Segmentation of Human Blood Vessels in Magnetic Resonance Angiography using Image Projections and Neural Networks

Rüdiger von Hackewitz
MSCN

School of Agricultural, Computational and
Environmental Sciences
University of Southern Queensland

June 2019

Submitted in partial fulfillment of
the requirements of the award of
Master of Science

Abstract

Magentic Resonance Angiography (MRA) is a medical imaging procedure that helps visualise blood cells in the human body. MRA scans are of importance to medical practitioners, when preparing for surgeries or to detect blood vessel malformations in the human body.

A full three-dimensional (3D) MRA scan is a set of stacked two-dimensional (2D) MRA images. The automated segmentation of blood vessel structures in a 3D MRA scan is an active field of research in the discipline of digital image processing.

In this thesis, a new approach for the segmentation of blood vessels in cerebral MRA scans is introduced, using image projections combined with neural networks. The imaging process decomposes the 3D source image into a set of 2D image projections, using the method of Maximum Intensity Projection (MIP). Subsequently, neural networks are trained to segment the set of 2D image projections into blood vessels and background. Derived from the 2D segmentation results, a skeleton of cerebral blood vessel structures is constructed in 3D space. Image post processing steps with dilution are applied to produce the final 3D blood vessel segmentation map.

The proposed method is computationally faster than many segmentation approaches that process the images directly in 3D space. The final model has been tested against 10 cerebral MRA source scans with ground truth data. An F1-Score of 0.93 can be achieved for records that are used during training or validation.

The scope of this work is limited to segmentation of blood vessels in cerebral MRA scans. However, the method offers a framework for dimensionality reduction that can be extended to 3D image segmentation processes in unrelated fields.

The discussed approach is only feasible for the automated segmentation of objects with very low density in 3D space.

Certification of Thesis

The work presented in this thesis is, to the best of my knowledge and belief, original work of *Rüdiger von Hackewitz*, except as acknowledged in the text. The material in this thesis has not been submitted, either in whole or in part, for a degree at this or any other university.

Rüdiger von Hackewitz Date

ENDORSEMENTS

Professor Yan Li Date

Dr Bo Song Date

Acknowledgements

Special thanks go to Yan Li for offering me the opportunity to undertake research on this body of work. When I contacted her in July 2018 and asked for research opportunities that involve image processing tasks, she kindly suggested that I undertake research on cerebral MRA scan images.

Bo Song helped me in the initial stages, provided access to the MRA source records and advised me on how to use software *ImageJ* for browsing and initial inspection of the 3D MRA scans. I am grateful to Bo for his patience and the initial zoom sessions to discuss the work and progress. Bo organised and completed the tedious task of segmenting manually two cerebral MRA source scans. Without the two manually segmented MRA scans, I would not have been in a position to draw the firm conclusions that are presented in this thesis. Bo, thanks for your input, repeated review of initial drafts and advice throughout this project!

Thanks go also to Geoff Breemer, who kindly provided me access to his thesis about the use of deep learning for blood vessel segmentation in cerebral MRA scans. His work provided valuable insights about the configuration and use of U-Nets.

Thanks to Xiaohui Tao for providing advice about the University's content and formatting requirements in dissertations, and sharing a LaTex University template, that has been used to format this thesis.

Thanks to my dear daughter Astrid, who helped with the proof-reading of the final manuscript.

Finally, I wish to appreciate my wife Edina for her patience and understanding while I was working fulltime on this research for over 6 months. You are special!

And last, but not least, thanks to Jesus Christ, my Lord and Saviour! Without him life would be meaningless.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Magnetic Resonance Angiography (short: MRA) uses magnetic waves to visualise the location of blood vessels in the human body. This medical imaging procedure was first developed in the 1970s and is useful for the analysis of blood flow in different body parts, such as the brain, heart, abdomen, liver and kidneys.

Figure 1.1: MRI / MRA Scanner (Source: CRA-Medical-Imaging (2018))

MRA and MRI (Magnetic Resonance Imaging) are closely related technologies, and both procedures can be undertaken with the same machine.

Today, MRI and MRA scans are often preferred over the use of X-rays, as they do not expose the body to harmful radiation.

1.1 Research Problems

MRA scans are a vital medical imaging procedure; they help locate a patient's major arteries and blood vessels in the body; MRA scans are also used for the treatment of vascular diseases, or to identify blood structure malformations such as emboli and stenosis, Barlinn and Alexandrov (2011).

MRA scans are a stack of 2D images, which are composed to produce a 3D image version; specialised software is required to view and analyse the images rendered by a Magnetic Resonance Imaging (MRI) device.

It can be challenging to visualise and manually segment blood vessel structures in a 3D MRA image. Radiologists benefit from software that can automatically separate a 3D image into two parts: blood vessel structures and background.

The focus of this research thesis is to investigate the segmentation of blood vessels in the human brain (cerebral MRA scans). The main arteries in the human brain were first described by Thomas Willis in the 17th Century. Named after him, the Circle of Willis (CW), Z.Vrselja (2014) is the major system of circular cerebral blood vessels that supplies the human brain constantly with sufficient blood and oxygen.

Identification and location of arteries in the CW are a critically important task, e.g. prior to brain surgeries, American-Cancer-Society (2019). It can be a challenging exercise for a radiologist to identify and manually label all the arteries of the CW in a cerebral MRA scan. Computer-aided segmentation of blood vessel regions in MRA images can assist with the analysis of cerebral MRA scans.

The automated segmentation of cerebral blood vessels must overcome several challenges: noise in the images needs to be filtered out; objects with non-vascular shape need to be excluded, and Computer hardware may reach its limits when processing high-resolution images in 3D.

1.2 Research Objectives

A new method - using image projections and neural networks - is proposed for the segmentation of cerebral blood vessels in MRA scans. A Computer program is designed, implemented and tested based on this innovative algorithm.

The new software should assist medical staff with identifying the structural elements of the CW in a patient.

The Computer software should be ready for deployment into a production system, so that the solution can be piloted. The final version of the product should create automatically - without any human intervention - the segmentation of cerebral blood vessels in MRA scans.

The solution is not intended as a replacement, but as a supplement for the services of an experienced radiologist. For example, the product may assist with the discovery of cerebrovascular diseases, AANS (2019).

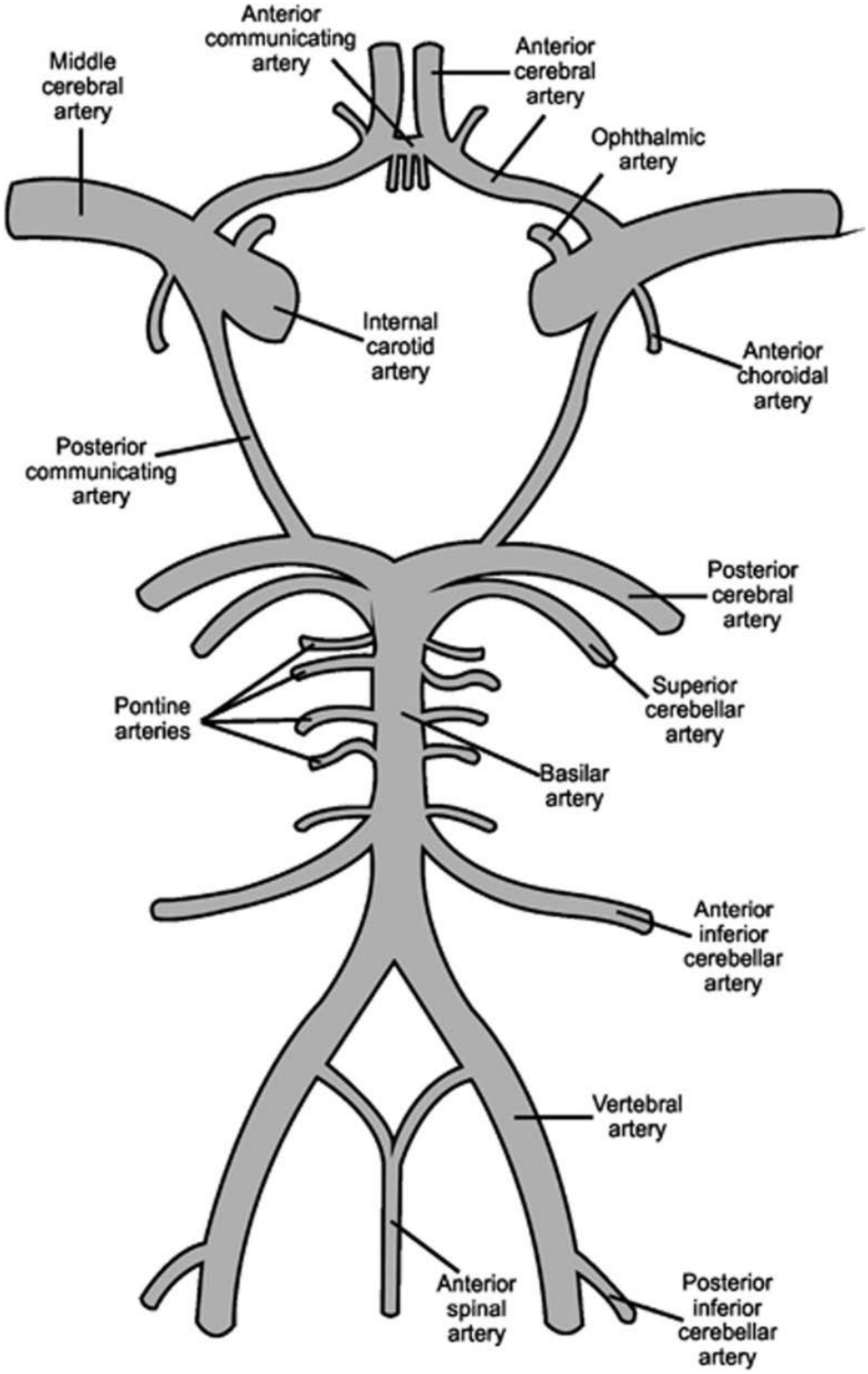


Figure 1.2: Circle of Willis,(Source: Z.Vrselja (2014))

The final outcome of this research should meet the following criteria:

- Results of the automated segmentation process should produce accurate

and reproducible segmentation maps against the ground truth data that have been provided for this project.

- The new product should complete the segmentation of cerebral blood vessels in a computationally fast and efficient way.

- The software should be available in a mainstream programming language and offer API connectors that are ready for deployment into the technical environment of a radiologist.

- The software needs to be configurable, so that it is able work on different MRI sanner models with different technical specifications (e.g. varying 3D image dimensions and resolutions)

1.3 Research Scope

The scope of this research is restricted to the analysis of MRA images of the brain (cerebral MRA scans). Blood vessel segmentation for MRA scans of other human body parts, such as the lungs or the abdomen, are not investigated in this research.

All available cerebral MRA scans are produced by the same MRI scanner version, a Siemens Sonata (Siemens (2018)). As such, results in this thesis cannot be transferred directly to MRA outputs that have been delivered by other machines. Adjustments to relevant hyper parameters and configuration settings are required (and supported by the software) so that the process can work on cerebral MRA scans of different devices.

Identifying specific blood vessel regions in the brain, such as components in the Circle of Willis, is not in scope of this project.

Results are evaluated using the data from 10 healthy patients; it is not tested if the product can identify and map cerebral blood vessel anomalies.

1.4 Data

To conduct this research, cerebral MRA scans from ten healthy individuals are analysed. All MRA source images are created by the Siemens MRI scanner, and all images have the same spatial resolutions.

In addition, ground truth records for each of the ten patient records are provided by the University of Southern Queensland. Ground truth data are the blood vessel segmentation maps that should be closely matched by the final product.

The F. Frangi et al. (2000) Filter in Matlab 2015b is used for the semi-manual segmentation of eight MRA datasets. A combination of 3D software packages is utilised for the complete manual segmentation of blood vessel segments in two of the ten available MRA scans.

The provided ground truth images are used for the training of neural networks, optimal configuration of hyper parameters in the segmentation process, and the final quality assessment of the newly developed segmentation method.

1.5 What to Expect in this Thesis

Rather than presenting the outcomes of this research and interpreting the results from the perspective of the finished product, the reader is taken on a journey from start to finish of this project. When reading through this thesis, the challenges along the way are discussed, and it is explained why some of the design decisions have been made, as pertaining to system architecture and tuning of image processing steps. In particular:

- Chapter 2 provides some background information about the beginnings of this project, and about the driving factors for segmenting MRA scans in 2D and not in 3D.
- A summary at the end of Chapters 3 to 9 provides key findings and results as they relate to the research phases. If short of time, the reader may start by reading the summaries at the end of each Chapter.
- In the final Chapter 10, strengths and weaknesses of the method are discussed. Additionally, an outlook is provided that outlines opportunities how the proposed method can be utilised and further improved.
- Several visual artefacts are provided, including 2D and 3D segmentation maps with visual overlays of predictions against ground truth data. Those images are moved to the Appendix B to avoid disruptions to the reader, whilst studying the main body of this thesis. Relevant image sections in the Appendix B are repeatedly referenced in this document to further understanding for some of the results and performance figures in the main body of this document.
- Consistent colour codes are applied throughout this thesis: ground truth data are represented in green, predictions for segmentation maps in red, and overlaps between ground truth and predictions in yellow.

Chapter 2

Background Information

2.1 Initial Research

The initial research was started with the objective to train a neural network that can automatically detect and classify blood vessel cells in an MRA scan of the human brain.

Review of literature about this topic reveals that signficant advances are being made in deep learning, and many neural network architectures are available that excel in the blood vessel segmentation of medical images.

Most existing algorithms for the segmentation of blood vessels are designed and tested in 2D space, but they perform poorly on commodity Computer hardware against 3D images.

In this thesis, the cerebral MRA scan of a patient is provided by an MRI device that produces a stack of 247 individual 2D images. Each 2D image version is quadratic with a side length of 320 pixels. As a result, each MRA source scan consists of over 25 Million (247x320x320) data points.

When some of the reputable machine-learning algorithms are applied against the available MRA data sets - each with over 25 Million data points - they can neither complete nor produce reasonable segmentation results in acceptable timeframes on commodity hardware.

As an alternative, the computing power of Cloud Services Providers such as Amazon Web Services (AWS), could be used to move training processes away from commodity Computer hardware (Apple MacBook Air, 8GB RAM). This approach bears risks: it requires some monetary investment, and there is no guarantee that the final trained neural network can produce useful results.

However, plenty of neural network architectures are available for image processing tasks in 2D, many of which perform well on commodity hardware. Some of those architectures are readily available on GitHub or are implemented as libraries of a package (Tensorflow, sci-kit).

The structure of blood vessels in the brain can be well represented by projecting the over 25 Million data points into 2D space. Less than 1 per cent of the 3D scan contains blood vessel cells, and vascular structures are still clearly visible when looking 'through' the 3D source scan from various angles.

Consequently, the idea is borne to decompose the 3D MRA scan into a set of 2D image projections, undertake the neural network segmentation process across those image projections, and then reconstruct a 3D-segmented image version of the source scan, based on the segmentation results of the 2D image projections.

2.2 Methodology

2.2.1 Dimensionality Reduction

Eliminating one of the three dimensions in the MRA scan is a dimensionality reduction method and a common technique in the statistical world. It simplifies data structures and reduces data volumes by sacrificing some of the information that is available in the original data set.

The question is: how much of source data information is lost by reducing its dimensionality? The proof-of-concept in Chapter 6 confirms that the initial 3D ground truth can be nearly perfectly reconstructed via the 2D segmented image projections.

Less than 1 per cent of an entire 3D MRA scan contains blood cell regions and, as a consequence, relatively small areas of a 2D image projections contain overlapping blood vessel streams.

Reducing the dimensionality of the MRA scan from 3D to 2D has two major advantages for the envisaged usage of neural networks:

- It is easier to visualise image segmentation results and overlays across a set of 2D image projections rather than their related 3D source scan.
- The training and processing of a neural network across the 2D image versions is the order of a magnitude faster than across their 3D counterpart.

Finally, it should be noted that the class imbalance between background and blood vessel elements is reduced in the 2D image projections. This proves to be useful for the subsequent machine learning processes across the 2D image projections.

2.2.2 Project Phases

Before starting with the training of neural networks across the decomposed 2D image versions, a feasibility study has to be undertaken. A proof-of-concept is developed that verifies that high-quality 3D segmentations of the original MRA scan can be reconstructed, once accurate segmentations of the 2D projections are available.

The proof-of-concept skips the training phase of neural networks for the 2D image projections. It works with the perfect segmentation map of the ground truth image, which has been made available at the outset this project. A

prototype is developed which helps to quantify the amount of the information that is lost by using this dimensionality reduction method.

Secondly, once the usefulness of this approach has been confirmed, a neural network is trained to optimise the prediction of blood vessel segments for each image projection.

In a third step, the entire process is put together: images are decomposed into 2D image projections, neural networks predict segmentation maps for each projection, and the 2D segmentation maps are then used to reconstruct a 3D segmentation map. Hyper parameters of the model are tuned and final post-processing tasks are applied to deliver an optimised blood vessel segmentation map against the ground truth data.

Finally, results of the segmentation processes are evaluated against unseen data, discussed, interpreted and recommendations are made how this process can be further improved.

On a high level, the project can be separated into four different phases:

- **Phase I**: Development and testing of protoype for decomposition of 3D MRA scan into 2D image projections with reconstruction of the 3D MRA segmentation map, based on 2D segmentation
- **Phase II**: Development of a neural network to create segmentation maps for blood vessel regions in each 2D image projection
- **Phase III**: Development and test of the end-to-end process for image decomposition, run of neural network and reconstruction of the 3D image segmentation, including the tuning of hyper parameters and post-processing tasks
- **Phase IV**: Testing of results against unseen data, interpretation of results and outlook into further research activities

2.3 MRA Data Scans

For this project, ten different cerebral MRA patient scans are dowloaded from the internet as provided by R. Kötter and Mazoyer (2001) .

Each 3D image scan is paired with a 3D blood vessel segmentation map (commonly called ground truth in the literature). Source and ground truth images have the same dimensions.

Eight of the ten ground truth images are created in a semi-automatic fashion by Song and Li (2016), using the Frangi filter in Matlab 2015b.

The remaining two 3D ground truth images are segmented manually by Bo Song (University of Southern Queensland), using software for the processing of spatial data, such as 3D slicer, Simpleware ScanIP and ImageJ.

The two manually segmented MRA scans have the highest quality and come closest to the result that should be achieved by the newly developed

algorithm. The eight segmentation maps created semi-automatically by the F. Frangi et al. (2000) method are of lesser quality.

With the limited number of available data sets, the following approach is selected to make optimal use of all data throughout this project:

- One of the two manually segmented MRA scans is set aside and not used at all throughout design, training, testing and parameter tuning of the segmentation method. It is only used at the end of the project to test the method against a formerly unseen record.

- The second manually segmented MRA scan is used as benchmark and validation record when training the neural networks, testing various architectures and tuning the hyper parameters.

- The remaining eight MRA scans - segmented semi-automatically - are of lesser quality and are mainly used during training of the neural network. They assist in obtaining a balanced average score for performance metrics across all nine records during training and testing.

Throughout this document, individual MRA scans are occasionally referred to by their ID. Table 2.1 lists all ten records, with segmentation method and purpose in this project:

MRA Source ID	*Segmentation Method*	*Purpose*
MNI 0590	Frangi	Training
MNI 0591	Frangi	Training
MNI 0592	Frangi	Training
MNI 0640	Frangi	Training
MNI 0643	Frangi	Training
MNI 0648	Frangi	Training
MNI 0656	Manually	Validation and Model Tuning
MNI 0657	Frangi	Training
MNI 0663	Frangi	Training
MNI 0664	Manually	Testing of the final model. *This record is not used during training and validation*

Table 2.1: MRA Source Records with Segmentation Method and Purpose

Throughout Project Phases I to III, focus is on the manually segmented MRA scan with the id MNI-0656. Segmentation maps, produced by the program, are compared repeatedly against the manually segmented ground truth image.

The manually segmented MRA scan MNI-0664 is only used once, during Phase IV. Its sole purpose is to measure and compare performance of the final

model against a formerly unseen record. It helps to understand how well the model generalises and if it avoids overfitting against the nine MRA records that are used for training and validation of the process.

2.4 Programming Language and Integrated Development Environment

All code developed for this thesis is built in Python 3 using Jupyter Notebook as Integrated Development Environment (IDE). It is developed and tested on a Mac OS Mojave 10.14.2, but should be able to operate on any other operating system that supports Python 3 (e.g. Linux, Window, AIX).

