Aortic Segmentation in Real-Time Flow Exercise Cardiac Magnetic Resonance Images using Convolutional Neural Networks

Mathilda Larsson

Master's thesis
2021:E40

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

CENTRUM SCIENTIARUM MATHEMATICARUM

Aortic Segmentation in Real-Time Flow Exercise Cardiac Magnetic Resonance Images using Convolutional Neural Networks

Mathilda Larsson

Principal supervisors: Johannes Töger and Einar Heiberg
Assistant supervisor: Ida Arvidsson
Examiner: Kalle Åström

Presented on 2021-06-15
Report submitted on 2021-06-07
Second report submission on 2021-06-18

Abstract

Background: Cardiovascular disease is common and it is therefore important to have tools to investigate pathophysiology and normal physiology of the cardiovascular system. One such tool is using cardiovascular magnetic resonance in order to measure blood flow in vessels, particularly while the participant imaged is exercising. This enables studying the blood flow during exercise which is especially important for people with diseases which only manifest during physical stress.

In order to measure the blood flow of the aorta during multiple heart cycles it is necessary to delineate the images depicting the aorta. Typically this is done manually, but since this is a time-consuming process it could benefit from a segmentation algorithm.

Aims: This thesis' aims to develop a semi-automatic segmentation model using a convolutional neural network. The training data consisted of real-time flow and gated images acquired during rest, and real-time flow images acquired during exercise were used for validation and testing.

Method: The model studied in this thesis is a kind of convolutional neural network called U-Net. Multiple variations of the U-Net architecture were explored by varying the number of input channels, the encoder depth of the network and using different loss functions. Additionally, the effects of using L2 regularization and adaptive learning rates during training were investigated. The performance of the models are measured using the accuracy, intersection over union and mean BF score of the segmented aorta class. Additionally, calculations of the sequence volumes based on the segmentations of the flow images were compared to the ground truth sequence volumes.

Results: In this project, the best results were produced using a U-Net model with encoder depth 4 and using both a velocity-encoded and magnitude image as input. Binary cross-entropy was found to be the best choice for loss function, but further testing could be done using Dice. Using L2 regularization and adaptive learning rates did not improve the result.

Conclusion: The networks developed in this thesis project prove that it is possible to train U-Net models on real-time flow and gated images acquired during rest in order to segment real-time flow images acquired during exercise. However, more studies into inter- and intra-observer variability of the data as well as improving the sequence volume calculations are needed before the algorithm could be of clinical use.

Acknowledgments

This project was carried out at the Cardiac MR Group in Lund and all the data is from research projects in the group and from the clinic at Skåne University Hospital. I would like to thank the entire Cardiac MR group for making this thesis possible, in particular Johannes Töger and Einar Heiberg for supervising the project. Additionally, I would like to extend my gratitude to the other master thesis students who have done their projects in the group this semester - you have all provided me with continuous support and important feedback on the project as well as on the report.

Abbreviations

2D PC CMR - two-dimensional phase contrast cardiac magnetic resonance
ANN - artificial neural network
CMR - cardiac magnetic resonance
CNN - convolutional neural network
CS - compressed sensing
ECG - electrocardiogram
EPI - echo planar imaging
ReLU - rectified linear unit

Contents

Chapter 1

Introduction

Stress cardiac imaging using exercise cardiovascular magnetic resonance (CMR) is a tool used to study physiology and assess pathophysiology. CMR imaging techniques provide images of high quality compared to other available methods and enables research into how the cardiovascular system functions in both healthy and unhealthy subjects during exercise. It can be performed by exercising outside the scanner bore on a treadmill before imaging or by exercising inside the CMR scanner bore. CMR has several advantages compared to other imaging modalities since it does not use ionizing radiation, fits most patient sizes and is non-invasive. However, due to the nature of imaging while the patient is exercising and therefore moving, the image quality is worse in exercise CMR images compared to CMR images acquired during rest. This is because of the creation of artefacts due to respiratory and physical motion [Craven et al., 2020]. To accommodate motion, the conventional gated CMR methods used for measuring blood flow in rest are replaced by real-time flow techniques which result in images of lower resolution.

One application of exercise CMR is measuring the blood flow in the aorta in order to make a diagnosis in case of disease [Craven et al., 2020]. For example, investigating the blood flow during exercise can be an important tool to detect cardiac disease in the New York Heart Association's class I category, as defined by [The Criteria Committee of the New York Heart Association, 1994], i.e. where the patient experiences no symptoms or limitations during rest. Blood flow measurements can also be used to investigate normal physiology, eg. cardiac output during exercise in healthy volunteers [Steding-Ehrenborg et al., 2013].

To measure the flow, delineation of the aorta in the exercise CMR images is necessary. However, manual delineation of cardiovascular images is time-consuming for a human to perform, especially since delineation is required over multiple cardiac cycles. The process could benefit from automatic segmentation using machine learning methods. Therefore, the aim of this project is to develop an automatic segmentation algorithm using a convolutional neural network for real-time flow exercise CMR images.

Chapter 2

Aims

The purpose of this thesis is to explore if convolutional neural networks with U-Net architectures can segment the aorta in real-time flow images acquired during exercise. The specific aims of the project were:

- To train on real-time flow images and gated images acquired during rest in order to validate and test on real-time flow images acquired during exercise.
- To evaluate how the performance of the U-Net networks are affected by varying the number of input channels and the encoder depth of the model as well as evaluate the performance differences when using cross-entropy, Dice and a combination of both as a loss function. Furthermore, the effect of using L2 regularization and adaptive learning with be evaluated separately.
- To use the segmentations of the U-Net model to calculate the sequence volumes in the validation and test set in order to compare them to the corresponding ground truth volumes.

Chapter 3

Background

The cardiovascular system is a system of organs which carry oxygen and nutrients to the tissues within the body as well as retrieve wastes such as carbon dioxide [Encyclopaedia Britannica, 2020]. This is done by blood flowing throughout the body in vessels, pumped using the muscular power of the heart. The blood is oxygenated in the lungs and are then pumped through the aorta to smaller vessels in the body. In order to study the flow in the aorta, the vessel needs to be imaged.

3.1 Magnetic Resonance Imaging

Magnetic resonance imaging (MRI) is a non-invasive technique which can be used to image tissues and organs in the body. The basis of the technique are the nuclei of hydrogen atoms [Weishaupt et al., 2008], each consisting of one proton which has a positive electric charge and spin [Weishaupt et al., 2008]. The latter causes the proton of the nuclei to rotate around its own axis. Due to its rotating mass with an electric charge, the proton possesses a magnetic moment which can be affected by external magnetic fields, such as the ones applied by an MRI scanner. The proton can also induce a current in an MRI scanner's receiver coil due to this property. Since there are hydrogen atoms in abundance throughout the body, MRI can be used to image the body for multiple purposes.

3.1.1 Two-dimensional Phase Contrast CMR

One of the applications of MR is imaging flow in the aorta. Two-dimensional phase contrast CMR (2D PC CMR) utilizes the proportional relationship between the velocity of the blood flow and the phase shift in the signal measured by the MR scanner [Nayak et al., 2015]. The phase shift is induced in hydrogen nuclei by applying a encoding magnetic field gradient. This results in a three component velocity vector which is aligned to define the xy-plane in the cross section of the aorta and the z-direction is aligned with the direction of the flow of the aorta. In order to analyze the vessel, two images are used. The first depicts the z component of the velocity vectors, see the left panel in Figure 3.1). Here, the flow of a pixel can be calculated as the product of the area corresponding to the pixel and the z component of the velocity. To determine which pixels are to be included in the total flow of the aorta, the aorta need to be delineated in the image. This can be done using the velocity image directly, or by using the corresponding magnitude image, which is the magnitude of the complex

signal containing the phase shift which is proportional to the velocity, see the right panel in Figure 3.1.

The lowest and highest velocity that is detectable in a flow measurement is defined by the velocity encoding, also known as the VENC [Lotz et al., 2002]. It is defined with the unit $[cm/s]$. For example, a flow image with a VENC = 150 can measure flow velocities corresponding to $\pm 150[cm/s]$.

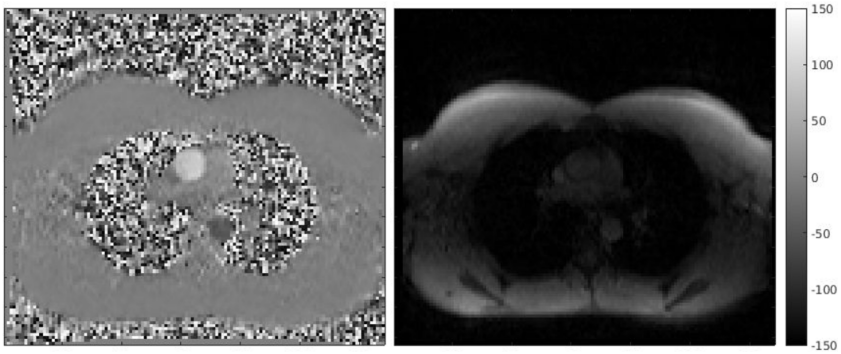

Figure 3.1: *Examples of a velocity-encoded image (left) and a magnitude image (right). The velocity encoding is* $VENC = 150$.

3.1.2 Gated Acquisition and Real-Time Flow

The traditional way of using 2D PC CMR is using gated acquisition. This method uses electrocardiogram (ECG) in order to synchronise acquisition during different parts of the cardiac cycles. Hence, the resulting image series consists of averages of multiple heartbeats. This averaging may lead to loss of data if there are differences between different heartbeats [Zhang et al., 2014]. The differences may occur due to physiological reasons such as arrhythmia but also due to movement. The latter is one of the reason to instead make use of real-time flow phase contrast imaging rather than ECG gated imaging techniques for acquisition while the patient is exercising. Gated acquisition is however preferable in some cases since the image quality is higher compared to real-time flow images. It can either be performed using breath-holding (which requires a regular cardiac rhythm) or without breath-holding [Jin and Simonetti, 2013]. The latter allows for imaging patients who cannot hold their breaths as well as studying physiology during free-breathing. Both methods require ECG gating. Real-time flow imaging on the other hand enables investigating the flow of the aorta during free breathing without need of ECG synchronization.

For the images used in this project, acquisition of the real-time flow images were accelerated using one of two methods: echo planar imaging (EPI) and compressed sensing (CS). Using the EPI technique, an entire image can be acquired in the short span of 20-100 milliseconds which enables capturing short physiological processes [Poustchi-Amin et al., 2001]. Compared to gated images, using EPI techniques can reduce motion artefacts and decrease imaging time which makes the method suitable to capture the blood flow during exercise.

Gated acquisition is limited by the Nyquist sampling rate which restricts how much the image acquisition time can be accelerated without causing aliasing artefacts [Geethanath et al., 2013]. CS provides a framework where images can be reconstructed much faster from undersampled measurements and in cases where there is significant aliasing. The core idea of the technique is to only acquire the coefficients

of a signal that are significant and making use of transform sparsity (i.e. a vector becoming sparse after applying a mathematical transform).

3.2 Convolutional Neural Networks for Image Segmentation

Convolutional neural networks (CNNs) have become increasingly popular to use for pattern recognition tasks during the past decade [O'Shea and Nash, 2015]. CNNs are a type of artificial neural network (ANN) which are specialised in encoding features in images into its architecture by training on image data sets. In general, an ANN consists multiple layers of nodes (also known as neurons) in a series of hidden layers, see Figure 3.2. The input is fed into the network in the input layer, where a series of mathematical operations are performed in the hidden layer until the network results in an output, eg. a prediction.

