialisation will reset these learning rates back to 10^{-4} which might explain this behaviour. The reset might have increased the learning rate enough for the system to leave a local minimum in the next iteration.

On the manually segmented training set the network trained 500 epochs with deep supervision and 500 epochs without deep supervision. The best performing network training on the full MS training data set was obtained after 70 deep supervision epochs and no training without deep supervision, which is what the segmentation results are based on. A graph showing the DI on the test set for the first 300 epochs can be seen in the right plot of Figure 4.2. Notice the starting DI being high at around 0.6, hinting at a successful pretraining where the network learnt to segment vessel structures well. Due to this, the network only had to fine tune its parameters to the manually segmented data rather than having to learn from scratch.

4.3 Segmentation performance for test data

4.3.1 Pretraining segmentation results

On the AS test data set containing 7 images, the pretrained I2I-3D network achieved 0.81 in mean DI and 3.37 in mean mHD. Segmentations for two of the 7 test images are shown in Figure 4.3. Aside from some unconnected islands of labellings, the

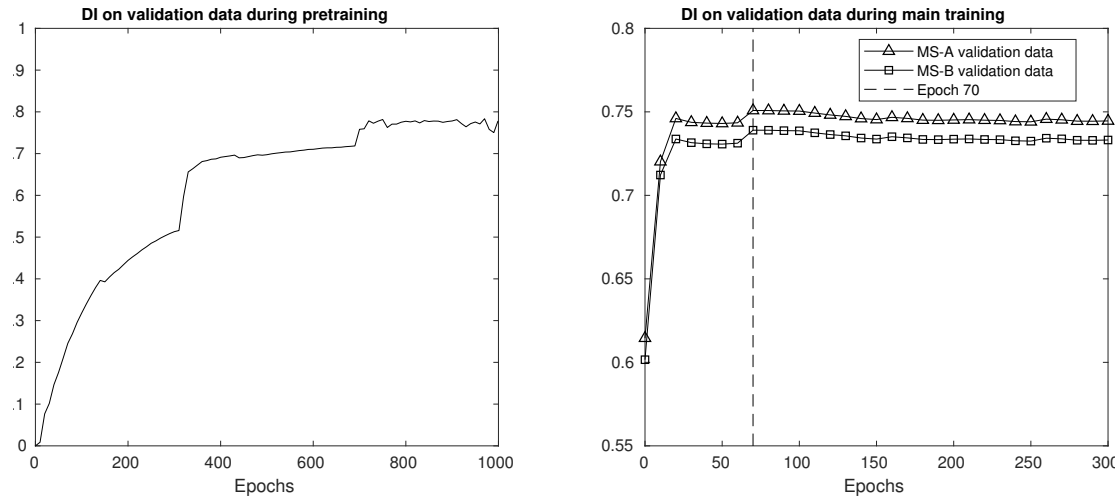

Figure 4.2: DI on validation data during pretraining on the AS data set (left) and main training on the MS data set (right). The final DI on the validation set in pretraining was 0.78. During main training on manual segmentations, peak performance was achieved at epoch 70.

network segmentations greatly resemble those from Medis QAngioCT Re. The network being able to segment vessels to this extent explains the high DI at the start of training on the main data set shown in the right plot of Figure 4.2.

4.3.2 Final segmentation results

To compare the segmentation performance of the different methods, similarity was measured between each of them and the manual segmentations, MS-A and MS-B, and the manual segmentations were also compared to each other. This was done to evaluate how the deep learning based method compares to the software used by professionals today as well as to how coherent the professionals are, also known as the *interobserver* variability.

Segmentations from one of the three cases are shown in Figure 4.4. Both the segmentations from the network only pretrained on VS, PS and AS data set, referred to as I2I-3D pretrained, and from the network trained on VS, PS, AS and the manually segmented data set MS (both MS-A and MS-B), referred to as I2I-3D fully trained, are shown. Segmentations for each of the methods on all three evaluation cases can be found in Appendix A.

From the 77 training images used to pretrain the network on vessel segmentation, the I2I-3D pretrained network performs admirably in mimicking Medis QAngioCT Re. Further, with the 25 manual segmentations in the main training the network learns to disregard small vessel structures, although not completely successfully, and therefore produces segmentations that resemble the manual segmentations to a higher degree than I2I-3D pretrained. From a visual inspection, it is clear that the incorporation of the small vessels is the largest source of dissimilarity between the