Python libraries are used to produce the graphs and 3D image versions in this thesis. All code and the ten MRA source data scans with ground truth data are stored in GitHub at `https://github.com/rvh1/mra_blood_vessel_segmentation/tree/master/data`. The code for this thesis is available to the public free of charge at `https://github.com/rvh1/mra_blood_vessel_segmentation`, with some restrictions as imposed by the included license agreement at README.md . All results and graphs in this thesis (with the exception of the training of the neural networks) are reproducible by executing the relevant code sections in the Jupyter Notebook files.

All code is developed, tested and executed on mid-level commodity hardware (MacBook Air 10.14.1, 1.7 GHz Intel Core i7, 8 GB 1600 MHz DDR3), without utilising the processing power of the GPU.

Chapter 3

Literature Review

The literature review covers elements as they relate to and are relevant for this thesis.

3.1 Hessian Matrix (Frangi)

The F. Frangi et al. (2000) method analyses the morphology of an image by computing a Hessian matrix with its eigenvalues and eigenvectors.

The Hessian matrix is a mathematical construct and captures the second derivatives of a data point in multidimensional space. This matrix can be used to identify local maxima, minima or shapes of objects in multidimensional space.

In their paper, the authors applied their algorithm to cerebral MRA data (two- and three-dimensional). Based on the eigenvalues in the Hessian matrix, the process decides whether a pixel (or voxel) is part of a tube-shaped region and should be classified as blood vessel or background. Tube-like structures will have a certain pattern of eigenvalues, and this information is used to filter out blood vessel regions in the MRA scans.

Eight out of the ten available MRA scans in this project have been semi-automatically segmented, using a Matlab implementation of the Frangi method. The method is an unsupervised segmentation process and does not rely on the availability of ground truth data.

3.2 U-Nets (Ronneberger)

Ronneberger et al. (2015) introduced ground-breaking work for the segmentation of images using an artificial neural network called U-Net. The algorithm has been developed specifically for the segmentation of medical images and the left part of the 'U' implements a convolutional neural network (CNN) that is typically used for object identification. At the bottom of the 'U', the right hand side starts to extract and build up the features and dimensions of the

originating source image with a corresponding binary segmentation map. The concept of this architecture is depicted in Figure 7.1.

U-Nets can produce segmentation maps of reasonable quality, if only a few ground truth images are available. As this project is faced with the challenge of a limited number of ground truth data, the U-Net architecture has been selected and is trained on the 2D image projections of the cerebral MRA scans.

3.3 Maximum Intensity Projection (MIP)

Maximum Intensity Projection (MIP) is discussed by Shifeng Zhao (2015) in the context of blood vessel segmentation of cerebral blood vessels in MRA scans.

Similar to the method proposed in this thesis, data are projected into the 2D plane. Rotating angles between 0 and 180 degrees are used to create image projections from different views.

Statistical information in the 2D projections is used to segment the data, and the results are projected back into the 3D image version.

In their paper, Shifeng Zhao (2015) recorded an average recall of 0.931 across nine MRA scans. Similar performance metrics are achieved by the segmentation method, proposed in this thesis (Table 9.2).

Shifeng Zhao (2015) recorded runtimes between 30 - 60 sec for the segmentation of a single MRA scan, by working with Matlab 7 on an Intel Core 2.4 GHz PC. The proposed method in this thesis could process the data slightly faster, in under 30 sec on similar hardware (Table 9.3).

3.4 Retinal Blood Vessel Segmentation (STARE and DRIVE Databases)

Many different algorithms have been proposed to segment retinal blood vessels. A beauty of the research to date is that most segmentation procedures are developed and benchmarked against the publicly available data sets in the STARE Clemson (2000) and/or DRIVE Utrecht (2007) databases. Unfortunately, to date, no similar database with ground truth data is available for cerebral MRA images.

Fraz et al. (2012) provided a useful overview about the various segmentation processes for retinal blood vessels. The paper distinguished between:

- Supervised classification methods, such as Artificial Neural Networks (ANN), Support Vector Machines (SVM) and K-Nearest-Neighbour (KNN) methods,

- Unsupervised classification methods, such as K-Means, Fuzzy C-Means (FCM), DBSCAN M.Ester (1996); and

- Morphological processing, such as watershed algorithm V.Grau (2004) and vessel tracing methods G.Hassan (2015)

Supervised methods work with a set of pre-segmented images (the ground truth); those images are used to train and optimise the blood vessel segmentation process against a baseline. The accuracy of the model is then tested against a second test data set that had not been seen previously by the training process. The outcomes of the trained algorithm against the test images are then compared against the ground truth images to evaluate the performance of the program.

Unsupervised segmentation methods work differently. They do not rely on pre- segmented images, but use structural information in the source image to identify the blood vessel flow. Such processes often use clustering algorithms such as DBSCAN M.Ester (1996), FCM J.Bezdek (1984) or morphological features such as Watershed V.Grau (2004) or F. Frangi et al. (2000) to segment an image.

According to Fraz et al. (2012), supervised segmentation processes produced overall better results than their unsupervised counterparts. This comes as no surprise as supervised methods have the luxury of using labelled images (ground truth) and can be trained, whereas unsupervised methods base their algorithms on the intrinsics of the data in the source image itself, with no or only minor input from pre-segmented images.

The images in the DRIVE database with retinal blood vessel images are coloured. However, most of the papers, investigated in the literature review, considered just the green channel when segmenting the image. Therefore, there should be no issue transporting the solution to the single-colour coded (grey) MRA images.

3.4.1 Line Detectors

Line detectors have the ability to trace lines in images, which can then be used to segment the path of blood vessels in an image.

The basic line detection algorithm uses a single line of fixed length, applies it in various directions of a pixel and chooses the direction of the line with relatively constant pixel values. There are some issues when using this algorithm for the tracing of blood vessels. It may not correctly classify pixels in close proximity to two different blood vessel lines, crossing blood vessels or blood vessels with very strong bends. All those scenarios are common features in images that depict blood vessels in 2D. Nguyen et al. (2013) introduced an improvement of the basic line classifier to overcome those problems in the classification of blood vessels. It is suggested to use two lines with differing lengths when tracing the blood vessel directions for pixels. The approach produced improved results in regions where blood vessels cross each other or are in close proximity.

3.4.2 DBSCAN with Neural Network

Density based spatial clustering of applications with noise (DBSCAN) was first introduced by M.Ester (1996). It is a well- established clustering algorithm and is very useful to identify and filter out regions with high pixel values, but low density. DBSCAN attempts to identify pixels in close proximity (high density) and groups them together.

Giraddi and Pujari (2014) used this algorithm for the segmentation of retinal blood vessels. They undertook an initial clustering of the medical image using DBSCAN. After completion of the pre-clustering process, the authors determined a set of features for candidate regions that maximise the accuracy for the separation of pixels against blood vessels and background. A total of seven features have been used for the classification of regions, including its maximum intensity (red/green), area size and eccentricity.

A neural network was then trained against those features to complete the segmentation of the areas into blood vessels and background. Different Neural Network (NN) architectures have been tested, and excellent results could be achieved with a single hidden layer. It was not surprising that such a simple NN architecture was sufficient for this process. The network expects only a seven- dimensional vector as input parameter and produces a boolean output.

3.4.3 Morlet Wavelet and Bayesian Classifier

Ghaderi et al. (2007) proposed the use of a Morlet Wavelet transformation to reduce noise and enhance vessel visibility in the source images. The Morlet Wavelet is a Gaussian function in the complex plain (2D) and is useful to 'smoothen' edges in images and reduce noise. Its values are best calculated using the Fourier transformation.

In a further step, Ghaderi et al. (2007) introduced a Bayesian classifier to calculate the conditional probability of whether a pixel is a blood vessel or not. This conditional probability took into consideration some of the features that have been calculated by the Morlet-Wavelet transformation. The process achieved reasonable results against the data in the DRIVE and STARE databases.

In addition, Ghaderi et al. (2007) tested the following alternative: in the image pre-processing step they still used the 2-D Morlet Wavelet. However, subsequently, they trained a Neural Network rather than a Bayesian classifier for the image segmentation.

This highlights the trend that neural networks are the preferred training method in most recent image segmentation procedures.

From a mathematical point of view, Morlet Wavelet transformation is a very interesting segmentation approach. The Morlet Wavelet is defined as function over the complex plain (2 dimensional), and this fits well with the idea for this project, to segment the 2D image projections of the 3D Source scan.

In a future study, it could be investigated if some of the aforementioned machine learning techniques and morphological processing methods for retinal blood vessels can be reapplied in the context of this thesis - for the segmentations of blood vessels in the 2D image projections.

3.5 Summary

A wealth of literature is available for the segmentation of blood vessels in medical images. Whilst a large number of unsupervised classification techniques produced excellent segmentation results, there is a recent trend towards supervised machine learning algorithms that employ the use of neural networks.

There is no one algorithm that works best for all blood vessel segmentation tasks. Various supervised and unsupervised classification methods have been proposed for 3D MRA scans and 2D retinal images.

Source and ground truth data for a set of 2D retinal images are made available to the public in the STARE Clemson (2000) and DRIVE Utrecht (2007) databases. Those two databases have been used as a baseline to develop and test many different blood vessel segmentation methods. Unfortunately, to date, no similar data sets with cerebral MRA scans are available to the public.

For this project, U-Nets - as described in Section 3.2 - are used for blood vessel segmentation of the 2D image projections. However, it might be feasible to use an alternative algorithm instead. Given the relatively small number of available ground truth data, an unsupervised classification method, such as Frangi from Section 3.1 or the Morlet Wavelet approach from Section 3.4.3, could also yield useful - or even better - 2D segmentation results.

Chapter 4

Image Projections

Before diving into the mechanics of image projections in this thesis, it is necessary to provide some background information about the nature of the MRA source data in 3D space.

4.1 Pixels vs Voxels

Pixels are the smallest unit available to address the colour component in a 2D image. Though often represented as dots, they are typically the squares in an image map and are colour coded in the RGB format (with three colour channels **R**ed/**G**reen/**B**lue).

Voxels (short for volumetric pixels) are the counterpart of pixels in 3D space. Voxels in MRA scans consist of a single colour channel (grey only), and the voxel is the smallest volumetric entity that can be assigned a signal strength by the electromagnetic field in the MRI scanner.

4.2 Format of MRA Source Scans

The 3D MRA scans are produced by creating a number of two-dimensional pictures (represented as 'slices') across a region in the human body, while the patient slides through the magnetic field.

Dimensions of the MRA scans vary for different machine types and manufacturers. The Siemens Sonata model has been used to produce the scans for this project. It works with the following technical specifications:

- Each image slice is quadratic with the dimension 320 x 320 pixels.
- A total of 247 image slices are created per MRA scan.
- The dimensions of a voxel in an MRA scan are 0.625 mm x 0.625 mm x 0.6 mm.

The voxel in the 3D scan of the Siemens Sonata is not a cube, but a rectangular cuboid. However, for simplicity's sake, voxels are depicted as cubes throughout this thesis.

The developed software takes into consideration that the resolution of the MRA scans needs to be configurable. However, the developed code has only been tested against the provided ten MRA scans from the Siemens Sonata model.

4.3 Image Projections

For this project, nine different parallel image projections are considered:

- Three Front Views: Parallel projection lines in direction of the three-dimensional axes (x, y, z), through the faces of the MRA rectangular cuboid

- Nine Edge Views: Parallel projection lines at a 45 degree angle through the edges of the MRA rectangular cuboid.

Note: projections from opposing directions are excluded as they produce mirrored 2D image versions, with no additional structural information.

Maximum Intensity Projection (MIP) is used to map the voxels with the highest value along a projection line into the 2D image projections.

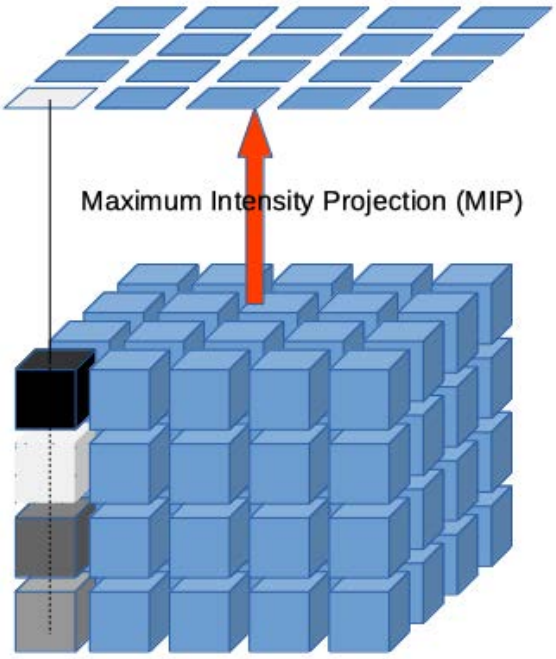
4.3.1 Front Views

As illustrated in Figure 4.1a , parallel image projections through the faces of a 3D cuboid preserve the dimensions of the face that they hit. Computationally fast tools are available in different programming languages to map voxels along an axis into a 2D image.

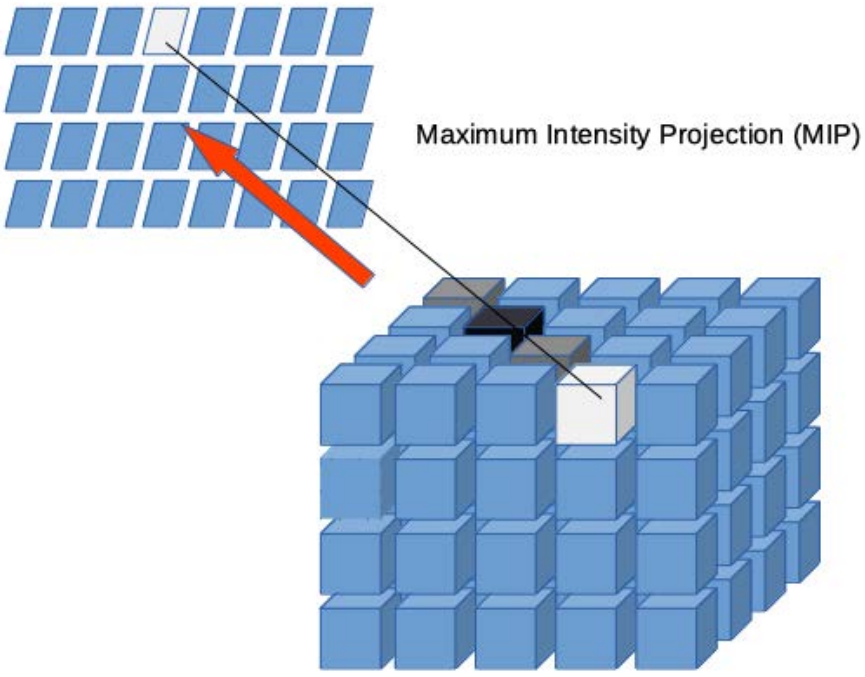
4.3.2 Edge Views

The edge view considers projections at a 45 degree angle into the edges of the rectangular cuboid (see also Figure 4.1b). This approach ensures that each voxel is on the projection line of a 2D image and no voxels are excluded from consideration when constructing the 2D projections.

Consequently, each voxel in a 3D MRA scan is reached by all nine projection lines, and this important fact is later used when reconstructing the 3D segmentation map and deciding whether a voxel is assigned to background or blood vessel structure.



(a) Front View



(b) Edge View

Figure 4.1: Front View and Edge View (Conceptual)

Edge View Implementation

The mapping of voxels alongside the edges of the cuboid is not straight forward. Index traversing is required when walking on a projection line through an image cuboid. Resultingly, such implementations are computationally expensive and take over a minute to complete on commodity hardware.

To eliminate this performance bottleneck, the idea is conceived to tilt the cuboids alongside an axis by 45 degrees, and embed this cuboid into an enlarged rectangular cuboid. Two computationally efficient 'Front View' projections are then executed against the enlarged cuboid to obtain edge views of the original 3D MRA source scan. This concept is illustrated in Figure 4.2.

By executing this process along the three axes, the six 'Edge View' projections (2 x 3) are obtained. Implementation of this idea shortens the time for the processing of image projections from minutes down to under ten seconds.

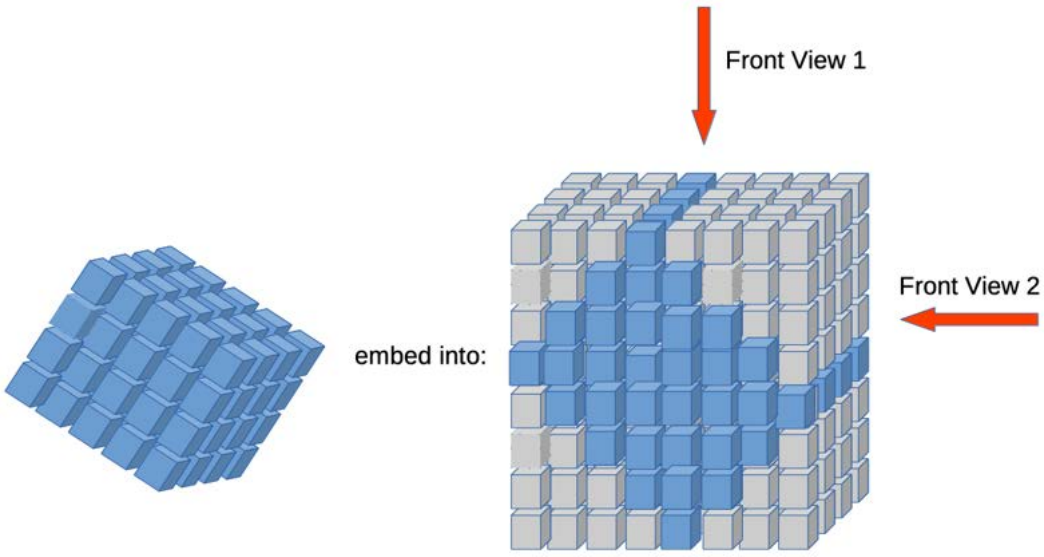


Figure 4.2: Edge to Front View Conversion

Width Stretching

Edge View projections do not preserve the relation between width and height in the projected 'Edge View' image. In reality, the new width of an Edge View projection is

$$w = \sqrt{(x-1)^2 + (y-1)^2} \tag{4.1}$$

However, the new image has a width of

$$w = x + y - 1 \tag{4.2}$$

As a consequence, the width of the new edge view projections is stretched by the factor

$$factor = \frac{x + y - 1}{\sqrt{(x-1)^2 + (y-1)^2}} \tag{4.3}$$

This phonomenon is illustrated in Figure B.3 for the eye/ear image projections of MRA scan MNI-0656. The resulting two edge projections B.3a and B.3a are stretched by the factor 1.416 (x and y have both a length of 320px, as the MRI model produces quadratic slices of length 320px).

4.3.3 Vertex Views

It should be noted that projections into the vertices (corners) of the rectangular cuboid are also possible by utilising the same technique as for the edge view projections. The original 3D source cuboid is tilted twice at 45 degrees alongside two different axes. The twice-tilted cuboid is then embedded into a further enlarged rectangular cuboid, and 'Front View' projections deliver the desired vertex view projections.

This technique would allow the inclusion of four additional 'Vertex View' projections into the process, and may be investigated in a future study. It is not included in this thesis.

4.4 3D Image Reconstruction

The second pivotal step in the image projection method is the reconstruction of the 3D segmentation map, based on the 2D image segmentations.

This approach reverses the steps described in Section 4.3. A projection line through the MRA source image is built for each predicted blood vessel pixel in the 2D segmentation map. Wherever a voxel with the highest signal strength in the 3D source image can be found on this projection line, it is assigned a 'hit count' indicating that it is deemed to belong to a blood vessel, based on the prediction of the respective 2D image projection. The concept is illustrated in Figure 4.3.

After completing this process for all nine segmented image projections, a new 3D raw segmentation map is constructed, with each voxel being assigned a number between 0 and 9.

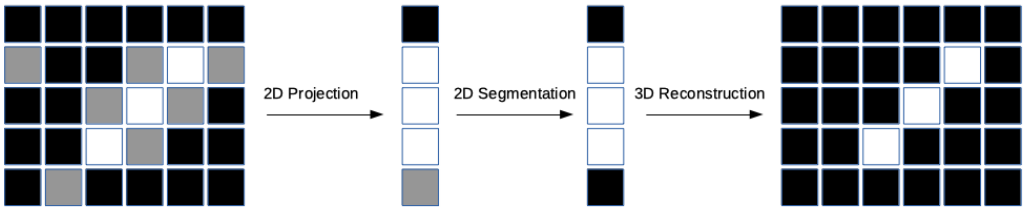


Figure 4.3: Image Slice Segmentation

A value of 0 means that the voxel in the new 3D map has not been classified as a blood vessel by any of the nine projections; such a voxel is assigned to the background. A value of 9 indicates that all nine image projections classify the voxel as a blood vessel and such a voxel is deemed to be part of the cerebral blood vessel system.