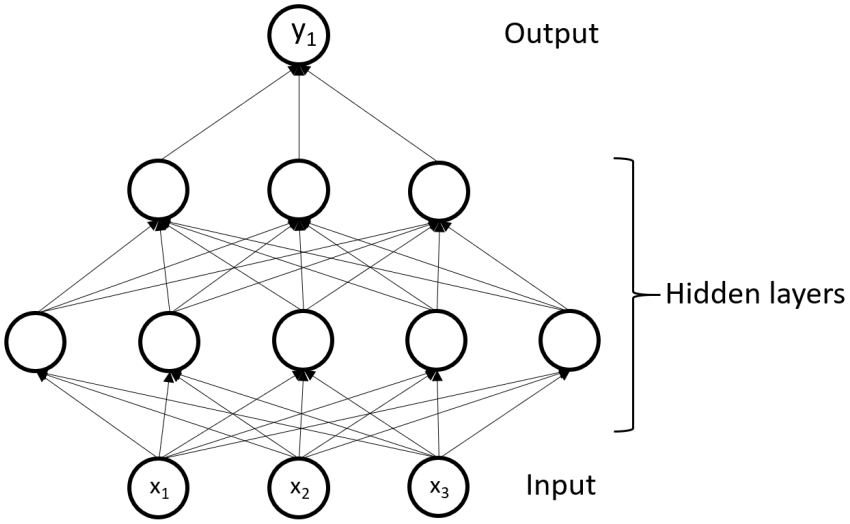

Figure 3.2: *Schematic of a basic ANN. Image by Filip Winzell, used with permission.*

The concept of a CNN is essentially the same as for an ANN, where the mathematic operation of the network is the convolution. The input can be an image and the output can be a segmentation of an object in the input image, eg. by labelling all pixels in an image to belong to either the background class or to the aorta class. The layers of a CNN each consist of a 3-dimensional structure with a height, width and depth where the height and width is defined by the size of the image (eg. 128x128 pixels) and the depth is the number of channels [Aggarwal, 2018]. In the input layer, the depth can for example be the number of color channels (eg. 3 for an RGB image or 1 for a grayscale image), while in the hidden layers the depth corresponds to the number of feature maps in that layer. The parameters of a CNN are called kernels (also known as filters) and typically have a square spatial dimension (height = width, most often an odd number) which is significantly smaller than the spatial dimension of the input. However, a kernel must have the same depth as the input as it is applied to. An important concept when it comes to CNNs is shared, sparse weights between the layers. Essentially, this concept means that having a network with small weights

are achieved by using fewer parameters which are shared by applying the same kernel multiple times over an input.

The convolutional operation makes up the first stage of a CNN layer. The output of this operation is a set of linear activations which are the input of the second stage where they are processed using a non-linear activation function. The third and final step of the layer is the pooling operation, which will result in decreasing the spatial dimensions.

3.2.1 The Convolution Operation

The convolution operation of the CNN layer consists of placing the kernel at multiple locations of the input (eg. the input image in the case of the first hidden layer) and performing a dot product of the overlapping parameters at each of those locations [Aggarwal, 2018]. In other words, for a filter of size 2x2x1, the dot product will be computed by placing it to match 2x2 pixels in a grayscale image. The result of the dot product will be a single value, see Figure 3.3. The next pixel value is calculated by sliding the filter one step to the right until the entire input has been processed.

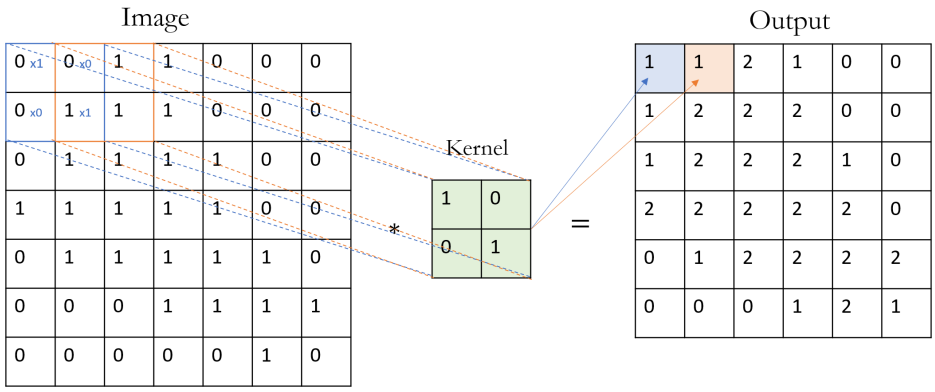

Figure 3.3: *Illustration of how a convolution is applied across an image using a kernel of size 2x2x1 and a stride of 1. Image by Filip Winzell, used with permission.*

Typically, the depth of the layer is not 1, as in the example above. Instead, multiple sets of independent kernels can be used to create an increasing number of feature maps, eg. using 3 independent kernels in a layer will result in an output with 3 feature maps. Setting the number of filters and their sizes, in a network determines the number of parameters. The motivation of using multiple kernels is that each one will be trained to represent a different kind of spatial pattern in a small rectangular area of the input, and hence many kernels will be needed to capture multiple shapes across the image. In the later stages of the CNN, the spatial size of the layer is typically smaller than the input image's while the depth is much larger to contain all the feature maps.

3.2.2 Padding

Following the example above in Figure 3.3, sliding the kernel over the image will result in an output of smaller resolution, eg. and input image of spatial size 7x7 will result in an output of size 6x6. This decrease in size is usually undesirable since it will lead to loss of information in the border of the image/feature map. The solution is to simply add padding around the image/feature map before applying the convolution

[Aggarwal, 2018]. How many padding pixels need to be added to preserve the size of the input depends of the size of the kernel and the stride. One common type of padding is zero padding where the padded pixels are equal to 0. Padding can also be used to give the output have a larger spatial size than the input, eg. when using an up-convolution (further explained in the Section "Upconvolution" below and in Figure 3.5).

3.2.3 Stride

The spatial resolution can also be decreased intentionally using a larger stride, for example to make the size of the network smaller [Aggarwal, 2018]. The stride determines the step size the kernel takes as it slides over the image. In the explanation of the convolutional operation above (see Figure 3.3), a stride of 1 was used in order to slide the kernel one step at a time, i.e. a convolution is performed at every possible spatial location. Performing a convolution at every other possible spatial location instead would mean using a stride of 2.

3.2.4 ReLU Activation

After applying the convolution for the entire feature map, the next step is to apply a non-linear function to each value [Aggarwal, 2018]. This is performed for every feature map, hence each feature map results in a single activated feature map of equal size. The dimensions of the layer therefore remain the same as before the activation function is applied. For CNNs, the rectified linear unit (ReLU) is typically used and it is defined as:

$$f(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

The reason why ReLU is used for CNNs (and other non-linear functions in ANN layers in general) is to achieve non-linear decision boundaries for the model. If all layers only contained linear operations (i.e. the dot product used in convolution), all the convolutional layers could be rewritten into a single linear operation which means having a model consisting of a single layer. Non-linearity therefore enables deeper networks and the ability of a CNN to capture complex relations in images.

3.2.5 Pooling

After the ReLU activation of each value, the pooling step replaces the pixel in the output of the layer with a summary statistic based on the nearby pixels [Aggarwal, 2018]. This is done for every activation map and the number of feature maps remain the same while the spatial dimensions of each feature map will decrease. One common approach is to use the maximum, also known as max-pooling (see Figure 3.4). By summarizing multiple input pixels, the size of the output will be smaller than the input image. Pooling is often applied with a stride > 1 which will result in an even bigger decrease in spatial size. The use of pooling also makes CNNs translationally invariant, i.e. less sensitive to where an object in an image is placed.

3.2.6 Upconvolution

The upconvolution operation is a mathematical operation where the output image has a larger spatial dimension than the input image, see Figure 3.5 [Ronneberger et al., 2015].

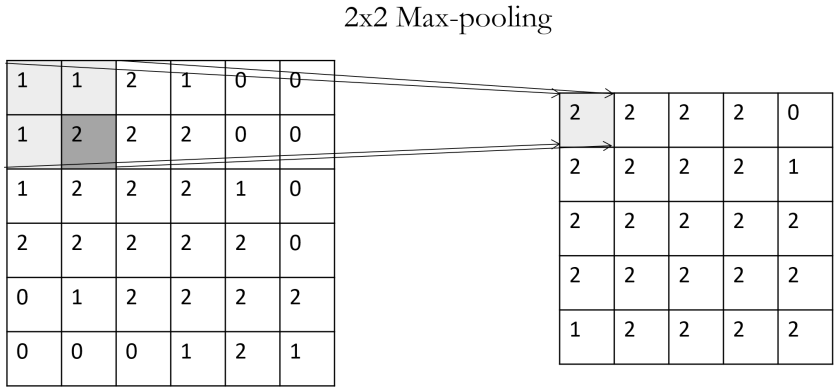

Figure 3.4: *Illustration of how max-pooling of size 2x2 and stride 1 is applied to a feature map. Image by Filip Winzell, used with permission.*

It also makes use of a kernel and padding, eg. a 3x3 kernel applied to a a 2x2 image with a padded border of 2 pixels will result in an image of spatial size 4x4.

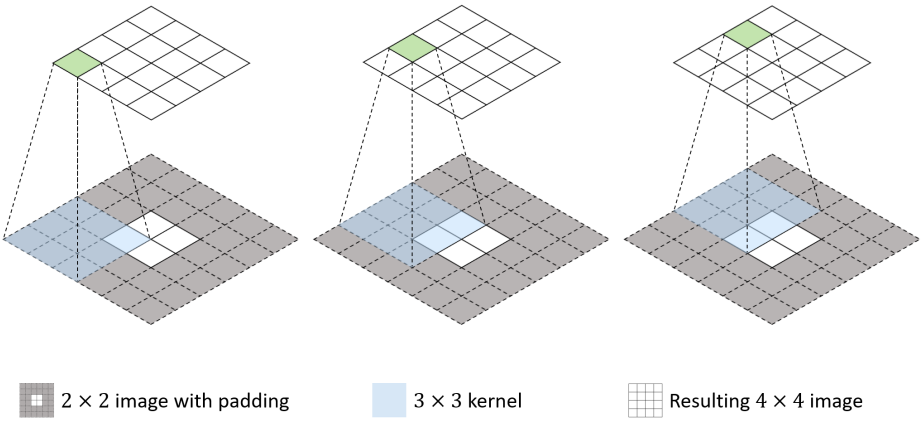

Figure 3.5: *Illustration of an upconvolution. Image by Ruben Bergengrip, used with permission.*

3.2.7 Training a CNN

The weights of the network (i.e. the values of the kernels) are calculated by training the CNN. The training uses backpropagation, which is a process where a loss function is used to update the weights based on the performance of the network. A common choice of loss function for binary segmentation tasks (tasks where there are two possible classes in the image, eg. aorta or background) is the binary cross entropy loss function [Jadon, 2020], which is formulated for an image with P pixels as

$$L_{CE} = -\frac{1}{P}\sum_{p=1}^{P} w_1 \cdot t_p \cdot \log(y_p) + w_2 \cdot (1 - t_p) \cdot \log(1 - y_p).$$

The idea is to compare the label t_p of pixel p (eg. $t_p = 1$ for the class "aorta" and

$t_i = 0$ for the class "background") to the output score y_p of the network. y_p is the output of the network and hence the probability the network has assigned to the pixel that it belongs to class 1, with $(1-y_p)$ being the probability that it belongs to class 0. w_k is the weight for class k in order to compensate for imbalance between the classes by weighting the classes according to the pixel frequency of each class, eg. when the number of pixels in class 1 correspond to 10% of the total number of pixels in the image and 90% make up class 0. The training is aimed at minimizing the loss, or in other words, making as many correct predictions as possible. If the network on the other hand has assigned a high probability of an incorrect prediction, the loss will be high.
Another kind of loss function is the Dice loss function

$$L_{Dice} = 1 - \frac{2(w_1 \sum_{p=1}^{P} y_p \cdot t_p + w_2 \sum_{p=1}^{P} (1-y_p) \cdot (1-t_p))}{w_1(\sum_{p=1}^{P} y_p^2 + t_p^2) + w_2 \sum_{p=1}^{P} (1-y_p)^2 + (1-t_p)^2},$$

which is based on the Dice coefficient which estimates the similarity between two given images [Milletari et al., 2016]. As for the cross-entropy function, w_k is the weight for each class k to account for the imbalance between the background and aorta class, y_p is the prediction that pixel p belongs to class 1, t_p is the ground truth label for pixel p and P is the number of pixels in the image.
A combination of both cross-entropy and dice can be formed as

$$L_{combo} = \gamma \cdot L_{Dice} + (1-\gamma) \cdot L_{CE},$$

where the scalar γ is used to determine the influence of one of the loss functions over the other.

