CHALMERS
UNIVERSITY OF TECHNOLOGY

Vascular Bifurcation Detection in Cerebral CT Angiography Using Convolutional Neural Networks and Frangi Filters

Master's Thesis in Complex Adaptive Systems

NILS JACOBSON

Department of Physics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021

Master's thesis 2021

Vascular Bifurcation Detection in Cerebral CT Angiography Using CNN and Frangi Filters

NILS JACOBSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2021

Vascular Bifurcation Detection in Cerebral CT Angiography Using Convolutional Neural Networks and Frangi Filters
NILS JACOBSON

Supervisor: Jonna Hellström, Mentice
Examiner and Supervisor: Giovanni Volpe, Department of Physics, Chalmers

Master's Thesis 2021
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2021

Vascular Bifurcation Detection in Cerebral CT Angiography Using CNN and Frangi Filters

NILS JACOBSON
Department of Physics
Chalmers University of Technology

Abstract

Segmentation and feature extraction are important tools for analysing and visualizing information in medical image data, particularly in vascular image data which relates to widely spread vascular diseases. Vessel segmentation is extensively featured in research, recently adapting deep learning trends in image processing. This paper aims to develop a vessel bifurcation detection method to support a seed point based segmentation approach. The suggested approach is a combination of classification, with a convolutional neural network, local vessel segmentation, with Frangi filters, and 3D morphological skeletonization. A small data set is produced for network training and evaluation. Results indicate a high classification accuracy which filters problematic samples for the Frangi filter. Thus, the combination can suggest quality branch seed points under most circumstances. Next step would be to expand the data set to enable further optimization and more rigid evaluation. In any case a combination of a high-performance classifier followed by qualitative assessment of local samples show potential.

Keywords: CNN, CTA, DenseNet, Vessel Segmentation, Vessel Branch, Vascular Bifurcation Detection

Acknowledgements

I am grateful for the opportunity to be involved in an interesting project at Mentice. My thanks to Jonna Hellström, my supervisor at Mentice, for great commitment, frequent discussions, and good suggestions. Thanks also goes to Eric Parker, from Mentice, Dallas, for collaboration and project specific insights. I would also like to thank my supervisor and examiner at Chalmers, Professor Giovanni Volpe, for literature suggestions. Finally, thanks go to family and friends for support and good times.

Nils Jacobson, Gothenburg, February 18, 2021

Abbreviations

ANN	Artificial Neural Network
BCE	Binary Cross Entropy
BN	Batch Normalization
CNN	Convolutional Neural Network
CTA	Computed Tomography Angiography
FFNN	Feed-Forward Neural Network
FCNN	Fully Connected Neural Network
GAN	Generative Adversarial Network
HU	Hounsfield Unit
MLP	Multi Layer Perceptron
MRA	Magnetic Resonance Angiography
NLL	Negative Log Likelihood
ReLU	Rectified Linear Unit
RNN	Convolutional Neural Network
SGD	Stochastic Gradient Descent

Contents

1

Introduction

Currently there is tremendous technical progress in the medical field, not least in areas of image processing and analysis. Applications include visualization and simulation of the human body, and diagnostic assistance. More specifically an application is simulation of systems for training medical professionals in performing difficult procedures. Developing and maintaining proficiency in such procedures requires massive amounts of practice.[1][2] Simulations provide opportunities to practice without having to risk patients' health or use expensive animal trials.[3]

Endovascular surgery are minimally invasive procedures performed to treat a wide range of vascular conditions such as aneurysm, stroke, and embolism. The procedures involve inserting a catheter into an peripheral vessel and with a wire guide it to the vessel section in need of surgery.[4] This requires a lot of experience and training which partially can be obtained through simulations.[1] Endovascular surgery can be simulated with an optical sensor used to interface medical tools and catheters to a virtual 3D model of the vascular system.[5] Apart from educational practice a future potential use of this simulation system is to prepare for an impending medical emergency case, for example a patient with ischemic stroke. The surgeon could practice in the simulator on a patient specific anatomy while the patient is being anaesthetised and prepared for the operation.[5] A requirement to enable such usage is being able to quickly make a 3D model of an actual patient's specific vascular anatomy.

3D models of vascular anatomies used in simulations of endovascular surgery are modelled from CT- and MR-angiograms. Manual segmentation of blood vessels in angiograms is time consuming and require expert radiologists. To quickly build a 3D model, vessel segmentation must be automated. Automated vessel segmentation can, in addition to speed up model creation, make models cheaper and more readily available. There are various approaches to automate vessel segmentation. Currently approaches based on deep learning show the best robustness and accuracy.[6]

CTA is standard procedure for stroke patients as they are admitted to a hospital. The blood vessels of main interest in stroke patients are the carotid and cerebral arteries.[7] Cerebral and coronary arteries supply the most vital organs, the brain, and the heart, respectively. Hence, even small vessels are of interest when segmenting

these arteries which make the requirements for segment stricter than elsewhere in the vessel tree. Existing methods leave room for further progress and improvements in this field.[8]

Against this background this thesis is carried out in collaboration with Mentice, which has developed a semi-automatic method for segmenting the carotid and cerebral arteries. As a part of Mentice's method, seed points are needed in protruding vessels from bifurcations along a known vessel. Specifically, this thesis will attempt to develop and suggest a solution for the sub problem of detecting vascular bifurcations. Branch detection is not only relevant in the context of this project. Other methods that use seed points, for instance [9], also requires branch detection to find new seed points and recursively segment the vessel tree. In Figure 1.1 a vascular model with example seed points place along vessels branching from a known marked vessel can be seen.

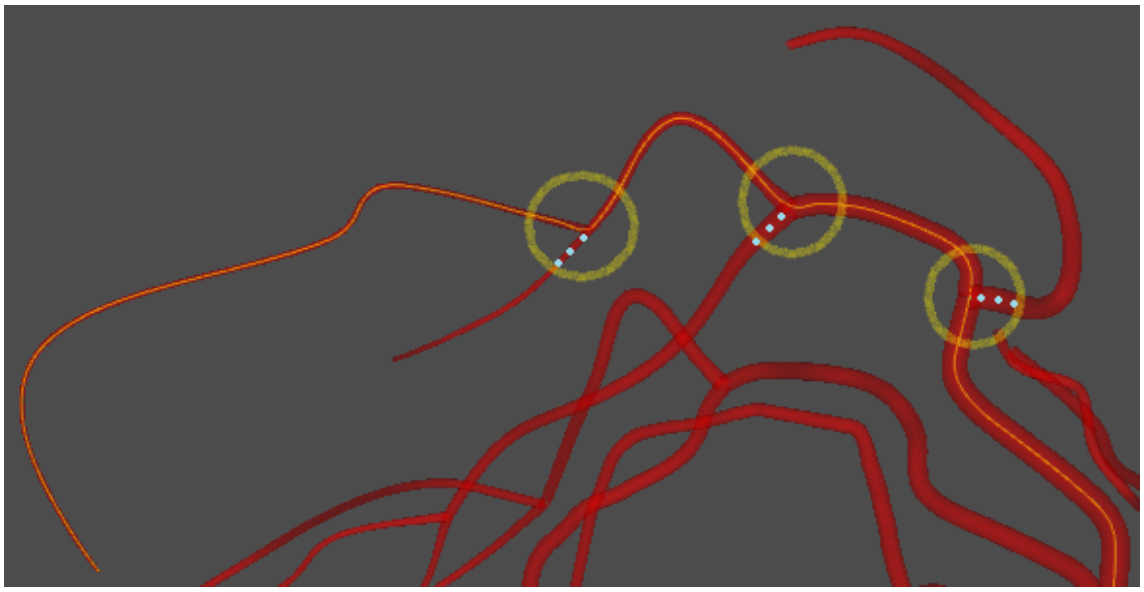


Figure 1.1: Part of a model of the cerebral vascular tree. The yellow line marks a known vessel, yellow circles mark bifurcation locations along the known vessel and the tiny blue dots are seed points along the vessel branches.

1.1 Problem Description

Overarching the problem addressed in this thesis is an opportunity to improve the availability of 3D models for specific patients' vascular anatomy to include in surgery simulators for practice and preparations. In this case the focus is segmentation of carotid and cerebral arteries, which are the vessels of interest with stroke patients. To realize this opportunity, automatic vessel segmentation must be performed quickly and with sufficient quality. One approach for segmenting arteries in standard procedure stroke CTA requires a sub-task to be solved to succeed. The sub-task consists of detecting bifurcations along an arbitrary known vessel and marking the protruding vessels. This sub-task is the problem addressed

in this thesis.

1.1.1 Aim and Research Questions

The aim of this thesis is to develop and propose a method for detecting vascular bifurcations along a blood vessel with priorly known centerline and radii in cerebral CTAs including the carotid arteries. Proposed method is to provide seed points in branches for a seed point based vessel segmentation method.

To reach this aim, the following research questions should be answered:

- Which methods are suitable for branch detection in CTA?
- Does the developed and proposed method perform sufficiently well?

1.1.2 Scope and Limitations

As the bifurcation detection approach suggested in this thesis is supposed to be compatible as a part of a particular vessel segmentation method there are some requirements on the input and output. Furthermore, to be feasible for the time frame of this project, there are some additional imposed limitations.

- Assume that there are vessels with prior known centerline and radii when specifying the input format.
- Proposed method will only be developed to work with CTAs, not MRAs or other imaging techniques.
- All used CTA data will be provided by Mentice
- It will be assumed that there only are two-way branches. This should not be a very restrictive constraint. It might exclude some complex branching structures such as the circle of Willis, this can be left for future work.
- The main focus will be on large branches of the carotid arteries. Furthermore, there is a limit to which size of branching vessels need to be detected. Endovascular surgery cannot effectively be performed on the smallest vessels. These vessels are therefore of no interest against the background of this project. Additionally the resolution and quality of the CT angiograms will inevitably limit the ability to detect vessels, particularly small vessels.
- Rigid verification of the method might not be possible due to shortage of ground truth material. Further verification will have to be done by manual trial on various samples.

1.2 Thesis Outline

In the introduction, **Chapter 1** , a brief background of the problem is given. The problem is defined with a purpose and scope.

Chapter 2 gives a deeper background understanding by introducing relevant medicine related concepts and prior vessel segmentation research. A few other theories applied in this thesis are described here as well.

Chapter 3 serves to describe the theory of deep learning. This is the main theory applied in this thesis. Thus, a separate chapter is justified.

Chapter 4 describes the choices made during development of the algorithm and why they were made. This includes how theories were applied and how data was obtained and processed.

In **Chapter 5** the results are presented along with the settings for which they were acquired.

The results are discussed in **Chapter 6** with respect to validity, prior research and future opportunities. This chapter is concluded with summarizing conclusions of this thesis.

2

Segmentation and Feature Extraction of Vascular Structures

This chapter introduces relevant background information and prior research on vessel segmentation and vascular feature extraction. The theoretical framework used later in the report is also presented here. However, the topic deep learning is attended to separately in Chapter 3.

2.1 Computed Tomography Angiography

The medical imaging technique used to visualize the lumen of blood vessels and other vascular structures is called angiography. In this thesis CTA is used, as it is standard procedure for stroke patients [7], as mentioned in the Introduction, Chapter 1. MRA (magnetic resonance angiography) is a similar technique based on MRI (magnetic resonance imaging). Qualitatively a difference is that bones are easier to discern from blood vessels in MRA than in CTA as mentioned in [10].

CT compute image slices from X-ray measurements. Stacking these image slices produce a 3D tensor of pixel values. Pixels are called voxels in 3D and constitute a 3D image. Various tissues and substances attune differently to X-rays and give rise to different intensities in the CT images. The intensities, called the linear attenuation coefficient, μ, are calibrated for individual CT-scanners, usually with water as a reference. To simplify comparison between different CT-scanners the intensity is usually converted to the Hounsfield units, HU, see the Hounsfield scale[11]. The Hounsfield scaled corresponds to a radiodensity and is defined by distilled water having 0 HU and air -1000 HU. The conversion from the linear attenuation coefficient to HUs is

$$HU = 1000 \cdot \frac{\mu - \mu_{water}}{\mu_{water} - \mu_{air}}.$$

Details about CT can be read in [12] or in an article by G. N. Hounsfield [13], who, together with A. M. Cormack was awarded the Nobel prize for the development of CT in 1979.

Blood is mostly water, which also is the main component in most body tissues. Thus, it is difficult to distinguish blood vessels in CT images. Hence, in CTA, a radio contrast fluid is injected into the vessels. As the contrast fluid blend with the blood the radiodensity rises and the vessels will be easy to discern from the background tissues.[7] Blood vessels with radio fluid are about 200-600 HUs. Most other tissues are as mentioned at 0 HUs. However, some soft tissues and cancellous bone can be in a similar range. Cortical bone is 500-1900 HU which also can overlaps with blood vessels in CTAs.[14]

2.1.1 DICOM File Format

CT images are usually saved as DICOM file format. The DICOM file header contains a lot of information. Details about DICOM format can be found in [15]. Header information relevant in this thesis is:

- **Pixel Spacing**, physical distance between/size of pixels in mm, two values, one for each dimension
- **Slice Thickness**, thickness of an image slice in mm
- **Rescale Slope**, slope when rescaling to HU
- **Rescale Intercept**, offset term according to Equation 2.1, when rescaling to HU

Information about the specific CT-scanner calibration is included and makes the conversion to HUs from intensity, I:

$$[\text{HUs}] = [\textbf{Rescale Slope}] \cdot I + [\textbf{Rescale Intercept}] \tag{2.1}$$

2.2 Cerebral Vascular Anatomy

Standard procedure stroke CTA usually include everything from the aorta arch and up in cranial direction, as seen in Figure 2.1. The aorta arch usually is only partially included as seen in yellow in the bottom of Figure 2.1. From the aorta mainly four branches lead to the brain. Firstly, the left and right carotid arteries, which each split into an external and internal branch. The internal branch passes through the temporal skull bone where the vessel is difficult to discern from the bone. Secondly, the left and right vertebral arteries which partially runs inside the vertebrae.[16] As the carotid arteries are the largest vessels supplying the brain, they are most important in the context of stroke and are vessels mainly focused on this thesis.

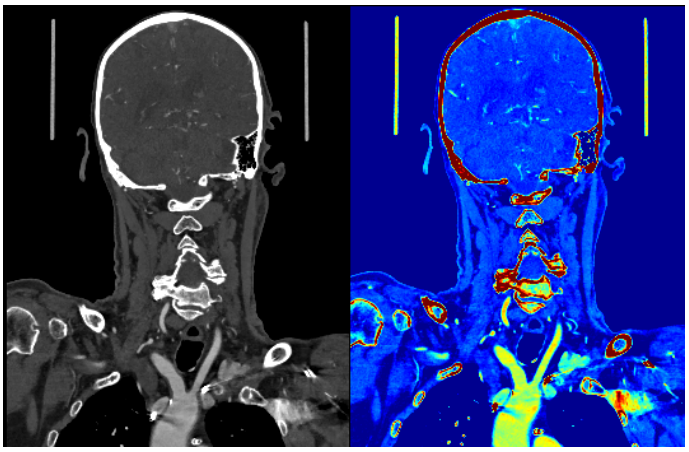


Figure 2.1: Full sagittal plane of a CTA from a standard procedure stroke scan. The image to the right is in a colormap which makes it easier to differentiate vessels from bone and background. This colormap will be used for CTA images in the rest of this thesis.

2.2.1 Geometrical Features of The Vascular Tree

There is prior research suggesting that the geometry of the vascular tree must follow physical rules for fluid mechanics. Murray's law, Equation 2.2, states the connection between the radius, r, of the vessel prior to a branch and radii, r_1, r_2,...,r_n , of the branching vessels as:

$$r^\gamma = r_1^\gamma + r_2^\gamma + ...r_n^\gamma \tag{2.2}$$

where γ depends on if the flow is laminar or turbulent. Suggestions from [17] are $\gamma = 3$ for laminar and $\gamma = 7/3$ for turbulent flow. There are also constraints, Equation 2.3, on the possible angles between two branches that depend on the radii. With angles, θ_i and θ_j as introduces in Figure 2.2, with $i, j \in 1, 2, i \neq j$, the constraints are:

$$\cos(\theta_i) = \frac{r^4 + r_i^4 - r_j^4}{2r^2 r_i^2} \tag{2.3}$$

Furthermore, the geometry is affected by the metabolic growth of the arteries during angiogenesis. In [18] all these theories are used to generate synthetic vessel trees. However, this synthetic data lack realistic artefacts, noise and background structures found in real CTAs which are not feasible to generate and thus were not pursued further in this thesis.

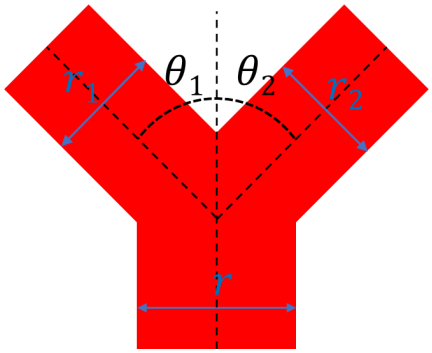


Figure 2.2: Schematic model of a vessel bifurcation with notations introduced.