An appropriate threshold value (N-HITS) for the number of 'projection hits' discriminates between background and blood vessel for the remaining voxels in the 3D raw segmentation map.

The raw 3D segmentation map is then further improved in a post-processing step to produce the final 3D segmentation map for the blood vessel system in cerebral MRA scans.

4.5 Weaknesses of Projection Method

Using the MIP method along a projection line has some major weaknesses.

The process identifies only the voxel with the strongest intensity on a projection line. Blood vessels with less intensity are not identified by this approach.

Consequently, when the projection line crosses a blood vessel region, it misses neighbouring voxels that are still part of the same blood vessel region but have slighty less signal strength.

As a result, the image projection method is not able to reconstruct the full blood vessel stream in 3D based on the segmentations in the 2D projections. It is only capable to construct the 'skeleton' of the cerebral blood vessel system, as illustrated in Figure 4.3. This limitation is considered throughout design and implementation of the solution. The introduction of image dilution in a post-processing step tackles this issue.

In addition, different blood vessel streams may cross through a projection line. The image projection will only capture the blood vessel stream with the highest intensity. This issue is addressed by viewing the 3D MRA scan from different angles.

Finally, the MIP method is only possible because the density of blood vessels in the cerebral MRA scans is very low (on average less than 0.5 per cent). This method would not be suitable for objects with high density in 3D space.

4.6 Summary

Parallel image projections into the 3D MRA source scan are used to produce nine 2D image projections of the source scan. This dimensionality reduction method allows computationally more efficient image processing in 2D space.

After segmenting the nine 2D image projections into background and blood vessel regions, a skeleton of blood vessel structures in 3D space is reconstructed.

In an image diluation process, the final 3D segmentation map is built by padding the skeleton of the image vessel regions to an appropriate size.

The proof-of-concept in Chapter 6 confirms that only minimal information of the source data is lost by this process.

This method is only feasible because a very small percentage of the 3D source image - less than 1 per cent - contains blood vessel cells.

Chapter 5

Performance Metrics

Performance metrics are required to measure and compare the quality of predicted blood vessel segmentation maps against their corresponding ground truth data.

The metrics, defined in this section, are calculated and tabulated for image processes throughout this document. They help quantify and compare the prediction quality for various segmentation runs against their ground truth counterparts.

The voxels in the 3D ground truth image are encoded as binary values (positive/negative). Consequently, four different scenarios are possible for the voxels in a prediction map:

- TP (True Positive): Voxel predicted to be positive and classified as blood vessel in the ground truth
- TN (True Negative): Voxel predicted to be negative and classified as background in the ground truth
- FP (False Positive): Voxel predicted to be positive but classified as background in the ground truth
- FN (False Negative): Voxel predicted to be negative but classified as blood vessel in the ground truth

This information is commonly depicted in a binary confusion matrix. Table 5.1 illustrates such a matrix as concept.

GT \ Pred.	*Positive*	*Negative*	*Totals*
Positive	**7**	**4**	11
Negative	**2**	**987**	989
Totals	9	991	1000

Table 5.1: Binary Confusion Matrix (Concept)

In Table 5.1, seven voxels have been predicted correctly as blood vessels (TP), four voxels have been predicted falsely as background (FN), two voxels have been predicted falsely as blood vessels (FP), and the 987 voxels have been classified correctly as background (TN).

Figure 5.1 provides conceptualised ground truth and prediction maps for the numbers in Table 5.1. It should be noted that the correctly classified background pixels (TF=989) are not visualised throughout this thesis, as they are of less interest when assessing the quality of a predicted segmentation map.

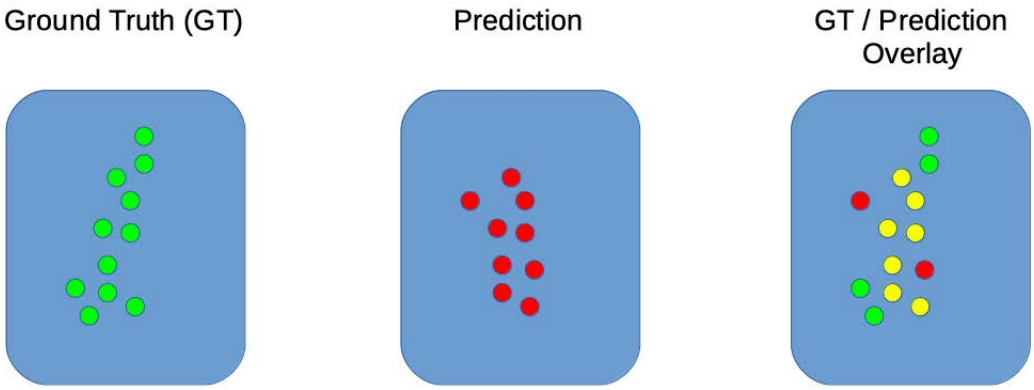


Figure 5.1: Ground Truth / Prediction Overlay

The colour codes in Figure 5.1 are used throughout the whole document: ground truth data are depicted in green, predictions in red, and matches between ground truth and prediction in yellow. It will be the aim to maximise the number of matching (yellow) data points in this project, whilst keeping the numbers of 'False Positives' (red) and 'False Negatives' (green) low.

There is a strong class imbalance in the data. Less than 1 per cent of all voxels in a cerebral MRA scan are blood vessels.

5.1 Precision, Recall and F1-Score

Precision is the conditional probability that a voxel that has been predicted to be a blood vessel is actually a blood vessel according to the ground truth data. Mathematically, this conditional probability can be written as:

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

For the confusion matrix in Table 5.1, the precision is calculated as 7/(7+2) = 0.78. In contrast, recall is the conditional probability that a voxel that belongs to a blood vessel is correctly classified as blood vessel by the predicted segmentation map. Formally, this can be written as:

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

For the confusion matrix in Table 5.1, the recall is 7 / (7+4) = 0.63.

To create an average combined score across precision and recall, the harmonic mean is used. The harmonic mean of precision and recall is also called F1-Score and calculated as follows:

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5.3}$$

The F1-Score for the data in Table 5.1 is 2x0.78x0.63 /(0.78+0.63) = 0.70.

It can be seen that the F1-Score is somewhere between precision and recall. For the harmonic mean, lower values attract a larger penalty than for the arithmetic mean.

Throughout this thesis, precision, recall and F1-Score are used as metrics to assess the quality of predicted segmentation maps against the ground truth data.

Precision and recall are equally weighted. The TP count (yellow dots) should be maximised, whilst minimising FP (red) and FN (green).

It should be noted that the value of TN (correctly predicted background voxels) is not considered in this thesis. This number is very large and overpowers the other three possible outcomes. Throughout this work, prediction maps are occasionally padded with zeros and such an increased number of artificial background pixels does not contribute to the overall quality of the predicted segmentation maps.

Precision and recall - paired with the F1-Score - are useful metrics in situations of unbalanced classes. It is a common metric in the literature and used frequently to compare the quality of algorithms for the segmentation of blood vessels.

Furthermore, precision is also called 'positive predicted value', and recall is also known under the term 'sensitivity'. To uphold linguistic consistency throughout the document, the terms precision and recall are used.

5.2 Binary Cross Entropy

Binary Cross Entropy (BCE) is used as a performance metric for the loss function of the neural networks. The neural networks operate on 2D image projections and report the value of a pixel not as a binary class value (0/1), but as a real number in the range from 0.0 to 1.0. As such, recall and precision are not appropriate measures to assess the performance of a neural network against ground truth data. Instead, the assigned value for a pixel can be interpreted as probability that it belongs to background or blood vessel.

Binary cross-entropy is calculated by formula 5.4.

$$BCE = -\sum_{i \in GT} [(y_i \, log(p_i) + (1 - y_i) \, log(1 - p_i))] \tag{5.4}$$

Identifier i denotes a pixel in the 2D ground truth image GT. y_i is the binary value (0/1) for pixel i in the 2D Ground Truth image. p_i is a real

number between 0.0 and 1.0; its value is predicted by the segmentation process for each pixel i in the source image. p_i can be interpreted as probability that pixel i is a blood vessel.

The neural network attempts to minimise the binary cross entropy (BCE) in Equation 5.4 by adjusting the values for p_i in a series of training runs.

A threshold value needs to be defined to discriminate between background and blood vessel in the final segmentation map. In this thesis, equal weight is attributed to recall and precision, and the threshold is set to 0.5. However, the threshold value is a configurable parameter and can be changed in the initialisation file. For example, it could be changed to 0.4 to favour the classification of blood vessels over the classification of background.

5.3 Summary

In this thesis, precision, recall and F1-Score are used as metrics to assess the performance of the predicted segmentation maps against the ground truth.

Equal weight is assigned to precision and recall, and the harmonic average between the two measures, the F1-Score, provides a balanced overall performance metric for the predicted segmentation maps against the ground truth.

Binary-Cross-Entropy (BCE) is used as loss function during training of the neural networks in 2D space. It is not further considered when assessing the quality of a prediction map against its ground truth.

Chapter 6

Phase I - Proof of Concept

The Proof of Concept in Phase I uses the 2D projections of the ground truth data to separate blood vessels and background in the 3D source image. The predicted segmentation maps are then compared with the provided 3D ground truth images.

For this phase, nine of the ten MRA scans are used (with the exception of record MNI 0664).

6.1 Ground Truth Projections

For each of the nine source MRA scans, nine ground truth projections are created. Unfortunately, the ground truth data are not provided as boolean data (0 or 1), but are encoded as single-coloured pixels in the range from 0 to 255.

In a first step, it has to be decided which value in the ground truth image should be used as a threshold to distinguish between background and blood vessel region.

3D ground truth images have been created with a range of different threshold values, and the outcomes can be inspected in Figure B.1. As can be seen, the threshold value of 100 delivers reasonable output, and this value is used throughout this thesis. It is a fundamental hyper parameter, is configurable and should be assessed by an experienced radiologist.

6.2 Raw 3D Segmentation Maps

Using the projection process described in Chapter 4, a segmentation map is constructed for each of the nine MRA source scans; the aggregated performance scores are computed across all nine MRA scans.

6.3 Selection of Hyper Parameter N-HITS

N-HITS is the threshold value for the number of hits by projection view. For example, if N-HITS is set to 7, then all voxels in the reconstructed 3D segmentation map are deemed blood vessels, if they are classified as blood vessel by at least 7 out of the 9 image projections.

Based on different settings for parameter N-HITS, the performance metrics against the ground truth data are calculated.

A diagram depicting a value of N-HITS against precision, recall and F1-Score is provided in Figure 6.1:

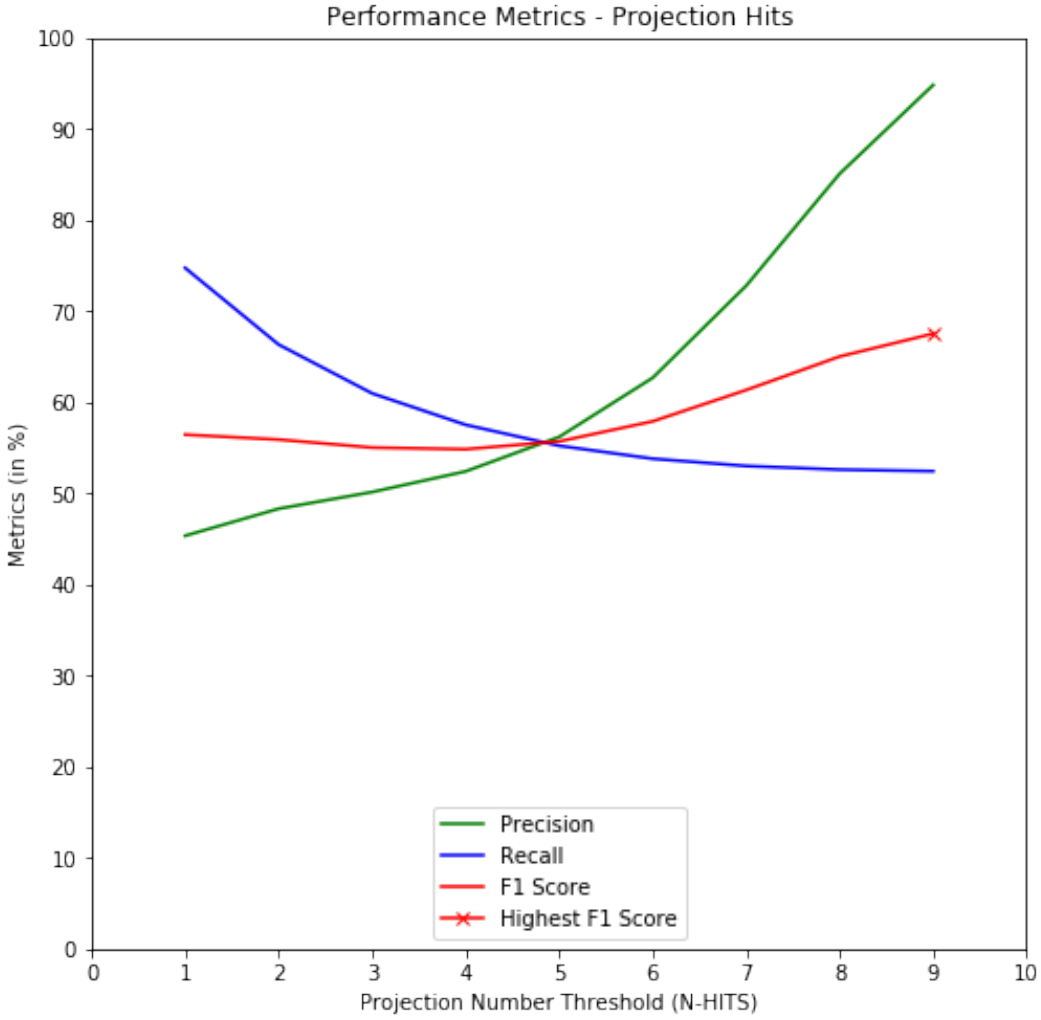


Figure 6.1: Parameter NHITS against F1-Score (Proof of Concept)

Precision increases, and recall decreases, as N-HITS increases, which is expected. When a large number of image projections predict that a voxel is a blood vessel, then the probability is high that this prediction is correct.

Contrary, it can be expected that most voxels, representing blood vessels, are hit by at least one image projection. Therefore recall should be highest

with N-HITS=1.

It can be seen that the F1-Score reaches its maximum at N-HITS = 9, and this is the value that is selected for this proof of concept.

Image projections can construct a useful skeleton of the blood vessel streams in the 3D scan. The aim is to increase the precision to a high value (minimising the number of FP). A low recall (e.g. below 0.4) is acceptable as long as *all* blood vessel regions are traced in the raw segmentation map. Figure 4.3 illustrates that it is not possible to build up a full blood vessel segmentation map just based on the data in the 3D raw segmentation map.

6.4 Post Processing

The application of mathematical morphological operations in image post processing is common for many segmentation algorithms. For example, Hassan et al. (2015) use a combination of dilution and erosion operations to finalise the blood vessel segmentation map for retinal images.

Similarly, in this project, the raw segmentation map is finalised in two subsequent post processing steps using dilution operations.

Figure 6.2 illustrates how the raw 3D segmentation skeleton is filled up to produce the final segmentation map. In a first step, any gaps (holes) in the voxel map are closed. In a second step, image dilution is applied to carefully expand the blood vessel segments.

The mechanical details for both steps are specified in Sections 6.4.1 and 6.4.2.

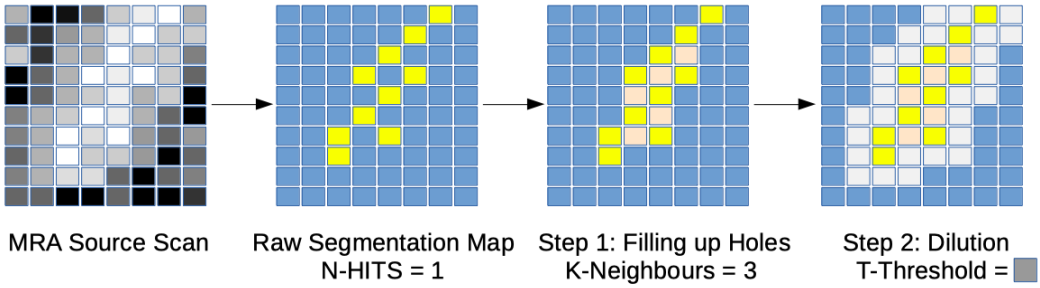


Figure 6.2: Two-Step Post Processing

6.4.1 Step 1: Filling Holes (K-Neighbours)

If voxels have many neighbouring voxels that are classified as blood vessels, then the likelihood is high that those voxels are also blood vessels. Filling up those holes in the blood vessel segmentation map is undertaken by checking two conditions:

1. The voxel has a signal strength above a certain threshold, which is set to 100 for this project in the configuration file.

2. The voxel is surrounded by K neighbours that are classified as blood vessels. Each voxel - with the exception of the border regions - has 26 neighbouring voxels (3^3 - 1).

Figure 6.3 displays precision, recall and F1-Score for each possible value of K from 1 to 26.

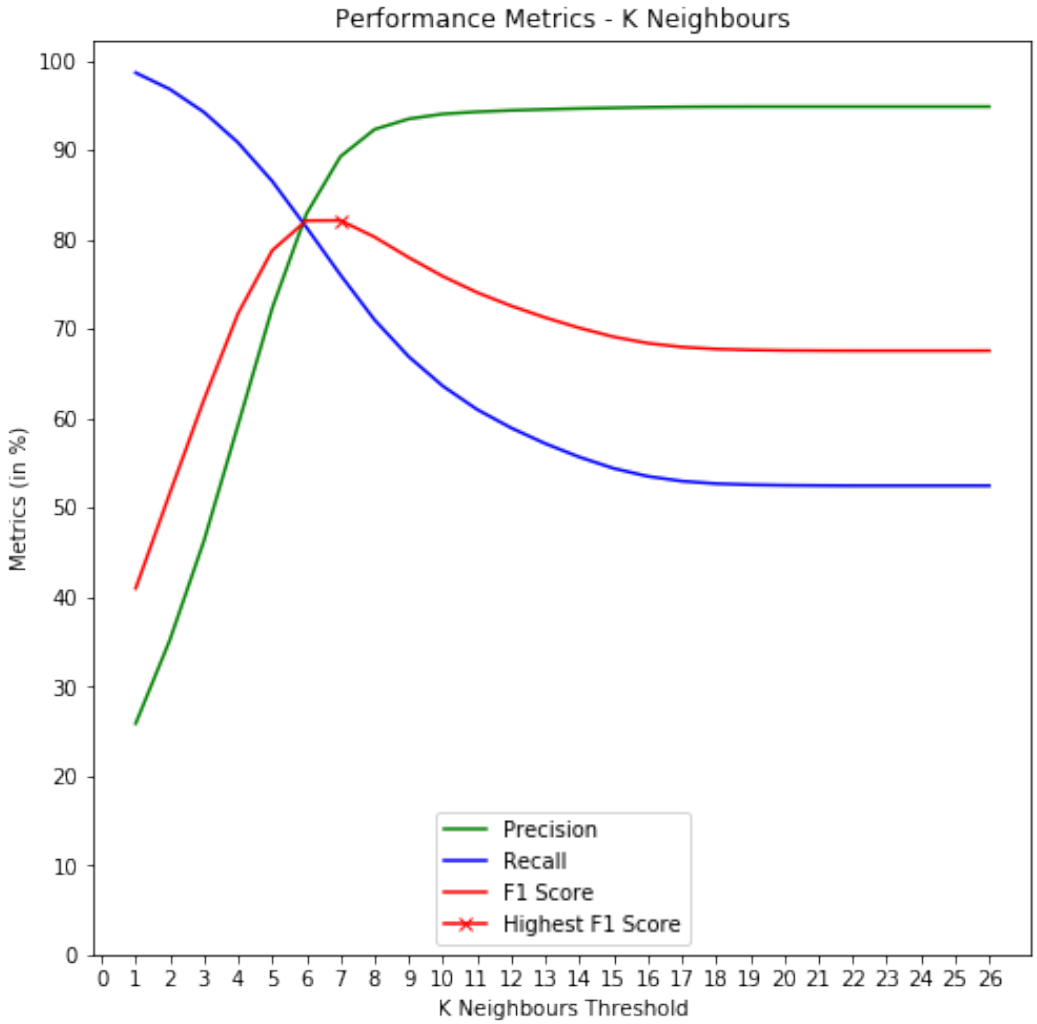


Figure 6.3: K-Neighbours against F1-Score (Proof of Concept)

It can be seen, that the best F1-Score can be achieved with K between 8 and 9; the value K=9 is selected. This improves the F1-Score for record MNI-0656 to over 0.8.