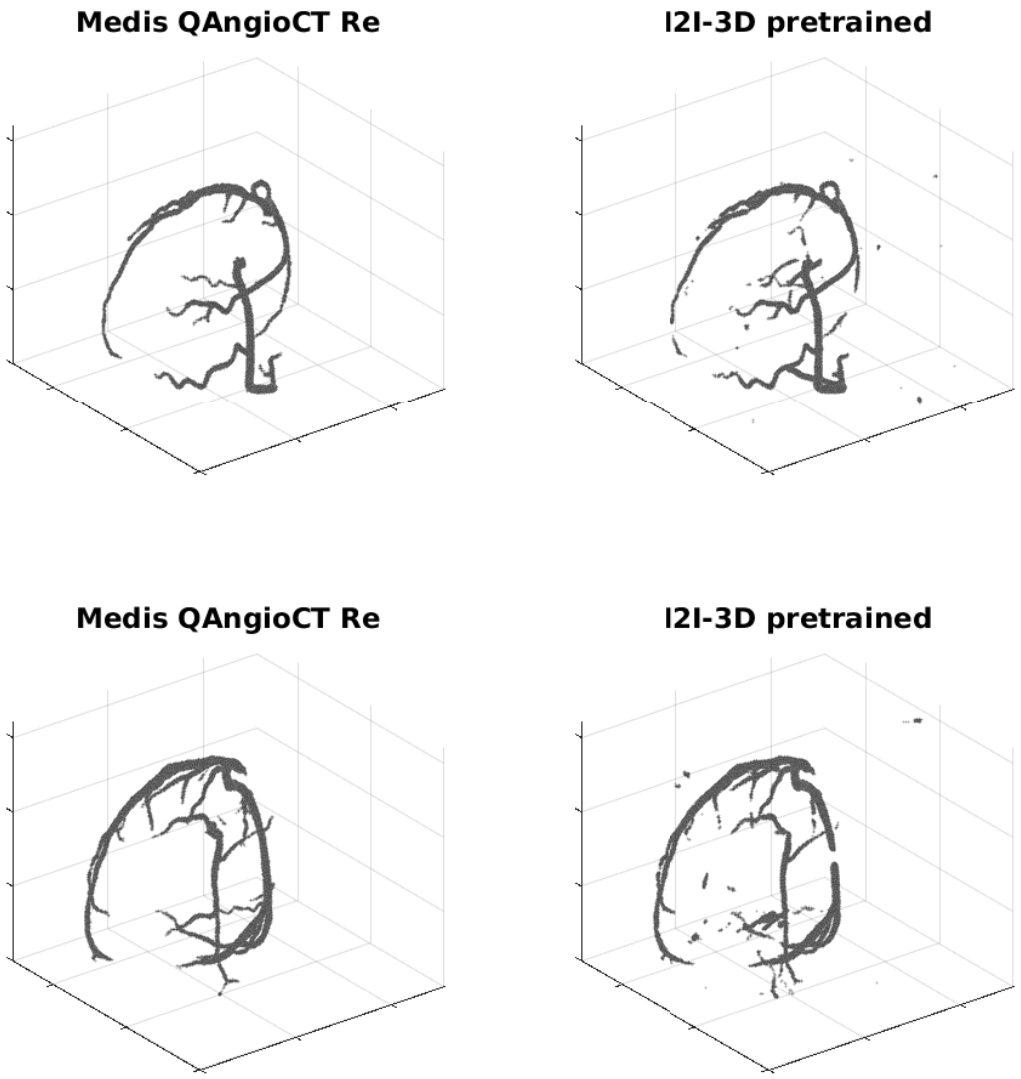

Figure 4.3: Segmentations for two volumes in the test data set of AS from Medis QAngioCT Re and the pretrained I2I-3D. The rows correspond to the two volumes, with the left figures being the Medis segmentations and the right figures being the pretrained I2I-3D segmentations.

manual segmentations and the I2I-3D pretrained and Medis QAngioCT Re.

Moving on to a comparison on a smaller scale, there is generally no significant difference in where the segmentation borders are placed between the manual segmentations, Medis QAngioCT Re and the fully trained I2I-3D, as exemplified by Figure 4.5, aside from how much of the aorta is part of the segmentation, i.e. the area in the bottom right corner of the images. A small difference that is visible is that Medis QAngioCT Re seems to favour a smaller vessel diameter, with more bright pixels outside of the red segmentation than the others, whereas I2I-3D is

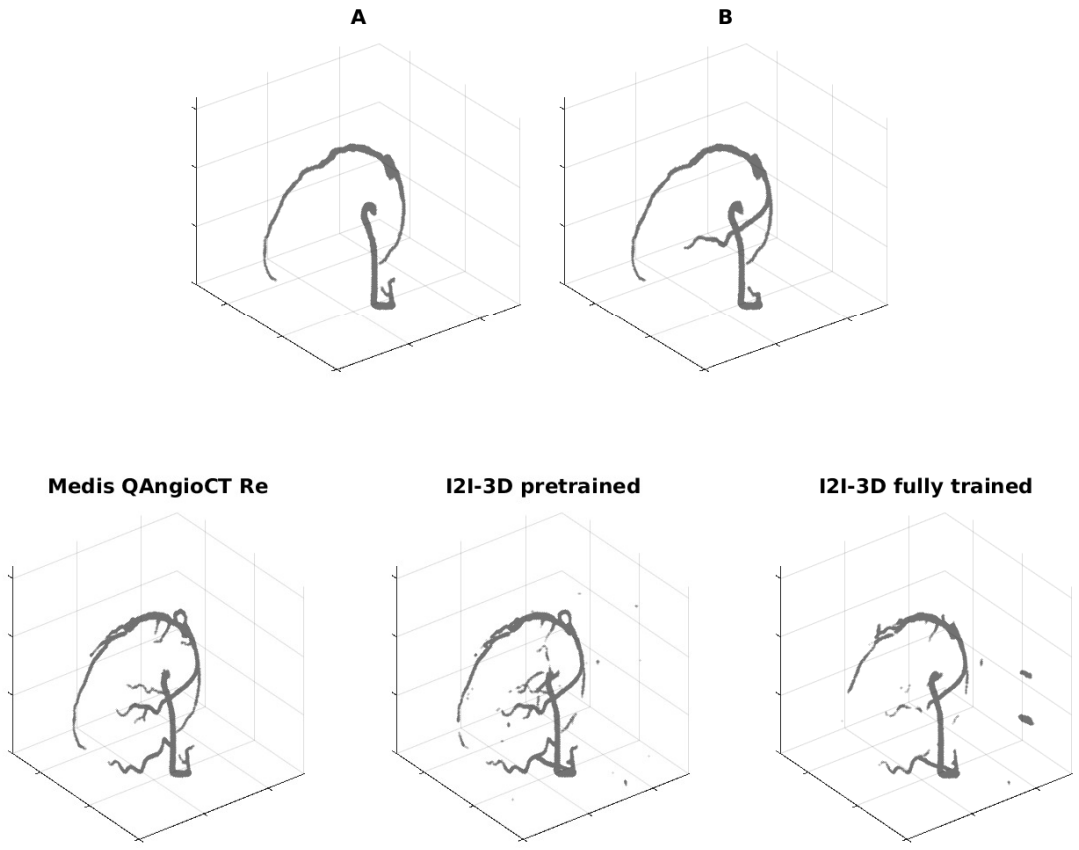

Figure 4.4: Segmentations on one volume in the test data from each of the segmentation methods. The top left figure shows the MS-A segmentation, the top right shows the MS-B segmentation, the bottom left shows the Medis segmentation, and the bottom middle and right figures show the pretrained and fully trained I2I-3D segmentations respectively. The fully trained I2I-3D segmentation contains some unconnected blobs, clearly not part of the coronary artery tree, but does a better job of resembling the two manual segmentations than those from the pretrained network and Medis QAngioCT Re. This is largely due to Medis QAngioCT Re not excluding small vessel from the segmentation.

more in line with the medical professionals. While this is generally only a small difference, it is significantly larger in some regions of the segmentation resulting in a quite discernible difference between the automatic methods, as shown in Figure 4.6. In this case, I2I-3D is much more in line with the medical professionals than Medis QAngioCT Re.