3.3 U-Net

U-Net is a CNN architecture first presented in 2015 for the purpose of segmentation of biomedical images [Ronneberger et al., 2015]. Since then, the U-Net architecture has successfully been applied to many datasets and the architecture has been modified for different tasks [Siddique et al., 2011].

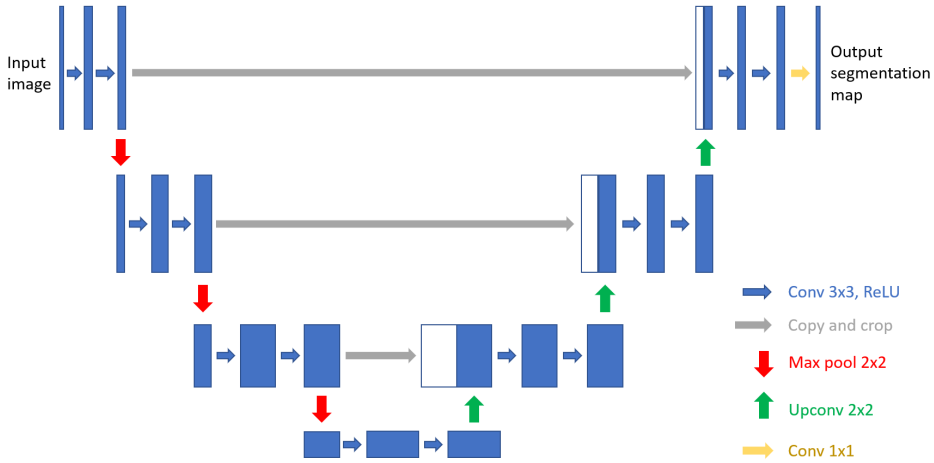

Figure 3.6: *Schematic of the architecture of the original U-Net model. Image by Filip Winzell, used with permission.*

The U-Net architecture gets its name from the U-shaped structure formed by its contracting and expanding path, see Figure 3.6. The first part, the contracting path, is similar to a typical CNN structure, with convolutions of size $3x3$ interleaved with ReLU activations and max-pooling operations of size $2x2$ using a stride of 2 to achieve downsampling [Ronneberger et al., 2015]. Every stage of downsampling in the contracting path results in doubling the number of feature channels and a decrease in spatial size due to the number of kernels used for each input and the use of max-poolinng.

In the expansive path, the feature maps are upsampled (i.e. the spatial resolution is doubled) with a $2x2$ up-convolution (leading to a halving of the number of feature channels). At each of these up-convolutions, the feature map in the contracting path which corresponds to the current upconvolution stage is cropped in order to be concatenated into the upsampled feature map. This kind of concatenation is called a skip-connection [Ma et al., 2020], where the upsampled output is combined with features of higher resolution from the contracting path [Ronneberger et al., 2015]. After the concatenation, two $3x3$ convolutions and ReLU activations are applied. In order to form a final segmented image with the correct number of channels, a $1x1$ convolution is applied in the final layer [Siddique et al., 2011].

3.4 Training, Validation and Testing

The available data is split into three data sets:

- Training data: The CNN models are trained using this data. The accuracy and loss is measured on this data after each iteration, i.e. after processing a mini-batch.
- Validation data: While training a model, the accuracy and loss is measured on the validation data after a set number of iterations. This is done to monitor overfitting which can be identified as when the training accuracy increases while the validation accuracy decreases. The performance of different models based on the validation data is used to determine which one seems to be the best and is to be used as a final model.
- Test data: A data set which the final model's performance is measured on, consisting of entirely unseen data for the model. No tuning of parameters is done after the test data has been used.

3.4.1 Training Parameters and Settings

There are multiple options for settings when training a CNN. The ones relevant for this thesis project are explained below:

- **Data augmentation**: Data can be augmented in order to improve the robustness of the model, especially to compensate for a lack of data. Typical augmentations used for images are rotations and translations within a defined range as well as normalizing pixel values. Details of the augmentations performed on the data in this project can be found in Section 4.3.
- **Mini-batches**: The training data can be divided into mini-batches which each holds a certain number of images. Eg. if the batch size = 20, 20 images are included in each mini-batch. The batch size defines the number of samples which are used before the training accuracy and loss are evaluated, i.e. how many

images are used in an update of the kernel weights (a pass). A larger batch size also requires more memory space on the GPU.

- **Epoch**: An epoch has been completed when each training sample has been used for a pass.
- **Iterations**: The number of passes needed to complete an epoch.
- **Learning rate**: How much the weights are updated after each iteration, denoted η. A high learning rate might lead to missing an optimal value (i.e. missing a minimum of the loss function) while a low learning rate will lead to training taking an unreasonably long time. The learning rate can either be constant throughout the training or be updated as the training progresses. The latter is done by multiplying the learning rate with a learning rate drop factor γ after a certain number of epochs have passed (denoted as the learning rate drop period T), which results in the learning rate decreasing.
- **Adam optimizer**: The weights / kernels ω of a neural network can be updated using different methods, one of the most popular being Adaptive moment estimation (Adam) [Kingma and Ba, 2017]. The current weight $\omega_k(t+1)$ is updated based on the previous weight $\omega_k(t)$ as well as a running average of the gradient $g_k(t)$ and running average of the squared gradient $(g_k(t))^2$ according to

$$\omega_k(t+1) = \omega_k(t) - \eta \cdot \frac{m_k(t+1)}{1-\beta_1^t} \cdot \frac{1}{\frac{v_k(t+1)}{1-\beta_2^t} + \epsilon},$$

$$\text{where } m_k(t+1) = \beta_1 m_k(t) + (1-\beta_1) \cdot g_k(t),$$

$$\text{and } v_k(t+1) = \beta_2 v_k(t) + (1-\beta_2) \cdot (g_k(t))^2.$$

  The running averages of the gradient and the squared gradient is modified by the learning rate η and the parameters β_1 and β_2. ϵ is a small number to make sure there is no division by 0.

  The use of Adam is popular since it avoids getting stuck in plateau regions of the error function and is able to escape local minima. The step size is also decreased near an optimum.

- **L2 regularization**: Regularization terms can be added to the loss function $L(\omega)$ in order to penalize large values of the weights ω [Krogh and Hertz, 1991]. This may be useful to avoid overfitting. One common type of regularization is L2 norm regularization which is defined as

$$L'(\omega) = L(\omega) + \alpha \cdot \frac{1}{2} \sum_k^K \omega_k^2,$$

  where $L(\omega)$ is the chosen loss function, eg. cross-entropy, and K is the total number of kernels. The value of α defines how large influence the regularization term will have on the modified loss function $L'(\omega)$.

3.4.2 Evaluation Scores

In order to evaluate the performance of the network, multiple evaluation scores are used. These are defined using the predicted values and the actual values of each pixel, which can be used to create a confusion matrix, see Figure 3.7. For this project, three scores based on the predictions and actual values of the pixels belonging to the aorta class were used: accuracy, intersection over union and mean BF score.

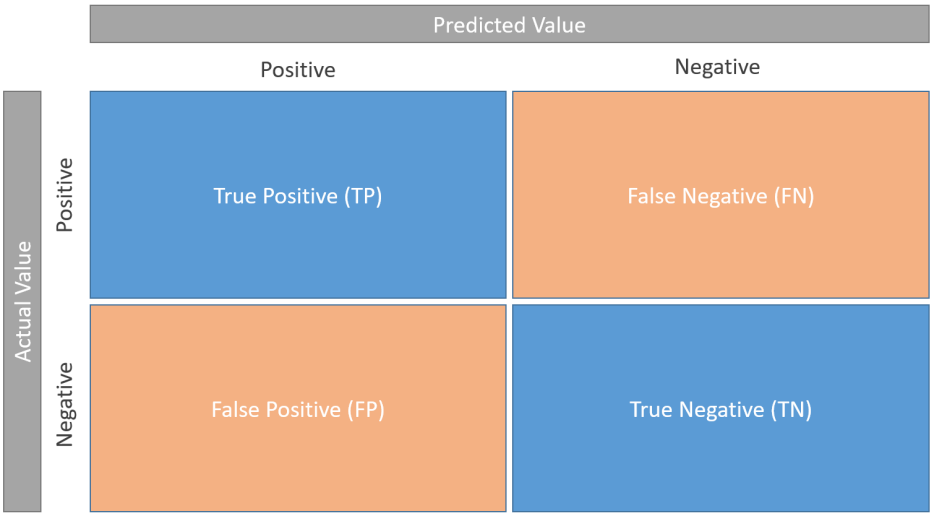

Figure 3.7: *The confusion matrix.*

The accuracy score for a class is the same as the recall for that class, which is defined using the true positives and false negatives, expressed as

$$\text{accuracy score} = \frac{TP}{TP + FN}.$$

The intersection over union (IoU) score

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

is similar to the accuracy, with the addition that it penalizes false positives.

The BF score BF_i for the aorta class in an image i is defined using the recall and precision values of that same image according to

$$\text{BF}_i = \frac{2 \cdot precision \cdot recall}{recall + precision},$$

$$\text{where } precision = \frac{TP}{TP + FP}$$

$$\text{and } recall = \frac{TP}{TP + FN}.$$

The BF score is used to evaluate how well the predicted boundary of the aorta pixels in an image matches the true boundary. The mean BF score

$$\text{Mean BF score} = \frac{1}{I} \sum_{i}^{I} \text{BF}_i,$$

is used for evaluation of multiple image and is defined as the mean of the BF scores for all the images I in the validation or test data set.

3.5 Related Work

To the best of the author's knowledge, no papers have been published with the aim of automatically segmenting the aorta in real-time flow CMR images during exercise. However, there have been related work sharing common elements. A semi-automatic nonrigid registration algorithm for segmentation of the aorta has been presented for real-time phase contrast CMR images which were acquired during rest and exercise [Odille et al., 2011]. For that algorithm to work, manual segmentation of the aorta in a reference frame was necessary to perform segmentation on the other frames in the sequence. Another semi-automatic algorithm for segmenting the aorta and pulmonary artery was developed using rigid motion tracking and active contour deformation [Bidhult et al., 2019]. This algorithm was developed for 2D PC CMR images acquired during rest and also required manual segmentation of the aorta and pulmonary artery in a reference frame. In another paper, the aorta lumen was segmented automatically in phase contrast ECG-triggered CMR images from patients with a dilated aorta as well as from healthy volunteers [Herment et al., 2010]. The algorithm uses a deformable surface model which requires a rectangular region of interest around the aorta provided by the user. A combination of an adapted U-Net and a recurrent neural network has also been used to segment the aorta in gated images during rest [Bai et al., 2018].