6.4.2 Step 2: Dilution

This result can be further improved by adding voxels at the edges of the blood vessel segments and smoothening the blood vessel areas. This is undertaken in a dilution process.

Dilution is a basic operation in image processing. In the context of this work, a structuring shape is specified that defines the neighbouring region of a voxel with value 1. All voxels within its area are assigned 1 (positive) if the voxel in the centre of the shape is positive.

Excellent results are achieved when selecting a cube with length 5 as the base shape. Whenever the centre voxel in this base shape overlaps with a positive voxel in the segmentation map, then all voxels within this base shape region are also included in the blood vessel map. By using a cube with side length of 5, the centre of the cube will expand to neighbours and neighbours-of-neighbours in the 3D segmentation map.

One further condition has to be met. A cut-off value T for the signal strength should be selected. Only voxels with signal strength above the threshold T in the source scan should be included in the blood vessel segmentation map.

The F1-Scores are calculated for signals strengths between 1 and 255, and the results are depicted in Figure 6.4.

The highest F1-Score is achieved with T=241. After applying the diluation process with this threshold value, the results and performance metrics for record MNI-0656 are recorded in Tables 6.1 and 6.2.

It can be confirmed that the proposed process produces an excellent segmentation map for the manually segmented cerebral MRA scan MNI-0656.

Pred. / GT	*Positive*	*Negative*	*Totals*
Positive	**45632**	**757**	46389
Negative	**1185**	**25245226**	25246411
Totals	46817	25245983	25292800

Table 6.1: Confusion Matrix for MNI-0656 - Final Segmentation Result against 2D Ground Truth Data

	Precision	*Recall*	*F1-Score*
MNI-0656	0.975	0.984	0.980

Table 6.2: Performance Metrics - Final Segmentation Result against 2D Ground Truth Data

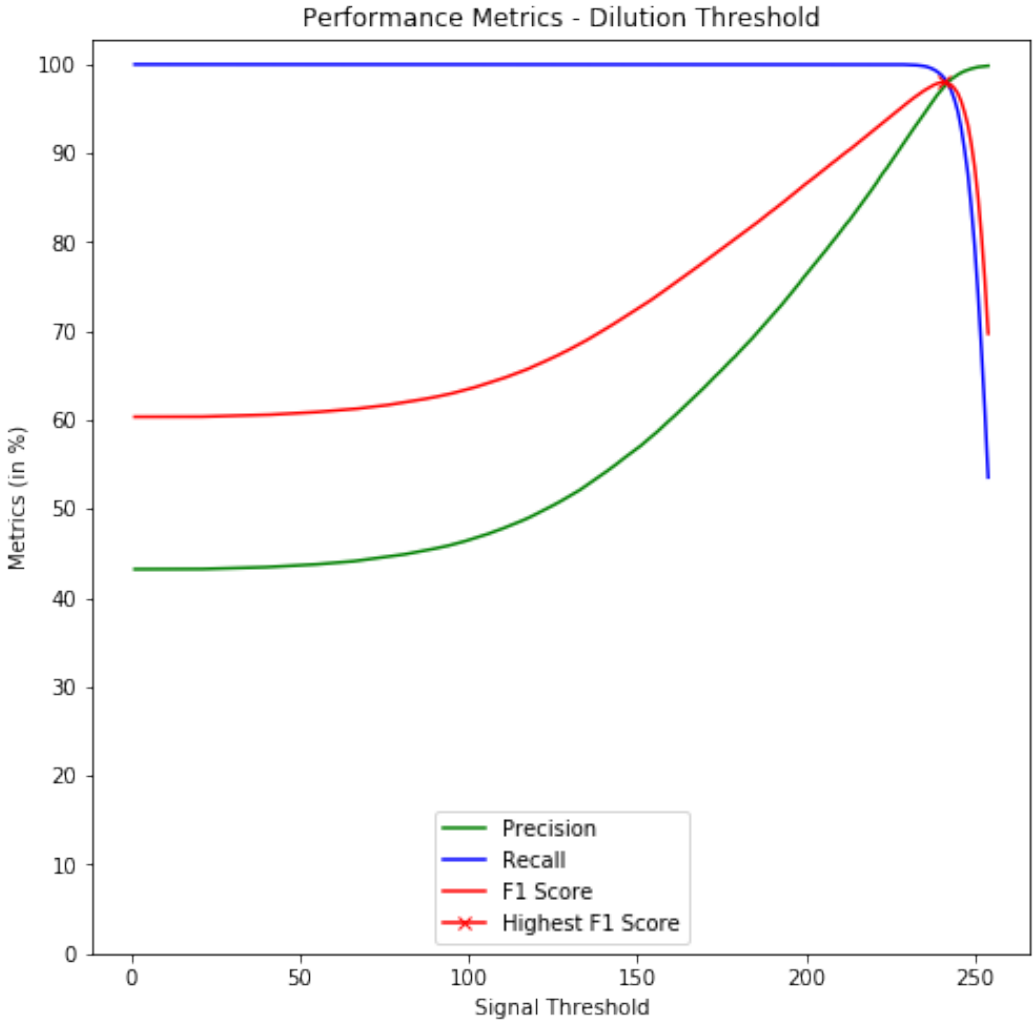


Figure 6.4: Dilution Threshold against F1-Score (Proof of Concept)

6.5 A Word about Performance Metrics

Sometimes, the results of performance metrics do not tell the full story about the quality of a segmentation result: Figure 6.5 illustrates that the final result with the performance metrics in Table 6.2 produces a nearly perfect match against the ground truth. Most voxels in Figure 6.5 are correctly classified (yellow), and only few voxels are misclassified at the borders of the blood vessel segments (red and green).

Those minor misclassifications at blood vessel border regions do not reduce the overall quality of the segmentation result in Figure 6.5.

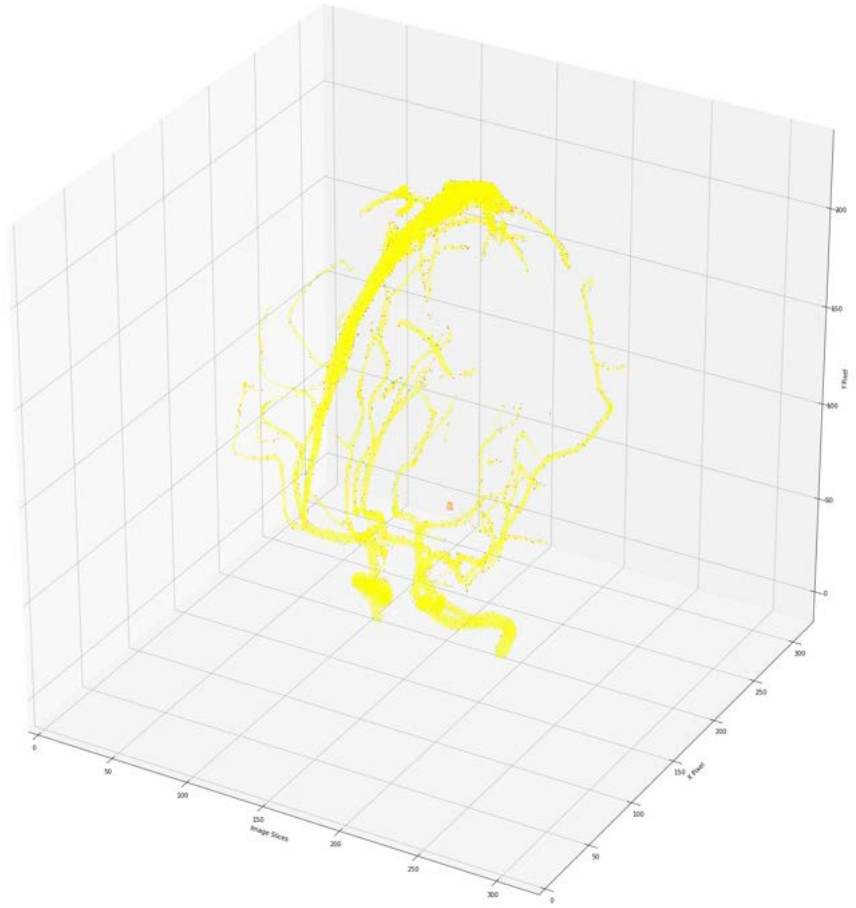


Figure 6.5: MNI-0656: Final 3D Segmentation Overlay (Proof of Concept)

6.6 Summary

Phase I delivers a feasability study with proof of concept for the image projection method from Chapter 4.

It is proven that image projections can be used effectively for the creation of high-quality 3D blood vessel segmentation maps, as illustrated in Figure 6.5 for record MNI-0656.

The final prediction map is created by applying two post processing dilution steps that refine the blood vessel skeleton in the 3D raw segmentation map.

Best results are achieved with the following hyper parameter settings:

- N-HITS = 9 (for construction of 3D raw segmentation map)
- K = 9 (1^{st} dilution step, filling up holes in the segmentation map)
- T = 241 (2^{nd} dilution step, slightly expanding the blood vessel areas)

However, some detail in the 3D ground truth is inadvertently lost during this process, even with fully complying 2D segmentation maps for the nine image projections. Those losses have no impact on the overall quality of the final prediction map, as demonstrated in Figure 6.5. Only a small number of voxels are misclassified at the blood vessel border regions.

Chapter 7

Phase II - Training of Neural Networks

Phase II encompasses the selection of an appropriate neural network architecture to segment images for each of the nine projections in 2D space. The selected neural network architecture is trained and validated against the 2D image projections.

Quality of segmentation results is assessed by comparing the predicted 2D segmentation maps with the corresponding counterparts of the 2D ground truth projections.

7.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have gained huge popularity since their effectiveness in image classification tasks has been realised.

The ground-breaking work of Alex Krizhevsky (2012) with the invention of the AlexNet is now a standard architecture based on CNNs and used to identify objects, such as cars or trees, in an image. Many new developments in the arena of image recognition use a variant of the AlexNet as the underlying neural network architecture.

Though not strictly an image segmentation task, the AlexNet and CNNs in general can be used very effectively for the segmentation of objects in an image. Using a CNN, each pixel (or voxel in the case of 3D images) is assigned a probability between 0 and 1 that it belongs to a certain object in an image. By visualising the resultant heat map, objects in an image, such as a cat or a dog, can be highlighted.

7.2 U-Nets

The use of CNN's for image segmentation tasks has been further advanced by Ronneberger et al. (2015), who developed a U-Net architecture. The left side of the U is the traditional CNN that is used for image classification and returns a real number between 0 and 1 for each image pixel. However, rather than terminating the process at this point, the output map (with real numbers between 0 and 1) is then expanded again. The right side of the U describes this process. It uses as input the convolutions at the same level, as on the left side of the U. Figure 7.1 illustrates this architecture.

The U-Net is useful because it can produce robust results, even when only few labelled images can be provided as input for the training process.

Breemer (2018) trained and tested a U-Net architecture successfully for the segmentation of the single MRA record MNI-0656. In his thesis, training and test data with ground truth records were generated through image augmentations of MRA scan MNI-0656.

The U-Net architecture has proven to work well on medical images, as shown by Ronneberger et al. (2015), who developed this neural network for the segmentation of biomedical images.

Given its current popularity and the small number of labelled source records for this project, it has been decided to work with a U-Net architecture on the segmentation of the cerebral MRA scans.

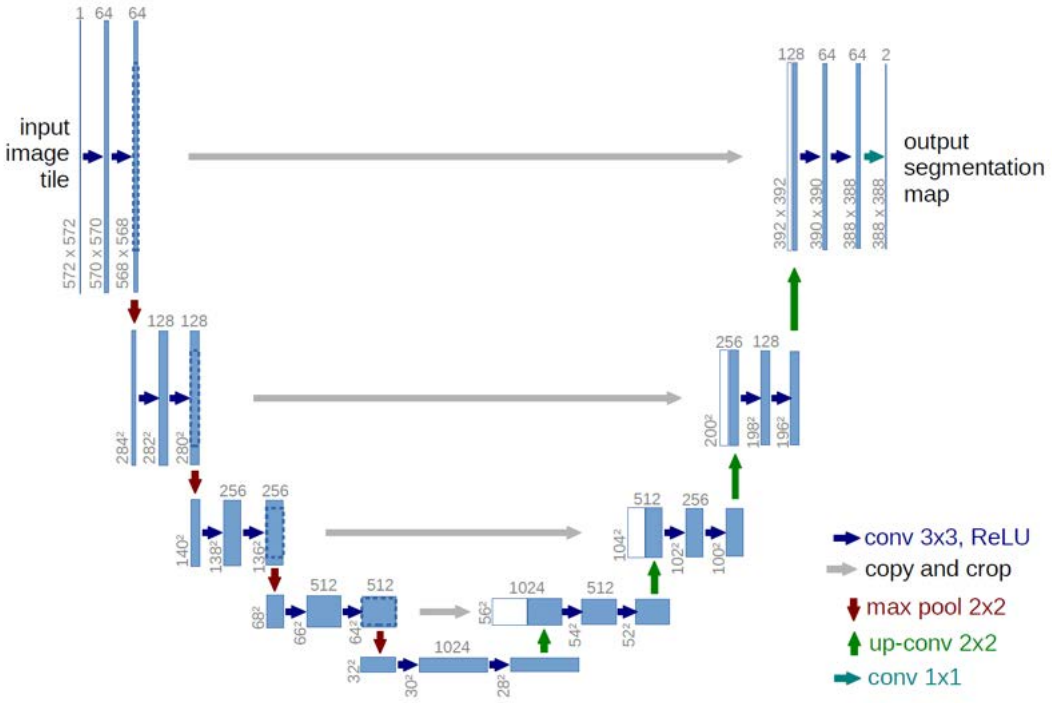


Figure 7.1: UNET Architecture (Source: Ronneberger et al. (2015))

7.3 Selecting U-Net Version

A number of U-Net implementations are available on the internet, such as

- A U-Net model is available for download on The University of Freiburgs website, which is based on the work of Ronneberger et al. (2015). It is tested on Ubuntu with Matlab 2014b.
- zhixuhao (2018) offers a ready-to-use U-Net version for Python 3, utilising the Keras framework and Tensorflow (Google's implementation for various neural network architectures). Both libraries can be downloaded from GitHub and are available to the public.
- Sterbak (2018) published and tested a U-Net version for segmentation of seismic 2D images. This version is available on the Internet (code snippets need to be integrated in purpose-built solution).

Several U-Net models were tested against the available 2D image projections. Some of those architectures cannot run efficiently on commodity hardware; others produce not the desired quality as output. It turns out that Sterbak (2018) is flexible, easy to configure and can produce good results against the 2D training data. The model, published by Sterbak (2018) is used as base architecture, and hyper parameters are tuned for this model to produce optimised output in the segmentation maps against the provided ground truth data of the cerebral MRA scans.

7.4 Single U-Net vs Multiple U-Nets

It is possible to train a single U-Net that segments all 81 images (9 image projections for each of the 9 MRA scans). This approach allows the training process to work with a relatively large number of labelled source images.

In addition, a single U-Net reduces overall complexity of the segmentation process. Coefficients for one U-Net model need to be maintained, loaded and processed for the segmentation of all 2D images. This has a positive impact on runtime performance for the overall segmentation process.

Working with a single U-Net produces reasonable results and is definitely an option.

Alternatively, different U-Net versions can be trained for each of the 9 image projections. This option has some drawbacks, as overall training times tend to be significantly longer (up from about 20 min to 2.5 hours). Also, processing times for a single MRA scan increases, as 9 different U-Net models need to be executed for processing of segmentation maps. Finally, parameters for 9 different U-Nets need to be maintained independently by the system.

Both options have been investigated, and despite the disadvantages of working with multiple U-Nets, it has been decided to train and configure 9 separate

U-Nets; one for each projection view. The key reasons for this decision are listed below:

- Metrics of a single U-Net cannot reach the same performance levels as those that are obtained by 9 separate neural networks (1 for each projection view).

- A projection-specific neural network can learn the intrinsics of a certain projection view and optimise the process accordingly, whereas a general neural network provides balanced results across all nine projection angles. This is best illustrated when comparing projections from top of the skull into the MRA scan with edge views from Figures B.2a and B.3b. Some of the blood vessel systems in 'Projection 1' are overpowered by high density voxels in the skull region. In contrast, blood vessel regions in 'Projection 5' (an edge view) are relatively clear with less interference.

- 2D Projection views have differing width and height dimensions. This information can be utilised in the training process of the U-Nets.

- 9 different U-Net versions increase the robustness of the overall segmentation: shortcomings in the segmentation process of a single U-Net version can be compensated by the other 8 neural networks.

7.5 CLAHE Enhancement

In an initial step, after creating the image projections of the source MRA scan, a Contrast Limited Adaptive Histogram Equalisation (CLAHE) process is applied, to enhance the contrast between blood vessel and background regions.

CLAHE uses a 'clip point' of the histogram for the redistribution of pixel values in an image region, of which all pixels above this clip point are redistributed. This might lead to distorted or overemphasised features in an image. To 'smoothen' the CLAHE contrast enhancement, Yakun Chang (2018) proposes the introduction of dual-gamma correction and test this new approach against a range of images in their paper.

For this project, the implementation of library cv2 for Python 3 has been used to enhance contrast between blood vessels and background. The result of this operation can be viewed in Figure B.6.

Furthermore, the CLAHE process adds some adaptivity to different MRI scanner versions. Blood vessel contrasts between different models are enhanced and to some degree normalised prior to feeding the data into the neural networks.

7.6 Image Resizing

Image projections have different sizes. For example, see Figures B.2, B.3 and B.4 for the front and edge projections of record MNI-0656.

Consequently the U-Net models need to be configurable for different image input sizes. Furthermore, image width and height of the 2D projections need to be a multiple of 64, to ensure compatibility between extracting and contracting paths of the U-Net.

This requirement is implemented by padding the 2D image projections with zeros (black), until width and height are a multiple of 64. Those padded images are fed into the neural network. After completion of the segmentation process, the resultant image segmentation maps are cropped back to their original size.

This approach also ensures that the segmentation process works for MRI models with different technical specifications (depth, width, height of 3D image). Those parameters are configurable in the initialisation file and are used by the image projection and segmentation process.

7.7 Training and Validation Data

The eight records, segmented using the Frangi filter, are used for training, and the manually segmented data set MNI-0656 is set apart for validation. The validation record is never used by the neural network for training purposes. Its sole purpose in the whole process is to verify the segmentation results against an unseen manually segmented data set with high-quality ground truth data.

Better performance metrics could be achieved by feeding record MNI-0656 into the training process, and using a resampling method with cross-validation to compute the loss function.

However, it is a deliberate decision to use the manually segmented record just for validation and not for training. It provides a level of confidence that the segmentation process will produce segmentation maps of comparable quality once it is executed against other manually segmented data sets.

7.8 U-Net Hyper Parameters

Table 7.1 provides the list of hyper parameters in the U-Net architecture that are used for training of the image projections. All nine image projections are trained with the same hyper parameter settings.

Parameter	*Setting*	*Comment*
Epochs	50	The neural network completes up to 50 training iterations across all training and validation records
Batch Size	1	Processing each record individually produced better results than running the MRA scans in two batches (of size 4 each)
Initialisation	He Initialisation	He et al. (2015) initialisation adds a level of randomness to the training process, which cannot be controlled in Python
Loss Function	Binary Cross Entropy	BCE is part of the Keras library and produced excellent convergent results; it works well in combination with precision and recall
Learning Rate	0.001	Different learning rates have been tested; this (default) setting produced best results with the ADAM optimiser; reducing the learning rate gradually, as the performance of the model did not improve, led to overfitting and did not generalise sufficiently any more against unseen data
Drop-Out Rate	0.05	Dropping out neurons randomly helped with regularisation and produced overall better results when testing against unseen data
Patience	10	If the loss function did not improve after 10 epochs, the learning process was stopped.
Metric	Accuracy	This setting has no impact on the training process; is only used in visualisation graphs
Optimiser	ADAM	ADAM produced better results than Stochastic Gradient Descent (SGD)

Table 7.1: Hyper Parameters for Training of U-Nets

7.9 Loss Functions

Binary Cross Entropy (BCE), as defined in Equation 5.4, is the loss function for the training process, and it converges for most runs after 20 epochs, without improving any further.

BCE can be further minimised by reducing the learning rate or including the manually segmented record MNI-0656 and variants thereof in the training data. However, in general, the tested U-Nets start to overfit against the training data once this is done.