The segmentation performance measured by the two similarity metrics, represented by the mean over the three images, are listed in Table 4.3 for the tested methods on both data sets, MS-A and MS-B. Performance metrics on both MS-A and MS-B are also visualised in Figures 4.7 and 4.8 respectively. These metrics tell the same story as the visual inspection; the pretrained I2I-3D and Medis QAngioCT Re perform close to identically, while the fully trained network achieves significantly better

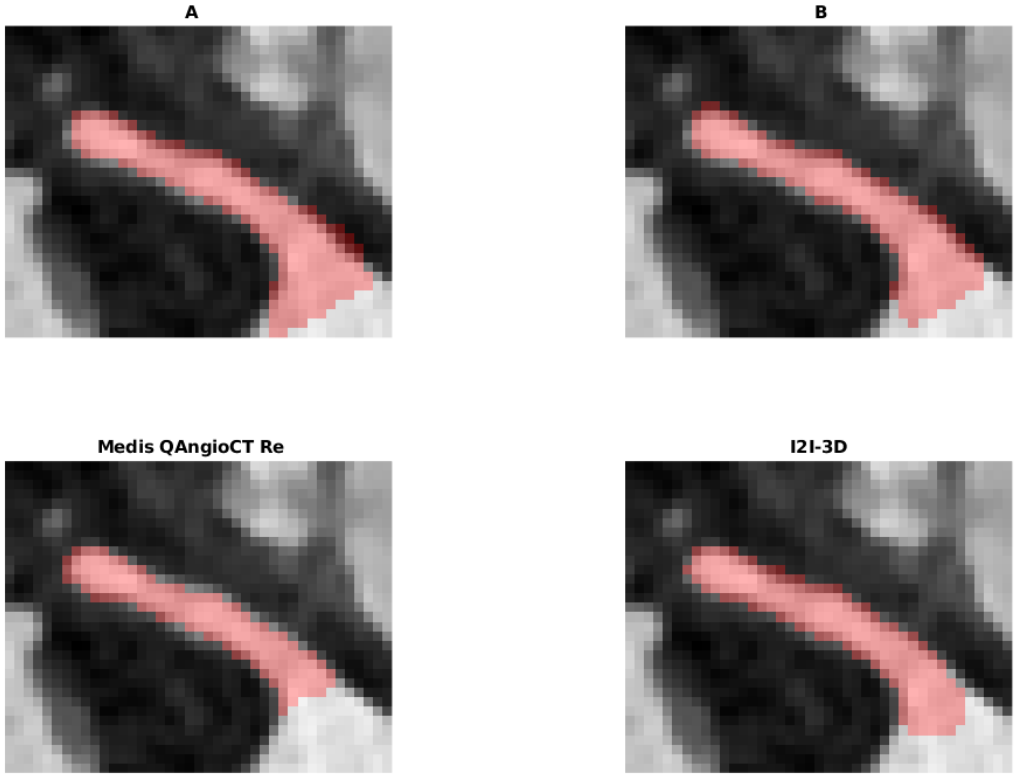

Figure 4.5: Image highlighting segmentation border placement for segmentations from professional A (top left), professional B (top right), Medis QAngioCT Re (bottom left), and the fully trained I2I-3D (bottom right). Area marked in red represents the segmentation.

similarity metrics. Over the three test images, the two professionals (i.e. the interobserver variability) achieve the highest similarity in both similarity measures. The fully trained I2I-3D network shows better performance and less variation than Medis QAngioCT Re on both MS-A and MS-B.

Table 4.3: Similarity metrics measuring segmentation performance. A high DI as well as a low mHD are desired. The I2I-3D network can not compete with the interobserver agreement, but outperforms the non-deep learning based automatic software in similarity to both medical professionals.

Segmentation method	**MS-A**		**MS-B**	
	DI	mHD	DI	mHD
Professional A	-	-	**0.89**	**2.18**
Professional B	**0.89**	**2.18**	-	-
I2I-3D fully trained	0.73	3.51	0.74	3.47
I2I-3D pretrained	0.57	4.63	0.58	4.57
Medis QAngioCT Re	0.58	4.69	0.60	4.64

In addition to the fully trained I2I-3D, two networks trained solely on segmentations from one of the professionals were also implemented, i.e. only having trained on MS-

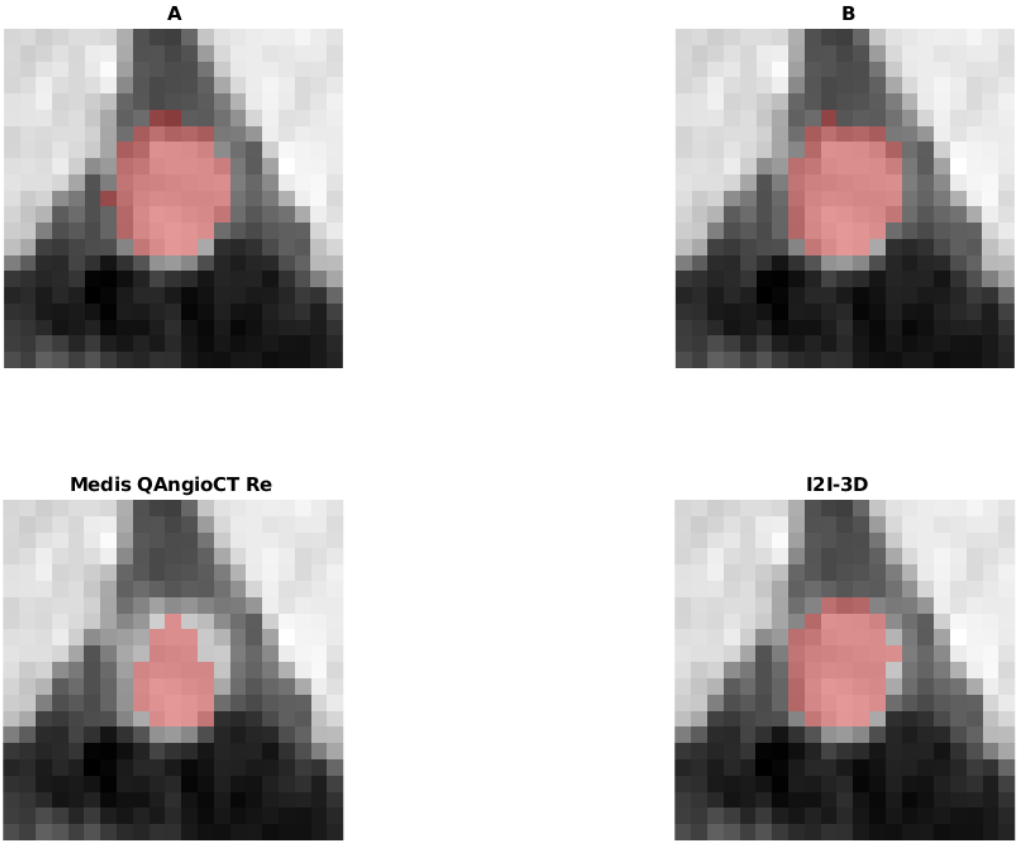

Figure 4.6: Image highlighting segmentation border placement for segmentations from professional A (top left), professional B (top right), Medis QAngioCT Re (bottom left), and the fully trained I2I-3D (bottom right). Area marked in red represents the segmentation.