2.3 Previous Work in Vascular Image Analysis

In an article from 2009 by D. Lesage, et al. [8], state-of-the-art of literature on vascular segmentation is reviewed. The most common task in vascular image analysis is segmentation of vessels and organs. Features such as branches, centerlines, aneurysms, etc. are usually easier to extract from a segmentation rather than directly from the images. A later review article from 2019 by F. Zhao, et al. [6] includes newer machine learning, especially deep neural networks, based methods for vessel segmentation. This section first treats vessel segmentation methods divided in non-machine learning based, in Section 2.3.1, and machine learning based, in Section 2.3.2. Deep learning theory is attended more thoroughly in Chapter 3. In Section 2.3.3 prior work in vessel feature extraction is covered.

2.3.1 Non-Machine Learning Segmentation Methods

Well cited non-machine learning methods for automatic vessel segmentation include: analysing eigenvalues of the Hessian to find principal directions [19]; cross sectional marching filters, modelling vessels as Gaussian intensity ridges [20]; utilizing mathematical morphological, including watershed [21] filters, for pre-processing or region growing; minimizing energy formulations with minimal paths[22], active contours[23, 24, 25] or graph based methods[26, 27]; and stochastic approaches with Markov marked point processes [28] or random walks [29]. Of these Frangi Filters[19], based on Hessian eigenvalues, are used in this thesis and paid attention in Section 2.3.1.1.

2.3.1.1 Frangi Filters

A. Frangi, et al.[19] in 1998 presented a filter for enhancing tubular vessel structures in 2D and 3D images. The response from this filter is in later literature often referred to as Frangi's vesselness measure and often incorporated when developing

new methods, for example in [30] as well as some listed in 2.3.1. Frangi filters are used in some recent state-of-the-art vessel segmentation software such as QAngioCT [31].

The idea of the filter is to analyse the eigenvalues of the Hessian matrix, $\mathbf{H}$, to find principal direction of the second order nature in the image. The Hessian is the Jacobian of the gradient, which in this case is

$$\mathbf{H}(I(\vec{x}, S)) = \nabla^T \nabla I(\vec{x}, S), \tag{2.4}$$

of the image, $I(\vec{x})$, at a point, $\vec{x}$, at various scales, S. In image processing derivation is often defined as convolutions with a Gaussians distribution, $N(\vec{x}, S)$, at scale S, which in the 3D case is:

$$N(\vec{x}, S) = \frac{1}{\sqrt{2\pi S^2}^3} \cdot \exp\left(-\frac{\|\vec{x}\|^2}{2S^2}\right). \tag{2.5}$$

Therefore, the derivative is written

$$\frac{\partial I(\vec{x}, S)}{\partial \vec{x}} = SI(\vec{x}) * \frac{\partial N(\vec{x}, S)}{\partial \vec{x}}. \tag{2.6}$$

Eigenvalues of the Hessian, $|\lambda_1| \leq |\lambda_2| \leq |\lambda_3|$, will for a tubular structure have these properties:

$$\begin{array}{c} |\lambda_1| \approx 0 \\ |\lambda_1| \ll |\lambda_2| \\ |\lambda_2| \approx |\lambda_3| \end{array} \tag{2.7}$$

A bright tube with a dark background is indicated by $|\lambda_2|, |\lambda_3| < 0$ and vice versa. In CTA, the vessels are always bright, so $|\lambda_2|, |\lambda_3| < 0$ is the correct criteria. Corresponding eigenvectors, $\hat{v}_1, \hat{v}_2, \hat{v}_3$, are orthonormal with $\hat{v}_1$ pointing in the direction with minimum intensity variation (along the vessel) and $\hat{v}_2$ and $\hat{v}_3$ span a plane with a cross section of the vessel.

Eigenvalues are used to define three measures terms which form the final vesselness measure. The first term,

$$R_B = \frac{|\lambda_1|}{\sqrt{|\lambda_2 \lambda_3|}}, \tag{2.8}$$

which respond to deviation from blob-like structures. The second term,

$$R_A = \frac{|\lambda_2|}{|\lambda_3|}, \tag{2.9}$$

respond to the deviation from a plate-like structure and enables distinguishing plate-like and line-like structures. Lastly third term,

$$S = \|\mathbf{H}\|_F = \sqrt{\sum_{j\leq} \lambda_j^2}, \tag{2.10}$$

called "second order structuredness" in [19], is added to stop random noise fluctuations in background from producing unpredictable responses. This term is the Frobenius matrix norm of the Hessian and adjusts the sensitivity to areas of high variance. Collectively these terms a function,

$$V(\mathbf{x}, s) = \begin{cases} 0, \text{ if } |\lambda_2| > 0 \text{ or } |\lambda_3| > 0 \\ 1 - \exp\left(-\frac{R_A^2}{2\alpha^2}\right) \exp\left(-\frac{R_B^2}{2\beta^2}\right) \left[1 - \exp\left(-\frac{S^2}{2\gamma^2}\right)\right], \text{ else,} \end{cases} \tag{2.11}$$

with parameters α, β and γ to tune the three terms R_A, R_B and S. By making calculation on different scale vessel of different size can be found. As the final vesselness measure the maximum response over all investigated scale is chosen:

$$V(\mathbf{x}) = \max_s V(\mathbf{x}, s) \tag{2.12}$$

2.3.2 Machine Learning Segmentation Methods

Machine learning based vascular segmentation methods has recently been very successful. In machine learning segmentation is a classification problem where pixels or voxels are labelled as vessel or non-vessel. Explored methods include conventional ANNs (FCNNs); K-nearest neighbour classifiers combined with Gaussian mixture models; unsupervised K-mean and Fuzzy C-mean clustering; support vector machines combined with marching filter methods; decision-tree-based combined with random forest; and deep learning with specifically CNNs. Often methods are combined with non-machine learning methods. More details can be found in [6], including references to more specific articles.

CNNs are very well suited for image processing and has stepped up as the main competitor in image classification in start of the 21st century. Hence, they are winning much popularity in medical image processing and have proven superior in both robustness and accuracy to many older methods. Limited work has been performed on coronary and cerebral due to lack of standard data bases [6]. Lack of data is a recurring issue in medical processing as machine learning methods, in

particular deep learning and CNNs, is driven by data for training and performance validation. Labelling ground-truth is difficult, time consuming and subject to human error.

To contest the lack of data, many projects explore generation of synthetic data. Synthetic data provides unlimited data with perfect ground truth. The difficulty is to produce an abstraction for all features found in real medical images. In [18] synthetic data is generated from known geometrical properties of vascular anatomy, as mentioned in Section 2.2.1. Even if the data was reasonably vessel-like only Gaussian-like noise was introduced making the image analysis substantially easier. Generative adversarial networks (GANs) are a deep learning method for generating novel samples with resemblance to existing samples. GANs are promising but requires some data to be trained from and even though the output samples might look authentic, they can lack important properties which are not easily discernible to the eye.

2.3.3 Vascular Feature Extraction

Extracting vessel features such as centerlines and branches involves finding vessels. Consequently, it is often performed after or synchronized with vessel segmentation, on the binary image.[6]

A method including similar steps as the project in this thesis is presented in [9]. A tubular based, Frangi inspired, method is used to segment the vessels. Centerlines and vessel thickness are thereafter traced with a method similar to Diffusion Tensor Imaging tractography [32]. Branch detection is used to find new seed points in connected vessels and recursively map the entire vessel tree. The method used for branch detection uses the known centerline and vessel thickness, radius, to apply an unsupervised clustering.

In [18] a CNN is applied for vessel segmentation, centerline prediction and bifurcation detection. Again, the centerline extraction makes use of prior segmentation and bifurcation detection rely on both segmentation and centerline predictions. However, the directions of the bifurcation branches aren't given.

2.3.3.1 Skeletonization of Segmented Vessels

Skeletonization is an imaging process where a thinning algorithm is applied repeatedly in a binary image until it does not change the image anymore. This leaves a thin line corresponding to the geometrical and topological shape of the thinned components like shown in Figure 2.3. A basic method for thinning is mathematical morphology structuring elements and hit-and-miss transforms.

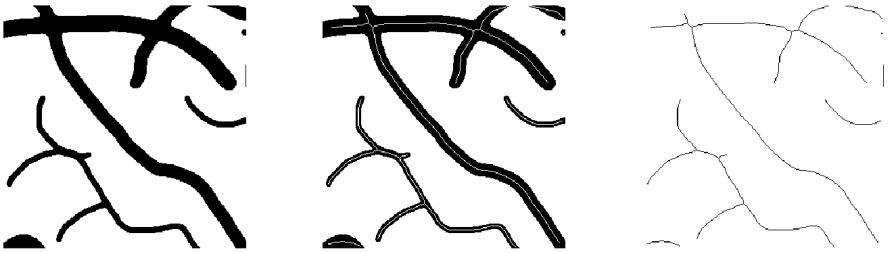


Figure 2.3: Skeletonization of 2D image **Left:** Binary image of segmented components **Middle:** Skeletonization overlayed **Right:** Only skeletonization

A skeletonization is a reasonable representation blood vessels' structure, as proposed in [33]. It offers readily accessible seed points, which not necessarily have to be the exact centerline of the vessel. However, in 3D, topological properties are more difficult to address. Lee, et al. [34] suggest a method for extracting medial surfaces and axis of 3D objects which is applied in this thesis. In 3D a common type of neighbourhood for each voxel includes 26 neighbours and is called a Moore neighbourhood [35]. In [34] these 26 neighbours are divided into eight overlapping octants, consisting of $2 \times 2 \times 2$ cubes. Thinning is then performed by labelling the octants and using the sum of octant labels to determine whether to keep the voxel.

3

Deep Learning

In this chapter the aim is to introduce the concepts of deep learning that was applied in this thesis. Particularly the focus is on convolutional neural networks (CNNs), but the basics of general artificial neural networks is also included.

Deep learning is a type of machine learning based on deep artificial neural networks (ANNs). Computer vision, image processing, speech recognition and other domains which find patterns in and interpret large amounts of complex data all have seen dramatic improvements recently by introducing deep learning algorithms. CNNs dominates where spatial arrangement of data is relevant such as images while recurrent neural networks (RNNs) thrive in processing sequential data such as text and speech.[36]

3.1 Artificial Neural Networks

The basis of artificial neural networks (ANNs) is inspired by the biological brain, which literally is a network of neurons. Braincells, *neurons*, are connected by projections called dendrites and axons which, through synapses, communicate with other neurons. Connections between neurons vary in strength depending on how useful the connection is.

A ANN is typically structured like in Figure 3.1, which depicts a fully connected feed forward neural network (FC-FFNN). The structure features artificial neurons arranged in *layers*. Information feed to the input layer propagates through the hidden layers until it produces an output in the final layer. A neuron, j, in layer, l, sends out an activation signal,

$$o_j^{(l)} = a\left(\sum_i w_{ij}^{(l)} o_i^{(l-1)} + b_j^{(l)}\right) \tag{3.1}$$

to neurons in the next layer, $l+1$, based on the received inputs, $o_i^{(l-1)}$, from the previous layer, $(l-1)$. Where $w_{ij}^{(l)}$ is the weight representing the strength of the connection and $b_j^{(l)}$ is a bias term. The activation function, a, which introduce non-linearity, is explained further in Section 3.1.1.

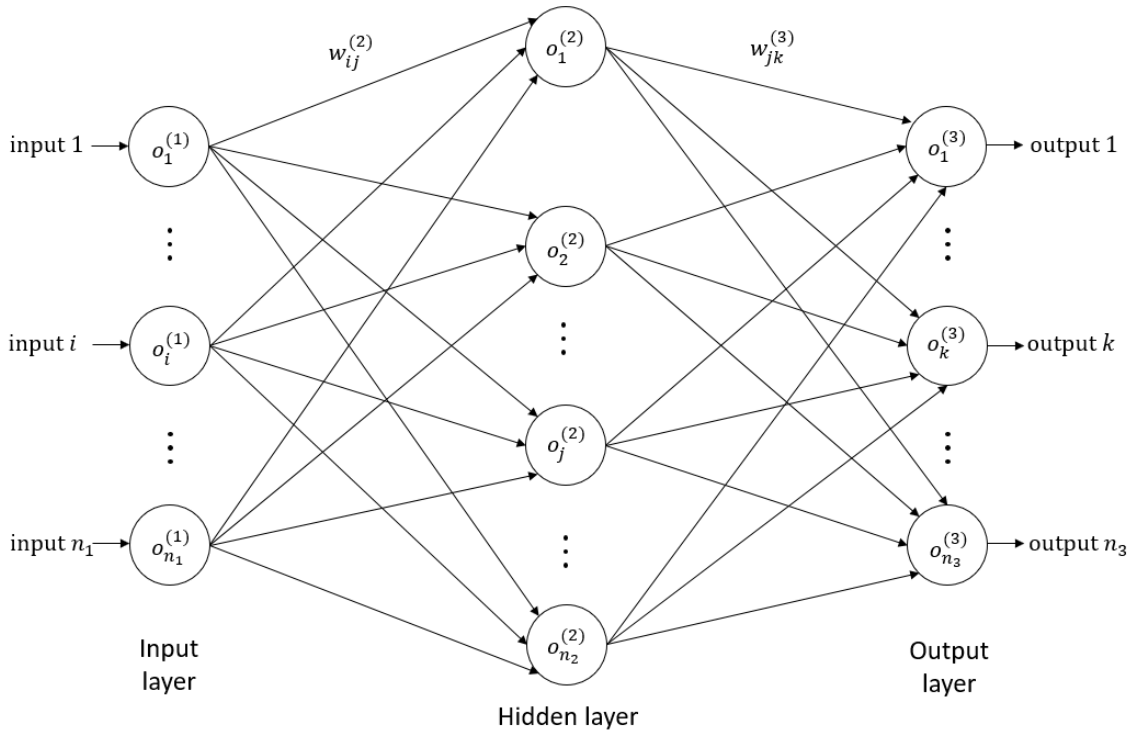


Figure 3.1: A FC-FFNN multilayer perceptron with one hidden layer. Maps from $\mathfrak{R}^{n_1}$ input space and $\mathfrak{R}^{n_3}$ in output space via a hidden layer with n_2 neurons.

A deep NN consists of many hidden layers and can model more complex function than a shallow NN. Information is feed between neurons through one-way weighted connections.

3.1.1 Activation Functions

Without activation functions the network would correspond to stacking linear affine mappings, $\vec{o} = A\vec{o} + B$, with matrices A and B. Through linearity this only creates a new affine mapping, thus not increasing the complexity. Activation functions provide a source of non-linearity which remedy this issue. The most common activation functions used in NNs include sigmoid functions, $a(x) = \frac{1}{\exp(-x)+1}$, $\tanh x$, rectified linear unit, $\mathrm{ReLU}(x) = \max(0, x)$, and softmax,

$$a_j(\vec{x}) = \frac{\exp(x_i)}{\sum_{j=1}^{M} \exp(x_j)}. \tag{3.2}$$

Softmax is calculated from the outputs of the entire previous layer and can be interpreted as a probability distribution. In multiclass classifiers there is often a softmax after the final output layer which signifies that the input $\vec{x}$ belongs to class j with probability a_j.

3.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are FFNN that feature *convolutional layers*. The general structure was first introduced by Fukushima et al. in [37]. Convolutional layers are especially suited for making use of spatial data like images which is proven by great results on classification of large public image data sets, such as ImageNet [38], starting with A. Krizhevsky's et al. contribution in [39].

3.2.1 Convolutional Layers

CNNs are driven by the cross-correlation operation, $\star$. This is not to confuse with the similar, but different, convolution operation. Cross-correlation is a sliding inner product between a filter kernel and the signal, in this case an image. In 2D, for each filter position, (x, y), a filter kernel, F_{kernel}, with size $d \times d$, $d = 2n + 1$, gives output

$$C(x, y) = F_{\text{kernel}} \star I(x, y) = \sum_{i=-N}^{N} \sum_{j=-N}^{N} F_{\text{kernel}}(i, j) \cdot I(x + i, y + j) \tag{3.3}$$

where $I(x, y)$ is the image pixel value at (x, y). Therefore, after sliding kernel over all filter position, separated by filter stride length s a new image is produced, called a *feature map*. A convolutional layer is made up of a number, k, filter kernels which is called the filter size. Therefore, a convolutional has k output channels, meaning it outputs k different feature maps.

A filter with kernel size, $d > 1$ can't be placed on the edge of an image, as seen in Equation 3.3. Feature maps will thus be smaller than the original image input. If this is not desirable zero padding the image can resolve this issue. Stride length $s > 1$ also will reduce the size of the image, but in this case, it is usually intended as a way to reduce the dimensionality of the input.

Functionally filter kernels in convolutional layers give respond to image structures which match the shape of the filter. Therefore, feature maps will contain information about features in the image, like edges and small shapes. This exploits local spatial relations and is invariant to translation as the same kernel is applied all over the image. With several consecutive convolutional layers, deeper layers will extract features from feature maps of feature maps. Thus, abstraction level of the feature maps increases through the network and can eventually represent complex objects.

3.2.2 Pooling Layers

To reduce dimensionality as the image data propagates through the network pooling is used. Dimensionality is reduced to let the network look at larger scale structures. There are two types of pooling: average and max pooling. Average pooling takes the average of a neighbourhood of pixels and converts into one pixel while max pooling

takes the maximum. An average pooling layer with kernel size $d \times d$ and stride s outputs the feature map

$$P_{avg}(x, y) = \frac{1}{d^2} \sum_{i=0}^{d-1} \sum_{j=0}^{d-1} I(i + sx, j + sy). \tag{3.4}$$

In max pooling the output is

$$P_{max}(x, y) = \max_{i,j \in \{0,1,2,\ldots,d-1\}} I(i + sx, j + sy). \tag{3.5}$$

Pooling can also be performed globally in an entire feature map. An example of where global pooling can be used is at the backend of a CNN to make the output 1D for a final FC-layer.