The main purpose of this training process is not just to minimise the loss function, but to guarantee that the neural networks do not overfit against the relatively small number of available training data.

In general, most runs can be stopped after 20 epochs. A maximum of 50 epochs is allowed per training run, and it can be seen that the loss for training and validation data converges after 20 epochs (see Figures A.2 and A.5), or alternatively an overfit against the training data starts to occur (see Figures A.3 and A.4).

7.10 Performance by Projection

After completing the training of the nine neural networks, performance metrics across each of the nine projections are recorded for each projection in Table 7.2.

Projection ID / UNet Model	*Precision*	*Recall*	*F1-Score*
UNet Model 1	0.756	0.843	0.797
UNet Model 2	0.820	0.945	0.878
UNet Model 3	0.796	0.662	0.723
UNet Model 4	0.819	0.619	0.705
UNet Model 5	0.706	0.877	0.782
UNet Model 6	0.788	0.639	0.706
UNet Model 7	0.788	0.720	0.752
UNet Model 8	0.706	0.862	0.776
UNet Model 9	0.843	0.566	0.677
Overall	0.771	0.740	0.755

Table 7.2: Performance by U-Net Model 1 to 9

The nine neural networks (one for each projection) produce relatively comparable results. It should be noted that Model 9 produces slightly worse numbers for precision, recall and F1 Score. Those numbers are easily improved by choosing the model of a separate run for U-Net 9. However, it has been decided to keep those figures to demonstrate the robustness of the overall system against a model that includes a neural network of lesser quality.

7.11 Performance by MRA Data Set

Performance across each of the nine MRA data sets is provided in Table 7.3.

All nine MRA scans produce similar results. It is remarkable that the result for the validation record MNI-0656 is in similar range as for the other eight data sets that are used for the training. This indicates that the process does not overfit and generalises sufficiently against manually segmented MRA scans.

MRA ID	*Ground Truth*	Purpose	*Precision*	*Recall*	*F1-Score*
MNI 0590	Frangi	Training	0.772	0.648	0.705
MNI 0591	Frangi	Training	0.773	0.776	0.774
MNI 0592	Frangi	Training	0.767	0.720	0.743
MNI 0640	Frangi	Training	0.731	0.771	0.751
MNI 0643	Frangi	Training	0.770	0.774	0.772
MNI 0648	Frangi	Training	0.770	0.776	0.772
MNI 0656	Manual	Validation	0.850	0.650	0.736
MNI 0657	Frangi	Training	0.756	0.777	0.766
MNI 0664	Frangi	Training	0.763	0.769	0.766
Overall	-	-	0.771	0.740	0.755

Table 7.3: Performance of 9 U-Net Models by MRA Record

7.12 2D Image Segmentations with Overlays

The results for the 9 U-Net segmentations of record MNI-0656 are provided in Figure B.7. In this Figure, yellow areas in the overlay image are classified correctly, green areas highlight blood vessel regions that have been missed by the segmentation process, and red areas highlight sections that are wrongly classified as blood vessels by the U-Nets.

7.13 Frangi vs. Manual

Figure B.9 illustrates that the blood vessels, produced by the Frangi method, are generally thinner than those in the two manually segmented ground truth data.

As a result, the neural networks produce blood vessel streams that are thinner than those in the manually segmented image. The neural networks learn by using the relatively thin blood vessel segments in the training data, and apply the results to the validation record MNI-0656, which is manually segmented.

The fact that the blood vessel segments in the image projections of MNI-0656 are generally too thin does not matter for the overall process. It is

important for the process to trace the blood vessel structures correctly in the image projections; the final thickness of the blood vessels will be adjusted during 3D image post processing.

7.14 Runtimes

Training of a single neural network with 50 epochs and the configuration settings in Table 7.1, takes between 10 to 15 minutes.

The entire nine neural networks can be trained in approx. 2 to 3 hours. Code to start the training process is provided in the accompanying Jupyter notebook files. The run is not deterministic, as some internal initialisations are set by Tensorflow (He et al. (2015) Initialisation); those settings cannot be controlled by the program in random seeds.

However, the model coefficients (weights) of the nine neural networks are provided in GitHub with the source files. By loading those weights (without running the training process), the reader is able to reproduce the exact performance figures that are presented in this thesis.

7.15 Summary

In Phase II of the project, nine different U-Nets, one for each image projection, are trained. The U-Nets deliver reasonable and relatively consistent results across all projections and test records.

It is critically important that the trained U-Nets can correctly trace blood vessel structures in the 2D image projections. High values for precision, recall and F1-Score are not important at this stage of the process. Figure B.7 illustrates that blood vessel structures are relatively well traced by the U-Nets for each of the nine 2D image projections of record MNI-0656.

The final U-Net models predict blood vessel structures that are generally too thin, as they are trained against ground truth data of the Frangi method. Manually segmented data generally have thicker blood vessel streams than those that are semi-automatically created via the Frangi method.

The manually segmented ground truth data of record MNI-0656 has thicker blood vessel structures and is just used for the validation of the trained models.

The proposed method is relatively robust; shortcomings in the segmentation process of a single U-Net version can be compensated for by the remaining neural networks.

Chapter 8

Phase III - End-to-End Segmentation Process

The required machinery is now available to build an end-to-end blood vessel segmentation process. In Phase III, all components developed during Phases I and II are used to configure the final model. This includes image projections, 2D image segmentations via U-Nets, 3D reconstructions and image post-processing steps.

The results are evaluated and hyper parameters are tuned for record MNI-0656 to produce results with optimised performance metrics against the manually segmented record.

8.1 Training and Validation

Record MNI-0656 is used for evaluation purposes. The other eight training records are used to establish aggregated performance metrics across all nine MRA data sets.

The final step of the dilution process is just optimised against the manually segmented record MNI-0656, as the blood vessel segments of ground truth data - created by Frangi - are generally too thin. See Section 7.13 for more details.

8.2 Image Segmentation

The image segmentation is applied exactly as described in Phase I for the Proof of Concept. The only difference is that the program now uses the nine 2D segmentation maps produced by the U-Nets in Phase II, rather than the 2D projections of the ground truth data.

8.3 Raw Segmentation Map

The process creates a raw segmentation map, which assigns each voxel in 3D a value between 0 and 9. As in Phase I of the project, performance metrics are now calculated for the data in the raw segmentation map, when parameter N-HITS has values from 1 to 9.

In summation, all voxels with values less than N-HITS are assigned to the background. The remaining voxels are considered to be part of the cerebral blood vessel structure.

The performance metrics for the possible values of N-HITS are depicted in Figure 8.1. Contrary to the results in Figure 6.1 in the Proof of Concept, this time the F1-Score decreases as N-HITS increases.

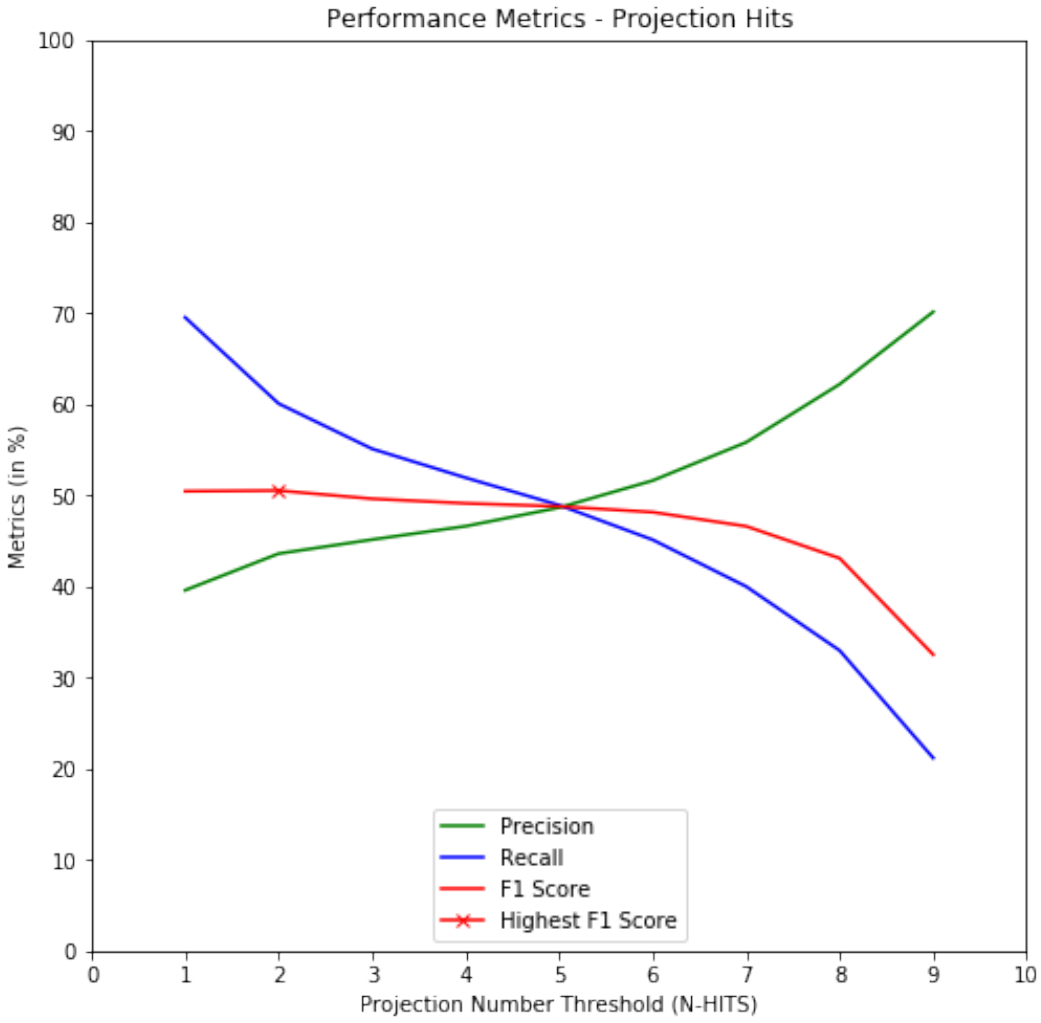


Figure 8.1: Parameter NHITS against F1-Score

As the N-HITS threshold increases, the precision increases and the recall decreases. The highest F1-Score is achieved with N-HITS=2. However, a higher value for precision is preferred, and it has been decided to work with

N-HITS=5. The F1-Score is relatively constant in the range from 1 to 5. After N-HITS=7, the F1-Score starts to decrease sharply.

The final decision to select N-HITS=5 was not solely based on the results in Figure 8.1; also considered were the 3D overlays of the prediction map with the ground truth for the values N-HITS=1 to 9. The results are provided in Figures B.10 to B.12. It can be seen that best balance between recall (minimising green areas in overlay) and precision (minimising red areas in overlay) are achieved when N-HITS has values between 4 and 6.

Going forward, N-HITS=5 is selected. However, similar results could be achieved when working with N-HITS=4 or 6.

At this stage of the process, a high precision is more important than a high recall. Additional pixels are added to the blood vessel segments during post-processing in a dilution step, improving the recall score.

The setting of the value for N-HITS is a balancing exercise, and the three performance metrics intersect somewhere at N-HITS=5 (all three metrics have values around 0.5).

It appears, should it be possible to improve the quality of the 2D image prediction maps (undertaken during Phase II), that the value for N-HITS should be increased. When 2D segmentation results are close to perfect, then the value for N-HITS should be set to 9 (as established in Section 6.3).

8.4 K-Neighbours

In the first post processing step, any holes in the segmentation map are filled. This first post processing step is specified in Section 6.4.1 and works with the configurable hyper parameter K.

The optimal threshold K for the number of neighbours in the raw segmentation map is determined by calculating the same graph that has been produced in Figure 6.3.

Figure 8.2 illustrates that the value of K=9 produces an optimised F1-Score for this model.

It appears that the hyper parameter for the K-Neighbours algorithm is only marginally influenced by the quality of the 2D segmentation process. It can be left unchanged.

The selection of K=9 is based on the results in Figure 8.2. The accompanying 3D overlay maps in Figures B.13 and B.14 illustrate that K=9 is a good choice.

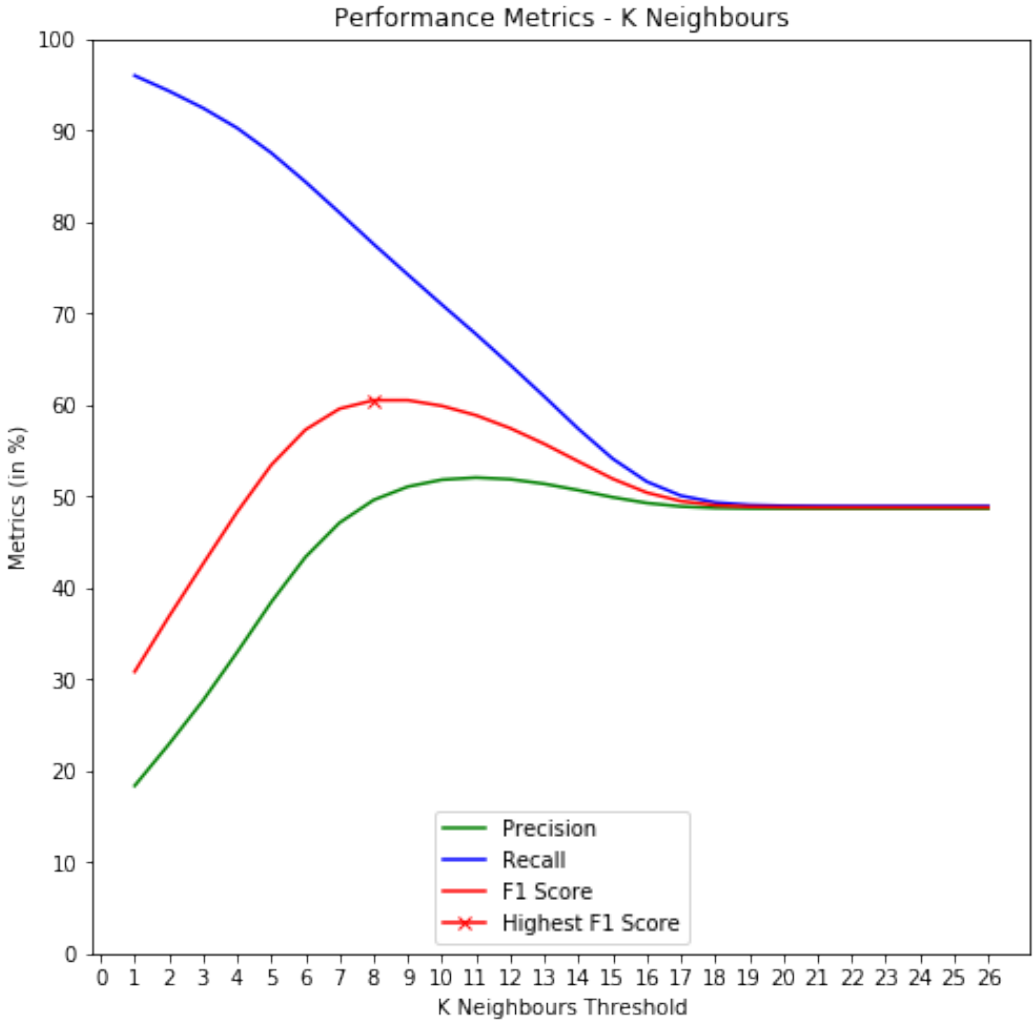


Figure 8.2: K-Neighbours against F1-Score

8.5 Dilution

In the final step, a dilution process is applied. This step is specified in Section 6.4.2 and requires the configuration of the hyper parameter T.

The parameter T is the threshold value for the signal strength of the MRI scanner. Only voxels with intensity above this threshold are considered in the image dilution process. Its optimal value is determined in Figure 8.3.

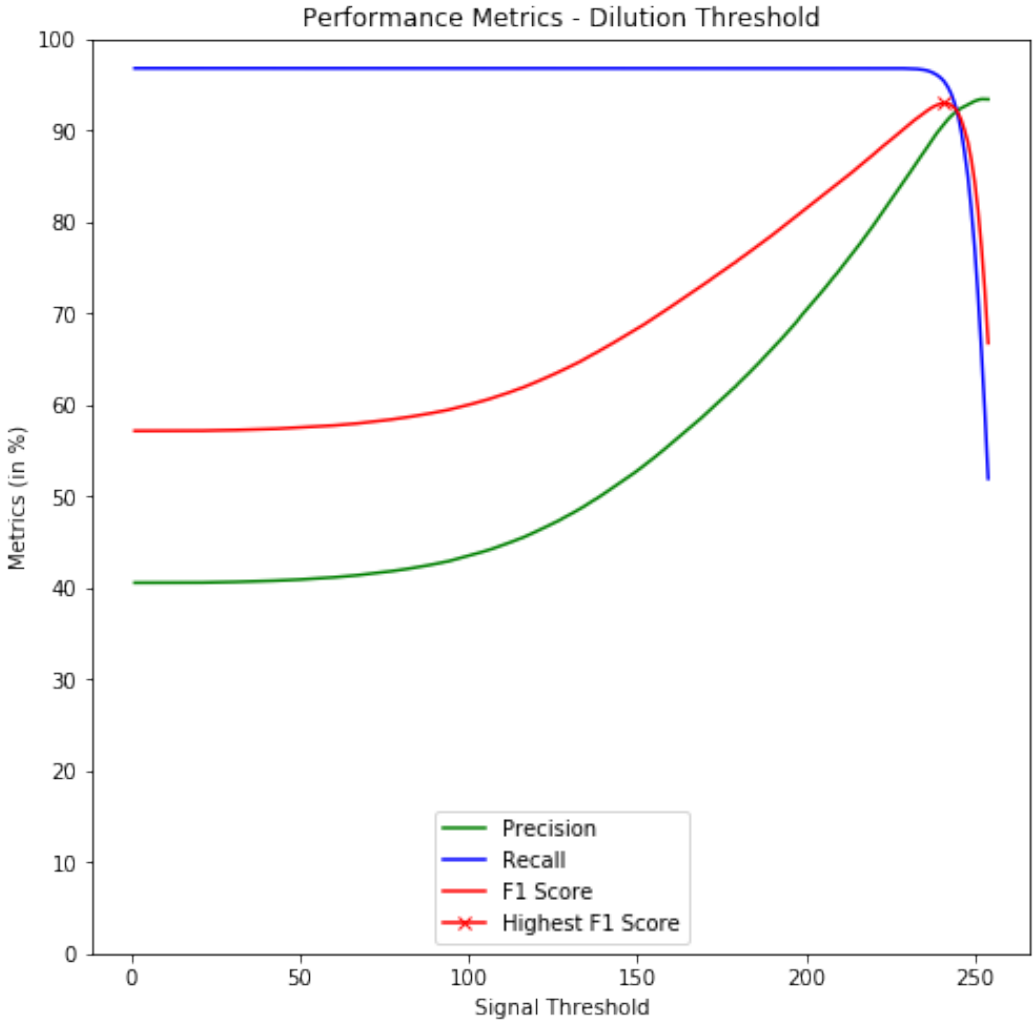


Figure 8.3: Dilution Threshold against F1-Score

It appears that the hyper parameter T for the dilution process is only marginally influenced by the quality of the 2D segmentation process. The value 241 optimises the F1-Score, and this is the same value that had been established during the Proof of Concept in Figure 6.4.

The selection of the parameter T=241 is based on the results in Figure 8.3; the accompanying 3D overlay maps in Figure B.15 illustrate that T=241 is a good choice.

8.6 Performance Metrics

The final performance metrics for record MNI-0656 are depicted in Tables 8.1 and 8.2. The accompanying segmentation map with overlays of ground truth data is provided in Figure 8.4.