A and MS-B respectively. However, neither of these produced results on par with the fully trained I2I-3D, and they are thus left out of the comparison.

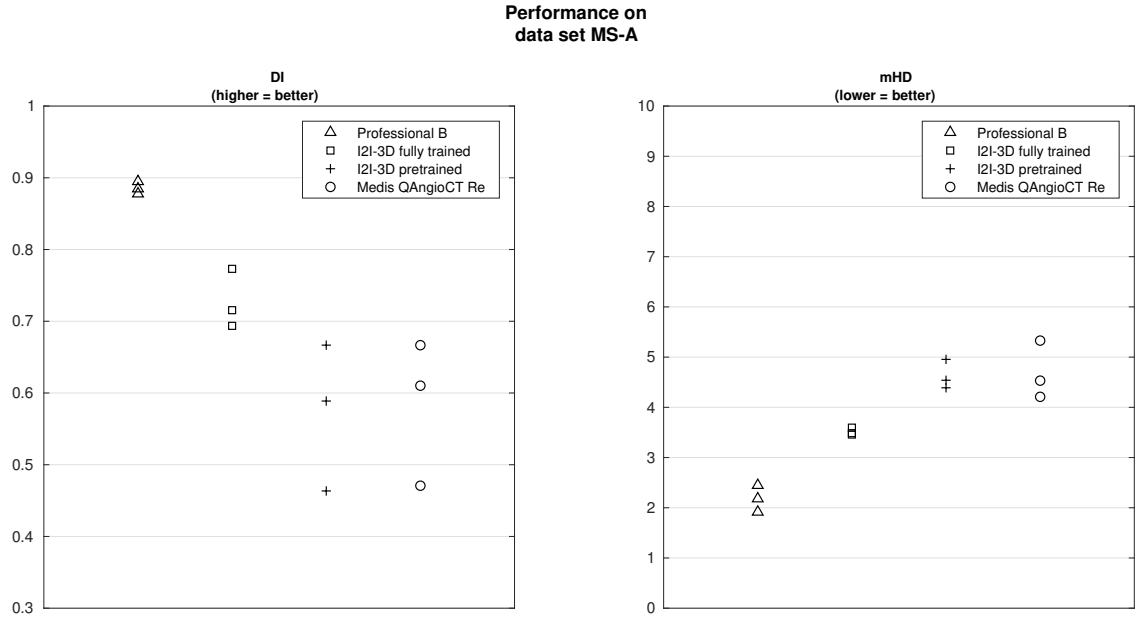

Figure 4.7: Performance of segmentation methods on the three images in data set MS-A. Professional B shows both the best performance and the least variation in DI. In mHD, B has the best performance, but I2I-3D shows the least variation. While professional B achieves the best results, the fully trained I2I-3D does outperform Medis QAngioCT Re, whereas the pretrained I2I-3D is close to Medis QAngioCT Re's performance.

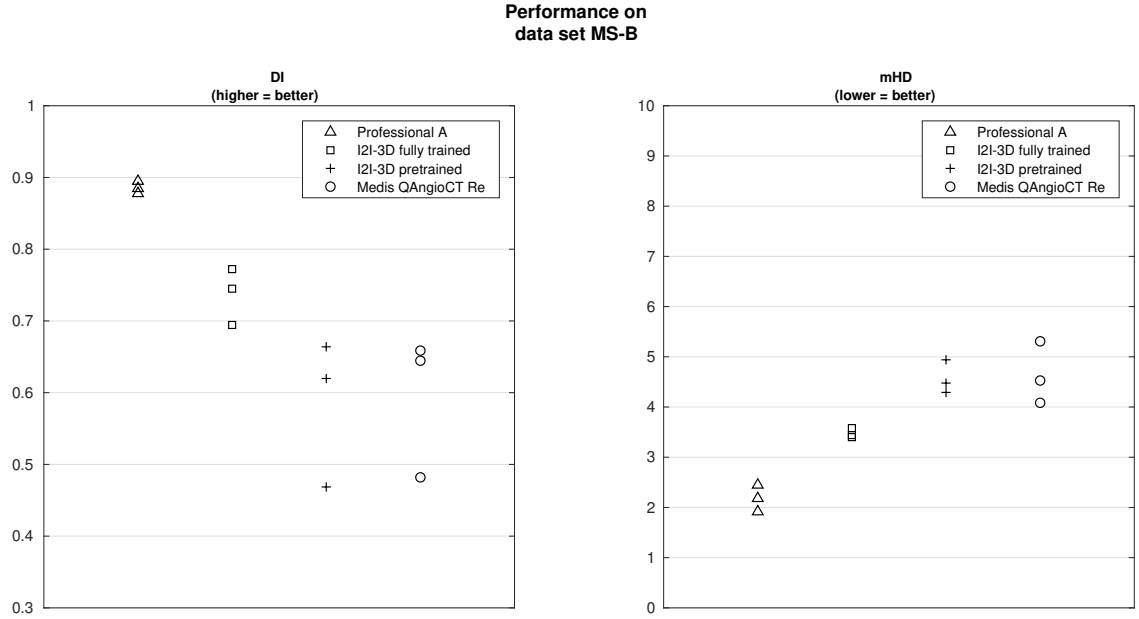

Figure 4.8: Performance of segmentation methods on the three images in data set MS-B. Once again, the professional shows the best segmentation performance, followed by the I2I-3D network. The fully trained network shows better performance and less variation than the Medis QAngioCT Re software. The pretrained I2I-3D is close to Medis QAngioCT Re in performance.

5
Discussion

5.1 Analysis of method

This study investigated the coronary artery segmentation performance of a version of the I2I-3D network implemented in PyTorch. By building the network from scratch in PyTorch rather than using the Caffe implementation provided by Merkow et al., a significant amount of time was consumed by implementing and testing the network. Had a different methodology been chosen, for example starting from the Caffe implementation, this project could most likely have reached further in the investigation of the network's potential. There could also have been time to investigate improvements to the architecture of the network. That said, there is also value in providing an implementation of the network in another framework[1], expanding its availability for further research. With the network showing promising performance, that value should not be neglected.

A limiting factor of this project is the relatively small data sets. With a test set of only three images in the main data set, one could certainly question the weight of the results. A clear indication that the data sets were too small was shown in Tables 4.1 and 4.2, where the performance was better on the AS and MS validation sets than on the respective training sets. Since the network has trained on the training set but never seen the validation images, one expects the performance to be higher on the training set than on the validation set. That fact that it is not may be an indication that some hard to learn features in the training sets were not represented in the validation sets. To do the evaluation thoroughly would thus require a larger data set. With the SCAPIS study progressing, that will become a reality in time. Working with medical data, as that provided by SCAPIS, requires that ethical concerns are taken into consideration. It should not be possible to connect any data sample to an individual taking part in the study. As such, the data used in this project was stripped of identification tags in information headers.