3.2.3 Batch Normalization

Batch normalization layer is added to normalize the outputs from the previous layer. This keeps the mean and variance of the inputs consistent during training. Varying input distributions, internal covariate shift, slows down training by having layers remap training targets to a new distribution. Thus, batch normalization increases the speed of the training. Batch normalization also keeps the values in ranges where there is less risk of activation functions saturating. Saturating activation functions make gradients vanish and thus essentially halts the learning. Bath normalization was originally introduced by S. Ioffe and C. Szegedy [40], where more details can be found.

3.2.4 CNN Architectures

A great number of variations in the network architecture of CNNs have been proposed over the years. Initially there are structures with basic convolutional layer building blocks like [38]. For image segmentation, U-shaped networks, which involve upsampling to original image size after features has been extracted are common, like U-net by Ronneberger et al. [41]. More complex building blocks, consisting of various constellations of convolutional filters, have shown promise for handling increasingly complex tasks rather than simply increasing network depths. This includes ResNet[42] and DenseNet[43] which have brought image classification up another step.

3.2.4.1 DenseNet

DenseNet suggested by Huang et al. in [43] is a densely connected convolutional network. The architecture significantly improves state-of-the-art classification on popular benchmark tasks such as CIFAR[44] and ImageNet[38] and requires less computations. Densely connected is a reference to the introduced dense block (see

Figure 3.3) in which the convolutional layers are connected to all subsequent convolutional layers. This allows reusing filters several times on different scales and levels of abstraction. Thus, trainable parameters is reused and the amount of convolutions necessary in later layers is reduced. The information flow in a DenseNet is shown in Figure 3.2. There is an initial convolutional layer which produce feature maps for the first dense block. Between each dense block there are transitional layers which change scale of the feature maps. Finally, the output from the last dense block is flatten by global average pooling and classified with a FC- and softmax-layer.

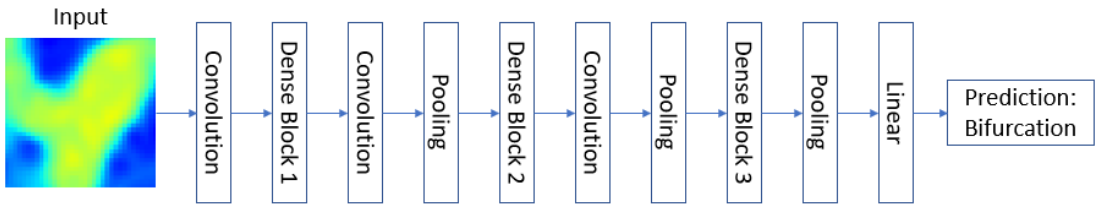


Figure 3.2: DenseNet structure with an initial convolutional layer, three dense blocks, transitional layers between dense blocks and a linear classifier layer. The transitional layers consists serve to change scale of the feature maps by convolution and pooling.

The special building block, the dense block, can be seen in Figure 3.3. A dense block consists of several convolutional layers with filter size, k, five layers with $k = 4$ in this case. Between each layer there is a BN-ReLU-Conv layers which consists of a batch normalization, a ReLU and a $1x1$ convolution. Each convolutional layer receives the output of all the preceding convolutional layers. Thus, each layer has a filter size k larger than the preceding layer. The parameter k is called the growth rate of the network.

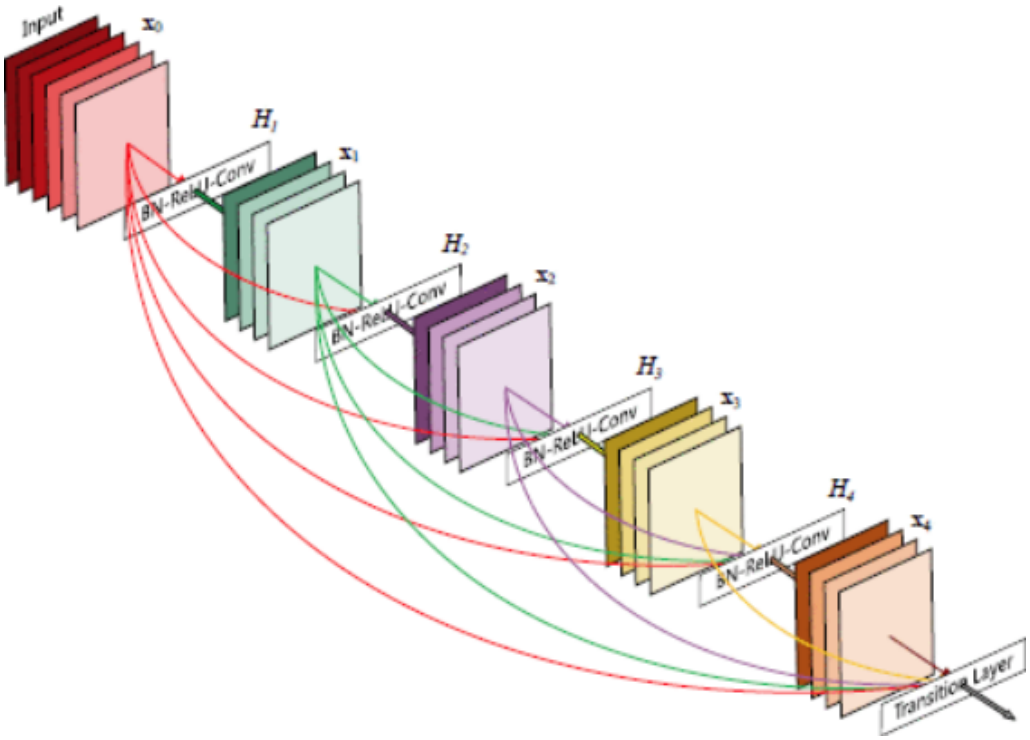


Figure 3.3: Dense block with 5 layers and growth rate $k = 4$. The output of each layer is included in the input and output of every other layer respectively. ©2017 IEEE. Reprinted, with permission, from G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks", *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. *The views expressed in this report are those of the author and do not necessarily represent those of any content provider.*

3.3 Learning

The process of optimizing network parameters, filters, weights, and biases, is called training. Optimization is done in respect to target output (labels) representing the ground truth of the input training data and the model outputs to mini the error between the two.

3.3.1 Loss Function

In order to quantify the error between network output, $\vec{x}$, and target output, $\vec{y}$, a loss function, $L(\vec{x},\vec{y})$ is defined. As the error decreased the loss function should go towards zero. Thus, training of a network is the minimization of the loss function. Defining an optimal loss function can be difficult and must specifically adapted for the problem at hand. In a two class classification networks binary cross entropy (BCE) loss, L_{BCE}, is commonly used.

$$L_{BCE}(\vec{x}, \vec{y}) = -\sum_{i=1}^{n} \left[y_i \cdot \log(x_i) + (1 - y_i) \cdot \log(1 - x_i) \right]. \tag{3.6}$$

The terms inside the sum in Equation 3.6 will produce zero if $x_i = 1$ and $y_i = 0$ or vice versa. Thus, the loss is minimizing the loss requires the network output to clearly belong to class $x_i = 1$ or class $x_i = 0$. So Equation 3.6 is a negative log likelihood (NLL), quantifying the difference between two probability distributions.

3.3.2 Backpropagation

Loss calculated in the output layer must be propagated backwards through the network in order to update weights accordingly. This process is called backpropagation and involves computing the gradients of the loss. Gradients are then used to update weights in the order which the loss decreases fastest,

$$w_{ij}(t_n) = w_{ij}(t_{n-1}) - \eta \frac{\partial L}{\partial w_{ij}}, \tag{3.7}$$

where t_n is step n and η is the learning rate. Backpropagation was introduced by Rumelhart et al. [45].

3.3.3 Optimizer

Exactly how the gradients and backpropagation is carried out is defined by the optimiser. Gradient descent (Equation 3.7) is a simple description of such optimiser. In computers stochastic gradient descent (SGD), which is a stochastic approximation of gradient descent, has advantages in terms of computational speed.

Adam[46] is a currently popular optimizer which use SGD with adaptable learning rates for each weight. This allows for faster convergence and easier setup than SGD.

3.3.4 Epochs and Steps

During training the network is passed training samples from the training data *batch* repeatedly in a loop. For each *step* one sample, a subset of samples (called *mini batch*) or the entire batch is feed to the network. After each step, the weights are update according to the loss function and optimizer. An epoch is when one iteration of the loop has been completed, i.e., each training sample has been passed to the network once.

3.3.5 Overfitting

Overfitting is when the network has learnt a specific set of training data very well but perform poorly on new data. At this point the network has learned details of the

training data but have not learned general features which generalize to other data. To notice overfitting tendencies, the network is usually evaluated on a validation data set, separate from the training data set, each epoch. Performance on the validation set will give information about how well the network generalizes. Regularisation methods aim to prevent overfitting from happening and promotes the network to learn to generalize.

3.3.6 Data Augmentation Transforms

One regularisation method is data augmentation. In data augmentation various natural transforms are applied to manipulate training data to create greater variation without disturbing important features. Thus, important features which are not affected by transforms will be learned while details which are varied by transforms will not. Augmentation can be done either offline by duplicating the data set and augmenting the copy; or online. Online augmentation is of course very powerful as it provides unique data each epoch. However, the transforms must be computed each epoch, which with more complex transforms increases training time.

For images flipping along axes, rotations, translations, scaling, noise, blurring, and affine transforms are common transforms. An elastic deformation transforms [47] can be useful in medical imaging as they can smoothly vary shapes of organs. In this thesis, altering the path of vessels without distorting branches and radii could be useful.

An elastic deformation image transform can be performed by introducing a rectilinear grid and then alter the grid point locations with random displacements. The original grid points in the image are resampled to the displaced grid points. Preferable interpolation method which avoids introducing disconuities, like the B-spline based interpolation suggested in [48], is used.

Specifically which augmentation transforms were applied in this these can be found in Sections 4.3.3.5 and 5.1.4.

4

Development of Bifurcation Detection Method

This chapter describes what I have done to obtain the results presented in this thesis. How and why I applied described methods is also explained. The work can be divided in three steps: first acquiring and preparing data, second developing a neural network model, and third extracting bifurcation seed points.

4.1 Overview of Approach

The target of this project was to find bifurcations from a known vessel path in cerebral CTA. Information about the known vessel in question can be used to disregard most of the CTA and only consider a vicinity of the vessel. My general idea was to traverse the vessel centre line in tangential direction while looking for branching vessels in radial direction. Branch detection in [9] is based on a similar idea. For a vessel vicinity I adopted an approach of sliding windows in the form of 3D boxes along the centre line of the vessel, see boxes in Figure 4.5 and Figure 4.6. I trained a CNN to classify between boxes containing and not containing bifurcations. After classification vascular structures in the boxes containing bifurcations can be segmented using the Frangi's vesselness measure [19]. By thinning the segmented structures, seed points along the general path of the vessel and branching vessels are obtained.

4.2 Data Acquisition and Preparation

This section describes the CTA data and how I processed it to produce a training set.

Performance of a CNN is largely decided by the data used while training the network. With suggested approach the input to the CNN is 3D boxes extracted from CTA along a vessel path. This is a quite specific input format which means it doesn't exist any publicly labelled or unlabelled data sets. One way of obtaining data sets for training would be to generate synthetic data, mentioned in Section 2.2.1 and Section 2.3.2. However, training a GAN or creating an abstraction of features to be

able to generate synthetic data is not feasible in the time scope of and data available in this project. Therefore, this project included extracting and labelling data sets from CTA. This limits the amount of available data for training, but with good data augmentation even a small data set can give good performance.

4.2.1 Description of CTA Images

I extracted the data sets used in this project from five anonymized CTA scans available at Mentice. In this report each CTA will be referred to as CTA#1, CTA#2, ..., CTA#5, respectively. The CTAs had been scanned with standard procedure stroke protocol including everything cranial of the aorta arch in the patients. CTAs come from a few clinics in Europe. All images came in DICOM-file format, which is described in Section 2.1.1. Pixel spacing of the CTAs were in the range 0.4 mm to 0.6 mm and slice thickness 0.2 mm to 0.6 mm. However, the results should be similar with any standard procedure stroke CTAs in Hounsfield units with similar resolution. As described in Section 2.1 a 3D image is created by stacking a series of 2D CTA images in sequence.

4.2.2 Obtaining Prior Known Vessels

As described in 1.1, the centerline and radii of a vessel path is assumed to be given. In this project I manually marked the centerline and radii to facilitate reproducibility. There are automatic methods for marking centerlines and radii, brought up in Section 2.3.3, but they generally aren't robust enough to not warrant manual validation in which case manual marking can be performed from the start. The manual marking was done to emulate the output of Mentice's automatic method. How I did the manual marking is shown in Figure 4.1. I centered points along a vessel by visually inspecting 2D image slices along each 3D image axis. At each point I matched a circle's radius to vessel width in corresponding 3 perpendicular image slices (coronal, axial and sagittal planes).

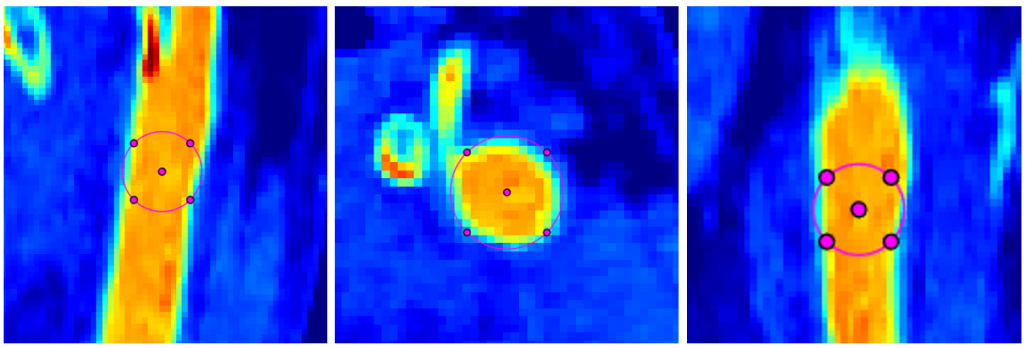


Figure 4.1: Centerline point with responding vessel radius marked with purple in a CTA. The 3 images are 3 perpendicular planes; **Left:** Coronal plane, **Middle:** Axial plane, **Right:** Sagittal plane. The marked centerline point and radii is the same in all images, the level zoom into the CTA is different in all 3 images.

The centerline I interpolated from marked points fitting a Catmull-Rom spline [49] as seen in Figure 4.2, correspondingly the radii was linearly interpolated. Together the centerline and radii make up a tubular vessel-like structure a part of which a polyhedron rendering is on display in Figure 4.3.

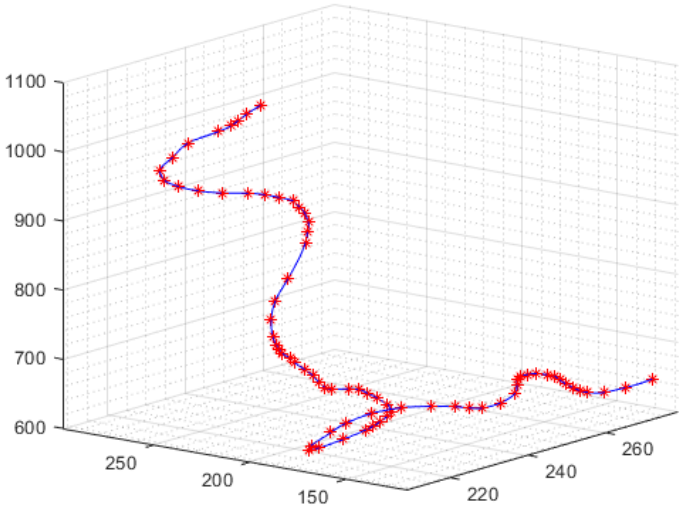


Figure 4.2: The centerline interpolated with a Catmull-Rom spline (blue curve) fitted to the manually marked centerline points (red *). The axes ticks are in voxel coordinates.

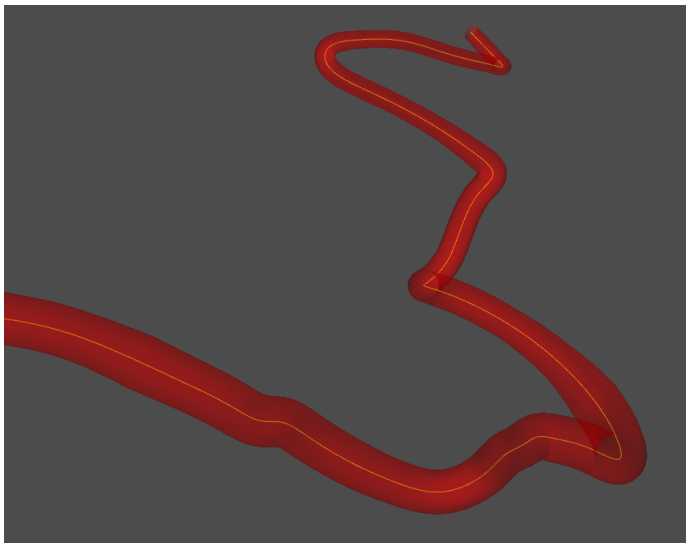


Figure 4.3: Part of rendering of marked vessel centerline and radii as a polyhedral in 3D. Not based on the same vessel as Figure 4.2.