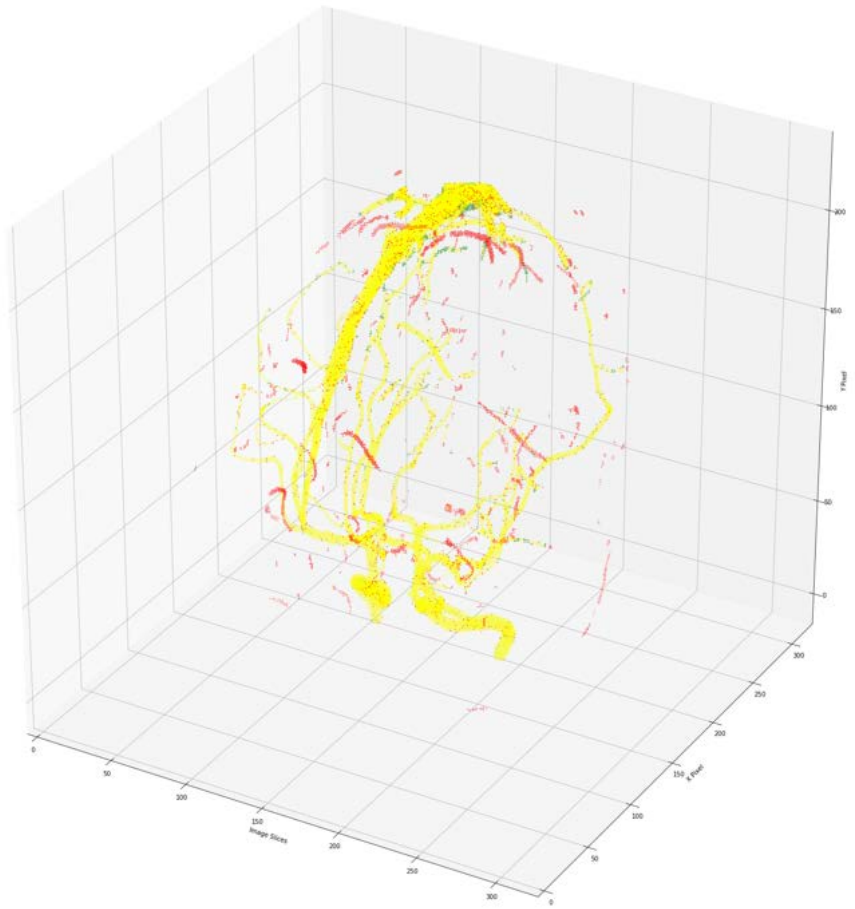


Figure 8.4: MNI-0656: 3D Segmentation Overlay (Using Final Model)

GT \ Pred.	*Positive*	*Negative*	*Totals*
Positive	**44236**	**4505**	48741
Negative	**2153**	**25241906**	25244059
Totals	46389	25246411	25292800

Table 8.1: Confusion Matrix for MNI-0656 - Final Segmentation Result

	Precision	*Recall*	*F1-Score*
MNI-0656	0.908	0.954	0.930

Table 8.2: Performance Metrics for MNI-0656 - Final Segmentation Result

Performance metrics of precision, recall and F1-Score are all above 0.9 and Figure 8.4 illustrates that the segmentation process produces excellent results. Only a few blood vessel segments are wrongly classified (red colour), and very few blood vessel cells have been missed by the process (green colour).

Some of the minor red areas in the 3D Overlay may actually be blood vessel cells that are just missed by the manual segmentation process. This requires review by a radiologist.

8.7 Performance against Ground Truth Projections

It is an interesting question to investigate how well the process performs, if perfect ground truth data are used in the segmentation.

Table 8.3 illustrates how well the segmentation process works, if the U-Net segmentation step predicts (theoretically) a perfect segmentation map. Performance data are slightly lower for N-HITS=5, when compared with N-HITS=9.

The data for N-HITS=9 are from Table 6.2.

N-HITS	*Precision*	*Recall*	*F1-Score*
N-HITS=5	0.954	0.984	0.969
N-HITS=9	0.975	0.984	0.980

Table 8.3: MNI-0656 3D Performance against GT with N-HITS=5 and 9

As the lower precision score for N-HITS=5 in Table 8.3 indicates, some wrongly classified blood vessel regions are now included in the predicted segmentation map, even though the 2D segmentations are perfect.

By lowering the N-HITS threshold from 9 to 5, some inaccuracies are introduced into the 3D-2D decomposition and reconstruction processes, that cannot be remedied even by perfect segmentation maps for the 2D image projections. The recall is not affected by this change.

Consequently, the focus should be on improving the image segmentation process for the 2D image projections (currently undertaken by U-Nets). Once the quality of the 2D segmentation process from Phase II is improved, it is possible to increase the value of N-HITS in the final model. This in turn will improve the overall quality of the final 3D prediction map.

8.8 Workflow Overview

Figure 8.5 depicts the architecture of the overall image segmentation process. It classifies all voxels in a 3D source scan into background or blood vessel regions. Figure 8.5 encapsulates the entire imaging process in a single Workflow diagram.

A key element of this new approach is that the 3D source image is not used for the segmentation of the 2D image projections. Only during post-processing, when the nine 2D segmentation processes are completed, the information in the 3D MRA Source scan is used again, to build up the skeleton of a raw 3D segmentation map. In a post-processing step, the raw segmentation map is then finalised by filling up holes and applying image dilution.

The nine U-Nets complete the 2D segmentation process on the image projections without further input from the MRA source scan.

This concludes Phase III, the training and tuning of the blood vessel segmentation process for cerebral MRA scans. In Phase IV, the workflow process in Figure 8.5 is applied to an unseen cerebral MRA scan to assess the generalisation capabilities of this new method.

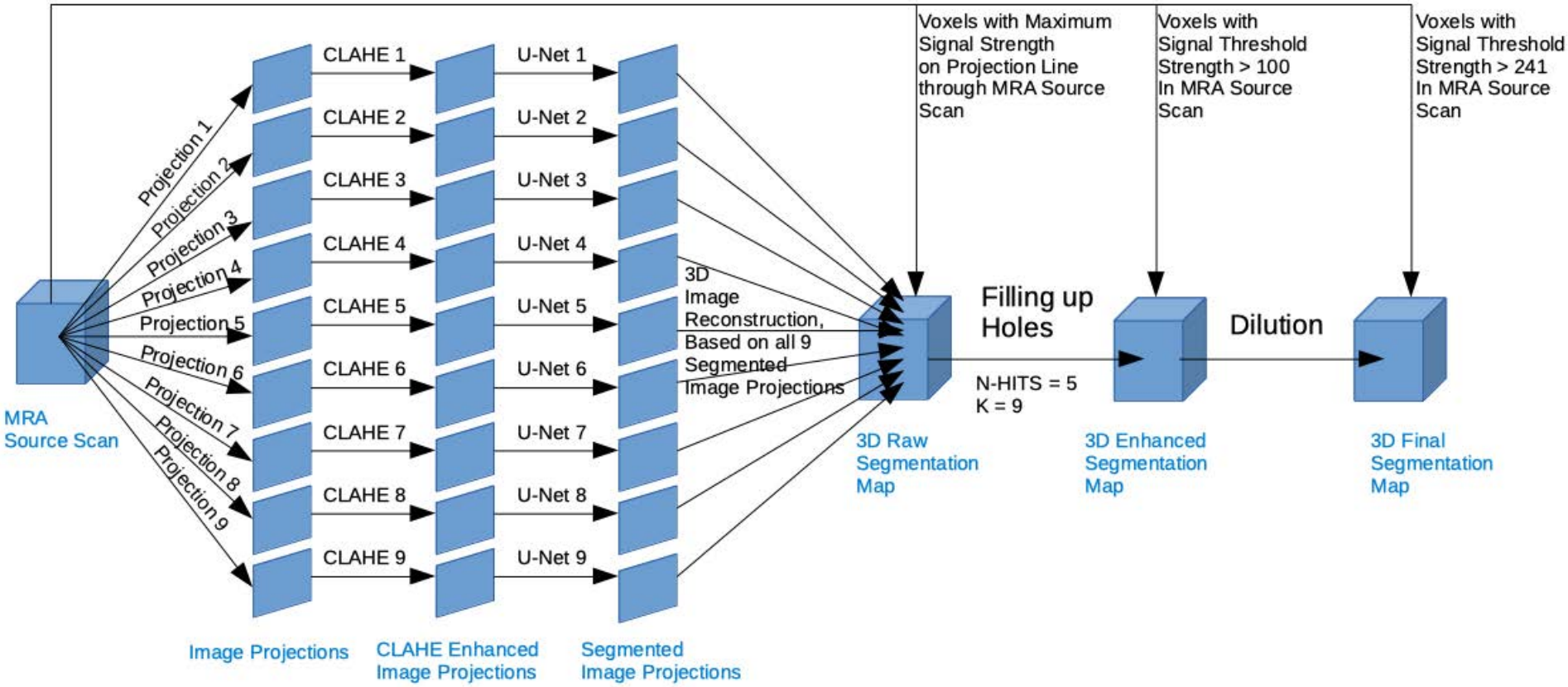


Figure 8.5: High-Level Architecture Diagram for Blood Vessel Segmentation

8.9 Summary

Phase III undertakes the end-to-end tuning of the blood vessel segmentation process for record MNI-0656. A 3D overlay of the final prediction map for record MNI-0656 against its ground truth data is provided in Figure 8.4. The final imaging process for this manually segmented MRA record produces an excellent result, with an F1-Score of 0.93.

To follow are the hyper parameter settings for the final model, and compared with the values, chosen for the proof of concept in Phase I:

- N-HITS = 5 (N-HITS = 9 in Phase I for Proof-of-Concept)
- K = 9 (unchanged)
- T = 241 (unchanged)

There appears to be a correlation between the optimal value for hyper parameter N-HITS and the quality of the 2D image segmentation process from Phase II. The optimal value for N-HITS increases when the performance metrics for the 2D segmentation results improve.

Chapter 9

Phase IV - Final Test against Unseen MRA Scan

Phase IV is the pivotal moment of this project. By investigating the performance of the segmentation process against a formerly unseen MRA scan, it is possible to assess whether the segmentation process generalises and if it can still produce reasonable results against new and unseen data sets in a production environment.

Record MNI-0663 has been manually segmented and the source data are produced by the same MRI scanner, the Siemens Sonata model. No adjustments to the hyper parameters are needed.

9.1 U-Net Performance Metrics

Table 9.1 compares the U-Net segmentation results of record MNI-0663 with those of record MNI-0656. In total, 9 projections for each MRA scan are segmented, and the 2D visual overlays are presented in Figure B.8. In comparison to the 2D segmentation results of record MNI-0656 (see Figure B.7), the segmentations for MNI-0663 contain slightly more green areas. This is also reflected in the lower recall score for MNI-0663 in Table 9.1. However, MNI-0663 has higher precision and the F1 Scores are nearly identical.

Overall, both 2D projection maps produce relatively comparable results, which indicates the trained U-Nets should produce reasonable 2D segmentation results for unseen cerebral MRA scans of the Siemens Sonata model.

	Purpose	*Precision*	*Recall*	*F1-Score*
MNI-0656	Validation	0.850	0.649	0.736
MNI-0663	Final Test	0.894	0.605	0.722

Table 9.1: Performance Metrics of Nine 2D Projections (U-Nets) - MNI-0663 vs MNI-0656

9.2 Final 3D Performance Metrics

Finally, record MNI-0663 is segmented using the workflow process, as depicted in Figure 8.5. Hyper parameter settings from Phase III are applied and the nine trained U-Net models with the results from Section 9.1 are used to complete the 3D segmentation process for MNI-0663.

Table 9.2 provides the overall performance metrics for the complete 3D segmentation process against record MNI-0663, and compares the results with those of record MNI-0656.

	Purpose	*Precision*	*Recall*	*F1-Score*
MNI-0656	Validation and Tuning	0.908	0.954	0.930
MNI-0663	Final Test	0.942	0.822	0.878

Table 9.2: Final 3D Performance Metrics - MNI-0663 vs MNI-0656

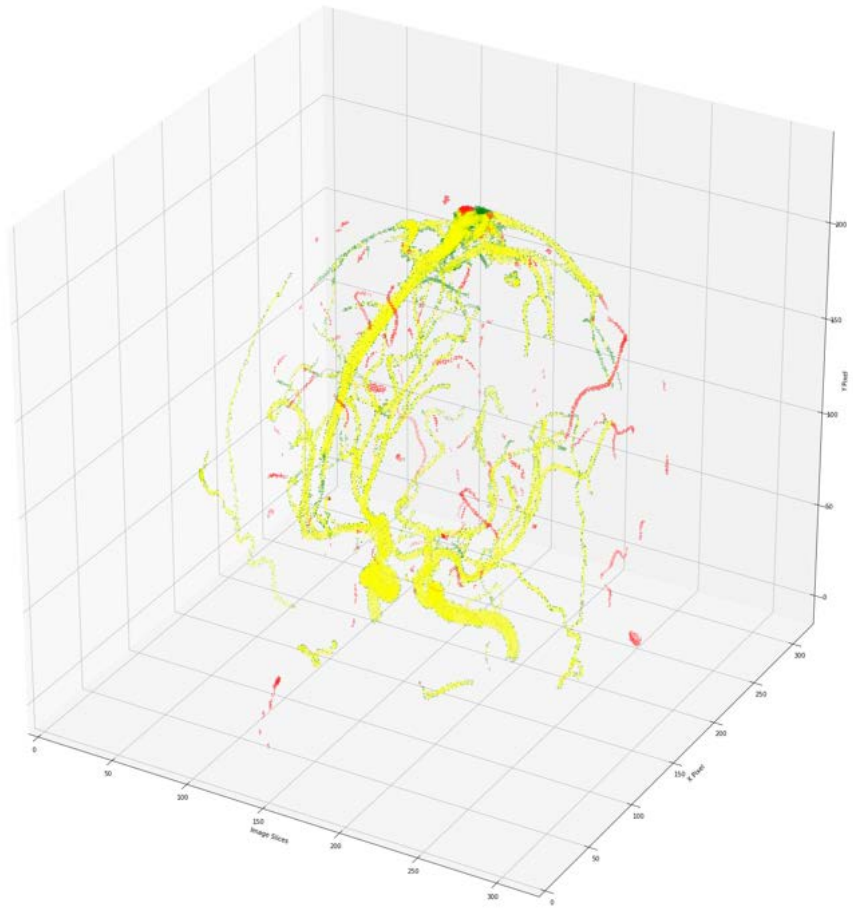


Figure 9.1: MNI-0663: 3D Segmentation Overlay (Using Final Model)

The results for the two records are in comparable ranges. Precision for the final test record MNI-0663 is even higher than for the validation record MNI-0656. However, recall in the final test record is significantly lower, and the F1-Score is slightly lower. This might indicate a slight overfit of the model against the validation record.

The 3D overlay of the prediction map for record MNI-0663 against its manually segmented ground truth data is provided in Figure 9.1. This Figure illustrates that the proposed model provides a useful segmentation of an unseen MRA scan. Some green patches (reflected in lower recall) can be found in the overlay for record MNI-0663.

A complete comparison of the two segmentation maps for the two records - MNI 0656 and MNI 0663 - with 3D overlays and ground truth data, is provided in Figure B.16. This Figure illustrates that segmentation maps for both manually segmented records are of similar quality. The misclassified areas in both overlay maps (red / green) are comparable, though slightly more green patches can be found for the overlay of record MNI-0663.

The performance metrics for the two records are in similar range as those reported in Shifeng Zhao (2015)'s paper. Her approach also involves Maximum Intensity Projections (MIP's). However, in her paper, image projections are segmented using morphological features rather than deep learning.

9.3 Amplification of Recall in Final Score

Table 9.1 records the metrics for the 2D segmentation process, whereas Table 9.2 reports the final performance in 3D space after completion of the full workflow process, as depicted in Figure 8.5.

At first, it can be observed that recall scores for the 2D segmentation results in Table 9.1 are not very high (both below 0.65). However, after construction of the 3D segmentation map, the recall scores are both lifted significantly above 0.8. This indicates that the post-processing step is critically important for the creation of a high-quality segmentation map in 3D.

Relatively small discrepancies in recall for the 2D segmentation (0.605 vs 0.649) are amplified in the final 3D segmentation map (0.822 vs 0.954). Even though the difference in the recall score (0.044) appears to be small, it indicates that vital blood vessel regions are missed by the U-Nets when segmenting the 2D image projections for the final test record. Those misses are amplified when building up the full 3D blood vessel streams in the post processing step.

It is important that the 2D segmentation process can map out a full skeleton of all cerebral blood vessel components. The absolute value for the recall is not as important for the 2D segmentation process.

The blood vessel skeleton for the 2D image projections has been better mapped out for the validation record, and therefore its higher final recall score (0.954 vs 0.822). Any misses of blood vessel regions in the 2D segmentation process are amplified in the final 3D segmentation map, as illustrated in the outcome for the final test record MNI-0663.

9.4 Runtime Performance

Table 9.3 provides details about the runtime performance of the process, as measured on an Apple MacBook Air, 8GB RAM.

No	*Step*	*Description*	*Time*
1	Loading Source Images	Reading the 247 image slices of the MRA scan from hard disk into memory as consolidated 3D image	1 sec
2	2D Image Projections	Creating 9 2D image projections and applying CLAHE contrast enhancement to each of the 9 image projections	2 sec
3	2D Image Segmentations	Running 9 different UNet segmentation processes individually against each 2D image projection	5 sec
5	3D Image Reconstruction	Reconstructing 3D raw segmentation map from the 9 2D image segmentations	8 sec
6	3D Image Post-Processing	Applying K-Neighbour algorithm and 3D image dilution	3 sec
Total			**19 sec**

Table 9.3: Runtime for the Processing of a Single MRA Scan

Currently, an entire MRA scan, comprising 247 quadratic image slices of length 320px, can be processed in under 20 seconds on mid-to-low level Computer hardware.

Runtimes in Table 9.3 exclude the warm-up time when first starting the program. Warm-up includes loading of weight coefficients for the nine neural networks from hard disk and initialisation of classes for the processing of U-Nets. The initial startup-time takes between 30-50 seconds on commodity hardware.

Significant time has been spent to improve the runtime performance for the image projections. More details about the concepts utilised are provided in Section 4.3, and code details are included in the Appendix Sections A.3 and A.4.

9.5 Robustness

Multiple U-Net versions can be trained, just by repeatedly kicking off the training process for the neural networks in the accompanying Jupyter Notebook file. An entire training run for all nine neural networks completes in about 2.5 hours on an Apple MacBook Air with 8GB RAM, without utilising the GPU of the Computer.

Overall performance of the neural networks is in very similar range for all runs. However, occasionally the odd outlier in one of the nine training runs can occur.

It is notable that the overall results in the 3D segmentation maps are only marginally impacted by the quality of a single U-Net model. Poor performance of a single model against one of the image projections can be alleviated by the other 8 U-Nets. By taking into consideration a balanced score across all 9 2D image projections, reliable results with similar overall performance metrics can be achieved for all neural network versions that are trained and tested in this project.

Readers interested in exploring further, may wish to train their own set of individual UNets, utilising the provided Jupyter Notebook file, and compare the results with the figures presented in this document.

9.6 Overview Hyper Parameters

Table 9.4 contains the comprehensive list of hyper parameters that are configured for the segmentation process. Values can be adjusted in an associated initialisation file, for instance, when working with other MRI models and differing 3D image resolution settings.

The hyper parameters used for the training of the neural networks are described in Table 7.1. Here, it is assumed that a set of 9 neural networks is already trained. The U-Net configuration files used to present the figures in this thesis are provided in GitHub together with the source code files.

Parameter	*Setting*	*Purpose*
NoSlices	247	Number of image slices per scan. This parameter is device-specific.
ImgWidth	320	Image width of 2D image slice. This parameter is device-specific.
ImgHeight	320	Image height of 2D image slice. This parameter is device-specific.
PxlThreshold	100	Threshold for signal strength of voxel to be considered as candidate for blood vessel. This parameter is device-specific.
ModelPath	data/models	Path to the model coefficients of the nine neural networks; may need to be adjusted depending on the U-Net models that should be used. Different U-Net coefficients may be trained and used for different MRI scanners.
Threshold	0.5	Threshold used to define whether pixel is assigned blood vessel or not. Each pixel is assigned a value between 0 and 1 by the U-Net. Both options are assigned equal probability. There should be no need to change this setting.
MinHits	5	This parameter is called N-HITS in this thesis. It needs to be set, based on the quality of the U-Net segmentations. If better segmentation results for the 2D image projections can be achieved, then the value of this parameter should be increased.
MinNeighbours	9	Parameter K-Neighbours has been optimised against the available source data. It appears to be largely uneffected by the quality of the 2D segmentation results.
DilutionThreshold	241	Dilution threshold T has been optimised against the available source data. It may need adjustments if testing with a different MRI model.

Table 9.4: Hyper Parameters in Segmentation Model

9.7 Use of Manually Segmented Ground Truth Data

With the sparse use of the two manually segmented MRA scans, the performance figures reported in this thesis are on the lower end and could be further

improved, simply by making more use of both manually segmented data sets during training and validation.

For example, using one or a combination of the following approaches improves the overall performance scores (e.g. F1-Scores can be above 0.9):

- Both manually segmented records MNI-0656 and MNI-0663 are included in the training and hyper paramter optimisation process.
- MNI-0656 and MNI-0663 are both used during training of the U-Nets. For support of generalisation, cross validation is used instead. After each epoch, training and validation data sets are split randomly.
- Semi-automatic Frangi segmentations (in total 8 records) are excluded completely. Instead, image augmentations of the two manually segmented records are created. Image augmentation is a common technique to artificially increase the number of training data when only a limited number of source / ground truth data is available.