Another data set that could have been used as either another pretraining data set or another evaluation set is from the Rotterdam Coronary Artery Algorithm Evaluation Framework which provides data to compare methods for coronary artery segmentation and stenoses detection [23]. Using this data would have been the next step in the investigation into I2I-3D's potential, given more time. With facts in hand, that would have been a better use of time than that spent trying to implement mixed

[1] Available at `https://github.com/frefdrk/i2i3d-pytorch`

precision learning.

Why attempts to implement mixed precision were unsuccessful is not known. As the network was not able to learn with mixed precision, there might have been a problem of vanishing gradients, i.e. gradients being too small to affect parameters in the update. One parameter in mixed precision learning, the loss scaling factor, should be tuned to combat this problem and that might not have been done correctly.

5.2 Analysis of results

With the I2I-3D network architecture having performed well on vessel wall boundary detection in prior research, it is expected that it is also able to learn to segment full vessel structures. However, with relatively low filter sizes of only 4, 8, 16 and 32 layers in the four respective filter blocks it is surprisingly effective at the task, able to outperform a commercial automatic software in similarity metrics measured to manual segmentations. On one hand, this is showing impressive performance. On the other hand, when analysing the segmentations in Figure 4.4 by eye one can see that Medis QAngioCT Re is better at keeping vessel structures intact, and its main fault is including the small, clinically irrelevant vessels of a diameter below 2 mm. The fully trained network, having learnt to not include these, has the upper hand in that regard. Even with the impressive numbers in Table 4.3, one should not be too hasty in celebrating the performance of the network when compared to QAngioCT Re. The similarity metrics do not tell the complete story.

A more fair comparison of how the two automatic methods compare in full coronary artery vessel segmentation is shown in Figure 4.3. Clearly, the network can produce segmentations similar to Medis QAngioCT Re, but that was not the goal of this study. This study aims to reach beyond the Medis QAngioCT Re's segmentations and produce segmentations as close to the manual ones as possible. The network suceeds in that aspect. During the training on the manual segmentations, the network learns to disregard the small vessels that Medis QAngioCT Re includes, although not perfectly, which improves the similarity indices to manual segmentations quite drastically. The network also learns to place the segmentation border more in line with the manual segmentations than Medis QAngioCT Re, as shown in Figures 4.5 and 4.6. These are both improvements that could set apart a deep learning approach from regular automatic approaches. However, it seems that what the network learns when training on the manual data unfortunately also results in there being broken vessels, as can be seen in the left most vessel in Figure 4.4. There are ways to combat this issue. Using more manually segmented training data could be a first step, for example. Post-processing on the probabilistic network output could also be applied. As an example, applying Frangi filtering techniques to detect vessel-like structures in the probabilistic output might produce more sophisticated segmentations than simply thresholding at 0.5. Other alternatives for post-processing include using connected component analysis to distinguish large connected components (vessels) from small connected components (non-vessel

blobs), using conditional random field modeling to produce segmentations from local information in the probabilistic output rather than thresholding every voxel at a fix threshold, and to simply exclude any structures smaller than 2 mm in diameter.

Nevertheless, using a state-of-the-art network architecture, designed for use on medical images and for vessel (wall) segmentation in the form of I2I-3D, gives promising results. While not quite keeping up with medical professional segmentations, the similarity measures favour the deep learning approach over commercial segmentation software. A deep learning method for coronary artery segmentation, as suggested in this thesis, definitely seems viable.

In addition to this, it is important to remember that these results were achieved using a rather limited data set. With 77 automatic coronary artery segmentations in pretraining and 25 manually corrected segmentations for the main training, these results show great potential in regards to what could be possible to achieve with a larger data set. That said, any results obtained with such a small evaluation set, three images in this case, should be taken with a grain of salt as pointed out previously. Nevertheless, the strong ability of the U-shaped, interconnected fully convolutional network with integration of deep supervision, but otherwise quite bare-boned structure, to segment these tiny structures in the evaluation data from limited and highly unbalanced data sets should be encouraging for future work.

None of the data used to train the network is published online. It was generously made available from the SCAPIS project for this specific study, meaning that testing the repeatability of the results achieved in the thesis is limited to those connected to the SCAPIS project.

Finally, a point regarding the proposed method for coronary artery segmentation in this thesis should be made clear; the proposed deep learning tool aims to assist medical professionals in their work rather than to replace them altogether. The segmentation results and, in continuation, any diagnoses based on them should be inspected by a professional.

5.3 Subjectivity in manual segmentations

The similarity metrics between the medical professionals segmentations verify the subjectivity in the process of producing manual segmentations mentioned in Section 3.1.4. With a Dice index of 0.89 and a modified Hausdorff distance of 2.18 in the test data set, it is clear that they differ, although not to a huge degree. The question of how a neural network handles such subjectivity in the training data is interesting. Looking at Figure 4.4, the network looks to be more similar to the segmentation from professional B than that from professional A in the presented case, which the similarity metrics agree with. Whether the subjectivity in the data is a problem or not is difficult to answer. Since medical professionals don't completely agree on what the ground truth is in terms of segmentations, the most one can hope of from the network is that it produces segmentations that are more similar to the

group of medical professionals than any one professional from that group. With the professionals each having their own subjective ideal segmentation, the network will never be able to agree completely with all of them. This needs to be understood by anyone wanting to use a deep learning tool in this fashion. Alternatively, one could construct a separate network for each individual. The subjectivity would still be present, but there would only be the subjectivity of one individual to learn. This was attempted in this project, but did not result in segmentations on par with the network trained on both MS-A and MS-B on either of the MS-A or MS-B test data sets. Either way, if the tool is to be used to produce an automatic segmentation to then manually correct, having the tool produce some sort of average segmentation in relation to a large group of medical professionals seems like the way to go from an engineer's point of view.

5.4 Future work

With the SCAPIS study advancing, more and more segmentation data will be produced. This could certainly benefit deep learning algorithms which are so deeply dependent on training data. With a starting point in the proposed I2I-3D network, there are also additions and changes to be investigated. A successful implementation of mixed precision training would allow for a deeper network which potentially could improve the network performance. The architecture does not include either batch normalisation layers or drop-out layers, as Merkow et al. did not include these in their original architecture, both of which have shown to have a positive impact on learning [44] [6]. Modified versions of the ReLU activation function, such as the Leaky ReLU which has a non-zero response for negative inputs, have also shown improvements compared to regular ReLU in CNN applications [45].