Chapter 4

Methodology

Matlab 2021a [Mathworks Inc, 2021] was used for all programming tasks of the project, such as pre-processing of images as well as training and evaluation of the neural networks. The U-Net models were set up using built in Matlab functions and were trained on Nvidia RTX 3090 Titan and Nvidia Titan RTX GPUs.

4.1 Data Sets

The ground truth delineations were made by physicians and medical students of the Cardiac MR Group as parts of research projects or in the clinic. The images were acquired using either Philips or Siemens MR scanners. The software Segment [Heiberg et al., 2010] was used to make these delineations. The images and delineations were extracted from .mat files into image files suitable for use in the training and evaluation of the networks. There were three kinds of images available for the project. The first being real-time flow images acquired *during exercise* using CS or EPI techniques, which is the data which this project aimed to segment. There were also real-time flow images acquired *during rest* using CS or EPI techniques, as well as gated images acquired during rest. An overview of the data used is shown in Table 4.1.

The real-time flow images during exercise corresponded to 9 participants, which were split into a validation and a test data set. Since the aim is to train networks which should perform well on exercise data, as many exercise images as possible should be used for validation and testing. Since there were only 9 participants imaged during exercise, all exercise images were chosen to be used for validation and testing. The validation data consisted of 4 participants and the test data of 5 non-overlapping participants. The training set consists of real-time images during rest and gated images, see Table 4.2. All of the data sets include images (a velocity encoded image and a magnitude image) and ground truth delineation. The total number of participants were 41 and some of them have been imaged with multiple imaging techniques and therefore have sequences in both the rest and exercise categories.

4.2 Data Normalization

The purpose of data normalization is to standardize images and compensate for the different VENCs that have been used when acquiring different image sequences. Normalization of the velocity encoded images were performed by calculating the median

	CS exercise	EPI exercise	CS rest	EPI rest	Gated (rest)
#images	4734	5609	2606	4098	1610
#participants	9	9	9	21	29

Table 4.1: *The number of images and participants in each category based on imaging technique and if the images were acquired during exercise or during rest.*

	Training	Validation	Testing
#images	8314	5004	5339
#participants	41	4	5

Table 4.2: *The number of images and participants in the training, validation and testing sets.*

VENC for all sequences in the data sets and adjusting each pixel value according to

$$p_{norm} = \frac{p_{VE} \cdot VENC_{image}}{VENC_{median}},$$

where p_{norm} is the normalized pixel, p_{VE} is the original pixel in the velocity encoded image and $VENC_{image}$ is the VENC of the current image.

4.3 Data Augmentation

The images were also augmented using translation and rotation. Images were translated to make sure that the aorta was not exactly in the center of all images. The cropping was made by setting the spatial size of the cropped image to correspond to 100 x 100 mm and finding the center of the aorta. Before cropping, the aorta mask was offset ±10 mm from the aorta center, see Figure 4.1. The rotation of each image was then made by rotating it randomly within a range of ±20°. This range was chosen to approximate how much a participant could move their body while exercising.

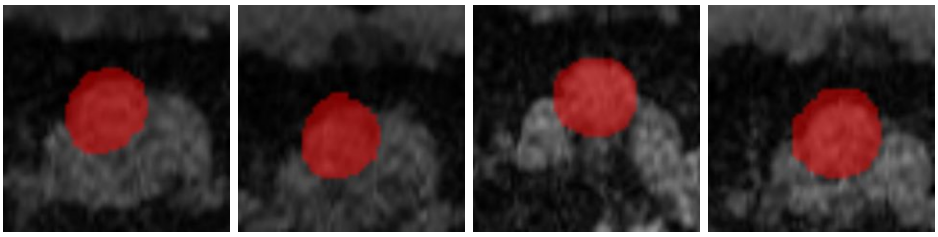

Figure 4.1: *Example images with overlay of the aorta illustrating the cropping and translation performed on the data.*

4.4 CNN Architecture

In order to have a reference to evaluate the influence of using different values and settings when training the networks, a baseline network was created. The baseline network's settings were chosen partially based on the original design of the U-Net

model where the encoder depth is equal to four and the loss function is binary cross-entropy. Using two feature channels for the input was also chosen as well as training the network for 10 epochs and with a constant learning rate of $\eta = 0.001$. All the settings are shown in Table 4.3.

Table 4.3: *Parameters for the baseline network. Italicized parameters are modified and studied individually in this project.*

# Input Feature Channels	*2*
Encoder depth	*4*
Loss function	*Cross-entropy*
L2 regularization	*None, i.e.* $\alpha = 0$
Adaptive learning rate	*None*
Learning rate	$\eta = 0.001$
Epochs	10
Minibatch size	20
Optimizer	Adam

4.4.1 Number of Input Channels

The input images to the U-Net model had a spatial size 64x64 pixels. This size was chosen since it was close to the cropped size of the images before resizing, which was $50 - 60$ pixels. The number of input channels was one, two or three, see Table 4.4.

Table 4.4: *Parameter for the feature channel input training setups.*

		baseline	
# Input Feature Channels:	1	2	3

The first channel was the magnitude image (see the right panel in Figure 3.1) and using a second channel meant an addition of the velocity-encoded flow image (see the left panel in Figure 3.1). When using three input channels, the third channel was a type of angiography image (Figure 4.2). Each pixel p_a of the third image is calculated as

$$p_a = 2 \cdot |p_{VE}| \cdot p_m$$

using the values in the corresponding pixel p_m in the magnitude and p_{VE} in the velocity encoded flow image. The purpose of this calculation is to find the blood vessels in the image (hence the reason of referring to it as an angiography image). This is possible since the areas of the body which does not have a flow will either have a lower pixel value in the magnitude image (corresponding to darker pixels) or a lower pixel value in the flow image, which will result in a lower value in the angiography calculation of that pixel. The aorta pixels will both have a higher value in the magnitude image as well as in the velocity image, resulting in a higher value in the angiography image.

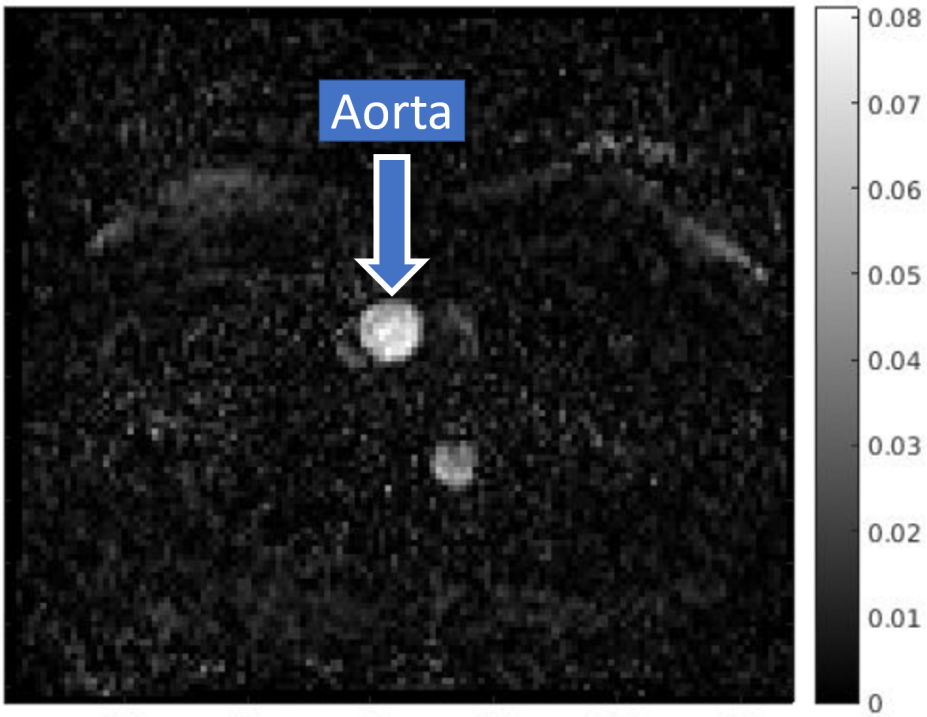

Figure 4.2: *Example of the angiography channel used for the three channel input.*

The three channel image can be visualized as an RGB image, see Figure 4.3.

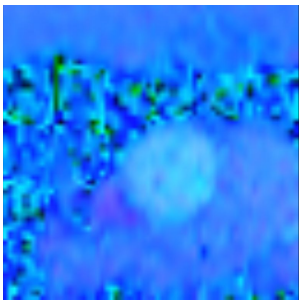

Figure 4.3: *Example of a 3 channel image.*

4.4.2 Encoder Depth

The number of downsampling operations in the contracting part of the U-Net model defines the encoder depth of the network, i.e. if there are four downsampling stages and four corresponding upsampling stages the network has an encoder depth equal to four. The original U-Net model has an encoder depth equal to 4, see Figure 3.6. In this thesis, using an encoder depth of three or five is also explored, see Table 4.5. The larger the encoder depth, the more complex the network gets. This means that more complex patterns can be detected but this also increases the risk of overfitting the model. A larger encoder depth means a larger model and hence also longer computational time for training as well as longer execution time.

		baseline	
Encoder Depth:	3	4	5

Table 4.5: *Parameter setups for the networks trained with different the encoder depths.*

4.4.3 Loss Function

For the combination layer, the value of γ was set to 0.5. This means that the influence between the binary cross-entropy and Dice parts of the loss function are equal.

Table 4.6: *Parameter setups for the networks trained with different the encoder depths.*

		baseline	
Loss Function:	Dice	Cross-entropy	Combination

4.5 Training Settings and Parameters

4.5.1 L2 Regularization

The values of α used for the different networks are displayed in Table 4.7.

Table 4.7: *Parameter setups for the networks trained with L2 regularization with different values on α.*

	baseline					
$\alpha =$	0, no regularization	0.0001	0.0005	0.001	0.01	0.1

4.5.2 Adaptive Learning Rate

With a learning rate drop period of $T = 2$ epochs, the following learning rate drop factors were used: $\lambda = 0.1, 0.2, 0.5, 0.7$ with the initial learning rate being $\eta_0 = 0.001$. These were compared to the baseline network were the learning rate was constantly $\eta = 0.001$.

Table 4.8: *Parameter setups for the networks trained with different learning rate drop factors λ. The initial learning rate η_0 was the same as well as the learning rate drop period T.*

	baseline				
$\eta_0 =$	0.001	0.001	0.001	0.001	0.001
$T =$	-	2	2	2	2
$\lambda =$	-	0.1	0.2	0.5	0.7

4.6 Flow Quantification

Three scores are used to measure the network's performance: accuracy, IoU and mean BF score (as described in Section 3.4.2). The network is also evaluated based on volumes which are calculated for each sequence. A sequence s includes multiple images i. The number of images in a sequence s is denoted I, though I is a number that differs between the sequences depending on how many images were delineated manually. There are three different kinds of sequence volumes:

- V^s_{GT} : The ground truth volume for the sequence s calculated in Segment using the manual delineations.
- V^s_{Calc} : The volume for sequence s calculated by a script where the delineations will have been resized in order to match the input of the neural network. Due to the resizing, this volume will differ slightly to the ground truth V^s_{GT}.
- V^s_{Segm} : The volume for a sequence based on the segmentations made by the network. The calculations are made using the same script as V^s_{Calc} and they therefore share the same error due to resizing.