However, after some consideration, the aforementioned methods have not been used for this project. They can produce overly optimistic performance numbers that disguise the fact that intrinsic image patterns across the two manually segmented source scans and image augmentations are learnt by the neural network. This will lead to poorer results when generalising against unseen data sets.

The performance numbers in Table 9.2 are conservative in relation to record MNI-0663. Recall, precision and F1-Score for this record can be improved once MNI-0656 and MNI-0663 are both used for training, validation and hyper parameter tuning of a final model.

9.8 Summary

Phase IV concludes this project. The developed model - with the architecture as depicted in Figure 8.5 - is tested against the formerly unseen manually segmented MRA scan MNI-0663. A 3D overlay of the prediction map against its ground truth is provided in Figure 9.1.

The predicted segmentation map produces good performance metrics against the ground truth. Table 9.2 compares the results with those that are achieved during Phase III, when training and validating the model. This suggests that the model does not or only marginally overfit against the training data. Side-by-side 3D overlays for both records - MNI-0656 and MNI-0663 - are provided in Figure B.16.

Better performance metrics are achievable, by using all 10 MRA scans for training and validation of a final model.

Excluding warm-up times for the initialisation and starting of the software, a 3D MRA scan of the Siemens Sonata model can be segmented in 20 seconds on commodity hardware.

Chapter 10

Conclusion and Future Directions

10.1 Major Contributions

10.1.1 Research Conclusions

Referring to the research objectives in Section 1.2, it can be concluded that all expectations are fully met by this research work:

- The Computer program creates reproducible segmentation maps of high quality against the available ground truth data.

- The new solution can complete the segmentation of cerebral blood vessels in a computationally fast and efficient way.

- Python is a mainstream programming language and should work in most operating systems. The output folder in the file system can be used as interface to transport the produced 3D image segmentations into the technical environment of a radiologist.

- The software is configurable and able to work with different MRI sanner models, for example, varying 3D image dimensions and resolutions. However, this has not yet been tested and verified in practice.

In summary, the following key contributions are being offered by this research:

10.1.2 Dimensionality Reduction - Framework

The proposed new model, using parallel image projections, offers a framework for the segmentation of 3D images in 2D space. It produces 2D image projections prior to undertaking the image segmentation task. This framework can be used for the 3D segmentation of objects in unrelated fields.

Also, methods other than U-Nets can be used for the segmentation of images in 2D. For example, the Frangi method could be plugged into this framework instead of U-Nets to segment the blood vessels in 2D space.

10.1.3 Simplified Evaluation of 3D Segmentation Results in 2D Space

Overall, it is easier to assess the quality of segmentation results in 2D space. Evaluation of image segmenations in 3D requires specialised software.

This advantage should not be underestimated. Evaluations of 3D image segmentations sometimes rely too heavily just on the performance metrics, simply because it is challenging to graph the overlays in 3D.

Image projections simplify the process of assessing the quality of a 3D segmentation by looking at 2D segmentation results of the related image projections.

10.1.4 New Method for Creation of Ground Truth Data in 3D Space

Manual creation of segmentation maps is a tedious exercise for images in 3D space. Consequently, the nine image projections themselves could be used for the manual segmentation of cerebral MRA scans. Rather than using complex 3D software for the cumbersome manual segmentation of scatterplots in 3D, the nine 2D image projections of a 3D MRA scan could be manually segmented by a radiologist, using methods discussed in this thesis in Chapter 6. The manually segmented 2D projections can then be used to reconstruct a 3D ground truth map. This approach could simplify the manual creation of new high-quality ground truth data in 3D space.

10.1.5 High-Quality Segmentation Results

The proposed method achieves excellent performance metrics against the ground truth of the two manually segmented records. Overlay images of ground truth and prediction maps confirm that the proposed process does not miss major blood vessel structures nor does it add many non-blood vessel elements.

Precision and recall are above 0.9 for manually segmented MRA scans that are used during network training/validation and hyper-parameter tuning.

10.1.6 Excellent Runtime Performances

The proposed segmentation approach achieves excellent runtime performance. It can process a single MRA scan in about 20 seconds on commodity hardware.

Most other products, especially those that process the data in 3D, require runtimes in the order of minutes for the processing of a single MRA scan Shifeng Zhao (2015).

10.1.7 Robustness

The process is robust against minor deficiencies in the 2D segmentations of the U-Nets. By using a combined score across 9 2D segmentations, some inaccuracies can be alleviated.

The process does not rely on the segmentation of a single segmentation run. It averages the results across 9 runs and helps reduce overall variance of performance metrics against unseen data.

10.2 Limitations and Future Work

10.2.1 High Density Data

It should be noted that this method is not suitable for segmentation of objects that have high density in 3D space. The method works well for this project, as less than 1 per cent of the 3D space in a cerebral MRA scan is occupied by blood vessel cells.

10.2.2 Loss of Information

Some information of the data in 3D space is inadvertently lost when using the MIP method for the 2D decomposition. This information can not be recovered when reconstructing the 3D segmentation map and is discussed in Sections 6.4 and 6.6. If very high performance metrics are required, then this method may not be appropriate.

10.2.3 Need for Image Post Processing

Image projections can just produce a raw segmentation map, as illustrated in Figure 4.3. It is essentially a skeleton of the blood vessel structures that need to be covered by the final segmentation map. Some sophisticated post-processing, such as image diluation, is required to obtain a high-quality final image segmentation. In its own right, the image projection method cannot produce final high-quality segmentation results. It may not always be clear how a post-processing method can improve on the data in the raw segmentation map.

10.2.4 Future Work

Discussed are some of the options to further improve the method described in this thesis. Some of those options are already mentioned in this document:

- Vertex projections could be included. Further information is provided in Section 4.3.3. This increases the number from 9 to 13 image projections. The method may further improve overall performance metrics without any amendments to the 2D image segmentation process via U-Nets.
- In the post-processing step (Section 6.4), the relationship between K and the pixel threshold could be further investigated. Step 1 (Section 6.4.1) assumes a high value for K (number of neighbouring blood vessel voxels) and the low value 100 as a threshold for the signal strength. Step 2 (Section 6.4.2) assumes the low value of 1 for K, but a high value of 241 for the signal strength. By modelling the value for the signal strength as a function of K, the two post-processing steps could be collapsed into a single step. For K=1, the value 241 could be selected as a threshold for the signal strength, and as K increases, this threshold value should decrease until it reaches 100 at K=9.
- Rather than using N-HITS as parameter to control the precision of the segmentation process, the pixel probability value (currently set to 0.5) could be utilised to manage this instead. Initial tests indicate that leaving N-HITS =9 and lowering the pixel probability value instead, could produce excellent results.
- The projection lines could be built to just run halfway through the 3D image. This would allow the inclusion of projection images from opposing directions and could potentially improve overall quality of the 3D segmentation results.
- The optimal value of N-HITS depends on the quality of the 2D image segmentation process. It may be investigated how improvements in the 2D image segmentation process influence the selection of an optimised value for the N-HITS parameter. The results of this investigation help quantify the effectiveness of 2D image segmentation improvements on the overall 3D segmentation result.

Improving 2D Segmentation Results

Focus should be on improving the results in the 2D image segmentations. In the absence of sufficient volumes with ground truth data, unsupervised classification techniques should be considered. This could include the F. Frangi et al. (2000) method and other techniques that are used for segmentation of blood vessels in the human eye. An overview about those methods is provided by Fraz et al. (2012) and in the Literature Review in Chapter 3. One of those methods could be considered to replace the neural networks in this thesis.

Piloting Solution

A pilot of the solution is ready to be configured for deployment into a production environment. Produced results could be reviewed by a radiologist, and based on feedback, the process could be further enhanced.

Transfer of the Solution to other Fields

This method is not restricted to segmentation of cerebral blood vessels. It can also be trained and retuned for the segmentation of blood vessels in MRI / MRA scans of other human body parts; it may produce results of similar quality.

10.3 Discussions

Throughout the work on this project, it became clear that preconceived ideas and methods needed to be adjusted, depending on available hardware, time, budget and quality of ground truth data.

10.3.1 Neural Networks

The use of neural networks is still a key component of this project, but not the heart of the entire process. Image projections with dimensionality reduction are the glue that hold the entire process together; the U-Nets could be easily replaced by another blood vessel segmentation method in 2D.

For this project, the use of image projections, paired with an unsupervised segmentation algorithm across the 2D image projections, could be a viable alternative. By analysing the morphological features in the MRA data, there would be less reliance on the availability of ground truth data.

10.3.2 Performance Metrics

Performance metrics have their value. However it is easy to make the mistake of merely focusing on performance metrics to assess the quality of a segmentation result. This may lead to wrong or misleading conclusions.

For example, a segmentation map may produce a relatively low recall, but has well coverage of all the blood vessel areas in the ground truth image. A second segmentation may produce higher values for precision, recall and F1-Score. Still, the overall result of the second segmentation may be less useful if entire blood cell regions are missing or misclassified.

It is a pitfall to evaluate the quality of a segmentation process purely based on its performance metrics. Inspections of colour-coded segmentation maps and overlays with ground truth data are a vital element in the appraisal of image map predictions.

10.3.3 Availability of Ground Truth Data

The availability of quality ground truth data is critical, especially when working with supervised machine learning methods, such as neural networks. Unfortunately, the entire work is based on only two manually segmented MRA scans and eight additional semi-automated ground truth data. This severely hampered the usefulness of neural networks in this project.

It would be helpful, if in the future, a body of cerebral MRA scans with approved ground truth records could be made available to the public.

For example, major improvements could be achieved in the automated segmentation of retinal blood vessels (blood vessels in the human eye). Research is largely based on images in the STARE Clemson (2000) and DRIVE Utrecht (2007) databases. Both databases are available to the public and contain a large set of retinal images and corresponding ground truth data; those images are used for the training and validation of retinal blood vessel segmentation processes.

A similar initiative for cerebral MRA scans could encourage further research in the field of the automatic segmentation of blood vessels in the human brain.

10.3.4 Team Sport

This research requires the support of a medical professional, such as a radiologist. A Data Scientist cannot work in isolation. The results in this thesis need to be reviewed by medical practitioners. The cooperation with professionals in their respective field of expertise is essential to produce high quality outcomes. Data science is a team sport.

References

American Association of Neurological Surgeons AANS. Cerebrovascular disease, 2019. URL https://www.aans.org/Patients/Neurosurgical-Conditions-and-Treatments/Cerebrovascular-Disease.

Geoffrey E. Hinton Alex Krizhevsky, Ilya Sutskever. Imagenet classification with deep convolutional neural networks, 2012. URL https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

American-Cancer-Society. Tests for brain and spinal cord tumors in adults, 2019. URL https://www.cancer.org/cancer/brain-spinal-cord-tumors-adults/detection-diagnosis-staging/how-diagnosed.html.

Kristian Barlinn and Andrei V. Alexandrov. Vascular imaging in stroke: comparative analysis. *US National Library of Medicine*, pages 340–8, 2011. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3250265.

Geoff Breemer. Segmenting human cerebral blood vessels in magnetic resonance data using deep learning. *Unpublished research work, University of Southern Queensland*, 2018. URL https://github.com/GeoffBreemer/DLToolkit.

Clemson. Structured analysis of the retina (stare), 2000. URL https://www.isi.uu.nl/Research/Databases/DRIVE.

CRA-Medical-Imaging. Magnetic resonance imaging / magnetic resonance angiography, mri / mra, 2018. URL https://www.craimaging.com/procedures/mrimra.

Ro F. Frangi, W.J. Niessen, Koen Vincken, and Max A Viergever. Multiscale vessel enhancement filtering. *Med. Image Comput. Comput. Assist. Interv.*, 1496, 02 2000.

M.M. Fraz, P. Remagnino, A. Hoppe, B. Uyyanonvara, A.R. Rudnicka, C.G. Owen, and S.A. Barman. Blood vessel segmentation methodologies in retinal images – a survey. *Computer Methods and Programs in Biomedicine*, 108

(1):407 – 433, 2012. ISSN 0169-2607. doi: https://doi.org/10.1016/j.cmpb.2012.03.009. URL http://www.sciencedirect.com/science/article/pii/S0169260712000843.

R. Ghaderi, H. Hassanpour, and M. Shahiri. Retinal vessel segmentation using the 2-d morlet wavelet and neural network. pages 1251–1255, Nov 2007. doi: 10.1109/ICIAS.2007.4658584.

G.Hassan. Retinal blood vessel segmentation approach based on mathematical morphology, 2015. URL https://www.sciencedirect.com/science/article/pii/S1877050915028355.

Shantala Giraddi and Jagadeesh D. Pujari. Exudates detection with dbscan clustering and back propagation neural network, 2014. URL https://www.semanticscholar.org/paper/Exudates-Detection-with-DBSCAN-Clustering-and-Back-Giraddi-Pujari.

Gehad Hassan, Nashwa El-Bendary, Aboul Ella Hassanien, Ali Fahmy, Shoeb Abullah M., and Vaclav Snasel. Retinal blood vessel segmentation approach based on mathematical morphology. *Procedia Computer Science*, 65:612 – 622, 2015. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2015.09.005. URL http://www.sciencedirect.com/science/article/pii/S1877050915028355. International Conference on Communications, management, and Information technology (ICCMIT'2015).

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

J.Bezdek. Fcm: The fuzzy c-means clustering algorithm, 1984. URL http://www.sciencedirect.com/science/article/pii/0098300484900207.

M.Ester. A density- based algorithm for discovering clusters in large spatial databases with noise, 1996. URL https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf.

Uyen T.V. Nguyen, Alauddin Bhuiyan, Laurence A.F. Park, and Kotagiri Ramamohanarao. An effective retinal blood vessel segmentation method using multi-scale line detection. *Pattern Recognition*, 46(3):703 – 715, 2013. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2012.08.009. URL http://www.sciencedirect.com/science/article/pii/S003132031200355X.

Arthur Toga Alan Evans Peter Fox Jack Lancaster Karl Zilles Roger Woods Tomas Paus Gregory Simpson Bruce Pike Colin Holmes Louis Collins Paul Thompson David MacDonald Marco Iacoboni Thorsten Schormann Katrin Amunts Nicola Palomero-Gallagher Stefan Geyer Larry Parsons Katherine Narr Noor Kabani Georges Le Goualher Dorret Boomsma Tyrone

Cannon Ryuta Kawashima R. Kötter, John Mazziotta and Bernard Mazoyer. A probabilistic atlas and reference system for the human brain: International consortium for brain mapping (icbm), 2001. URL https://royalsocietypublishing.org/doi/abs/10.1098/rstb.2001.0915.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL http://arxiv.org/abs/1505.04597.

Yun Tian Pengfei Xu Zhongke Wu Qingqiong Denget Shifeng Zhao, Mingquan Zhou. Extraction of vessel networks based on multiview projection and phase field model, 2015. URL https://www.sciencedirect.com/science/article/pii/S1877050915028355.

Siemens. Siemens sonata, 2018. URL https://www.healthcare.siemens.com.au/magnetic-resonance-imaging.

Wen P. Ahfock T. Song, B. and Y. Li. Numeric investigation of brain tumor influence on the current distributions during transcranial direct current stimulation. ieee transactions on biomedical engineering, 2016.

Tobias Sterbak. U-net for segmenting seismic images with keras, 2018. URL https://www.depends-on-the-definition.com/unet-keras-segmenting-images.

University Medical Center Utrecht. Digital retinal images for vessel extraction (drive). 2007. URL https://www.isi.uu.nl/Research/Databases/DRIVE.

V.Grau. Improved watershed transform for medical image segmentation using prior information, 2004. URL http://citeseerx.ist.psu.edu.

Peng Ke Hyoseob Song Jungmee Hwang Yakun Chang, Cheolkon Jung. Automatic contrast-limited adaptive histogram equalization with dual gamma correction, 2018. URL https://ieeexplore.ieee.org/document/8269243.

zhixuhao. Implementation of deep learning framework – unet, using keras, 2018. URL https://github.com/zhixuhao/unet.

Z.Vrselja. Function of circle of willis, 2014. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3982101.

Appendix A

Program code

Python 3 has been selected as programming language. In general, the author has no preference for one language over another, but the reason to work with Python was largely driven by the fact that Python offers out-of-the-box an imense number of libraries with support for machine learning and image processing. For example, plenty of code with implementations of various U-Net Architectures is availabe for Python on GitHub. It appears that Python 3 is the language of choice for many Data Scientists.

All code in this section has been written from scratch and is the author's own work. The only exception is the base architecture of the U-Net model, which uses an implementation by Sterbak (2018).

In total less than 1,000 lines of Python source code had to be written for this thesis. The code and utilised MRA data sets for this thesis are made available to the public in Github at `https://github.com/rvh1/mra_blood_vessel_segmentation`

A.1 Python Libraries

The following is a complete list of Python libraries that are utilised for the implementation of the solution.

Limited use has been made of specialised Python libraries. The majority of the code is implemented in standard Python. The three key libraries for this project are Keras, Tensorflow and Numpy.

Library	*Purpose*
keras	Framework for Machine Learning Libraries, used as API for the low level calls of the Tensorflow library
tensorflow	Popular Google library, used for implementation and training of deep learning models,such as convolutional neural networks (CNN)
numpy	Key library for efficient processing of mathematical operations on multidimensional arrays; many methods, such as amax for image projections, are implemented directly in C code and run more efficiently numpy than standard Python code
math	Supporting some basic mathematical operations
cv2	Computer-Vision (CV) library, used for basic image processing tasks, such as reading/writing files and CLAHE enhancement of 2D image projections
scipy	Library with many useful functions for support of machine learning processes; used to run the 3D image dilution process very efficiently (post-processing)
logging	Library used to support logging functionality in the application
os	Support of basic OS operations, such as creating and deleting subdirectories in the debug folder
configparser	Reading configuration settings in the initialisation file MRASegmentation.ini

Table A.1: Complete List of Python Libraries Used in Project

The following libraries are not required to run the segmentation process, but are used in Jupyter Notebook for visualisation of the results.

A range of 3D visualisation tools have been tested, most of them are not practical for use in this thesis, or they are too slow to process and display several 1000 data points in reasonable timeframes.

The traditional matplotlib library does an acceptable job for this project. However, more specialised 3D software should be used for a detailed analysis of the predicted segmentation maps.

Library	*Purpose*
matplotlib	popular library for visualisation of data in graphs and plots
mplot3d	ships with matplotlib; used for visualisation of 3D scatterplots (comparing predicted segmentation maps and colour-coded overlays against the ground truth)

Table A.2: Python Graphing Libraries

A.2 Project Source Files

A.2.1 Python Program Files

The following Python source code files are required to run the MRA segmentation process.

File Name	*Purpose*
MRASegmentation.py	Main Python source file to run the whole segmentation process
MRASegmentation.ini	Vital initialisation file that contains configuration and hyper parameter settings; it needs to be set prior to starting the program (in particular directories for logging, debugging and neural networks need to be set)
ImageProjections.py	Code implements the 2D image projections and 3D MRA reconstructions
UNetModels.py	Code to train the 9 neural networks and/or execute the UNet segmentation processes
FinalSegmentationMap.py	Completing post processing steps and building up the final 3D image segmentation map
support.py	Supporting functions, such as reading/writing files, and calculation of performance metrics

Table A.3: Complete List of Python Source Files Used in Project

A.2.2 Jupyter Notebook Files

Jupyter Notebook is used as Integrated Development Environment (IDE). The following files are included in the submission and can be executed to reproduce the results and graphs that are presented in this thesis.

File Name	*Purpose*
1a - UNET Training.ipynb	Training of the nine U-Nets, one against each projection view; model parameters are stored in the models directory which is a configurable parameter in MRASegmentation.ini; Debug mode needs to be enabled for training of U-Nets
1b - UNET Evaluation.ipynb	Evaluating the results with performance metrics, reported by MRA scan and by projection view; included are a number of graphical 2D image projections, depicting overlays of predicted segmentation maps with ground truth data
2 - Model Tuning.ipynb	Tuning of the end-to-end segmentation process, with hyper parameter optimisations for N-Hits, K-Neighbours and Signal Strength threshold T; graphing of 3D segmentation maps
3 - Final Model Test.ipynb	Testing of the finalised model against record MNI-0663 (held back and only used in the final test); graphing of 3D segmentation maps
A1 - Supplements.ipynb	Supplementary tests and runs with graphical 3D outputs for all 10 MRA scans from Table 2.1 (Source vs GT vs Prediction vs Overlay); output from this notebook is not included in the document; it is useful for medical practitioners to assess the quality of the segmentation results across all 10 records

Table A.4: Complete List of Jupyter Notebook Files Used in Project

A.3 2D Parallel Image Projections

The code is optimised for runtime performance, not readability. All image projections are CLAHE enhanced, prior to feeding them into the neural networks.