These points, together with hyperparameter optimisation and post-processing possibilities previously mentioned clearly show that there is still a lot of potential to investigate. With deep learning based segmentation of coronary arteries being the first step in the development of a larger tool, this should be done thoroughly.

Building on this deep learning approach to construct a deep learning based tool able to detect stenosis and vascular plaque is a vision of the Computer Vision and Medical Image Analysis group at the Department of Electrical Engineering at Chalmers University of Technology. Using the proposed I2I-3D structure can be encouraged based on the results of this thesis.

6
Conclusion

In a large-scale study such as SCAPIS, it is important that the automatic segmentation tools used are able to produce high-quality segmentations in order to minimise the need for laborious and time consuming manual correction. The field of automated image analysis has been moving away from traditional image processing techniques in favour of deep learning for some time now. In the case of image segmentation, fully convolutional networks are leading the way, with U-Net inspired networks such as I2I-3D being front runners in medical applications.

The main research question of this thesis is whether or not a deep learning method for coronary artery segmentation in CTA images is viable. To answer that question, a version of the state-of-the-art deep convolutional neural network architecture named I2I-3D was applied to data from the SCAPIS study. The network was implemented in the PyTorch deep learning framework, with pretraining on three separate data sets. To investigate the viability, the network segmentations were compared to both commercial segmentation software and manual segmentations performed by two medical professionals. The comparison was made by establishing how similar the different methods were to each of the manual segmentations through two separate similarity metrics. One metric, Dice index, measures the segmentation overlap, i.e. counts correctly and incorrectly classified voxels and gives a single value corresponding to the performance. The other metric, modified Hausdorff distance, determines how close in space two segmentations are, i.e. determining similarity in overall shape.

What can be concluded from the results is:

1. The I2I-3D network is able to learn well from the limited data set but can not perform on par with manual segmentations.
2. The I2I-3D network beats the commercial software Medis QAngioCT Re in both similarity measures on both evaluation data sets. Manual inspection of the segmentation shows that the network loses some important structures of the coronary tree, making it harder to determine a clear winner between the two. The largest source of dissimilarity between the manual segmentation and Medis QAngioCT Re segmentations is that the latter includes small vessel structures, while the former does not. I2I-3D does however perform better in segmentation border placement than Medis QAngioCT Re.
3. The results are promising for future development, as there are clear additions and changes to investigate for potential improvements, such as batch normalisation layers, drop-out layers and post-processing of the probabilistic output from the network.

The main research question of this thesis was: *Is deep learning viable for coronary artery segmentation?* With the network outperforming the commercial software in both similarity metrics, the viability has been demonstrated. However, a visual inspection of the segmentations show that there is still room for improvement. With more data and implementation of the proposed additions possibly improving the performance further, the future for a deep learning based coronary artery segmentation tool looks bright.

Bibliography

[1] Geneva: World Health Organization. World health statistics 2017: Monitoring health for the SDGs, sustainable development goals. 2017.

[2] G. Bergström et al. The Swedish Cardiopulmonary Bioimage Study: Objectives and Design. *Journal of internal medicine*, 278(6):645–659, 2015.

[3] A. Broersen et al. FrenchCoast: Fast, robust extraction for the nice challenge on coronary artery segmentation of the tree. In *Proceedings of MICCAI Workshop" 3D Cardiovascular Imaging: a MICCAI segmentation Challenge*, 2012.

[4] P. Fischer O. Ronneberger and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR*, abs/1505.04597, 2015.

[5] J. Merkow et al. Dense volume-to-volume vascular boundary detection. *CoRR*, abs/1605.08401, 2016.

[6] Y. Bengio Y. LeCun and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[7] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.

[8] N. Srivastava et al. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[9] A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] C. Lee. Deeply-supervised nets. In *Artificial Intelligence and Statistics*, pages 562–570, 2015.

[12] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[13] K. Fukushima and S. Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[14] S. Ji et al. 3D convolutional neural networks for human action recognition.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(1):221–231, Jan 2013.

[15] O. Russakovsky et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[16] UNC Vision Lab ImageNet. Large scale visual recognition challenge 2017 (ilsvrc2017), 2017. URL `http://image-net.org/challenges/LSVRC/2017/results`. [Online; accessed 3 May 2018].

[17] S. Beucher et al. The watershed transformation applied to image segmentation. *Scanning Microscopy-Supplement*, pages 299–299, 1992.

[18] Y. Huang and D. Chen. Watershed segmentation for breast tumor in 2-D sonography. *Ultrasound in Medicine and Biology*, 30(5):625–632, 2004.

[19] A. Frangi et al. Multiscale vessel enhancement filtering. In William M. Wells, Alan Colchester, and Scott Delp, editors, *Medical Image Computing and Computer-Assisted Intervention — MICCAI'98*, pages 130–137, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[20] M. E. Martinez-Perez et al. Segmentation of blood vessels from red-free and fluorescein retinal images. *Medical image analysis*, 11(1):47–61, 2007.

[21] H. K. Yuen et al. Comparative study of hough transform methods for circle finding. *Image and vision computing*, 8(1):71–77, 1990.

[22] M Schaap et al. Standardized evaluation methodology and reference database for evaluating coronary artery centerline extraction algorithms. *Medical Image Analysis*, 13(5):701 – 714, 2009. Includes Special Section on the 12th International Conference on Medical Imaging and Computer Assisted Intervention.

[23] Consortium for Open Medical Image Computing. Rotterdam coronary artery algorithm evaluation framework, 2018. URL `http://coronary.bigr.nl/stenoses/results/results.php?type=testing&subs=public&sinfo=no&vendor=all`. [Online; accessed 23 May 2018].

[24] D. Ciresan et al. Deep neural networks segment neuronal membranes in electron microscopy images. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2843–2851. Curran Associates, Inc., 2012.

[25] N Feng et al. Toward automatic phenotyping of developing embryos from videos. *IEEE Transactions on Image Processing*, 14(9):1360–1371, 2005.

[26] E. Shelhamer J. Long 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.

[27] Ö. Çiçek et al. 3D u-net: learning dense volumetric segmentation from sparse annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 424–432. Springer, 2016.

[28] O. Jimenez del Toro et al. Cloud-based evaluation of anatomical structure segmentation and landmark detection algorithms: Visceral anatomy benchmarks. 35(11):2459–2475, 2016.

[29] A. Norlén et al. Automatic pericardium segmentation and quantification of epicardial fat from computed tomography angiography. *Journal of Medical Imaging*, 3(3):034003, 2016.

[30] Y. LeCun et al. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer, 1998.