4.2.3 Extraction of Samples Along Vessel Path

I only considered a vicinity of the known vessels to be able to leverage vessel centerlines and radii. There are various ways of choosing a vicinity. The use of CNN suggests the use of 2D or 3D images. Cross section 2D slices along the vessel path can been feed to a RNN like a sequential time series. Another alternative is to mask everything but the vessel and vicinity and feed the entire masked CTA to a 3D CNN. This would be able to use more large-scale spatial information such as where along a certain vessel type usually bifurcations are located. However, this would require plenty of samples of each kind of vessel type, respectively. Marking and labelling vessels manually is very time consuming which makes this less viable in the scope of this project. It would also require a large CNN which takes longer to train. Instead, I choose to use 3D boxes along the vessel path. I feed the small 3D boxes to a 3D CNN DenseNet as described in Section 3.2.4.1 and will be specified later in this chapter. While the CNN cannot learn large scale features bifurcations are a distinguished feature which should be possible to categorize by itself. There are several bifurcations along each marked vessel which results in more training samples per vessel. The required CNN can be made smaller which makes for fast training. Smaller boxes are also a less complex task than large, masked boxes which further lowers the necessary network size.

4.2.3.1 Box Orientation, Size and Resampling

I extracted boxes along the blood vessels as in 4.5. A local orthonormal coordinate system was defined, as drawn in Figure 4.4, with origin in current point along the centerline of the vessel. Two axes, $i = 1, 2$ are perpendicular to each other and the centerline in current point and the third axis, $i = 3$, tangent to the centerline in current point. Exact orientation of the axes beyond previous description doesn't matter as data augmentation mirrors and rotates the 3D box, data augmentation is described in more detail in Section 3.3.6, Section 4.3.3.5 and Section 5.1.4. The axes are scaled by pixel spacing and slice thickness specified in the DICOM-file, so a unit step is 0.5 mm along each axis. Boxes are aligned with this local orthonormal coordinate system and centered at the current point.

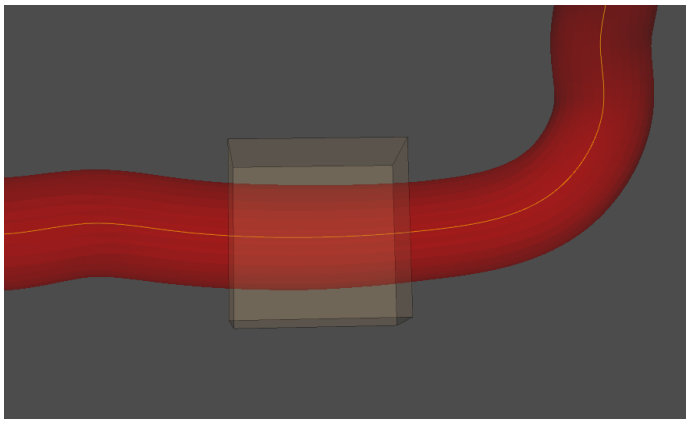


Figure 4.4: 3D model of sample box on vessel polyhedron model.

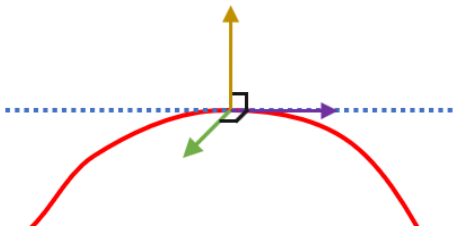


Figure 4.5: Local orthonormal coordinate system with origin in current vessel centerline point. The red line is a vessel centerline. The blue dashed line is the tangent to the centerline in current point. Axes $i = 1$, yellow arrow, and $i = 2$, green arrow, are perpendicular to the tangent. Axis $i = 3$, purple arrow, is parallel with the tangent.

The size of the box should be large enough to accommodate the largest bifurcations but should also be as small as possible to diminish the possibility of several bifurcations, other vessels, and structures within one box. As branching vessel radii, R_B, according to Equation 2.2, is related to main vessel radius, R_V, it makes sense to scale the box size by current vessel radius. Similarly, the step length between centerline points for which boxes are extracted should be as large as possible while small enough to not miss any bifurcations. In Figure 4.6 branching vessel radius, R_B, main vessel radius, R_V, box height, H, box width W and box step length, d, are defined. As the vessel is cylindrical the base of the 3D box is a square with side length W.

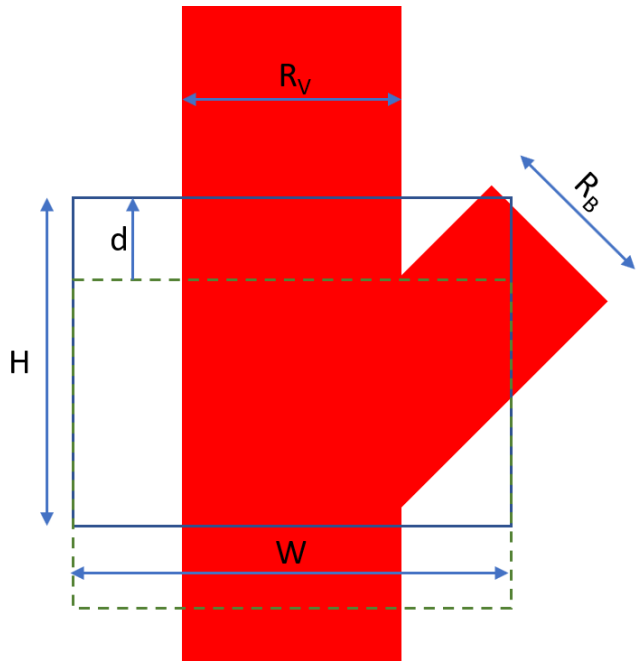
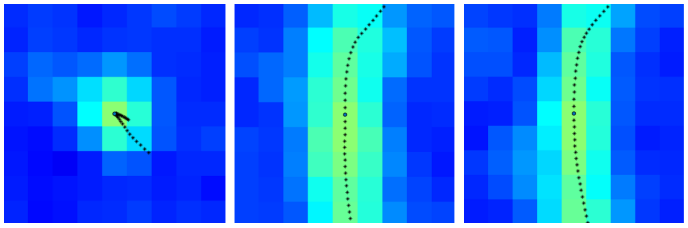


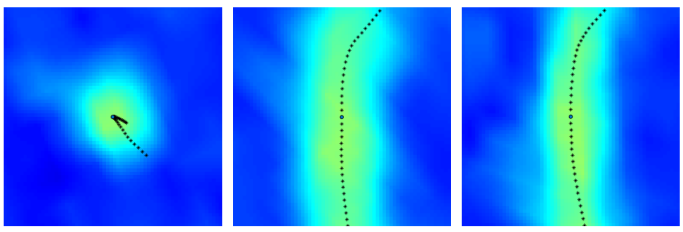
Figure 4.6: 2D model of vascular branch with sample box. The following measures are defined: branching vessel radius, R_B, main vessel radius, R_V, box height, H, box width W and box step length, d.

Equation 2.2 and 2.3 some further calculations were made. For a two branch bifurcation Equation 2.2 gives that with one radius, $r_1 \to 0$, going to zero the other radius, $r_2 \to r$, goes to the main vessel radius. In this limit Equation 2.3 we have $\cos(\theta_1) \to 0$ and $\cos(\theta_2) \to 1$ so $\theta_1 \to 90°$ and $\theta_1 \to 0°$. In the other extreme case, the branch radii are equal, $r_1 = r_2$. Equation 2.2 then give $r = 2^{1/\gamma} r_1 = 2^{1/\gamma} r_2$. By Equation 2.3 we the obtain $\cos(\theta_1) = \cos(\theta_2) = 2^{2/\gamma - 1}$. With the values for γ suggested in Section 2.2.1 for we obtain the lowest possible angle $\theta_1 = \theta_2 \approx 25°$ in the turbulent flow case. Bifurcations with a maximum total angle of $\theta_1 + \theta_2 \approx 90°$ and a large and small branch are very cover with a small box. The case which will decide lower limit of the box size is the bifurcation with equally large branches and total angle $\theta_1 + \theta_2 \approx 50°$. However, these results didn't match observations in real CTA data perfectly, so even if the equations can give pointers I had to confirm that the box size was suitable in real observations.

Lastly the CTA must be resampled in the box at a uniform grid align with the local coordinate system. The resolution of the of the grid is preferably as low as possible while still being able to discern bifurcations to reduce number of computations and memory usage. Eligible resolution depends on box size. Which resolution I used in this project is specified in Section 5.1.5. As box extraction will be a part of the algorithm to find bifurcations along a vessel, low computation time is desirable. Therefore, a fast sampling algorithm is preferable. Nearest neighbour interpolation was not good enough for very tiny vessels, so I used linear interpolation when resampling. In Figure 4.7 a tiny vessel is resampling at grid points from low to high resolution.



(a) Resampled at grid points without up sampling resolution.



(b) Resampled at grid points, up sampling to 64x64x64 resolution.

Figure 4.7: Tiny vessel resampled at grid points. The figure shows three slices which are mutually perpendicular. The images to the left are slices perpendicular to the centerlines respectively.

4.2.4 Labelling of Data Sets

I labelled the data sets by defining bounding boxes for each bifurcation along the vessel such as the one seen in Figure 4.8. I then classified extracted boxes as a bifurcation if a box fully contained a bonding box, else as a non-bifurcation. This way the labelling of extracted boxes was automated and did not have to be redone for various box sizes. In total I defined bounding boxes for about 100 branches from five different CTAs.

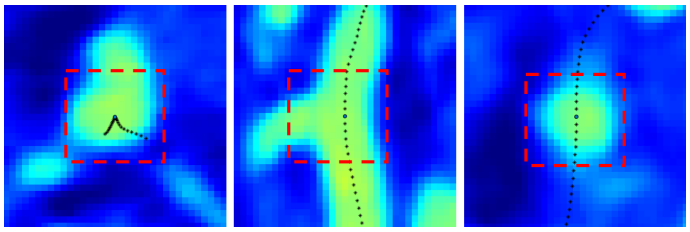


Figure 4.8: Bounding box place around a bifurcation. The figure shows three slices which are mutually perpendicular. The image to the left is a slice perpendicular to the centerline.

4.2.4.1 Bifurcation Samples

To get good class balance, I extracted boxes containing bifurcations first. For each bifurcation I extracted a number of boxes with random rotations around axis $i = 3$ as defined in Figure 4.4 and at random centerline positions such that the bounding box still was fully contained by the box. Using many copies of the same bifurcation allows for including more non-bifurcation boxes without the class balance suffering and each bifurcation to be included from various angles and offsets.

4.2.4.2 Specially Labelled Non-Bifurcation Samples

After I had extracted all boxes with bifurcations, I extracted boxes around specially labelled vessel points. These specially labelled points I had specified to make sure certain important non-bifurcation structures were included in the data sets. Which areas were important I determined by training a network on a data set without specially labelled points and examine which boxes the trained network had high loss on. Examples of types of samples that were specifically included in the data set:

- the internal carotid artery passing through the temporal skull bone
- the vertebral artery passing through the vertebra
- vessels sharply bending
- close parallel vessels

4.2.4.3 Attain Class Balance

To achieve good class balance I filled up the rest of the data set, containing bifurcation boxes and specially labelled non-bifurcation boxes, with random non-bifurcation boxes until the total number of non-bifurcation boxes matches the number of bifurcation boxes. The boxes I used to complete the data set was randomly picked after excluding sections of the centerline containing bifurcation bounding boxes and specially labelled non-bifurcations.

4.2.5 Rescaling and Clamping Intensities

I rescaled intensity levels of the CTA images to the dimensionless physical Hounsfield Scale which are described in Section 2.1.1. The translation between intensity level and HUs are defined by the DICOM-file header attributes rescale slope and rescale intercept. Much of the CTA images contains HU values which are way out of the of range of HU values for vessels and surrounding anatomical structures. These outlying values I clamped into a relevant range lessen their impact in the CNN model. As mentioned in Section 2.1.1, typical values for arteries containing radio contrast fluid in CTAs range from 200 to 600 HUs. To keep distinctive differences in HU values of background and other anatomical structures like bone I added a margin to the artery HU range. Much of the background is constituted by water, fat, connective tissue, and other bodily fluids which have HU values around 0. Air has a value of -1000 HUs, but it is not relevant to distinguish between air and background when looking for vascular bifurcations unless perhaps if looking in the lungs. Therefore, I used a lower threshold at 0 HU or lower for clamping intensities. The highest relevant HU values are bone tissue which has values in 400 to 1800 HUs. However, there are not any other relevant structures above 600 HUs which means the upper threshold for clamping can be set much lower than 1800 HUs. An upper threshold of 800 HUs leaves 200 HUs above the upper end of the artery HU range which is enough to distinguish vessels and the same 200 HUs as the difference between a lower threshold of 0 HUs and the lower end of the artery HU range.

4.2.5.1 Intensity Standardization

As explained in Section 3.2.3 standardization of the input data hastens the training process of NNs. This standardization also will have to be applied when samples are feed to the trained network model. For consistency in standardization between various sample boxes and CTAs, I used set values of standard deviation and mean. The alternative would be to compute standard deviation and mean for either entire CTA or each sample box. Due to variance in ratios between background and anatomical structures and noise the standard deviation varies greatly between different sample boxes and CTAs. Hence HU values would be mapped differently for each CTA or sample box. Set values for the standardization makes sure the values of vessel voxels will be in the same range after standardization and retains the physical property of HUs. Additionally, set values avoids having to compute standard deviation, σ, and mean, μ, inside the bifurcation detection algorithm.

I performed standardization according to Equation 4.1, where I chose S and D to scale the intensities from -1 to 1. After clamping intensities range from 0 to 800 this gives us

$$\mu = \frac{0 + 800}{2}$$

and

$$\sigma = 800 - \mu.$$

So summarily the intensity levels, I, were pre-processed as follows:

1. Converted to Hounsfield Units according to Equation 2.1
2. Clamped to range $[0, 800]$

$$I_{\text{clamped}} = \begin{cases} 0, I < 0 \\ 800, I > 800 \end{cases}$$

3. Standardize to

$$I_{\text{standardized}} = \frac{I_{\text{clamped}} - \mu}{\sigma} \tag{4.1}$$

4.3 Implementing the CNN

Implementing and training a 3D CNN is the main focus of this thesis. I chose CNN as a method mainly due to having higher potential in robustness, accuracy and stability than other methods viewed in Section 2.3.1 according to [6]. The alternative of 2D CNN connected to a RNN wasn't pursued due to being less examined in the context of medical image processing and most likely being more difficult to train. I used Python and the deep learning framework, PyTorch[50], for implementing and training the CNN. In addition to PyTorch, I used utilities from the packages Monai[51] and TorchIO[52]. Monai and TorchIO contain tools for deep learning in medical imaging based on PyTorch. In Table 4.1 the versions of used software is listed.

Table 4.1: Versions of software and packages used in the implementation of the network.

Python 3.6.12	PyTorch 1.6.0	Monai 0.3.0+60.gbd1e008	TorchIO 0.17.52

4.3.1 Pytorch, Monai and TorchIO

PyTorch is one among several popular deep learning frameworks. It provides a intuitive, Pythonic style interface to implementing and training deep learning. Other frameworks including Caffe, TensorFlow and Theano represent computations with

a static graph which is repeatedly applied to on the data set. A static graph can have some advantages when it comes to optimizing performance and scalability. PyTorch uses a dynamical approach which execute tensor computations one by one to dynamically form a network graph and does differentiation automatically for each computation step. Even so PyTorch is comparable in speed with the fastest frameworks. The dynamical approach allows for simple debugging and changing the computation graph at runtime without having to recompile. PyTorch can run its computations on the GPU by interfacing to NVIDIAs parallel computing platform, CUDA. Which lowers training time considerably. [50]

Monai [51] and TorchIO [52] are Python packages which implements many tools in PyTorch for deep learning in medical imaging. They integrate into the normal PyTorch workflow. Both packages extend PyTorch mainly with tensor transforms customized for medical images in 3D which are useful in data augmentation and pre-processing and are used in this project. Specifically what are used from each package is specified in Section 4.3.3.5.

4.3.2 Network Architecture

DenseNet, a Densely Connected Convolutional Network originally suggested in [43], is network architecture I chose in this project. The building blocks and structure of DenseNet are described in Section 3.2.4.1. Compared to earlier suggested popular architectures for medical imaging like U-net[53], ResNet[42], DenseNet shows promise for medical imaging applications according to [54] and [55]. Furthermore, DenseNet has shown great performance for image classification on popular data sets like CIFAR [44] and ImageNet [38].[43]

I implemented a 3D DenseNet with PyTorch in Python closely based on the implementation by Monai [56]. A detailed example of the network architecture I used is shown in Table 4.2. Various combinations of parameter settings were attempted, but all followed this structure. The parameters are explained in Section 4.3.2.1.

Table 4.2: A network architecture used for image boxes with resolution $64 \times 64 \times 64$. In the table "conv" includes the entire BN-ReLU-Conv sequence. The first convolution layer with $7 \times 7 \times 7$ kernel has filter size 32 (32 output channels). Each $3 \times 3 \times 3$ kernel convolution (dense layer) in the dense blocks has a filter size of $k = 4$, where k is the growth rate of DenseNet. The last layer is a fully connected softmax classification layer gives 2 output classes.