The process of calculating V^s_{Calc} and V^s_{Segm} are illustrated by Figure 4.4. They are calculated in the same manner, the only difference is which pixel values p_a are used in the calculation of the flow f_i for an image i in

$$f_i = r^2 \sum_{a}^{A} p_a.$$

r is the resolution [pixels/cm] for image i, p_a are the values of the pixels belonging to the aorta class and A is the number of aorta pixels in image i. V^s_{Calc} is specified by the manual delineations and V^s_{Segm} by the segmentations made by the network. Therefore, V^s in

$$V^s = \Delta t \sum_{i}^{I} f_i$$

denotes either V^s_{Calc} or V^s_{Segm}. Here, Δt [s] is the time between two image acquisitions and I is the the number of images in the sequence s.

In order to estimate the error caused by the resizing of the images, the differences between V^s_{GT} and V^s_{Calc} can be studied. To evaluate the performance of a network, the sequence volumes based on segmentations V^s_{Segm} are then compared to V^s_{GT}. Here it is important to note that the differences between V^s_{Segm} and V^s_{GT} are caused both by the image resizing error as well as the difference between segmentations made by the network and the manual delineations.

To estimate the difference between two methods for obtaining sequence volumes, the mean error (ME) and mean absolute error (MAE) were used. When comparing V^s_{GT} and V^s_{Calc} for a sequence, these errors will be denoted $ME^{GT:Calc}$ and $MAE^{GT:Calc}$. Similarly, when comparing V^s_{GT} and V^s_{Segm}, these will be denoted $ME^{GT:Segm}$ and $MAE^{GT:Segm}$.
The error e_s [%] for a sequence s is defined as

$$e_s = \frac{V^s_{GT} - V^s_{Calc}}{V^s_{GT}}$$

and is used to create a vector $\mathbf{e^S}$ containing all errors for all S sequences according to

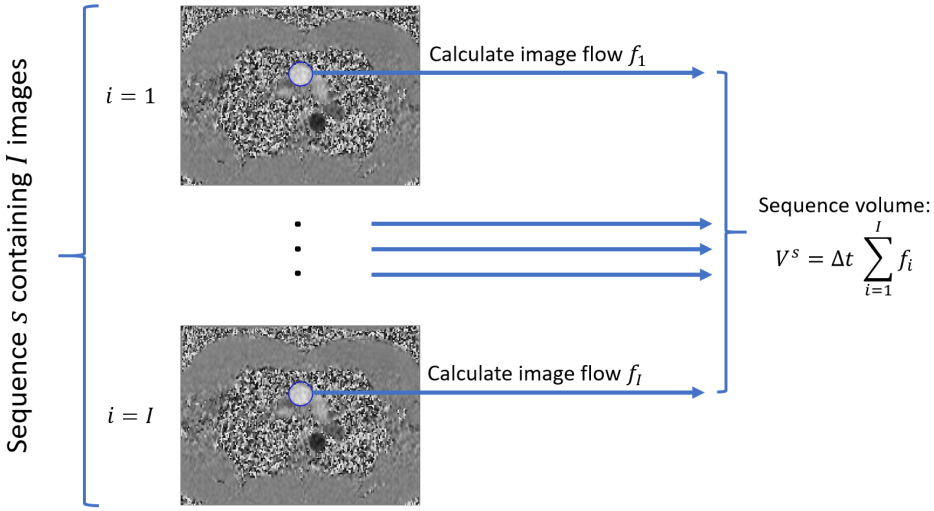

Figure 4.4: *Schematic of the calculation of sequence volume V^s for a sequence s containing I images.*

$$\mathbf{e^S} = \begin{pmatrix} e_1 & e_2 & \dots & e_S \end{pmatrix}.$$

This error vector is then used to define the mean error $ME^{GT:Calc}$ as

$$ME^{GT:Calc} = \mu(\mathbf{e^S}) \pm \sigma(\mathbf{e^S})$$

using the mean $\mu(\mathbf{e^S})$ and standard deviation $\sigma(\mathbf{e^S})$ of said vector $\mathbf{e^S}$. The mean absolute error

$$MAE^{GT:Calc} = \mu(\mathbf{a^S}) \pm \sigma(\mathbf{a^S})$$

is defined similarly using the absolute error

$$a_s = \frac{|V^s_{GT} - V^s_{Calc}|}{V^s_{GT}}$$

for a sequence s, which is used to form the absolute error vector

$$\mathbf{a^S} = \begin{pmatrix} a_1 & a_2 & \dots & a_S \end{pmatrix}.$$

The mean error and mean absolute error are defined in the same manner for $ME^{GT:Segm}$ and $MAE^{GT:Segm}$.

Chapter 5

Results

5.1 Flow Quantification

The mean error $ME^{GT:Calc}$ and mean absolute error $MAE^{GT:Calc}$ between the ground truth volumes and the calculated volumes are shown in Table 5.1 and in Figures 5.1, 5.2 and 5.3 for 106 sequences from 9 participants in the validation and test data, i.e. the real-time flow acquired during exercise data sets. The linear curve fitted to the values in Figure 5.1 has the slope $k = 1.0009$ and intercept $m = -2.491$, hence it is barely distinguishable from the identity line.

Table 5.1: *Error estimation between calculated sequence volumes V_{Calc} compared to ground truth volumes V_{GT}.*

$ME^{GT:Calc}$ [%]	0.21 ± 3.59
$MAE^{GT:Calc}$ [%]	2.96 ± 2.02

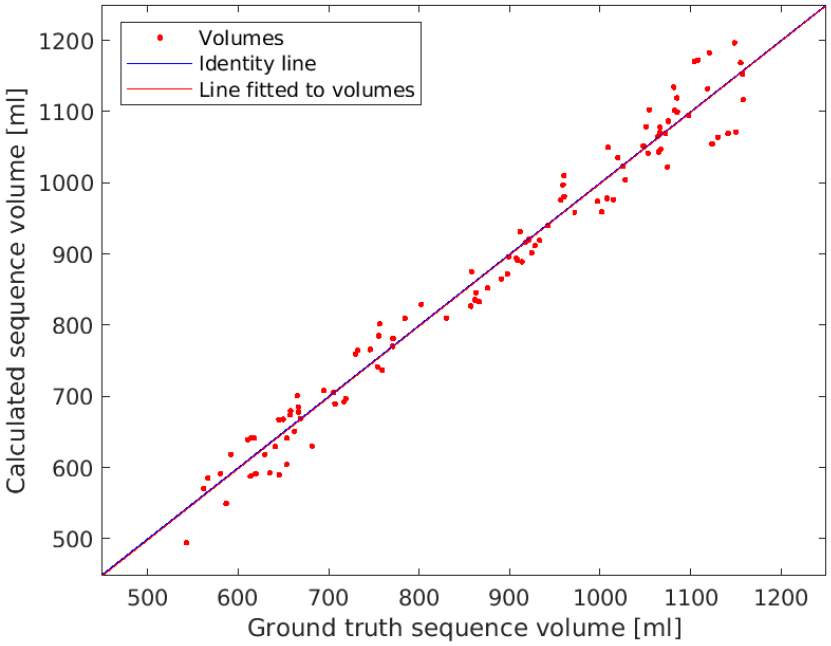

Figure 5.1: *Scatter plot for the ground truth sequence volumes and calculated sequence volumes. For the line fitted to the points, the slope* $k = 1.0009$ *and the intercept* $m = -2.491$.

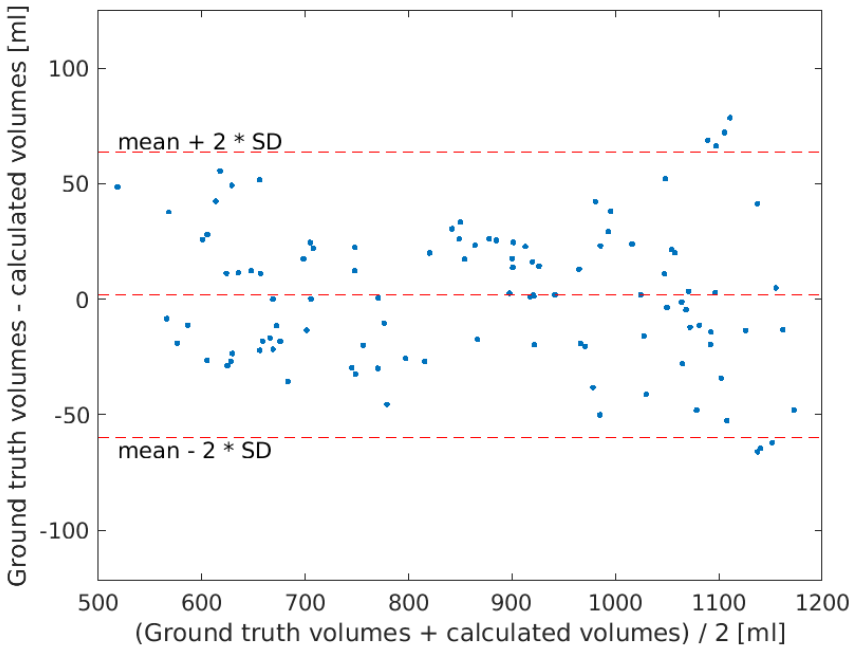

Figure 5.2: *Bland-Altman plot for the ground truth sequence volumes and calculated sequence volumes.*

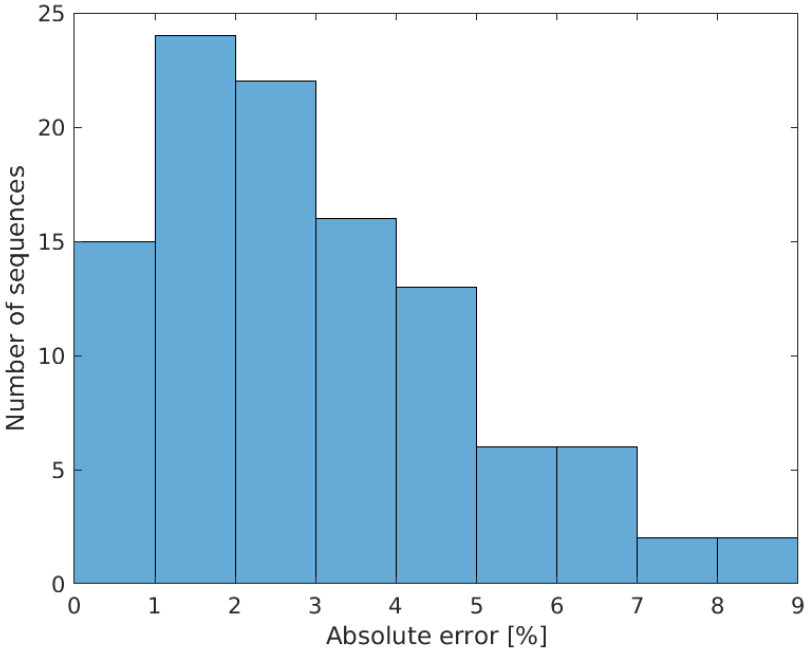

Figure 5.3: *Histogram depicting the absolute errors for the sequences.*

5.2 Number of Input Channels

The results for using 1, 2 or 3 feature channels as input to the network are shown in Figure 5.4 and Table 5.2. The training times and network sizes are displayed in Table 5.3.