[31] M. Eichelberg P. Mildenberger and E. Martin. Introduction to the dicom standard. *European Radiology*, 12(4):920–927, Apr 2002.

[32] E. Jones et al. SciPy: Open source scientific tools for Python, 2001–. URL `http://www.scipy.org/`. [Online; accessed 30 Apr 2018].

[33] The PyTorch Team. Pytorch, a year in..., 2018. URL `http://pytorch.org/2018/01/19/a-year-in.html`. [Online; accessed 30 Apr 2018].

[34] A Paszke. Automatic differentiation in pytorch. 2017.

[35] N. Ketkar. *Introduction to PyTorch*, pages 195–208. Apress, Berkeley, CA, 2017.

[36] jmerkow. Github user page, 2016. URL `https://github.com/jmerkow/I2I/`. [Online; accessed 30 Apr 2018].

[37] NVIDIA. Training with mixed precision :: Deep learning sdk documentation, 2018. URL `https://docs.nvidia.com/deeplearning/sdk/mixed-precision-training/index.html`. [Online; accessed 13 May 2018].

[38] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[39] A. A. Taha and A. Hanbury. Metrics for evaluating 3D medical image segmentation: Analysis, selection, and tool. *BMC medical imaging*, 15(1):29, 2015.

[40] K. Zou et al. Statistical validation of image segmentation quality based on a spatial overlap index1. *Academic Radiology*, 2(11):178–189, 2004.

[41] K. Zou et al. Three validation metrics for automated probabilistic image segmentation of brain tumours. *Statistics in medicine*, 23(8):1259–1282, 2004.

[42] R. de Luis-García R. Cárdenes and M. Bach-Cuadra. A multidimensional segmentation evaluation for medical image data. *Computer methods and programs in biomedicine*, 96(2):108–124, 2009.

[43] M. Dubuisson and A. K. Jain. A modified hausdorff distance for object matching. In *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, volume 1, pages 566–568. IEEE, 1994.

[44] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[45] B. Xu et al. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

A

Segmentations for the three test volumes in the MS dataset

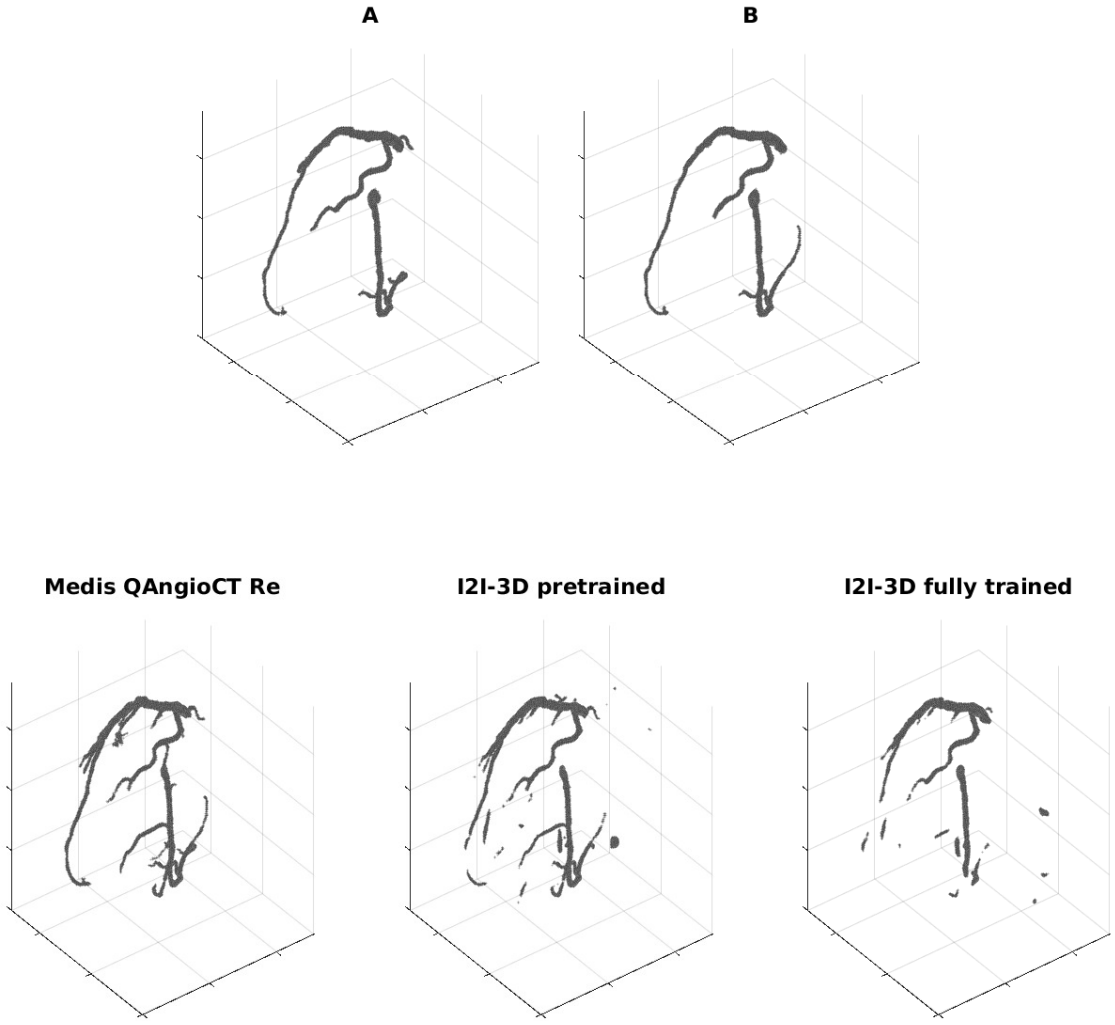

Figure A.1: Volume #1 in test set of the MS dataset.