Layers	Output Size	Building Block
Convolution	$32 \times 32 \times 32$	$7 \times 7 \times 7$ conv, stride 2, pad 3
Pooling	$16 \times 16 \times 16$	$3 \times 3 \times 3$ max pool, stride 2, pad 1
Dense Block (1)	$16 \times 16 \times 16$	$\begin{pmatrix} 1 \times 1 \times 1 \text{ conv} \\ 3 \times 3 \times 3 \text{ conv, pad 1} \end{pmatrix} \times 3$, stride 1
Transition Layer (1)	$16 \times 16 \times 16$ $8 \times 8 \times 8$	$1 \times 1 \times 1$ conv $2 \times 2 \times 2$ average pool, stride 2
Dense Block (2)	$8 \times 8 \times 8$	$\begin{pmatrix} 1 \times 1 \times 1 \text{ conv} \\ 3 \times 3 \times 3 \text{ conv, pad 1} \end{pmatrix} \times 5$, stride 1
Transition Layer (2)	$8 \times 8 \times 8$ $4 \times 4 \times 4$	$1 \times 1 \times 1$ conv $2 \times 2 \times 2$ average pool, stride 2
Dense Block (3)	$4 \times 4 \times 4$	$\begin{pmatrix} 1 \times 1 \times 1 \text{ conv} \\ 3 \times 3 \times 3 \text{ conv, pad 1} \end{pmatrix} \times 7$, stride 1
Classification Layer	1×1	global average pool flatten tensor 2D fully connected, softmax

4.3.2.1 Network Architecture Parameters

The architecture parameters I used are the same as in the implementation of DenseNet in Monai[56]. These are *initial features*, *growth rate* k and *block configuration*. *Initial features* is the filter size of the initial $7 \times 7 \times 7$ kernel convolution. Growth rate, k, is the growth rate defined in [43], e.g., how many filters are added in each layer. Block configuration is how many layers there are in each dense block meaning if the block configuration for a dense block is 5 it will consist of 5 dense layers which each has filter size k. In Table 4.2 the initial features are 32, the growth rate, k, 4 and the block configurations 3, 5, 7 for each dense block respectively.

4.3.3 Training the Network

To find the suitable weights for the network it must be trained. How a neural network is trained is explained in Section 3.3. In PyTorch networks are trained in a loop where a data set of training samples are repeatedly feed to the network, meanwhile computing gradients and updating the network weights accordingly through backpropagation.

4.3.3.1 Summary of Training Algorithm

This is a step-by-step summary of the training algorithm which parts will be described separately in following sections.

1. Divide data set in training and validation sets
2. Initialize network parameters
3. Select a batch of samples from the training data set
4. Augment the data batch
5. Calculate the network output of the data batch
6. Calculate loss between outputs and target ground truths
7. Backpropagate the gradients of the loss
8. Update network parameters with optimiser according to the gradients
9. Evaluate network performance on validation set
10. Go to step 3 and repeat until stopping condition is reached

4.3.3.2 Training and Validation Set

How a data set is divided in training and validation set is very important for performance of the network. Especially when using a small data set. There is a risk that important rare type of samples are absent in one of the sets. If the data set representation is poor in the training set the network will not learn the missing samples types and perform poorly. If sample types are not represented in the validation set the evaluation will show the network overperforming on the validation set. As the finished model chosen is based on performance on validation set this could result in a less optimal model, which only performs well on the overly simple validation set, being picked.

Another important issue is that the training and validation set must be completely independent from each other. For example, as the same branch is sampled at different locations and angles, all samples for a unique branch should be kept together and restricted to either training, validation set or the test set. Otherwise, evaluation on the validation set will show the network overperforming as the network has learned identical branches from the training set.

Accordingly, the samples I chose to include in training, validation and test sets were from separate CTA, respectively. Samples were picked so the same types of bifurcations are represented in each CTA.

4.3.3.3 Kaiming Initialization

I initialized network parameters with Kaiming initialization [57] as in the original DenseNet paper [43]. In Kaiming initialization all biases are initialized to zero. Batch normalization weights are initiated to 1. The weights, w_l, of a convolutional layer, l, are initialized based on a criterion for ensuring that gradients do not become exponentially large or small. In [57] this criterion is:

$$\frac{1}{2} \cdot n_l \cdot Var(w_l) = 1, \forall l,$$

where $n_l = k^2 \cdot d$ is the number of inputs to layer l, with the input $k \times k$ pixels and filter size d. Therefore, the initial weights will be sampled from a gaussian distribution,

$$w_l = N(0, \sigma_l),$$

with zero mean and standard deviation

$$\sigma_l = \sqrt{\frac{2}{n_l}}.$$

4.3.3.4 Mini Batches

During training data can be feed sample by sample, the entire data set in a batch or as subset of samples called mini batches. Feeding mini batches of data to the network allows more utilization of parallel processing on the GPU compared to one sample at a time. It also lowers variance of network parameter updates which can lead to more stable convergence. However, if batches are too large it is difficult to fit on the memory and the training can be slow. A suitable mini batch size depends for various networks and data sets. Common mini batch sizes used in training of image classification networks are 4, 8, 16 or 32.[58] For this project I found a mini batch size of 32 to work well.

4.3.3.5 Data Augmentation

In deep learning for medical imaging the size of training data is generally small which makes data augmentation particularly important. The size of the training set is virtually enlarged which is crucial to be able to generalize from a small data set. In this project I augmented the data online in every iteration of the training loop. Hence new samples were seen every cycle which made the network less susceptible to overfitting. Online augmentation clearly prolongs the training cycle. However, Monai[51] and TorchIO[52] implements the most computation heavy augmentation transforms on the GPU which speeds up the process immensely. If augmentation

had been performed offline, before the training loop, a large amount of memory would have to been allocated to store the extra augmented data sets.

I decided which transforms to apply and their parameter settings by visual inspection. The transform must preserve the ground truth of the samples and also output a sample which plausibly is an anatomy from a CTA. The augmentation transforms are described in Section 3.3.6 and Section 5.1.4. Transforms performed for data augmentation in this project were in order:

1. Random flip along an axis
2. Random motion artefacts [59]
3. Add random gaussian noise
4. Random spatial and elastic transforms
    (a) Random rotation along tangent to centerline
    (b) Small random rotations along axes perpendicular to centerline
    (c) Small random translations
    (d) Small random scaling
    (e) Random affine shearing transform
    (f) Random elastic deformation transform

Flipping along an axis is straightforward and gives 3^2 plausible samples from each sample in the data set. Implementation of random Gaussian noise is also straight forward. Flipping and Gaussian noise was implemented using `torchio.transforms.RandomFlip` and `torchio.transforms.RandomNoise` from TorchIO[[52]]. Motion artefacts simulates artefacts from the patient moving during the CTA scan. The TorchIO[[52] implementation, `torchio.transforms.RandomMotion`, based on Shaw et al.[59], was tested. Spatial transforms were implemented with `monai.transforms.Rand3DElastic` from Monai[[51]. Some of these spatial transforms resample outside of the sample image which means the image has to be padded. Border padding, using the value at the closest border, was used as it doesn't introduce any large gradients or implausible anatomical structures. Rotations were small for rotations along axes perpendicular to the centerline as the orientation of the centerline is a consistent feature of the samples separates the known vessel from the unknown bifurcations. Translations and scaling were included to simulate potential imperfections of the given vessels' centerlines and radii. Affine and elastic transformation represents variation in anatomies of different patients and vessel sections to create almost novel vessel anatomies. Elastic transforms are described in Section 3.3.6. Figure 4.9 shows a sample of a bifurcation, to the same sample all augmentation transforms have

randomly been applied in Figure 4.10

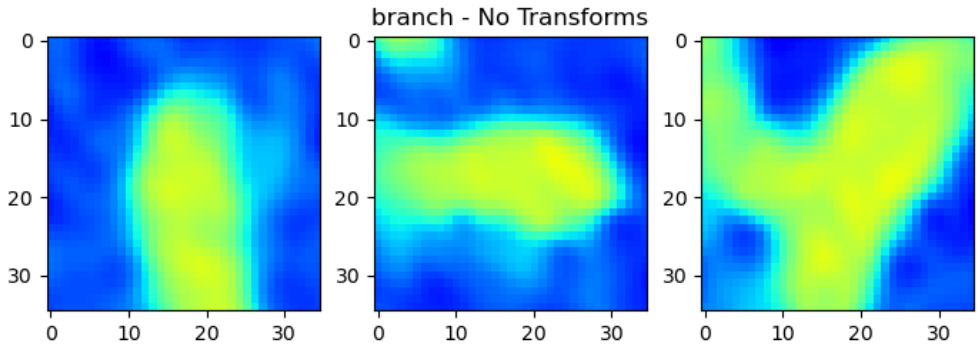


Figure 4.9: A sample of a bifurcation. The figure shows three slices which are mutually perpendicular. The image to the left is a slice perpendicular to the centerline.

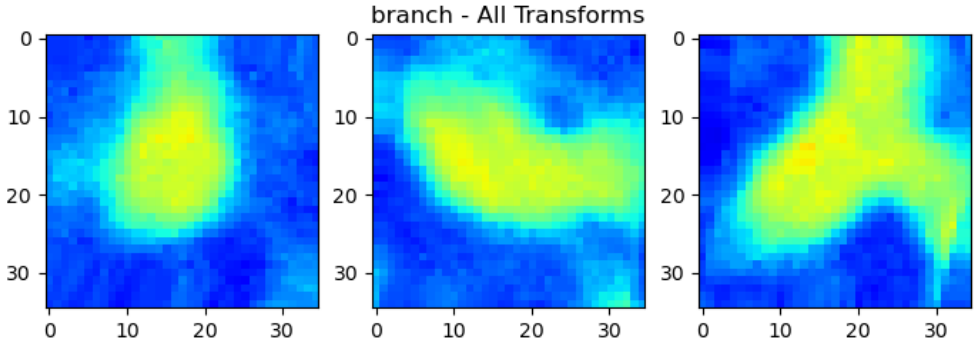


Figure 4.10: All augmentation transforms applied to the sample seen in Figure 4.9. The figure shows three slices which are mutually perpendicular. The image to the left is a slice perpendicular to the centerline. Parameters used in the augmentation are specified in Section 5.1.4

4.3.3.6 Loss Function and Optimiser

For classification networks typically a cross entropy loss is used as mentioned in Section 3.3.1. In multi-class classifications problems, as in [43] and in this thesis, a softmax (see Equation 3.2) layer is added after the fully connected layer. For softmax out, multi-class cross entropy is used as loss function which is negative log likelihood loss function. As the classification problem really is binary, with two classes, a sigmoid (see Section 3.1.1) in the output layer with a binary cross entropy loss function is basically the same thing.

In my implementation I use `torch.nn.CrossEntropyLoss` from PyTorch[50] which combines a logarithmic softmax layer with negative log likelihood loss. The output of logarithmic softmax is a probabilities

$$\hat{y}_i = \text{LogSoftmax}(x_i) = \ln\left(\frac{\exp(x_i)}{\sum_{j=1}^{M}\exp(x_j)}\right),$$

where x_i is the ith network output and the length of x is M, the number of classes. Negative log likelihood loss, $L(x, y)$, of network output x and target class vector y is given by

$$L(x, y) = -\frac{1}{M}\sum_{i=1}^{M}\hat{y}_i.$$

The optimizer I used was the Adam algorithm, implemented by PyTorch [50], which is widely used as a quick and efficient optimizer as well as being easy to use. Adam was suggested in [46] and is explained in Section 3.3.3. Basically, Adam is based on SGD but adaptively adjust learning rates individually for each weight.

4.3.3.7 Early Stopping

Network training iterations continues until the set stopping criteria is reached. The criteria usually are based on performance on the validation set which indicates how well the network generalizes from the training set. In this thesis the training was stopped when a moving average of the classification accuracy on the validation set did not increase anymore or if the moving average of loss on the validation set started increasing. Then the model with best classification accuracy on the validation set was chosen.

4.3.3.8 Optimising Training Setup

There are a lot of parameters which affects how quick the training process is and how well the trained network model will perform. One of the most important factors is picking good samples for training data. I iteratively improved the training data by adding new samples chosen based on previous attempts of trained network models. The goal for new samples were to resemble previous samples with high loss to emphasis learning these in the next training experiment. Other factors include division of training and validation set, mini batch size, various combinations of augmentation transforms and their parameters, initial learning rate and size of the various network layers. I found a setup with adequate performance by conducting series of training experiments featuring various setups.

4.4 Evaluating the Network

Predictions of the network is decided by which of the two output classes is given the higher score. If both outputs score similarly the sample might have to be marked down for further evaluation. I used samples with inconclusive classification scores

for improving the data set. If these samples were labelled inconsistently from other samples, I replaced them. Else, if the consistent, I added more similar samples to the data set to improve learning of those particular class features.

Performance of the network model I evaluated based on classification of a test data set taken from a CTA separate from the CTAs included in the training and validation sets. The measures I used for evaluation were precision, recall and F1-score (also called dice score). These are statistical measures of classification accuracy commonly used for evaluating classification models. Precision determines the accuracy of the network's positive predictions and is calculated:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.2}$$

where TP is the number of true positive and FP the number of false positive. Recall determines the accuracy of predictions for actual positive samples and is calculated:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.3}$$

where FN is the number of false negative. The F1-score is a harmonic mean which emphasises balance of between precision and recall and is calculated:

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

I also used loss and accuracy curves from the training in evaluation of the network. From the learning curves signs of overfitting and poor data representation in the training or validation sets can be found. For example, if the learning curves of the training data continues to improve while the curves of the validation data stops improving or deteriorates the network has overfitted the training data.

4.5 Finding Seed Points in Branching Vessels

After obtaining predictions from the classification network, seed points in the protruding vessel branch must be found. A CNN could potentially handle the prediction of seed points as well. However, classification CNNs are more proven in prior research and above all is a lot easier to label unambiguously. Hence, I went with classification CNN and adopted non machine learning approaches like Frangi filtering[19] and skeletonization. Additionally, the classification network handles cases where these methods alone would have given non-branch seed points. The approach I used in this project involves skeletonization of a vessel segmentation in the sample. Seed points could then be drafted seed points from the skeletonization curves.

4.5.1 Segmentation of Bifurcation Sample

In the sample classified as a bifurcation the vessels will emanate from the center of the box. By applying Frangi filters, as proposed by Frangi et al.[19] and described in Section 2.3.1.1, a measure for vesselness is obtained which can be used to label all voxels above a threshold as a vessel. Then the bifurcation can be segmented by selecting the connected component closest to the center of the sample box. In this thesis I applied Frangi filter on scales of 1, 3, 5 and 7 with filter parameters $\alpha = 0.5$, $\beta = 0.5$ and $\gamma = 50$. These parameter settings were suggested in [60] and gave satisfactory performance in this project. The threshold was 0.01 in the Frangi measure (see Section 2.3.1.1). A segmentation can be seen in Figure 4.11 with the segmentation overlaid with the original sample and in Figure 4.12 and Figure 4.13 in 3D together with skeletonization curve.

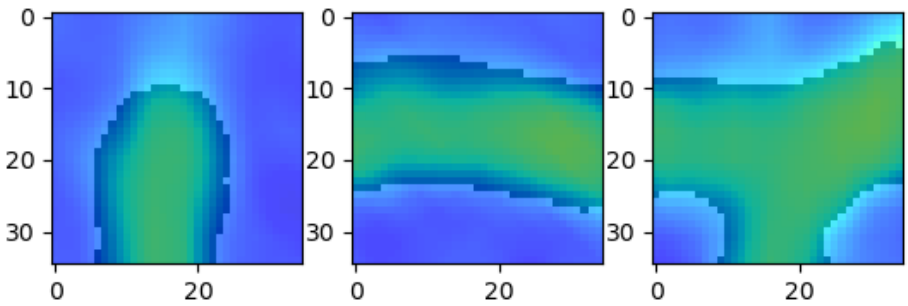


Figure 4.11: Segmentation overlaid with sample. Segmented by applying a threshold of 0.01 to a Frangi filter sample. Frangi filter applied on scale 1, 3, 5 and 7 with parameters $\alpha = 0.5$, $\beta = 0.5$ and $\gamma = 50$.

4.5.2 Skeletonization and Selecting Seed Points

Skeletonization of the segmented bifurcation I performed using an implementation from skimage[61] of a method suggested by Lee et al. [34] which is described in Section 2.3.3.1. I then selected seed points from the part of the skeletonization seen in Figure 4.12 and Figure 4.13. The bifurcation location can be found by finding a voxel in the skeletonization with three neighbours. A 3D Moore Neighbourhood [35] is used for this which includes the 26 closest surrounding voxels[34]. From the bifurcation voxel the average distance between each of the protruding curves and the original known centerline can be calculated. The curves with the lowest average distance to the centerline can be discarded as belonging to the originally known vessel. The remaining curve is kept as seed points for the branching vessel and constitutes the output of the bifurcation detection.