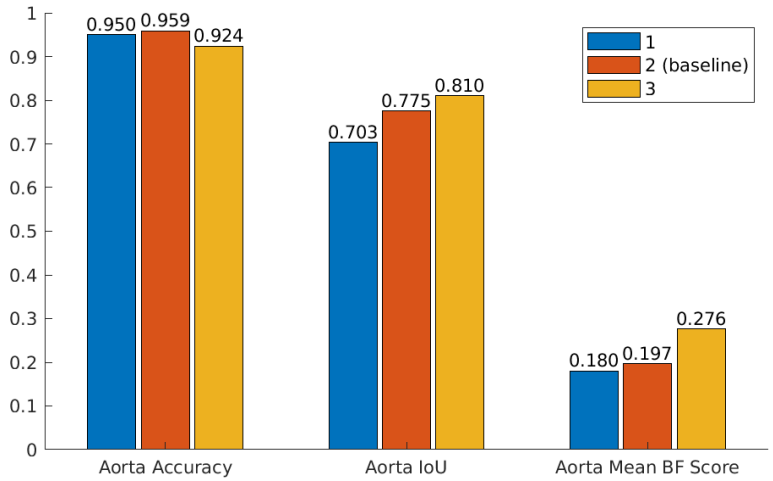

Figure 5.4: *Accuracy, IoU and Mean BF score for the aorta class for networks with different numbers of input feature channels.*

Table 5.2: *Using different numbers of input feature channels.*

		baseline	
	1 channel	2 channels	3 channels
$ME^{GT:Segm}$ [%]	3.50 ± 7.42	-0.17 ± 4.39	-0.21 ± 4.93
$MAE^{GT:Segm}$ [%]	6.45 ± 5.00	3.39 ± 2.75	4.17 ± 2.58

Table 5.3: *Training time and network size for the networks with different number of input channels.*

		baseline	
	1 channel	2 channels	3 channels
Training time [min]	16	21	17
Network size [MB]	116	117	116

5.3 Encoder Depth

Figure 5.5 and Table 5.4 display the results for the networks trained with an encoder depth of 3, 4 and 5. Table 5.5 show the training times and network sizes.

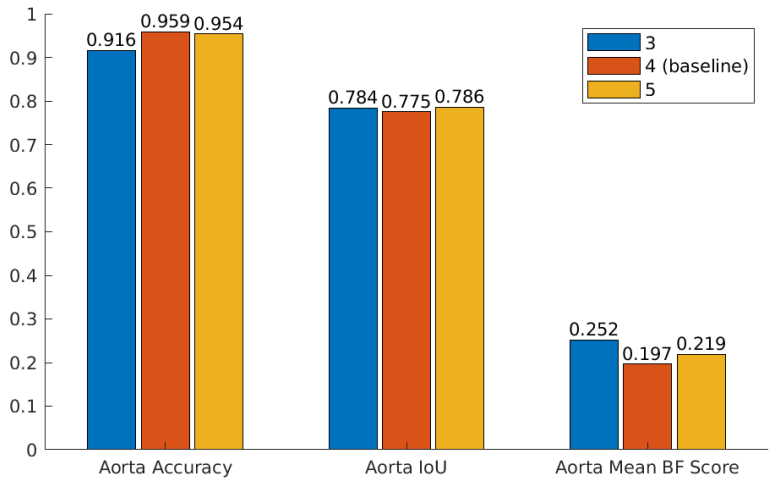

Figure 5.5: *Accuracy, IoU and Mean BF score for the aorta class for networks with varying encoder depths.*

Table 5.4: *Using different numbers of encoder depths.*

		baseline	
	3	4	5
$ME^{GT:Segm}$ [%]	1.19 ± 5.73	-0.17 ± 4.39	-1.06 ± 5.37
$MAE^{GT:Segm}$ [%]	4.74 ± 3.37	3.39 ± 2.75	4.63 ± 2.85

Table 5.5: *Training time and network size for the networks with different encoder depths.*

		baseline	
	3	4	5
Training time [min]	17	21	25
Network size [MB]	29	117	437

5.4 Loss Function

The results for using Dice, cross-entropy or a combination of the two as loss function when training the network is shown in Figure 5.6 and Table 5.6, with training times and network sizes displayed in Table 5.7. The training plots for the Dice loss function is found in Figure 5.7.

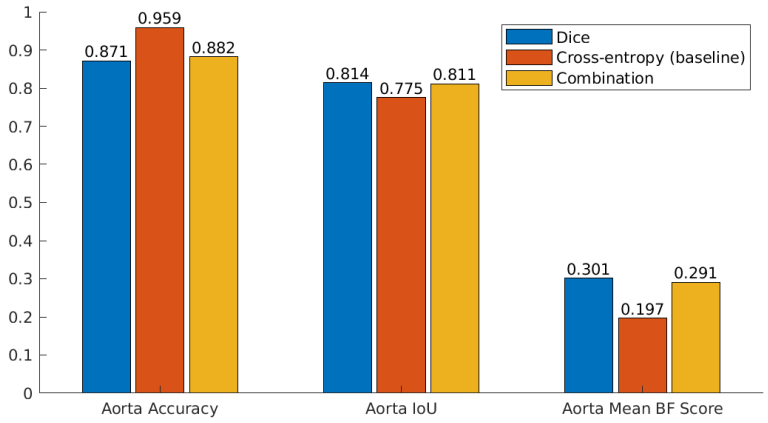

Figure 5.6: *Accuracy, IoU and Mean BF score for the aorta class for networks trained with different loss functions.*

Table 5.6: *Using different loss functions.*

	Dice	baseline Cross-entropy	Combination
$ME^{GT:Segm}$ [%]	-1.64 ± 4.87	-0.17 ± 4.39	-1.78 ± 4.89
$MAE^{GT:Segm}$ [%]	4.44 ± 2.52	3.39 ± 2.75	4.49 ± 2.58

Table 5.7: *Training time and network size for the networks trained with different loss functions.*

	Dice	baseline Cross-entropy	Combination
Training time [min] [%]	20	21	21
Network size [MB] [%]	105	117	114

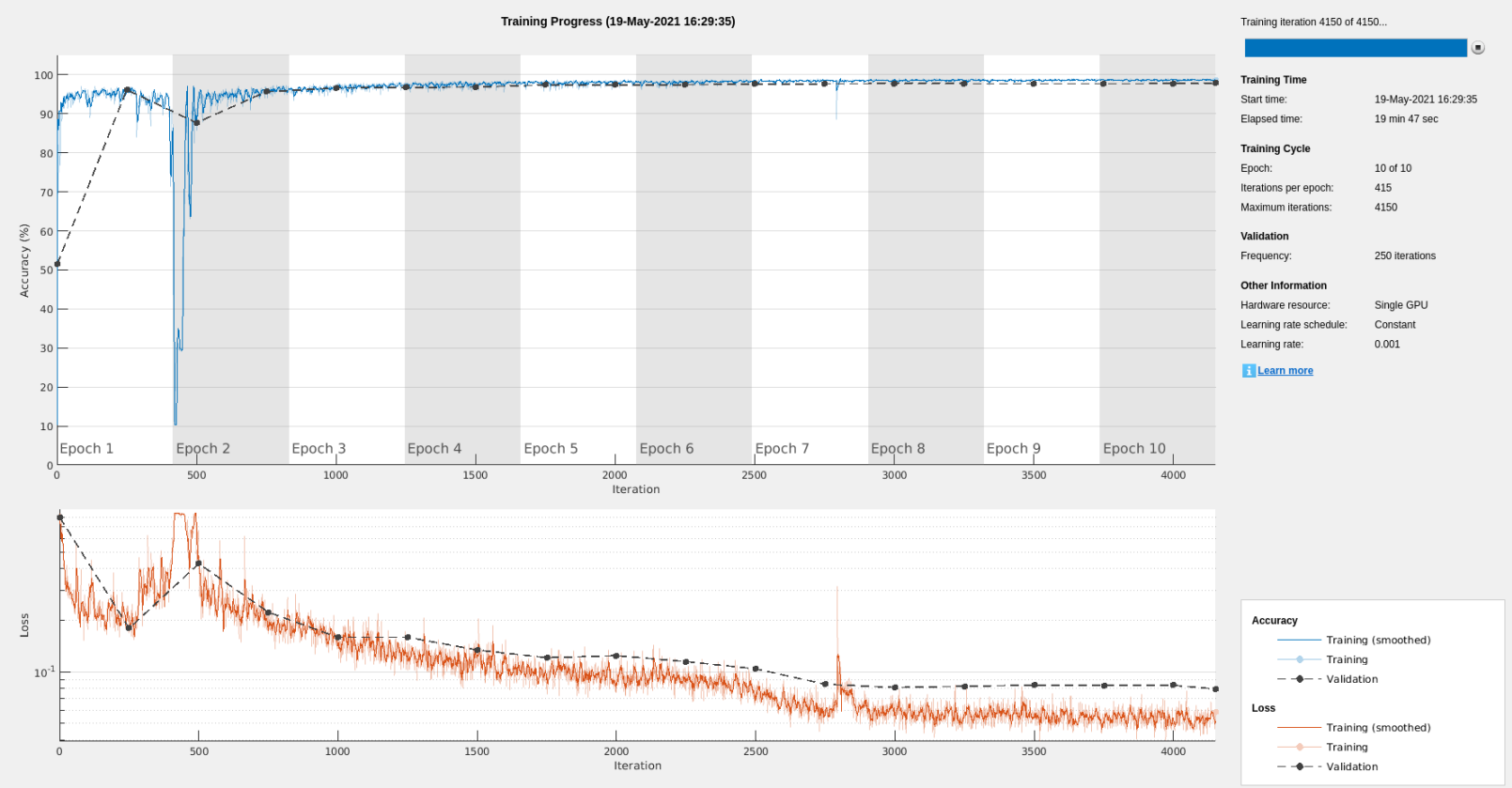

Figure 5.7: *Training plots for the model using the Dice loss function. The upper plot shows the training and validation accuracy, the lower plot shows the training and validation loss for the 10 epochs of training.*

5.5 L2 Regularization

The results of using no L2 regularization compared to using it with varying values of α are displayed in Figure 5.8 and Table 5.8. The various training times and network sizes are shown in Table 5.9.

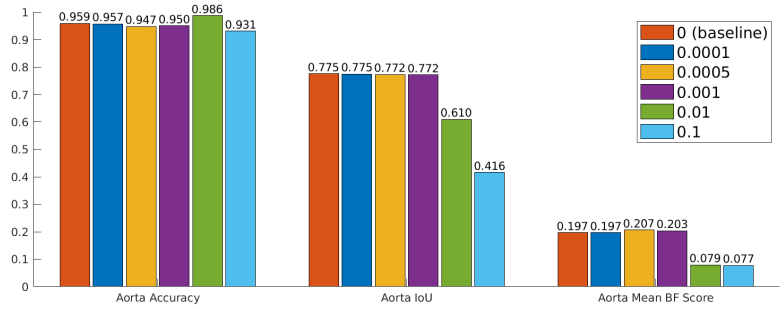

Figure 5.8: *Accuracy, IoU and Mean BF score for the aorta class for networks with different values of α.*

Table 5.8: *L2 regularization with varying values of α compared to baseline network without regularization.*

	baseline 0, no regularization	$\alpha = 0.0001$	$\alpha = 0.0005$	$\alpha = 0.001$	$\alpha = 0.01$	$\alpha = 0.1$
$ME^{GT:Segm}$ [%]	-0.17 ± 4.39	0.02 ± 5.29	0.68 ± 5.15	2.44 ± 5.35	-1.80 ± 7.06	-57.58 ± 22.42
$MAE^{GT:Segm}$ [%]	3.39 ± 2.75	4.24 ± 3.09	4.12 ± 3.11	4.31 ± 3.97	5.80 ± 4.33	7.58 ± 22.42

Table 5.9: *Training time and network size for the networks with and without L2 regularization.*

	baseline 0, no regularization	$\alpha = 0.0001$	$\alpha = 0.0005$	$\alpha = 0.001$	$\alpha = 0.01$	$\alpha = 0.1$
Training time [min]	21	21	20	20	27	20
Network size [MB]	117	116	96	104	79	67

,

5.6 Adaptive Learning Rate

The performance of networks with adaptive learning rates with different values of λ are shown in Figure 5.9 and Table 5.10. The training times and network sizes of these networks are shown in Table 5.11.