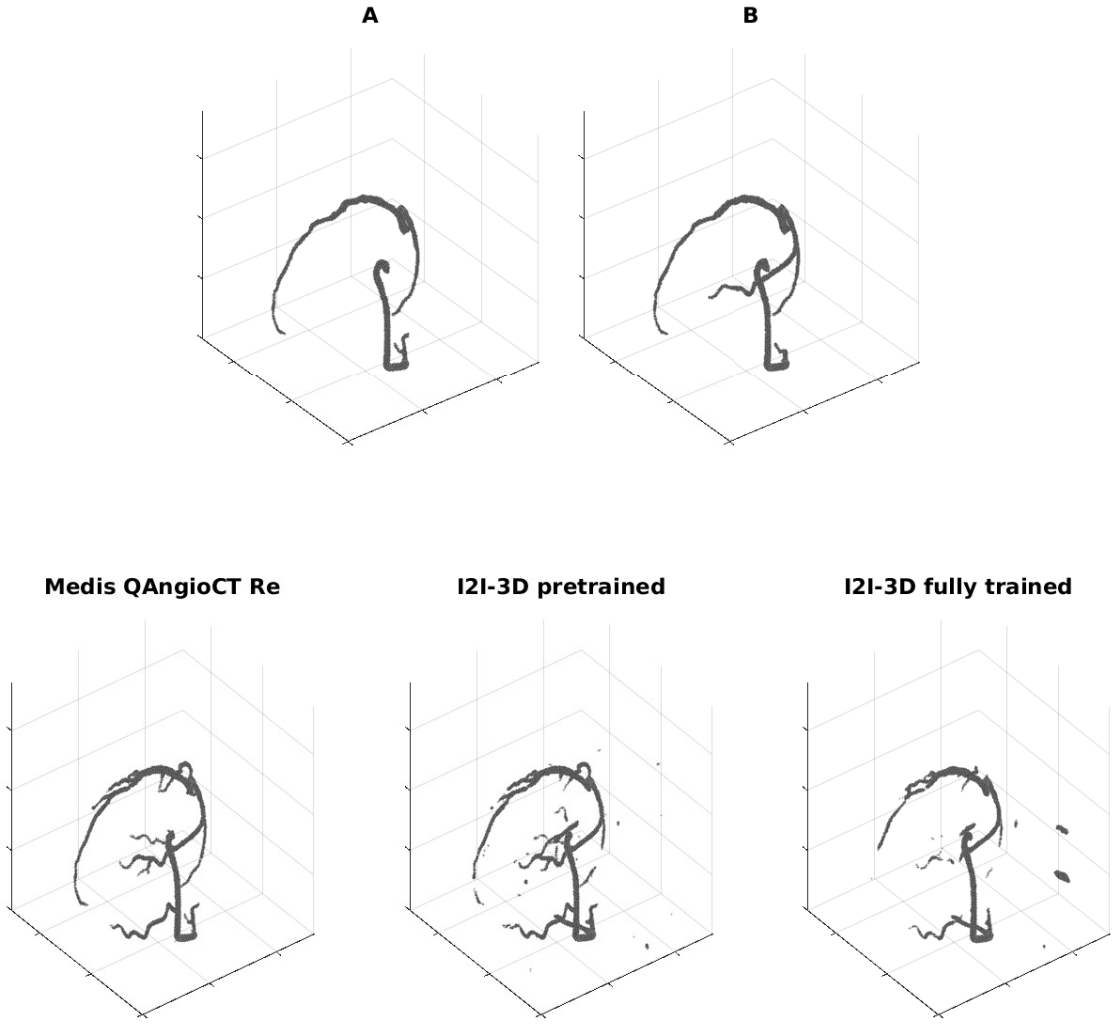

Figure A.2: Volume #2 in test set of the MS dataset.

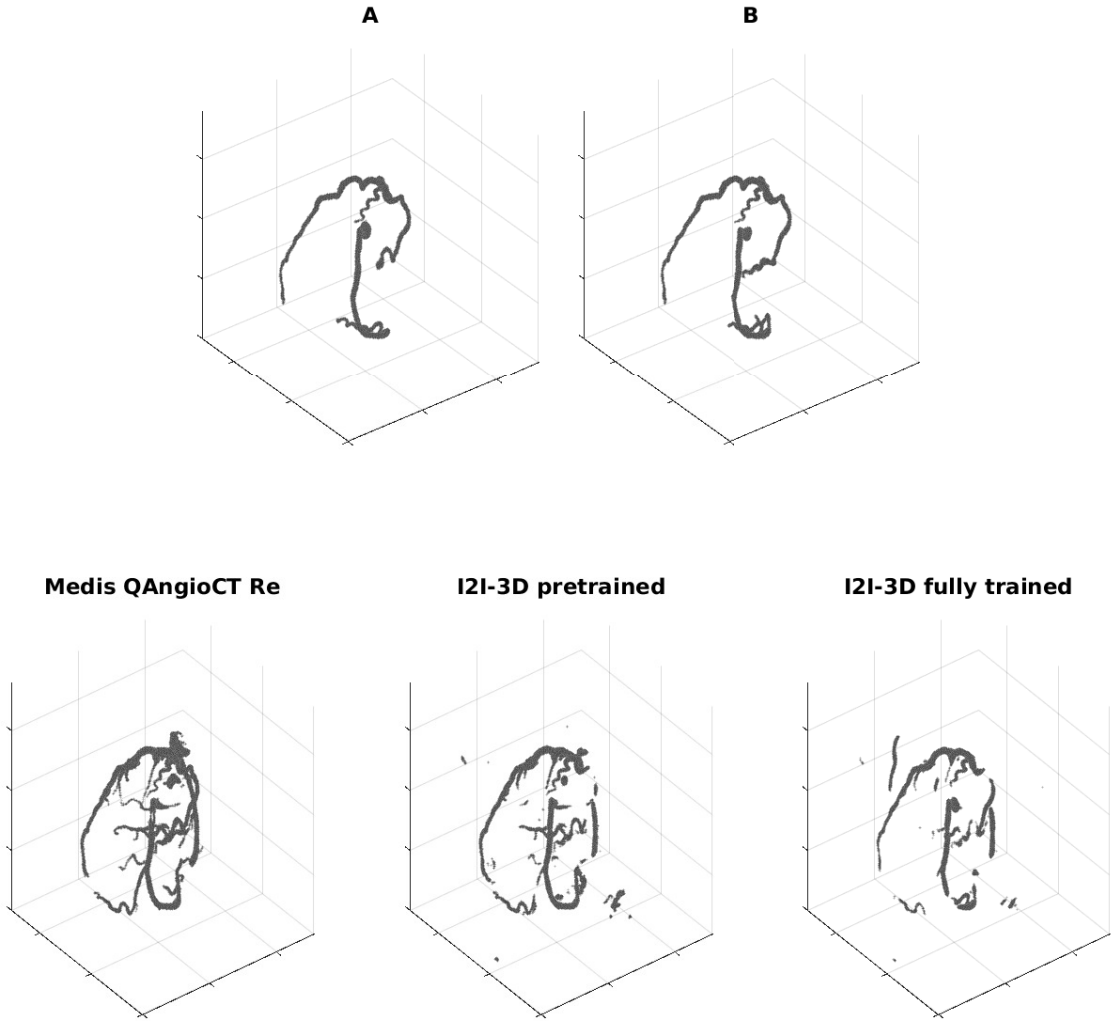

Figure A.3: Volume #3 in test set of the MS dataset.