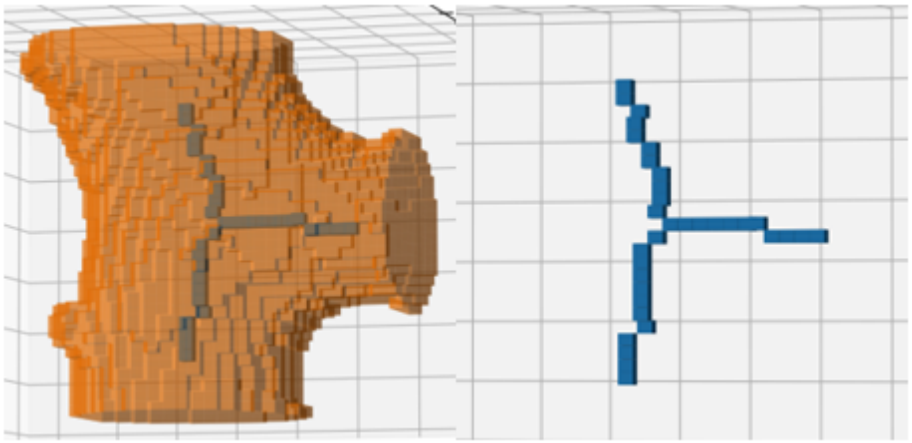


Figure 4.12: Skeletonization of segmentation in Figure 4.11. In the left image the segmentation is included.

In Figure 4.13 the sample contains two bifurcations. In this case the bifurcation location closest to the center of the sample should be used for suggesting seed points. The other bifurcation will be found in another sample further down the centerline.

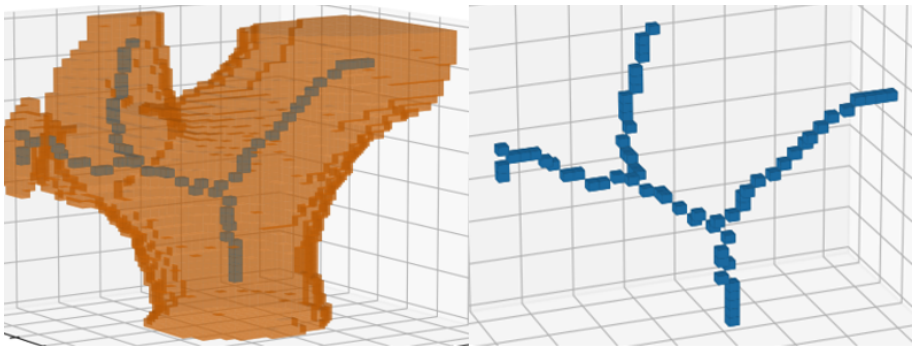


Figure 4.13: Skeletonization in sample containing two branches. In the left image the segmentation is included.

4.5.3 Evaluating Branch Seed Points

Evaluation of the branch seed points can depend on what application they're used for. As no data including ground truth of the seed points is featured in this thesis a comprehensive evaluation is difficult to perform. Hence quality of the seed point extraction was evaluated by visual inspection of choice samples.

4.6 Summary of Assembled Algorithm

Here Chapter 4 is summarized and listed in order of execution to describe the complete bifurcation detection algorithm. The algorithm assumes that the centerline and radii are given for the blood vessel to find bifurcations along.

1. Select points on the vessel centerline inter-spaced by a step length varying with vessel radius, see 4.2.3.1
2. Loop over selected points
   (a) Resample the CTA in a 3D box centered at the point, see Section 4.2.3.1
   (b) Rescale and clamp intensity, see Section 4.2.5
   (c) Classify sample as branch or non-branch with the classifier CNN, see Section 4.4
   (d) Segment by applying Frangi filter and a threshold, see Section 4.5
   (e) Skeletonize the segmented components, see Section 4.5.2
   (f) Find skeletonized voxels with 3 neighbours and select the one closest to the middle as bifurcation location, see Section 4.5.2
   (g) For each curve protruding from the bifurcation location; calculate the average distance to the originally known vessel centerline, see Section 4.5.2
   (h) Select and output the curve with largest average distance as bifurcation seed points, see Section 4.5.2

5

Results

In this chapter results from implementing and applying the methods in Chapter 4 are presented. Included are results from CNN training, performance of a trained CNN and behaviour of the seed point extraction scheme both unaided and in conjunction with the CNN output.

5.1 Network Training

As explained in Section 4.3.3.8, there are a multitude of parameters which influence the training. I attempted various setups of parameter settings to find suitable ones. In this section resulting choices from such experiments are presented as well as learning curves for the most significant experiments. The training setup I used for results in later sections is specified here.

5.1.1 Hardware Specifications

Everything in this thesis, including network training, was tested to run on a single NVIDIA Quadro M2000M GPU with 4 GB memory. However, most of the network training was done on a remote NVIDIA Tesla K80 GPU with 12 GB memory through Google Colab [62].

5.1.2 Training Parameter Setup

In training, the Adam algorithm [46], mentioned in Section 3.3.3, was used as optimizer. After some training runs I concluded an initial learning rate of 10^{-3} was working well. The loss function I used was NLL loss (see Section 4.3.3.6) and the mini batch size 32 (see Section 4.3.3.4). I picked the best model according to the stopping criteria specified in Section 4.3.3.7. However, for learning curves presented in this chapter, I continue the training after the best model had been picked to further visualize learning dynamics.

5.1.3 Network Architecture Setup

The network size I used for the results in Sections 5.2 and 5.3 was the one specified in Table 4.2, with 32 initial features, growth rate 4 and block configurations 3, 5

and 7 (these network parameters are explained in 4.3.2.1). Larger networks did not change the learning dynamics qualitatively. Very large networks were prone to overfitting the training data and performing worse on validation set. Smaller networks were not able to learn the training set well. This made the training performance generalized well to the validation sets, but overall, the performance on validation set was worse.

5.1.4 Data Augmentation Setup

I found that a suitable augmentation setup for the final data set was as presented in Section 4.3.3.5 excluding the random motion artefact transforms. This augmentation setup was used for the results in Sections 5.2 and 5.3.

To determine how various augmentation transforms affected the learning I performed numerous experiments. In Figure 5.1 learning loss curves are shown for some of the various augmentation setups. As can be seen augmentations clearly enable lower loss than without augmentation. Random flip transforms decrease the loss by a large margin. Adding Gaussian noise further decrease the loss. Random motion artefacts increase the loss slightly and hence is not included in the final augmentation setup.

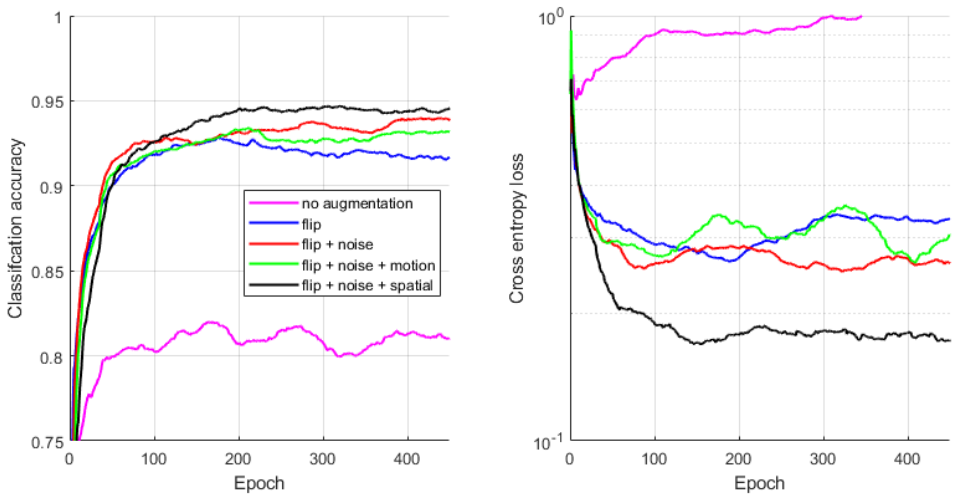


Figure 5.1: Accuracy (left) and loss (right) on validation set during training. Comparison of augmentation transforms. Clearly augmentation noticeably improves performance on validation set, i.e., generality. No augmentation (purple curve) stops improving after just a few epochs. Random flipping along axes (blue curve) drastically improves the generality. The addition of random Gaussian noise (red curve) perhaps slightly improves generality, but above all increases the learning rate in the start of training. No discernible improvement comes from adding random motion artefacts (green curve), arguably the generality instead deteriorates. Finally adding the random spatial transforms (black curve) boosts in generality. Loss and classification accuracy doesn't seem to correlate perfectly.

5.1.5 Data Set

After repeated training experiments and sample additions, the final labelled data set consisted of 663 branch samples taken from 103 unique branches and balanced out with 663 unique non-branch samples. Sample box size was $H = W = 1.5 * R_V$, using the notation specified in Figure 4.6. The resolution of sample boxes was $64 \times 64 \times 64$. Samples came from five different CTAs. Data samples were divided by CTA, the training set from three CTAs, the validation one CTA and the test set from one CTA.

	Training Set	Validation Set	Test Set
Data division 1	CTA#1, #2, #3,	CTA#4	CTA#5
Data division 2	CTA#1, #2, #5,	CTA#3	CTA#4
Data division 3	CTA#3, #4, #5,	CTA#1	CTA#2

Table 5.1: Three different divisions of the data set into training, validation and test sets. The CTA notation is introduces in Section 4.2.1.

Effects of different division of CTAs into training, validation and test sets can be seen in Figures 5.2, 5.3 and 5.4. The divisions are specified in Table 5.1.

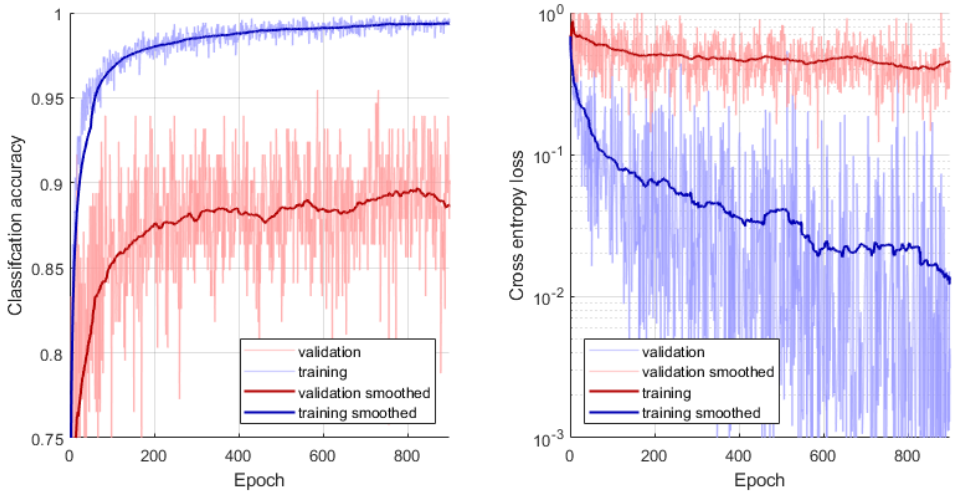


Figure 5.2: Learning curves for accuracy and loss for Data division 1 as presented in Table 5.1. The accuracy on the validation set reaches approximately 95%. After about 300 epochs the validation loss reaches a plateau while the loss for the training set keeps decreasing. At this point further learning doesn't generalize well to the training set, so there are presumably some types of samples present in the validation set which aren't in the training set making the validation set too difficult.

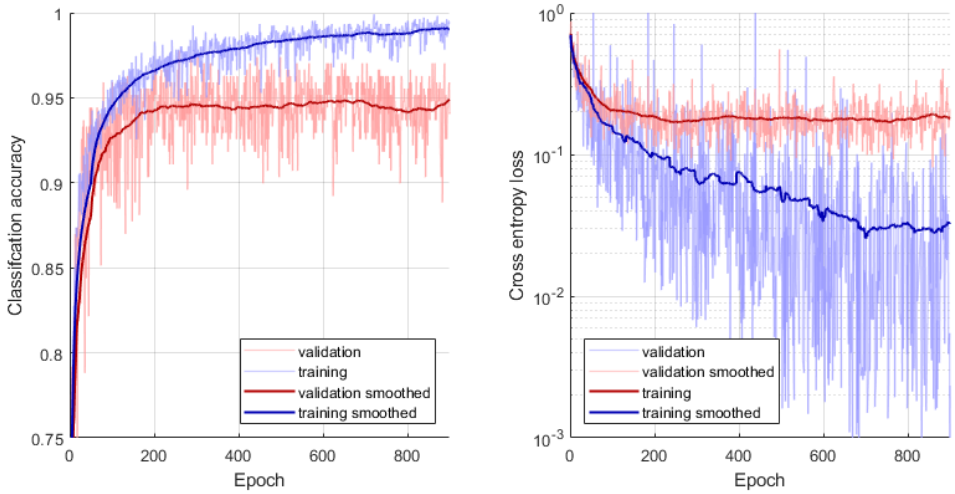


Figure 5.3: Learning curves for accuracy and loss for Data division 2 as presented in Table 5.1. Accuracy for the validation set does not exceed 90%. Validation loss does not seem to lower much as the training set loss decreases. This indicates poor representation of the data set in the training set. Makes it impossible to generalize well to particular samples only represented in the validation set.

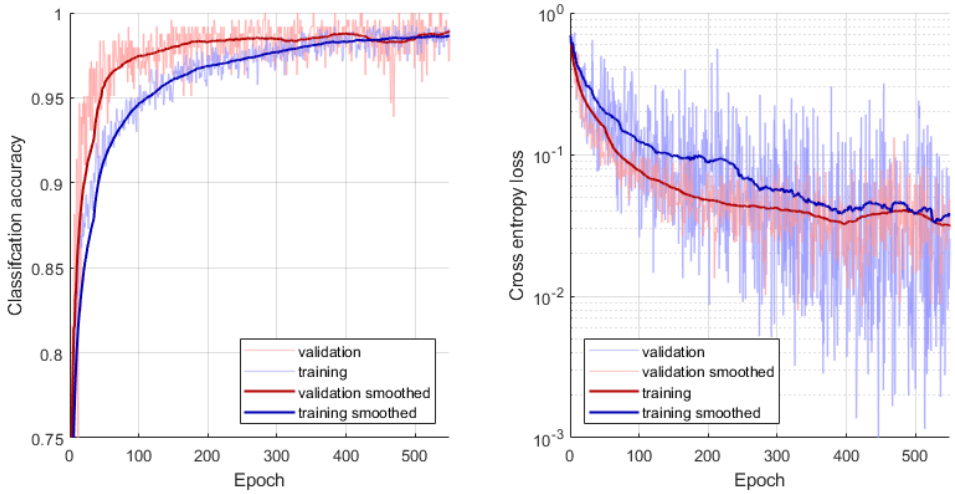


Figure 5.4: Learning curves for accuracy and loss for Data division 3 as presented in Table 5.1. Loss for the validation set is lower than for the training set but improves as the loss for the training set is minimized. This indicates that the learning generalizes well to the validation set. First after 500 epochs the curves for the training set catches up. However, 95% validation accuracy is reached very quickly, which could indicate that the validation set is too simple compared to the training set. But the training set is being enlarged by online augmentation which make it take longer to learn.

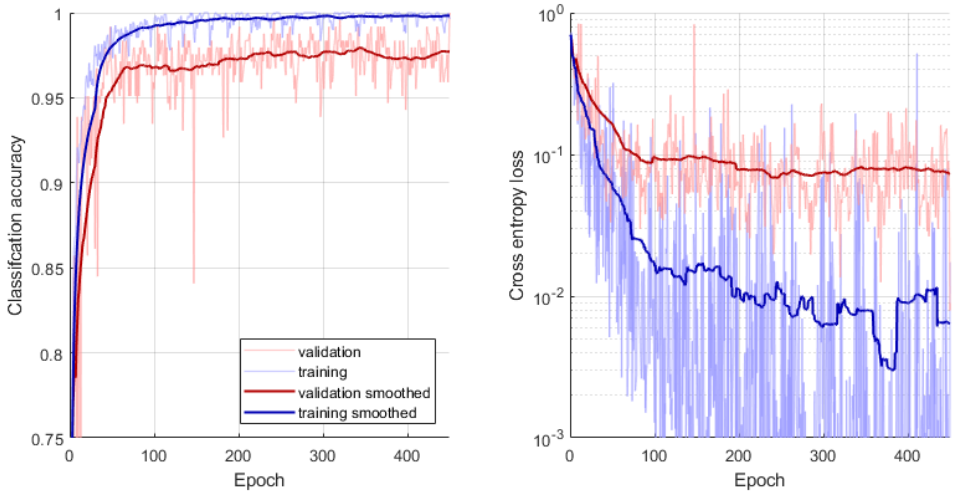


Figure 5.5: Same data set division as in Figure 5.4, division 3 according to Table 5.1. In this figure the only augmentation transform applied is random flip along axes. Here we can see that the validation accuracy is not higher than the training accuracy. This suggests that the initially low training accuracy in Figure 5.4 can be explained with augmentations increasing the learning time.

All three figures show that the accuracy clearly improves as the loss on the training set is minimized. However in Figure 5.2 the accuracy reached for the validation set is low, $\approx 90\%$, compared to $\approx 95\%$ in Figure 5.3, and $\approx 98\%$ in Figure 5.4. Learning on the training set is similar all three divisions of the data set. Overall, this indicates that some CTAs contain unique features which are not learned if those CTAs are not present in the training set. In all three figures improvement on the validation set flatten at about epoch 200 to 300. Improvement on the training set continues after this point. Further training after this point is overfitting but does not decrease performance on the validation set. In Figure 5.4 the loss curves of the training and validation sets show the most correlation out of the three figures, pointing to that the test set has a good representation of the data set. Accuracy initially is much higher for the validation set than the training set which could suggest that the validation set is too simple to predict and gives an overestimation of performance. However as explained in Figure 5.5 it has to be kept in mind that the training set is heavily augmented, thus featuring an enlarged data set which takes longer to learn. Figure 5.5 shows learning curves for the same data set division as in Figure 5.4 but with the only augmentation transform being random flip around axes.