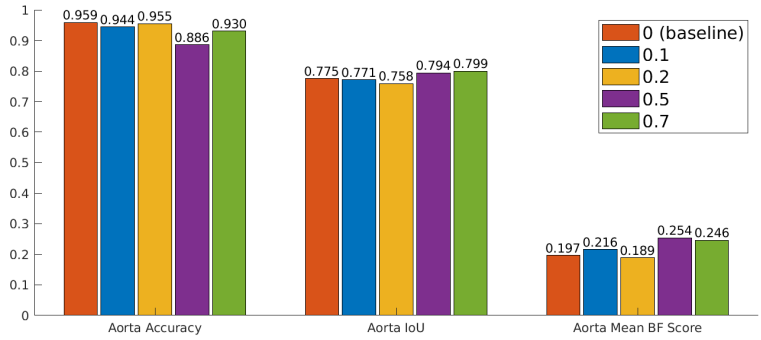

Figure 5.9: *Accuracy, IoU and Mean BF score for the aorta class for networks with varying learning rates compared to the baseline network with a constant learning rate.*

Table 5.10: *Learning rate drop period of $T = 2$ with different learning rate drop factors λ.*

	baseline				
	Constant learning rate	$\lambda = 0.1$	$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 0.7$
$ME^{GT:Segm}$ [%]	-0.17 ± 4.39	-0.86 ± 5.25	0.19 ± 5.41	-1.14 ± 4.90	-0.04 ± 4.82
$MAE^{GT:Segm}$ [%]	3.39 ± 2.75	4.40 ± 2.93	4.46 ± 3.01	4.27 ± 2.59	3.95 ± 2.70

Table 5.11: *Training time and network size for the networks with and without adaptive learning rates.*

	baseline				
	Constant learning rate	$\lambda = 0.1$	$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 0.7$
Training time [min] [%]	21	20	20	20	21
Network size [MB] [%]	117	115	115	115	115

5.7 Final Network and Results for Test Data

In order to choose the settings for the final network, mean value $\mu(\mathbf{a^S})$ of $MAE^{GT:Segm}$ was considered the most important score. This is because it is important to have a small bias in the sequence volumes and this is indicated by $\mu(\mathbf{a^S})$. $\mu(\mathbf{e^S})$ of $ME^{GT:Segm}$ is not as reliable to estimate this since it could be 0 if there are both negative and positive errors e_s. Hence, the setup with the smallest $\mu(\mathbf{a^S})$ for each tested parameter was chosen which results in the parameter settings for the final network, see Table 5.12. As this coincided to be the same settings as for the baseline network, no new network was trained. Still, this final network is used to evaluate the test data, see Tables 5.13 and 5.14 as well as Figures 5.11 and 5.12. Four example segmentations by the final network on the test data set are shown in Figure 5.10.

Table 5.12: *Parameters for the final network.*

# Input Feature Channels	2
Encoder depth	4
Loss function	Cross-entropy
L2 regularization	None, i.e. $\alpha = 0$
Adaptive learning rate	None

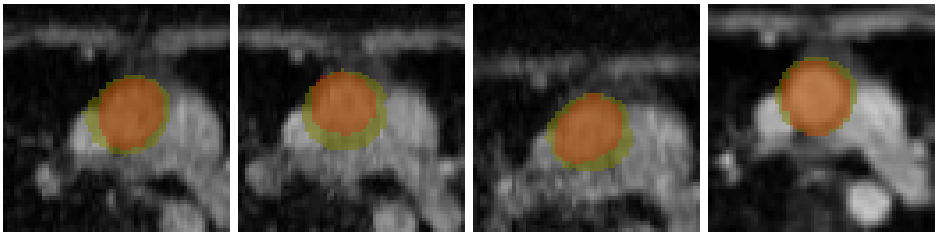

Figure 5.10: *Example images from the segmentation of the test data. The orange color overlay is the manual segmentation, the yellow is the segmentation made by the final network.*

Table 5.13: *Error estimation between segmented sequence volumes V_{Segm} compared to ground truth volumes V_{GT} for the test data.*

$ME^{GT:Segm}$ [%]	-2.18 ± 3.12
$MAE^{GT:Segm}$ [%]	2.92 ± 2.42

Table 5.14: *Accuracy, IoU and Mean BF score for the aorta class for the final network evaluated on the test data.*

Aorta accuracy	0.974
Aorta IoU	0.775
Aorta mean BF score	0.197

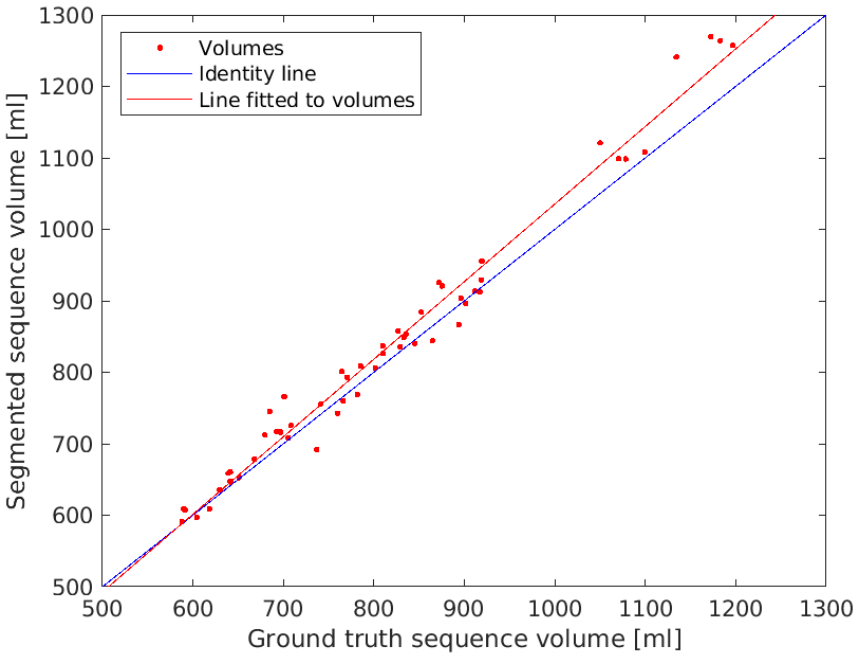

Figure 5.11: *Scatter plot of the ground truth sequence volumes vs. segmented sequence volumes for the test data. The linear fitted line is* $k = 1.0853$ *and* $m = -49.9157$.

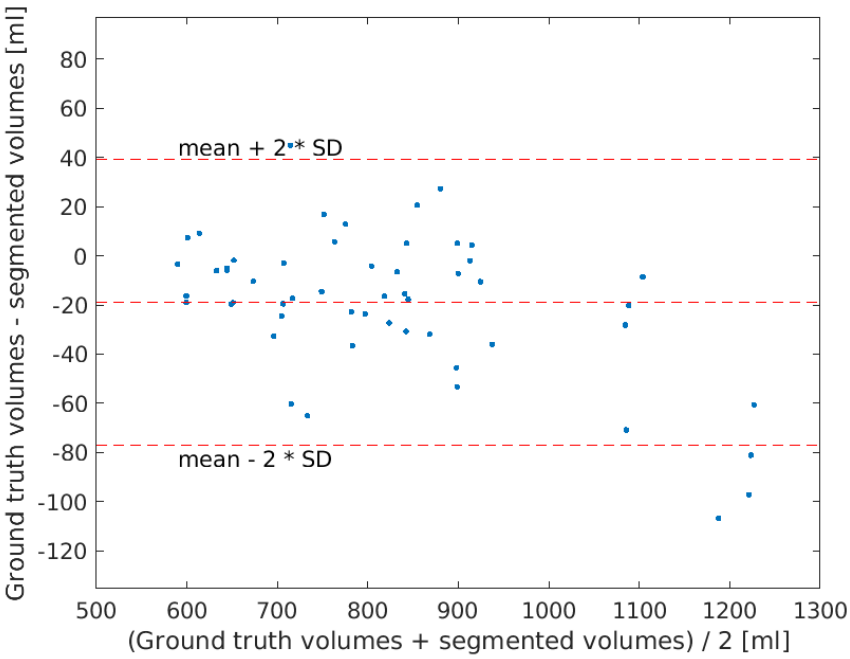

Figure 5.12: *Bland-Altman plot for the ground truth volumes vs. the segmented sequence volumes for the test data.*

Chapter 6

Discussion

A method for segmentation of the aorta in real-time flow images was developed by training U-Net models using various variations of the network architecture and training settings. The final network was successful at segmenting real-time flow images acquired during exercise while having been trained only on real-time flow images acquired during rest and on gated images. An in-depth discussion of the results for each training setting and network parameter used in the project as well as limitations and future work are presented below.

6.1 Flow Quantification

As shown in Table 5.1 and Figures 5.1 and 5.2, there is a difference between the ground truth sequence volumes and the corresponding calculated sequence volumes. Both are based on manual delineations, but the difference in volume stems from an error due to the resizing of images and masks used in the calculated sequences. The volume estimations seems to be fairly evenly distributed around their ground truth counterparts, with a slight tendency for larger differences in volumes as the volume increases, see the upper right in Figure 5.1. The ground truth and calculated volumes are also compared in the Bland-Altman plot in Figure 5.2 where the points should ideally be located along the horizontal axis at $y = 0$ if the sequence volumes were the same. However, in Figure 5.2, the points are spread out vertically. The tendency for larger differences can also be seen where the four values are above the mean + 2 standard deviations dotted line and three values are below the mean - 2 standard deviations. As shown by the histogram in Figure 5.3, 90 out of the total 106 sequence volumes (85% of the sequence volumes) have an absolute error $a_s \leq 5\%$ while the remaining 16 (15%) of the sequence volumes have and absolute error $5\% < a_s \leq 9\%$.

Hence, it is clearly established that the resizing of images in order to set the input spatial size of all images to the networks to 64x64 pixels do cause an error in the binary masks used to classify each pixel as belonging to the aorta or background class. This error manifests itself as the errors in the sequence volume calculations which makes the $ME^{GT:Segm}$ and $MAE^{GT:Segm}$ for the setups below less reliable.

6.2 Number of Input Channels

As shown in Figure 5.4, the baseline network with 2 input channels has the highest accuracy for the aorta class, while the IoU and mean BF score is higher for the network

using 3 input channels. The $ME^{GT:Segm}$ and $MAE^{GT:Segm}$ of the baseline network are similar to those of the network with 3 input channels, see Table 5.2. Surprisingly, the baseline network has a longer training time and slightly larger network size even though the network with 3 channels contain more parameters.