5.2 Network Performance on Test Set

In this section the performance of the obtained classification model is evaluated on a test set of samples. Data set was divided according to Data division 2 in Table 5.1. The network and training setup used in for the results presented in this section is as specified in Section 5.1. The rule used for classifying samples as a branch is that the prediction score for branch class is higher than for the non-branch class,

and vice versa for classifying non-branches.

Table 5.2 presents precision, recall and f1-score, by category, for the classification model's predictions on the test set. Precision for the branch class is 0.94, recall 0.96 and f1-score 0.95. For the no-branch class precision is 0.97, recall 0.95 and f1-score 0.96. Lower precision and higher recall for branch indicates a very slight tendency towards classifying non-branches as branches.

Table 5.2: Statistical evaluation of the predictions on the test data set performed by the classification model. Data set was divided according to Data division 2 in Table 5.1. Details of the setup are referred in 5.2. Precision, recall and f1-score are included by category and accuracy for all predictions.

Category	Precision	Recall	f1-score
no branch	0.97	0.95	0.96
branch	0.94	0.96	0.95

In Figure 5.6 the model is evaluated on sample boxes along a vessel centerline with labelled branches. Samples were take at distance intervals of $d = 0.2 * R_V$, which is a suitable step length in order to not miss any branches with the box size specified in Section 5.1.5, $H = W = 1.5 * R_V$. Sample box notations are specified in Figure 4.6. As seen in Figure 5.6 predictions match the ground truth well. The ranges of samples being predicted as bifurcations are wider than the actual samples labelled with bifurcation. Class scores are practically symmetrical around zero. This will be the case for a two-class classifier with cross entropy loss as the loss is minimized as a sum of losses (see Section 4.3.3.6).

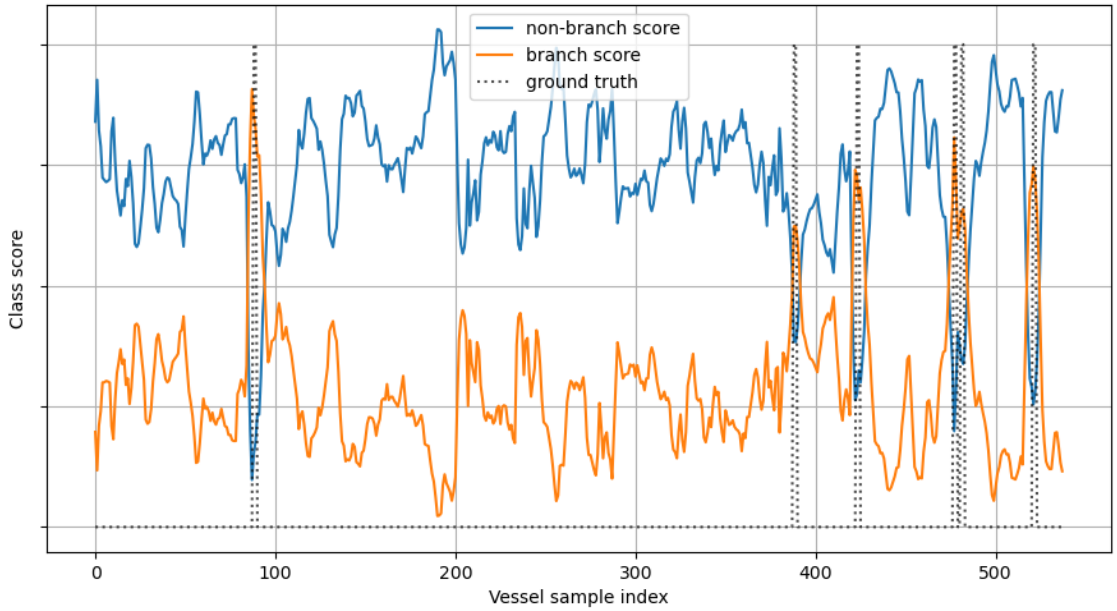


Figure 5.6: Classification model evaluated along a vessel centerline (such as seen in Figures 4.2 and 4.3) with labelled branches. The model's output class scores are plotted as a function of distance along centerline spline. (Orange curve) branch class score. (Blue curve) non-branch class score. (Gray dotted lines) is the ground truth which, in the figure, is high for a branch and low for a non-branch sample. Class scores are practically symmetrical around zero. Samples are classified as branches if the class score is higher for branch than non-branch. The ranges of samples being predicted as bifurcations are wider than the actual samples labelled with bifurcation. Details of the model are referred to in 5.2. Data set was divided according to Data division 2 in Table 5.1.

In table 5.2 present precision, recall and f1-score for using a the data set division in Figure 5.4. The predictions of the test set are perfect. However by reasons discussed in Section 5.1.5, it's possible that the test set could consist of easier sample, just as the validation set. Thus the results 5.2 should be considered as an potential overvaluation of the network performance until more rigid evaluation is performed.

Table 5.3: Statistical evaluation of the predictions on the test data set performed by the classification model. Data set was divided according to Data division 3 in Table 5.1. Details of the setup are referred in 5.2. Precision, recall and f1-score are included by category and accuracy for all predictions.

Category	Precision	Recall	f1-score
no branch	1.00	1.00	1.00
branch	1.00	1.00	1.00

5.3 Bifurcation Seed Points

Results presented in this section are seed points based on the bifurcation classified samples from Section 5.2. Some of these results has already been presented in Figures 4.12 and 4.13. In Figure 5.7 and Figure 4.12 typical bifurcations without much noise or background structures in the background are segmented and skeletonized. In these cases there only one way to pick seed points which clearly are within the branching vessel according to the selecting scheme in Section 4.5.2.

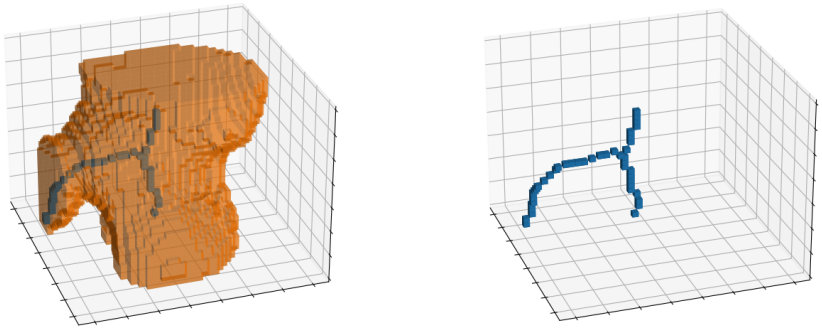


Figure 5.7: Segmentation, skeletonization and seed points of a sample correctly predicted as a bifurcation by the classification model. A typical case with only one relevant way to chose seed points.

Branch samples with much noise or background structures make the segmentation with Frangi filters more difficult and can result in false seed points suggestions in addition to the correct suggestion like shown in Figure 5.8. Using the suggested selection approach in Section 4.5.2 the seed point candidate closest to the center of the sample will be selected. It is very likely that the correct seed points will be selected in one of the branch-classified samples surrounding the bifurcation. For instance in Figure 5.8, the correct branch is chosen. However, in a later sample along the centerline one the false seed points will be closer to the center and are picked instead. False candidates will have to be handle separately and are not covered in the scope of this thesis.

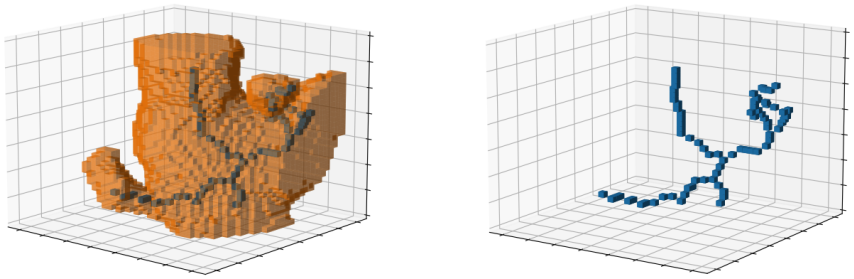


Figure 5.8: Segmentation, skeletonization and seed points of a sample correctly predicted as a bifurcation by the classification model. The segmentation is not good because of background structures and many reasonable seed point suggestions can be made from the skeleton. In this particular case seed points will be picked for the true branch as its closest to the center of the sample. However, if the sample were taken a bit further down the centerline the false seed points down to the left would be chosen.

An example of the branch classified samples far away from the actual branch, seen in Figure 5.6, is shown in Figure 5.9. The skeletonization does not show any branch, this is fine because it will be picked up in a later sample where the branch is more centered. Similar cases exist where correct seed points are indicated even if classification model incorrectly predicted a sample as a branch. Hence the wider range samples are being predicted as branches in 5.6 generally isn't an issue.

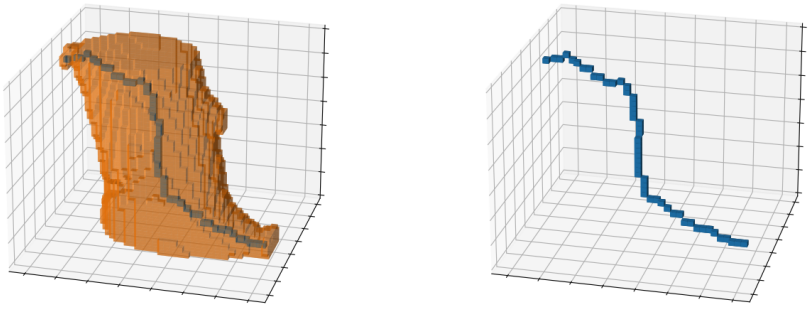


Figure 5.9: Segmentation, skeletonization and seed points of a sample incorrectly predicted as a bifurcation by the classification model. However, the sample is near enough to contain part of the bifurcation. In this case no seed points are indicated. However there are similar cases where correct seed points will be given even if the classification model gave a false positive bifurcation prediction.

If the samples have been miss-classified the segmentation will most likely suggest no branch seed points like in Figure 5.10. However in difficult areas with lots of background structures or noise false bifurcation seed points might be suggested as in Figure 5.11. Cases like this is rare and again false seed point candidates is not handled in the scope of this thesis.

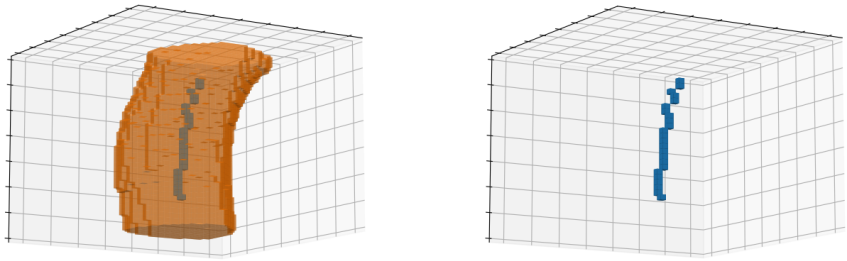


Figure 5.10: Segmentation, skeletonization and seed points of a sample correctly predicted as a non-bifurcation by the classification model. A typical case without any seed points indicated.

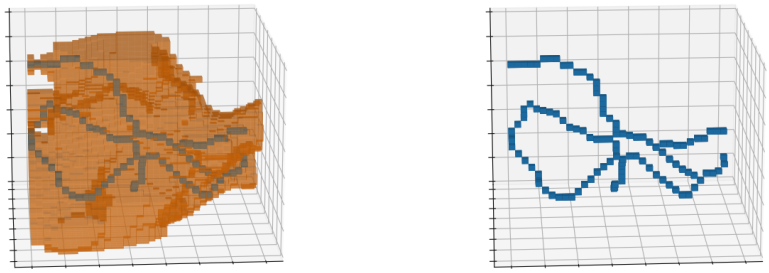


Figure 5.11: Segmentation, skeletonization and seed points of a sample correctly predicted as a non-bifurcation by the classification model. A sample from a vessel section passing through bone which give very poor segmentation with current Frangi filter settings. Indicates several possible choices of seed points. Such locations are typically not predicted as bifurcations by the classification model.

6

Discussion

In the discussion, the results from Chapter 5 are analysed in relation to the theory presented in Chapter 2 and Chapter 3, and the methods described in 4. Relevance and validity are discussed and suggestions for future research are made.

6.1 Relevance and Validity of the Algorithm's Performance

6.1.1 General Results and Limitations Imposed by Data

Results from the suggested algorithm manage to both detect and extract seed points for vascular bifurcations which resonate well with the purpose, stated in Section 1.1.1. However, it must be questioned what these results really imply. The algorithm was developed, trained, and evaluated solely on a data set produced and labelled as a part of this thesis. Hence subjective bias in the training data is likely to be present in the validation and testing data. However, the fact that the CNN generalized well to the validation data indicates that the data set was consistently designed and labelled. Moreover, the data set consisted of samples taken from only five CTAs. These five CTAs were not unusually similar in any particular way, thus, even if few, still provides a decent representation of a larger number of CTAs. Additionally, there's no particular reason to believe that performance transferring to test data taken from one of these CTAs wouldn't transfer to external CTAs. Although it clearly delimits the range of sample types available for training, consequently making some very specific types of samples impossible to learn.

Keeping the limitations imposed by the data set in mind, reaching over 90% classification accuracy even on one of the most unfavourable divisions of training, validation and testing set, in Figure 5.2, is a promising sign. Tendencies of overfitting are over the board low and generalization to validation data is decent for most data set divisions. Figure 5.6 shows that the model functions in the intended context, detecting branches along a given vessel centerline. Extracted seed points appear mostly good and the CNN clearly improves the potential of the seed point extraction by excluding difficult non-branch samples. It should be lifted

that there isn't any labelled ground truth for the bifurcation seed points. Seed points are therefore only evaluated by spot-checking samples manually and visual inspection. More testing should therefore be performed before drawing any larger statistical conclusion about the seed point quality.

6.1.2 An Abundance of Simple Samples

An observation can be made in Figure 5.2 that the loss on the validation set is quite high compared to that of the training set. According to NLL loss in Section 3.3.1 and Section 4.3.3.6 the network isn't able to easily discriminate between the two classes. However, the classification accuracy is still quite high, around 90%. A similar observation can be made in Figure 5.1 where variations in loss doesn't perfectly reflect in classification accuracy. The curve for no augmentations shows very high and increasing loss for the validation set. Still the classification for no augmentations increases and reach over 80%. This could indicate that the classification is extremely easy for many of the samples, being feasible even with very poorly optimized network with high loss. A possible explanation for this is that the data domain, by attributes, by clear majority consists of simple samples. Most branches are not especially difficult, and most non-branches contain a plain straight tube. Some precautions were taken to ensure inclusion of difficult rare samples, as explained in Section 4.2.4. The specially labelled non-bifurcation samples described in Section 4.2.4.2 were added to amend this issue. In earlier iterations of the data set, with less of these specially added samples, this issue was worse, so the precautions seem to have helped. However, being rare in a limited amount of manually labelled data, finding these samples is not feasible. The final data sets consequently contains only a few of these difficult samples. Additionally this theory could also possibly explain the flat stop of improvement at around epoch 200-300 on the validation sets seen in Figures 5.2, 5.3 and 5.4. At this point only the rare difficult samples are being miss-classified, which has none or low representation in the training set and thus cannot easily be learned. If this indeed is an explanation, more difficult samples would need to be added to the data set.

Another matter to address is the division of the data set in training, validation, and test sets. Figures 5.2, 5.3 and 5.4 can be seen as an inexhaustive cross validation. It appears that each set of CTA samples does not contain equally good representations of the data domain. Figure 5.2 show that learning CTA#1, #2, #3 (in Table 5.1) doesn't generalize very well to samples CTA#4. By contrast, learning CTA#3, #4, #5 generalize very well to samples CTA#1 as seen in Figure 5.3 and also to CTA#2 as seen in Table 5.3. This indicates either that CTA#4 has some unique types of samples or that CTA#1, #2, #3 (especially CTA#1 and #2) are under representative of the data domain; or both. Similar conclusions about other CTA#1 could be drawn by expanding the cross validation. However, the main point being made by this discussion is that data set division is important to consider and that, as in previous paragraph, more rare difficult samples from more CTAs should be included in the data set.

6.1.3 Network Predictions

Prediction accuracy of a trained network can be quite high on the test data set as can be seen in Tables 5.2 and 5.3. In Figure 5.6 the class scores for all samples along an entire vessel centerline is shown. There are cases where the separation between the classes is very high. These are the kind of samples to consider when to expanding the data set. Another point is that the samples close to branches tend to be predicted as branches. In particular the second and third branch from the right in Figure 5.6 are very close to each other and samples in between aren't predicted as non-branches. This does not necessarily pose a large issue, but the reason is likely that samples nearby a branch partially contain the branch. It makes sense that the network cannot handle these very well as it is unlikely that those kinds of samples are added to the data set. It is also in some cases subjectively difficult to judge exactly where to draw the line between a sample containing a branch and a sample not containing a branch. To remedy this issue quotas in the data set should be allocated for nearby branch samples. However, first a strict policy of how to judge if the branch is outside or inside a sample should be in place.

It's difficult to draw any conclusions from Tables 5.2 and 5.3 as they show good balance between recall and precision. No trends between training runs were discernible to favour recall over precision or vice versa. With more test data available perhaps these values would be interesting. To consider is that a classification model never will be able to guarantee 100% accuracy in class predictions. Consequently, the target algorithm would preferably feature some analysis to verify a suggested branch. In this case it would be prioritized for the branch detection method to have high recall for the branch class as it would be easier to exclude miss-classified non-branches than finding missed branches. However, this obviously depends on the exact application. In some cases, it could be more important to have high precision for the branch class, i.e., that the predicted branches very likely are branches.

6.1.4 Inconclusive Seed Point Propositions

Figures 5.8 and 5.11 both show samples were the skeletonization of the Frangi segmentation indicate multiple possible branch points. Figure 5.11 is a non-branch sample, thus the classification network is tasked to prevent this issue. Figure 5.8 however is a branch and there could be examples where the resulting skeleton is even more inconclusive. The reasons for this happening are disturbances and structures in the surface of the vessel wall. Samples with such disturbances produce and uneven surface in the segmentation which will trigger additional lines to be found in the skeletonization. Optimally a superior segmentation method or skeletonization method would be used to prevent this issue. However, a solution which comes with a trade-off is to smooth the surface. The trade-off with smoothing is that smaller vessels branching from large vessels will disappear. Consequently, this could be applicable if only bifurcations with similarly size vessels are of interest.

6.1.5 Comparison to Related Work

While there is much work done on vessel segmentation and vascular feature extraction, the quite specific scope of this thesis limits what comparisons can be performed. Like many other vascular image analysis methods proposed in [6] and [8], a combination of methods is used. In [9] branch detection is performed with a similar purpose as in this thesis (brought up in Section 2.3.3). They can perform a more rigid evaluation by visual scores from cardiology experts and branch detection in fresh patient CTAs. However, the results on display show mostly quite easy branch detection situations where the algorithm proposed in this thesis also would perform well. In bifurcation with several simultaneous branches seems to be handled better by their method though. They focus on coronal arteries, which present a smaller range of variation than focusing on carotid arteries from the aorta arch all the way up to the brain.

Augmentations, aided by Monai[51] and TorchIO[52], are effective; allowing learning and generalizing from a very small data set of only 100 unique labelled branches and in total 1300 samples. This goes to show why data augmentation is widely used in medical imaging. Random motion artefacts rather worsened (see Figure 5.1) the generality than improved it as suggested by [59]. Most likely the reason for this is that motion artefacts appearing exactly at vessels is quite rare and thus was not represented in the limited data set. Another reason could be that the motion artefacts introduced by [59] was meant for MRI and might have lower realism in CTA.

6.2 Suggestions for Future Work

As made clear in Section 2.3 the field of vascular image processing is large. There are several methods and combinations of methods which potentially could be investigated to add or replace parts the approach suggested in this thesis. However concrete suggestions closely connected to the results in this thesis will be included in this section.

Most importantly the data set used in this thesis must be expanded to include, well selected, samples from more CTAs. This would certainly improve generality of the network. Moreover, it will increase the possibility for rigid validation and evaluation. With a larger test set analysis of the ROCs (receiver operating characteristics) and PRCs (precision-recall curves) would be interesting to provide greater insight of classification properties and optimal thresholds for class scores.

On the same topic of training data, it would be interesting to include synthetic data sets. Especially development of a GAN for generating synthetic CTA data would be useful, not only for this specific project, but for vessel segmentation and medical imaging in general. However even simpler synthetic data could be used for transfer learning to focus learning on some specific vessel features before training on the real CTA data. Perhaps this could prevent learning some irrelevant features in CTA.

The suggested approach, as summarized in Section 4.6, has a lot of parameters. Even though some optimization has been done in this work, rigid optimization had to be left out in order complete an acceptable algorithm. Therefore, there is a lot of optimization that can be explored. Sample box size is very resource consuming to optimize as it requires generating several data sets. Each data set must be thoroughly probed for flaws before uses in training. However larger sample boxes would include more global information to discern vessels from vessel-like noise. A closely related task is to study how the resolution of the sample boxes affect the predictability. The architecture of the DenseNet, which is not altered much from the original, is also a relevant subject for further optimization.

Frangi filtering is a quite basic vessel segmentation technique with observed flaws like classifying objects connected to vessels like vessels.[6] A more advance method for vessel segmentation could reduce probability of cases like Figure 5.8. With access to labelled segmented vessel CTAs an additional CNN could be trained for local segmentation. A less extensive change would be to use an adaptive scheme for selecting Frangi filter parameters α, β and γ. The suggested fixed parameter values compromise for being able to detect small vessels but exclude nearby noise particles. During trials, manually selected filter parameters were able to handle cases like the sample in Figure 5.8 perfectly.

A possible expansion of the use area would be to train the network to detect vascular bifurcations in a sliding window sample all over the CTA, not limited to a vessel centerline. This would allow independently providing initialization seed points for vessel segmentation methods which require seed points, including Mentice's algorithm.

7

Conclusion

The aim of this work was to develop and propose a method for vascular bifurcation detection in cerebral CTAs including the carotid arteries. Previous studies indicate that feature extraction such as bifurcation detection often is preceded by vessel segmentation. It was also indicated that CNN show great potential for vessel segmentation and often are combined with other methods. In addition to detecting bifurcations the aim was to provide seed points in the branches for a seed point based vessel segmentation method. Consequently, the proposed method was chosen to exploit information available from the seed point based vessel segmentation method and use a CNN for bifurcation detecting combined with Frangi filters and 3D morphological skeletonization for seed point extraction. The branch detection was posed as a classification problem, thus the high performing DenseNet architecture was chosen as the employed CNN. Training data for the DenseNet was manually extracted and labelled from five CTAs making up a quite small data set of about 1300 samples. Online data augmentation on the GPU was applied to enable generalizing from such a tiny data set. The chosen augmentation transform was enough to nearly make overfitting a none factor. Assisted by the classification network, excluding most difficult samples, Frangi filters was able to locally segment branches which could be skeletonized to present seed points.

Results are promising with classification network provided a high classification accuracy of 90-100% with a similar f1-score. Extracted seed points were also good in most cases. However, no rigid validation was available due to low amount of data. There were also discrepancies of representability between the five CTAs where one or two contained unique sample types. Thus, the division in training and validation had a large effect which was seen in a inexhaustive cross-validation. Future studies could work to expand and improve the data set, particularly improve the balance between difficult and easy samples. With more data the performance can be validated and improved. There are also more sophisticated methods than Frangi filters and 3D skeletonization for seed point extraction which could be adopted. Possibly the CNN could be trained to find branches from scratch over an entire CTA which could be used to provide initial seed points for a seed point based segmentation method.

Bibliography

[1] R. Aggarwal et al. "Virtual Reality Simulation Training can Improve Inexperienced Surgeons' Endovascular Skills". In: *European Journal of Vascular and Endovascular Surgery* 31.6 (2006), pp. 588–593. ISSN: 1078-5884. DOI: https://doi.org/10.1016/j.ejvs.2005.11.009. URL: http://www.sciencedirect.com/science/article/pii/S1078588405007069.

[2] Kent R Van Sickle, E Matt Ritter, and C Daniel Smith. "The pretrained novice: using simulation-based training to improve learning in the operating room". In: *Surgical innovation* 13.3 (2006), pp. 198–204.

[3] Max Berry et al. "Endovascular training with animals versus virtual reality systems: an economic analysis". In: *Journal of Vascular and Interventional Radiology* 19.2 (2008), pp. 233–238.

[4] Rodney A. White et al. "Endovascular interventions training and credentialing for vascular surgeons". In: *Journal of Vascular Surgery* 29.1 (1999), pp. 177–186. ISSN: 0741-5214. DOI: https://doi.org/10.1016/S0741-5214(99)70359-9. URL: http://www.sciencedirect.com/science/article/pii/S0741521499703599.

[5] Peter Schneider. *Endovascular skills: guidewire and catheter skills for endovascular surgery*. CRC press, 2019.

[6] Fengjun Zhao et al. "Segmentation of blood vessels using rule-based and machine-learning-based methods: a review". In: *Multimedia Systems* 25.2 (2019), pp. 109–118.

[7] Kanako K Kumamaru et al. "CT angiography: current technology and clinical use". In: *Radiologic Clinics* 48.2 (2010), pp. 213–235.

[8] David Lesage et al. "A review of 3D vessel lumen segmentation techniques: Models, features and extraction schemes". In: *Medical Image Analysis* 13.6 (2009). Includes Special Section on Computational Biomechanics for Medicine, pp. 819–845. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2009.07.011. URL: http://www.sciencedirect.com/science/article/pii/S136184150900067X.

[9] Suheyla Cetin et al. "Vessel tractography using an intensity based tensor model with branch detection". In: *IEEE transactions on medical imaging* 32.2 (2012), pp. 348–363.

[10] Michael S Vaphiades and Joseph A Horton. “MRA or CTA, that’s the question”. In: *Survey of ophthalmology* 50.4 (2005), pp. 406–410.

[11] Tami D DenOtter and Johanna Schubert. “Hounsfield Unit”. In: *StatPearls [Internet]*. StatPearls Publishing, 2019.

[12] Gabor T Herman. *Fundamentals of computerized tomography: image reconstruction from projections*. Springer Science & Business Media, 2009.

[13] G. N. Hounsfield. “Computerized transverse axial scanning (tomography): Part 1. Description of system”. In: *The British Journal of Radiology* 46.552 (1973). PMID: 4757352, pp. 1016–1022. DOI: 10.1259/0007-1285-46-552-1016. eprint: https://doi.org/10.1259/0007-1285-46-552-1016. URL: https://doi.org/10.1259/0007-1285-46-552-1016.

[14] Sanjana Patrick et al. “Comparison of gray values of cone-beam computed tomography with hounsfield units of multislice computed tomography: An in vitro study”. In: *Indian Journal of Dental Research* 28.1 (2017), p. 66.

[15] Peter Mildenberger, Marco Eichelberg, and Eric Martin. “Introduction to the DICOM standard”. In: *European radiology* 12.4 (2002), pp. 920–927.

[16] Keith L Moore and Arthur F Dalley. *Clinically oriented anatomy*. Wolters kluwer india Pvt Ltd, 2018.

[17] HR Williams et al. “Minimum mass vascular networks in multifunctional materials”. In: *Journal of the Royal Society Interface* 5.18 (2008), pp. 55–65.

[18] Giles Tetteh et al. “Deepvesselnet: Vessel segmentation, centerline prediction, and bifurcation detection in 3-d angiographic volumes”. In: *Frontiers in Neuroscience* 14 (2020).

[19] Alejandro F Frangi et al. “Multiscale vessel enhancement filtering”. In: *International conference on medical image computing and computer-assisted intervention*. Springer. 1998, pp. 130–137.

[20] Joes Staal et al. “Ridge-based vessel segmentation in color images of the retina”. In: *IEEE transactions on medical imaging* 23.4 (2004), pp. 501–509.

[21] Serge Beucher et al. “The watershed transformation applied to image segmentation”. In: *Scanning microscopy-supplement-* (1992), pp. 299–299.

[22] Lu Wang et al. “Interactive retinal vessel extraction by integrating vessel tracing and graph search”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2013, pp. 567–574.

[23] Tony F Chan and Luminita A Vese. “Active contours without edges”. In: *IEEE Transactions on image processing* 10.2 (2001), pp. 266–277.

[24] Tim McInerney and Demetri Terzopoulos. “T-snakes: Topology adaptive snakes”. In: *Medical image analysis* 4.2 (2000), pp. 73–91.

[25] Yuanzhi Cheng et al. “Accurate vessel segmentation with constrained B-snake”. In: *IEEE Transactions on Image Processing* 24.8 (2015), pp. 2440–2455.

[26] Yuri Boykov and Gareth Funka-Lea. “Graph cuts and efficient ND image segmentation”. In: *International journal of computer vision* 70.2 (2006), pp. 109–131.

[27] Christian Bauer et al. “Segmentation of interwoven 3d tubular tree structures utilizing shape priors and graph cuts”. In: *Medical image analysis* 14.2 (2010), pp. 172–184.

[28] Caroline Lacoste, Gérard Finet, and Isabelle E Magnin. “Coronary tree extraction from X-ray angiograms using marked point processes”. In: *3rd IEEE International Symposium on Biomedical Imaging: Nano to Macro, 2006.* IEEE. 2006, pp. 157–160.

[29] Leo Grady. “Random walks for image segmentation”. In: *IEEE transactions on pattern analysis and machine intelligence* 28.11 (2006), pp. 1768–1783.

[30] S. R. Aylward and E. Bullitt. “Initialization, noise, singularities, and scale in height ridge traversal for tubular object centerline extraction”. In: *IEEE Transactions on Medical Imaging* 21.2 (2002), pp. 61–75.

[31] Alexander Broersen et al. “FrenchCoast: fast, robust extraction for the nice challenge on coronary artery segmentation of the tree”. In: *Proc. of MICCAI Workshop” 3D Cardiovascular Imaging: a MICCAI segmentation Challenge.* 2012.

[32] Peter J Basser et al. “Fiber tract following in the human brain using DT-MRI data”. In: *IEICE TRANSACTIONS on Information and Systems* 85.1 (2002), pp. 15–21.

[33] Ingela Nystroem. “Skeletonization applied to magnetic resonance angiography images”. In: *Medical Imaging 1998: Image Processing.* Vol. 3338. International Society for Optics and Photonics. 1998, pp. 693–701.

[34] Ta-Chih Lee, Rangasami L Kashyap, and Chong-Nam Chu. “Building skeleton models via 3-D medial surface axis thinning algorithms”. In: *CVGIP: Graphical Models and Image Processing* 56.6 (1994), pp. 462–478.

[35] Edward F Moore et al. “Gedanken-experiments on sequential machines”. In: *Automata studies* 34 (1956), pp. 129–153.

[36] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[37] Kunihiko Fukushima and Sei Miyake. “Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition”. In: *Competition and cooperation in neural nets.* Springer, 1982, pp. 267–285.

[38] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition.* Ieee. 2009, pp. 248–255.

[39] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[40] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[41] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[42] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[43] Gao Huang et al. “Densely connected convolutional networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[44] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[45] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.

[46] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[47] Hoo-Chang Shin et al. “Medical image synthesis for data augmentation and anonymization using generative adversarial networks”. In: *International workshop on simulation and synthesis in medical imaging*. Springer. 2018, pp. 1–11.

[48] James Shackleford, Nagarajan Kandasamy, and Gregory Sharp. “Chapter 3 - Multimodal B-Spline Registration”. In: *High Performance Deformable Image Registration Algorithms for Manycore Processors*. Ed. by James Shackleford, Nagarajan Kandasamy, and Gregory Sharp. Boston: Morgan Kaufmann, 2013, pp. 45–74. ISBN: 978-0-12-407741-6. DOI: https://doi.org/10.1016/B978-0-12-407741-6.00003-7. URL: http://www.sciencedirect.com/science/article/pii/B9780124077416000037.

[49] Cem Yuksel, Scott Schaefer, and John Keyser. “On the parameterization of Catmull-Rom curves”. In: *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*. 2009, pp. 47–53.

[50] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019, pp. 8026–8037. URL: https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

[51] The MONAI Consortium. *Project MONAI*. Dec. 2020. DOI: 10.5281/zenodo.4323059. URL: https://doi.org/10.5281/zenodo.4323059.

[52] Fernando Pérez-García, Rachel Sparks, and Sebastien Ourselin. *TorchIO: a Python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning*. 2020. arXiv: 2003.04696 [eess.IV].

[53] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[54] Tomoki Uemura et al. "Comparative performance of 3D-DenseNet, 3D-ResNet, and 3D-VGG models in polyp detection for CT colonography". In: *Medical Imaging 2020: Computer-Aided Diagnosis*. Ed. by Horst K. Hahn and Maciej A. Mazurowski. Vol. 11314. International Society for Optics and Photonics. SPIE, 2020, pp. 736–741. DOI: 10.1117/12.2549103. URL: https://doi.org/10.1117/12.2549103.

[55] Toan Duc Bui, Jitae Shin, and Taesup Moon. "3d densely convolutional networks for volumetric segmentation". In: *arXiv preprint arXiv:1709.03199* (2017).

[56] The MONAI Consortium. *Project MONAI*. 2020. URL: https://github.com/Project-MONAI/MONAI/blob/master/monai/networks/nets/densenet.py (visited on 12/14/2020).

[57] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[58] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[59] Richard Shaw et al. "MRI k-Space Motion Artefact Augmentation: Model Robustness and Task-Specific Uncertainty". In: *Proceedings of The 2nd International Conference on Medical Imaging with Deep Learning*. Ed. by M. Jorge Cardoso et al. Vol. 102. Proceedings of Machine Learning Research. London, United Kingdom: PMLR, Oct. 2019, pp. 427–436. URL: http://proceedings.mlr.press/v102/shaw19a.html.

[60] Alejandro F Frangi et al. "Model-based quantitation of 3-D magnetic resonance angiographic images". In: *IEEE Transactions on medical imaging* 18.10 (1999), pp. 946–956.

[61] Stefan Van der Walt et al. "scikit-image: image processing in Python". In: *PeerJ* 2 (2014), e453.

[62] Ekaba Bisong. "Google Colaboratory". In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*. Berkeley, CA: Apress, 2019, pp. 59–64. ISBN: 978-1-4842-4470-8. DOI: 10.1007/978-1-4842-4470-8_7. URL: https://doi.org/10.1007/978-1-4842-4470-8_7.