6.3 Encoder Depth

Figure 5.5 shows the baseline network with an encoder depth of 4 having the highest accuracy while the network with encoder depth 5 has the highest IoU and encoder depth 3 producing the highest mean BF score. The baseline network had the best $ME^{GT:Segm}$ and $MAE^{GT:Segm}$, see Table 5.4. Furthermore, the sizes of the trained networks increased significantly with a higher encoder depth as can be seen in Table 5.5, with the network of encoder depth 5 is more than 3 times the size of the baseline network. The size of a network may be important to the viability of using it in a clinical setting, since a larger network means longer execution times which may be infeasible for practical use.

6.4 Loss Function

The baseline network using the cross-entropy loss function had the best accuracy while using the Dice loss function resulted in the best IoU and mean BF score for the aorta class, see Figure 5.6. $ME^{GT:Segm}$ and $MAE^{GT:Segm}$ were similar for the three networks, see Table 5.6. The training times were almost identical for the three networks and the differences in network sizes were small, see Table 5.7. As can be seen in the training plots in Figure 5.7, the network using Dice loss has not converged after 10 epochs. A possible extension of this project could be to train this network for more epochs and compare that result to the baseline.

6.5 L2 Regularization

For the networks used to compare the use of L2 regularization to the baseline without, the best accuracy was gained with a $\lambda = 0.01$ (see Figure 5.8). The baseline network and network with $\lambda = 0.0001$ are tied for the best IoU. The network with $\lambda = 0.0005$ had the best mean BF score. Concerning $ME^{GT:Segm}$ and $MAE^{GT:Segm}$, both the baseline network and $\lambda = 0.0001, 0.0005$ had similar results while $\lambda = 0.1$ clearly produces much poorer results. The training times for the networks were similar (20-27 minutes) but the network size was significantly smaller for larger values of α, see Table 5.9. This is expected, since a large value of α penalizes large weight sizes more than small values of α do.

6.6 Adaptive Learning Rate

The baseline network with a constant learning rate produced the best accuracy, while the network with $\lambda = 0.7$ gave the best IoU and the network with $\lambda = 0.5$ resulted in the highest mean BF score (see Figure 5.9). According to Table 5.10, the $ME^{GT:Segm}$ and $MAE^{GT:Segm}$ are similar for all networks. The training times and network sizes are almost identical (20-21 minutes and 115-117 MB, see Table 5.11), which is expected since the learning rate should not influence the network size.

6.7 Final Network and Results for Test Data

As can be seen in the scatter plot in Figure 5.11, the calculated sequence volumes of the test data set are generally too large compared to the ground truth volumes. This can also be seen in the Bland-Altman plot in Figure 5.12, where the difference between the ground truth and calculated volumes are generally negative. This is visible too in Table 5.13 with the negative mean $\mu(\mathbf{e^S})$ of $ME^{GT:Segm}$. Noticeably, the standard deviation $\sigma(\mathbf{e^S})$ of $ME^{GT:Segm}$ and $\sigma(\mathbf{a^S})$ of $MAE^{GT:Segm}$ are larger for the test set than it was for the validation set. The aorta accuracy on the test set was 0.974 (see Table 5.14) which is smaller than for the validation set (0.959) but had the same IoU of 0.775 and the same mean BF score of 0.197.

6.8 Limitations in the Project and Future Work

There are some limitations that one should bear in mind when analyzing the results and when considering how to proceed in the task of automatically segmenting the aorta in real-time flow CMR images acquired during exercise. Firstly, only U-Net models have been explored in this thesis and while U-Net networks have provided good results in similar fields, it is possible that other CNN architectures (or other methods entirely) can produce better results.

Secondly, the reliability of the $ME^{GT:Segm}$ and $MAE^{GT:Segm}$ for the setups in this project are somewhat unreliable, as explained in Section 4.6 and shown in Section 5.1. The problems caused by image resizing can be bypassed altogether by cropping the images without resizing them. This would result in the image not corresponding to an actual size of 100 x 100 mm, but instead the images could be cropped to simply correspond to 64 x 64 pixels which would lead to varying actual sizes.

As for many other segmentation tasks, the models could be improved by using more data for training, validation and test. This includes adding more subjects and sequences, but also to include data from other hospitals, MR scanners and acquisition methods in order to improve robustness. If more data could be obtained for participants during exercise it would be possible to use exercise data for training as well as for validation and testing. Before a future model could be considered for clinical use, it is necessary to further analyze the quality of the manual delineations, such as investigating inter- and intra observer variability. Even if a CNN model could hypothetically reach perfect accuracy, IoU, mean BF scores and volume calculations, the network would be useless if the data it is trained on is not adequate.

6.9 Conclusion

In conclusion, this project has shown that it is possible to train a U-Net model on images acquired during rest (gated and real-time flow) and use it to segment real-time flow images acquired during exercise. The effect of using different numbers of input channels and encoder depths of the architecture were studied, and the best results were given using both the velocity-encoded and magnitude images as input and using an encoder depth of 4. Using cross-entropy as a loss function scored the best results, though using Dice could possible improve the model if trained longer. Using L2 regularization or adaptive learning rate did not improve performance. Before the final model trained in this project could be used clinically it is necessary to add more data as well as study how inter and intra observer variability affects the performance of the model.

Bibliography

[Aggarwal, 2018] Aggarwal, C. C. (2018). *Neural Networks and Deep Learning.* Springer Link. Accessed at `https://link-springer-com.ludwig.lub.lu.se/chapter/10.1007/978-3-319-94463-0\8`.

[Bai et al., 2018] Bai, W., Suzuki, H., Qin, C., Tarroni, G., Oktay, O., Matthews, P. M., and Rueckert, D. (2018). Recurrent neural networks for aortic image sequence segmentation with sparse annotations. *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018.*

[Bidhult et al., 2019] Bidhult, S., Hedström, E., Carlsson, M., Töger, J., Steding-Ehrenborg, K., Arheden, H., Aletras, A. H., and Heiberg, E. (2019). A new vessel segmentation algorithm for robust blood flow quantification from two-dimensional phase-contrast magnetic resonance images. *Clin Physiol Funct Imaging 39(5):327-338.*

[Craven et al., 2020] Craven, T., Tsao, C., Gerche, A. L., and et al. (2020). Exercise cardiovascular magnetic resonance: development, current utility and future applications. *J Cardiovasc Magn Reson 22, 65.*

[Encyclopaedia Britannica, 2020] Encyclopaedia Britannica (2020). Human cardiovascular system. `https://www.britannica.com/science/human-cardiovascular-system`. Accessed on 2021-05-03.

[Geethanath et al., 2013] Geethanath, S., Reddy, R., Konar, A. S., Imam, S., Sundaresan, R., Babu, D. R., and Venkatesan, R. (2013). Compressed sensing mri: A review. *Critical Reviews in Biomedical Engineering 41(3):183-204.*

[Heiberg et al., 2010] Heiberg, E., Sjögren, J., Ugander, M., Carlsson, M., Engblom, H., and Arheden, H. (2010). Design and validation of segment - freely available software for cardiovascular image analysis. *BMC Med. Imaging 2010:10:1.*

[Herment et al., 2010] Herment, A., Kachenoura, N., Lefort, M., Bensalah, M., Dogui, A., Frouin, F., Mousseaux, E., and Cesare, A. D. (2010). Automated segmentation of the aorta from phase contrast mr images: validation against expert tracing in healthy volunteers and in patients with a dilated aorta. *J Magn Reson Imaging 31(4):881-8.*

[Jadon, 2020] Jadon, S. (2020). A survey of loss functions for semantic segmentation. *2020 IEEE International Conference on Computational Intelligence in Bioinformatics and Computational Biology.*

[Jin and Simonetti, 2013] Jin, N. and Simonetti, O. (2013). Free-breathing real-time flow imaging using epi and shared velocity encoding. *SCMR 2013.*

[Kingma and Ba, 2017] Kingma, D. and Ba, J. (2017). Adam: A method for stochastic optimization. *arXiv 1412.6980*.

[Krogh and Hertz, 1991] Krogh, A. and Hertz, J. A. (1991). A simple weight decay can improve generalization. *NIPS'91: Proceedings of the 4th International Conference on Neural Information Processing Systems*, pages 950–957.

[Lotz et al., 2002] Lotz, J., Meier, C., Leppert, A., and Galanski, M. (2002). Cardiovascular flow measurement with phase-contrast mr imaging: basic facts and implementation. *RadioGraphics 22:651-671*.

[Ma et al., 2020] Ma, J., Song, Y., Tian, X., Hua, Y., Zhang, R., and Wu, J. (2020). Survey on deep learning for pulmonary medical imaging. *Front Med. 14(4):450-469*.

[Mathworks Inc, 2021] Mathworks Inc, Natick Massachusetts, U. S. (2021). Matlab 2021a.

[Milletari et al., 2016] Milletari, F., Navab, N., and Ahmadi, S.-A. (2016). V-net: Fully convolutional neural networks for volumetric medical image segmentation. *arXiv 1606.04797*.

[Nayak et al., 2015] Nayak, K., Nielsen, J., Bernstein, M., Markl, M., Gatehouse, P. D., Botnar, R. M., Saloner, D., C, C. L., Wen, H., Hu, B. S., Epstein, F. H., Oshinski, J., and Raman, S. V. (2015). Cardiovascular magnetic resonance phase contrast imaging. *J Cardiovasc Magn Reson. 17(1):71*.

[Odille et al., 2011] Odille, F., Steeden, J., Muthurangu, V., and Atkinson, D. (2011). Automatic segmentation propagation of the aorta in real-time phase contrast mri using nonrigid registration. *J. Magn. Reson. Imaging, 33: 232-238*.

[O'Shea and Nash, 2015] O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv*.

[Poustchi-Amin et al., 2001] Poustchi-Amin, M., Mirowitz, S. A., Brown, J. J., McKinstry, R. C., and Li, T. (2001). Principles and applications of echo-planar imaging: A review for the general radiologist. *RadioGraphics 21:767-779*.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *In: Navab N., Hornegger J., Wells W., Frangi A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science, vol 9351*.

[Siddique et al., 2011] Siddique, N., Sidike, P., Elkin, C., and Devabhaktuni, V. (2011). U-net and its variants for medical image segmentation: theory and applications. *eprint arXiv:2011.01118*.

[Steding-Ehrenborg et al., 2013] Steding-Ehrenborg, K., Jablonowski, R., Arvidsson, P., Carlsson, M., Saltin, B., and Arheden, H. (2013). Moderate intensity supine exercise causes decreased cardiac volumes and increased outer volume variations: a cardiovascular magnetic resonance study. *J Cardiovasc Magn Reson. 15(1):96*.

[The Criteria Committee of the New York Heart Association, 1994] The Criteria Committee of the New York Heart Association (1994). *Nomenclature and Criteria for Diagnosis of Diseases of the Heart and Great Vessels*. Little, Brown Co., 9th edition.

[Weishaupt et al., 2008] Weishaupt, D., Köchli, V., and Marincek, B. (2008). *How Does MRI Work?* Springer, 2nd edition.

[Zhang et al., 2014] Zhang, S., Joseph, A., Voit, D., Schaetz, S., Merboldt, K., Unterberg-Buchwald, C., Hennemuth, A., Lotz, J., and Frahm, J. (2014). Real-time magnetic resonance imaging of cardiac function and flow-recent progress. *Quant Imaging Med Surg. 4(5):313-29.*

Master's Theses in Mathematical Sciences 2021:E40
ISSN 1404-6342

LUTFMA-3455-2021

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/