The edge views have been implemented, by embedding tilted versions of the 3D MRA scan into enlarged 3D 'black boxes' and applying the computationally efficient amax operations of the numpy library. For more details about the conceptual approach see Section 4.3.2.

```
#-----------------------------------
# Author:       Rudiger von Hackewitz
# Date:         December 2018
#
# Running the 2D image projections


import numpy as np
from utils.support import create_black_image
import cv2

class MRAProjection():
    def __init__(self, ThreeDImg):
        self._3d_img = ThreeDImg

        self._d = ThreeDImg.shape[0]
        self._h = ThreeDImg.shape[1]
        self._w = ThreeDImg.shape[2]

        # build up image directory with image projections and CLAHE enhancements
        self._img_projs = dict()


    # CLAHE-enhance the image projections
    def do_clahe(self, img):
        clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
        output = clahe.apply(img.astype(np.uint8))
        return output.astype(np.uint16)
```

```
    def run_projections(self):

        # 1st Projection
        smax1  = np.amax(self._3d_img, axis=0)
        out1   = self.do_clahe(smax1)
        self._img_projs[1] = [smax1, out1]

        # 2nd Projection
        smax2  = np.amax(self._3d_img, axis=1)
        out2   = self.do_clahe(smax2)
        self._img_projs[2] = [smax2, out2]

        # 3rd Projection
        smax3  = np.amax(self._3d_img, axis=2)
        out3   = self.do_clahe(smax3)
        self._img_projs[3] = [smax3, out3]

        ################################
        # Now we run the edge projections
        ################################

        black_box = np.zeros((self._d,self._h + self._w - 1,self._h + self._w - 1), np.uint16)
        for i in range(self._h):
            for j in range(self._w):
                if j-i < self._w+1:
                    black_box[:,self._h+(j-i)-1,i+j] = self._3d_img[:,i,j]
                else:
                    black_box[:,self._h+self._w-(j-i)-1,i+j] = self._3d_img[:,i,j]

        # Projection 4
        smax4  = np.amax(black_box, axis=1)
        out4   = self.do_clahe(smax4)
        self._img_projs[4] = [smax4, out4]

        # Projection 5
        smax5  = np.amax(black_box, axis=2)
        out5   = self.do_clahe(smax5)
        self._img_projs[5] = [smax5, out5]

        black_box = np.zeros((self._h,self._w+self._d-1,self._w+self._d-1), np.uint16)
        for i in range(self._d):
            for j in range(self._w):
                if j-i < self._w+1:
                    black_box[:,self._d+(j-i)-1,i+j] = self._3d_img[i,:,j]
                else:
                    black_box[:,self._d+self._w-(j-i)-1,i+j] = self._3d_img[i,:,j]

        # Projection 6
        smax6  = np.amax(black_box, axis=1)
        out6   = self.do_clahe(smax6)
        self._img_projs[6] = [smax6, out6]

        # Projection 7
        smax7  = np.amax(black_box, axis=2)
        out7   = self.do_clahe(smax7)
        self._img_projs[7] = [smax7, out7]


        black_box = np.zeros((self._w,self._d + self._h - 1,self._d + self._h - 1), np.uint16)
        for i in range(self._d):
            for j in range(self._h):
                if j-i < self._h+1:
                    black_box[:,self._d+(j-i)-1,i+j] = self._3d_img[i,j,:]
                else:
                    black_box[:,self._d+self._h-(j-i)-1,i+j] = self._3d_img[i,j,:]

        # Projection 8
        smax8  = np.amax(black_box, axis=1)
        out8   = self.do_clahe(smax8)
        self._img_projs[8] = [smax8, out8]

        # Projection 9
        smax9  = np.amax(black_box, axis=2)
        out9   = self.do_clahe(smax9)
        self._img_projs[9] = [smax9, out9]
```

A.4 3D Image Reconstructions

The 3D MRA reconstruction of the segmented 2D image projections is achieved by reversing the methods implemented in Section A.3. The projection identifies voxels with maximum signal strength on the projection lines and flags them as candidates for blood vessels.

Function *reconstruct3DMRA* returns a raw 3D segmentation map: each voxel in this 3D map has a value between 0 and 9, depending on the number of projections that classify the voxel as blood vessel.

```
110
111     def reconstruct3DMRA(self, so_img_p): # 12 seconds
112         # initialise empty 3D 'black box' that will be segmented in the process below
113         black_box = np.zeros(shape=(self._d,self._h,self._w), dtype=np.uint16)
114
115         # Projection 1
116         p=0
117         ix = np.equal(so_img_p[p], 1)
118         idx = np.where(ix)
119         # list of indices that flag blood vessel locations
120         indices = list(zip(idx[0],idx[1]))
121
122         for idx in indices:
123             y = idx[0]
124             z = idx[1]
125             max_val = np.max(self._3d_img[:,y,z])
126             for i in np.where(np.greater_equal(self._3d_img[:,y,z], max_val))[0]:
127                 black_box[i,y,z] = black_box[i,y,z] + 1
128
129
130         # Projection 2
131         p=1
132         iy = np.equal(so_img_p[p], 1)
133         idy = np.where(iy)
134         # list of indices that flag blood vessel locations
135         indices = list(zip(idy[0],idy[1]))
136
137         for idy in indices:
138             x = idy[0]
139             z = idy[1]
140             max_val = np.max(self._3d_img[x,:,z])
141             for i in np.where(np.greater_equal(self._3d_img[x,:,z], max_val))[0]:
142                 black_box[x,i,z]= black_box[x,i,z] + 1
143
```

```
        # Projection 3
        p=2
        iz = np.equal(so_img_p[p], 1)
        idz = np.where(iz)
        # list of indices that flag blood vessel locations
        indices = list(zip(idz[0],idz[1]))

        for idz in indices:
            x = idz[0]
            y = idz[1]
            max_val = np.max(self._3d_img[x,y,:])
            for i in np.where(np.greater_equal(self._3d_img[x,y,:], max_val))[0]:
                black_box[x,y,i]= black_box[x,y,i] + 1

        # Projection 4
        p=3
        xd = self._d
        yd = self._h + self._w - 1

        tilted_src = np.zeros((xd,yd,yd), np.uint16)
        tilted_blk = np.zeros((xd,yd,yd), np.uint16)

        for i in range(self._h):
            for j in range(self._w):
                if j-i < self._w+1:
                    tilted_src[:,self._h+(j-i)-1,i+j] = self._3d_img[:,i,j]
                    tilted_blk[:,self._h+(j-i)-1,i+j] = black_box[:,i,j]
                else:
                    tilted_src[:,self._h+self._w-(j-i)-1,i+j] = self._3d_img[:,i,j]
                    tilted_blk[:,self._h+self._w-(j-i)-1,i+j] = black_box[:,i,j]

        ix = np.equal(so_img_p[p], 1)
        idx = np.where(ix)
        for idx in list(zip(idx[0],idx[1])):
            x = idx[0]
            y = idx[1]
            max_val = np.max(tilted_src[x,:,y])
            for i in np.where(np.greater_equal(tilted_src[x,:,y], max_val))[0]:
                tilted_blk[x,i,y] = tilted_blk[x,i,y] + 1
    # Projection 5
    p=4
    ix = np.equal(so_img_p[p], 1)
    idx = np.where(ix)
    for idx in list(zip(idx[0],idx[1])):
        x = idx[0]
        y = idx[1]
        max_val = np.max(tilted_src[x,y,:])
        for i in np.where(np.greater_equal(tilted_src[x,y,:], max_val))[0]:
            tilted_blk[x,y,i] = tilted_blk[x,y,i] + 1

    for i in range(self._h):
        for j in range(self._w):
            if j-i < self._w+1:
                black_box[:,i,j] = tilted_blk[:,self._h+(j-i)-1,i+j]
            else:
                black_box[:,i,j] = tilted_blk[:,self._h+self._w-(j-i)-1,i+j]

    # Projection 6
    p=5
    xd = self._h
    yd = self._w + self._d - 1

    tilted_src = np.zeros((xd,yd,yd), np.uint16)
    tilted_blk = np.zeros((xd,yd,yd), np.uint16)

    for i in range(self._d):
        for j in range(self._w):
            if j-i < self._w+1:
                tilted_src[:,self._d+(j-i)-1,i+j] = self._3d_img[i,:,j]
                tilted_blk[:,self._d+(j-i)-1,i+j] = black_box[i,:,j]
            else:
                tilted_src[:,self._d+self._w-(j-i)-1,i+j] = self._3d_img[i,:,j]
                tilted_blk[:,self._d+self._w-(j-i)-1,i+j] = black_box[i,:,j]
```

```
218
219         ix = np.equal(so_img_p[p], 1)
220         idx = np.where(ix)
221         for idx in list(zip(idx[0],idx[1])):
222             x = idx[0]
223             y = idx[1]
224             max_val = np.max(tilted_src[x,:,y])
225             for i in np.where(np.greater_equal(tilted_src[x,:,y], max_val))[0]:
226                 tilted_blk[x,i,y] = tilted_blk[x,i,y] + 1
227
228         # Projection 7
229         p=6
230         ix = np.equal(so_img_p[p], 1)
231         idx = np.where(ix)
232         for idx in list(zip(idx[0],idx[1])):
233             x = idx[0]
234             y = idx[1]
235             max_val = np.max(tilted_src[x,y,:])
236             for i in np.where(np.greater_equal(tilted_src[x,y,:], max_val))[0]:
237                 tilted_blk[x,y,i] = tilted_blk[x,y,i] + 1
238
239         for i in range(self._d):
240             for j in range(self._w):
241                 if j-i < self._w+1:
242                     black_box[i,:,j] = tilted_blk[:,self._d+(j-i)-1,i+j]
243                 else:
244                     black_box[i,:,j] = tilted_blk[:,self._d+self._w-(j-i)-1,i+j]
245
246
247         # Projection 8
248         p=7
249         xd = self._w
250         yd = self._h + self._d - 1
251
252         tilted_src = np.zeros((xd,yd,yd), np.uint16)
253         tilted_blk = np.zeros((xd,yd,yd), np.uint16)
254
255         for i in range(self._d):
256             for j in range(self._h):
257                 if j-i < self._h+1:
258                     tilted_src[:,self._d+(j-i)-1,i+j] = self._3d_img[i,j,:]
259                     tilted_blk[:,self._d+(j-i)-1,i+j] = black_box[i,j,:]
260                 else:
261                     tilted_src[:,self._d+self._h-(j-i)-1,i+j] = self._3d_img[i,j,:]
262                     tilted_blk[:,self._d+self._h-(j-i)-1,i+j] = black_box[i,j,:]
263
264         ix = np.equal(so_img_p[p], 1)
265         idx = np.where(ix)
266         for idx in list(zip(idx[0],idx[1])):
267             x = idx[0]
268             y = idx[1]
269             max_val = np.max(tilted_src[x,:,y])
270             for i in np.where(np.greater_equal(tilted_src[x,:,y], max_val))[0]:
271                 tilted_blk[x,i,y] = tilted_blk[x,i,y] + 1
272
273         # Projection 9
274         p=8
275         ix = np.equal(so_img_p[p], 1)
276         idx = np.where(ix)
277         for idx in list(zip(idx[0],idx[1])):
278             x = idx[0]
279             y = idx[1]
280             max_val = np.max(tilted_src[x,y,:])
281             for i in np.where(np.greater_equal(tilted_src[x,y,:], max_val))[0]:
282                 tilted_blk[x,y,i] = tilted_blk[x,y,i] + 1
283
284         for i in range(self._d):
285             for j in range(self._h):
286                 if j-i < self._h+1:
287                     black_box[i,j,:] = tilted_blk[:,self._d+(j-i)-1,i+j]
288                 else:
289                     black_box[i,j,:] = tilted_blk[:,self._d+self._h-(j-i)-1,i+j]
290
291         return black_box
292
293
```

A.5 Image Post Processing

This code implements the two post processing steps, described in Section 6.4.

In a first step, the holes in the raw segmentation map are filled up, and in the second step a dilution process is applied to create enlarged and smoothened segmentations for the blood vessel regions.

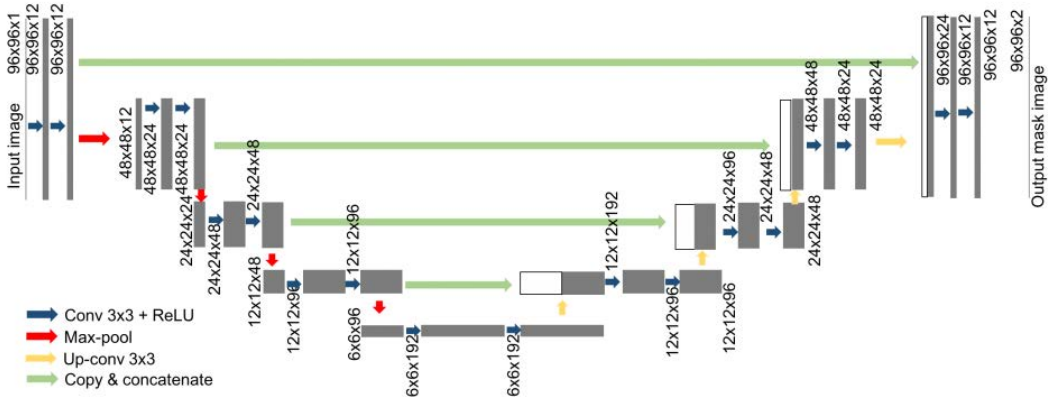
```
#-----------------------------------
# Author:       Rudiger von Hackewitz
# Date:         December 2018
#
# Finalisation of the process by adjusting some of the pixels against data in the 3D source file


import numpy as np
from scipy.ndimage import generate_binary_structure, binary_dilation

class SegmentationMap():
    def __init__(self, SegMapRaw, mra_src_3D, min_hits, min_neighbours, pxl_threshold,dilution_threshold):
        self._3d_raw = SegMapRaw
        self._mra_src_3D = mra_src_3D
        self._min_hits = min_hits
        self._min_neighbours = min_neighbours
        self._pxl_threshold = pxl_threshold
        self._dilution_threshold = dilution_threshold


    def finalise (self):
        m1 = self.cut_hits()
        m2 = self.consider_min_neighbours(m1)
        return m2


    def cut_hits(self):
        return (self._3d_raw > self._min_hits - 1)


    def consider_min_neighbours (self, b_map):
        n_map = (b_map * 0) # initialise empty map with zeros

        idx = np.where(b_map == 1)

        offset = [-1,0,1]
        for x,y,z in zip(idx[0],idx[1],idx[2]):
            for ox in offset:
                for oy in offset:
                    for oz in offset:
                        try:
                            n_map[x+ox,y+oy,z+oz] = n_map[x+ox,y+oy,z+oz] + 1
                        except IndexError:
                            pass # Ignore when pixels on the boundaries of the cube raise index errors

        added = n_map * (self._mra_src_3D.astype(int) > self._pxl_threshold).astype(int)
        k_neighbours = np.sign(b_map + (added >= self._min_neighbours).astype(np.uint16))

        # finally - apply dilution with set threshold for signal strength in the source image
        # all neighbours + neigbhours of neighbours are candidates for dilution:
        struct = generate_binary_structure(3, 2)
        fixed = binary_dilation(k_neighbours, structure=struct)


        #fixed = binary_dilation(k_neighbours)
        final = fixed * (self._mra_src_3D.astype(int) > self._dilution_threshold).astype(int)

        return final
```

A.6 U-Net Architecture

A U-Net Architecture has been applied to the 2D image projections, as proposed by Sterbak (2018); he used this model for Segmentation of seismic images; some of the hyper parameters in the proposed architecture are optimised for the use of blood vessel segmentation in this thesis. All hyper parameters for the U-Nets are fully documented in Table 7.1

A total of nine U-Nets are trained (one for each projection view). Image width and height are configurable parameters in the architecture, as they vary between different projection views.

Some elements of Sterbak (2018)'s Python code (using Python 3 and Keras framework with Tensorflow) had to be modified to cater for the specific needs of this project. However its core is taken from the internet (under the MIT license agreement) and is therefore not included in this code section.



- The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3×3 convolutions, each followed by a batchnormalization layer and a rectified linear unit (ReLU) activation and dropout and a 2×2 max pooling operation with stride 2 for downsampling. At each downsampling step we double the number of feature channels. The purpose of this contracting path is to capture the context of the input image in order to be able to do segmentation.
- Every step in the expansive path consists of an upsampling of the feature map followed by a 2×2 convolution ("up-convolution") that halves the number of feature channels, a concatenation with the correspondingly feature map from the contracting path, and two 3×3 convolutions, each followed by batchnorm, dropout and a ReLU. The purpose of this expanding path is to enable precise localization combined with contextual information from the contracting path.
- At the final layer a 1×1 convolution is used to map each 16- component feature vector to the desired number of classes.

Figure A.1: U-Net Architecture (Source: Sterbak (2018))

A.6.1 U-Net Loss Functions

Below are the representative loss functions for 4 out of the 9 training runs.

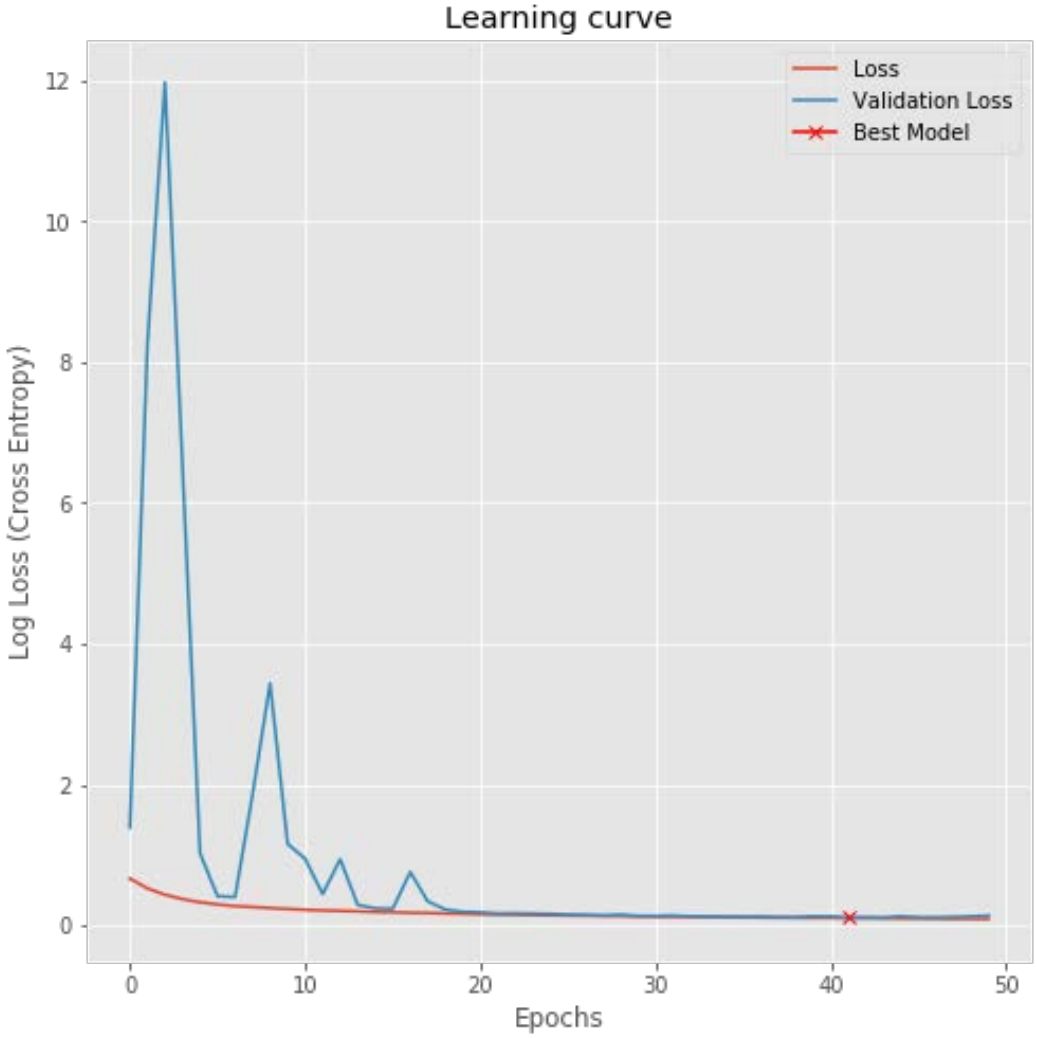


Figure A.2: Projection 2 - Loss Function

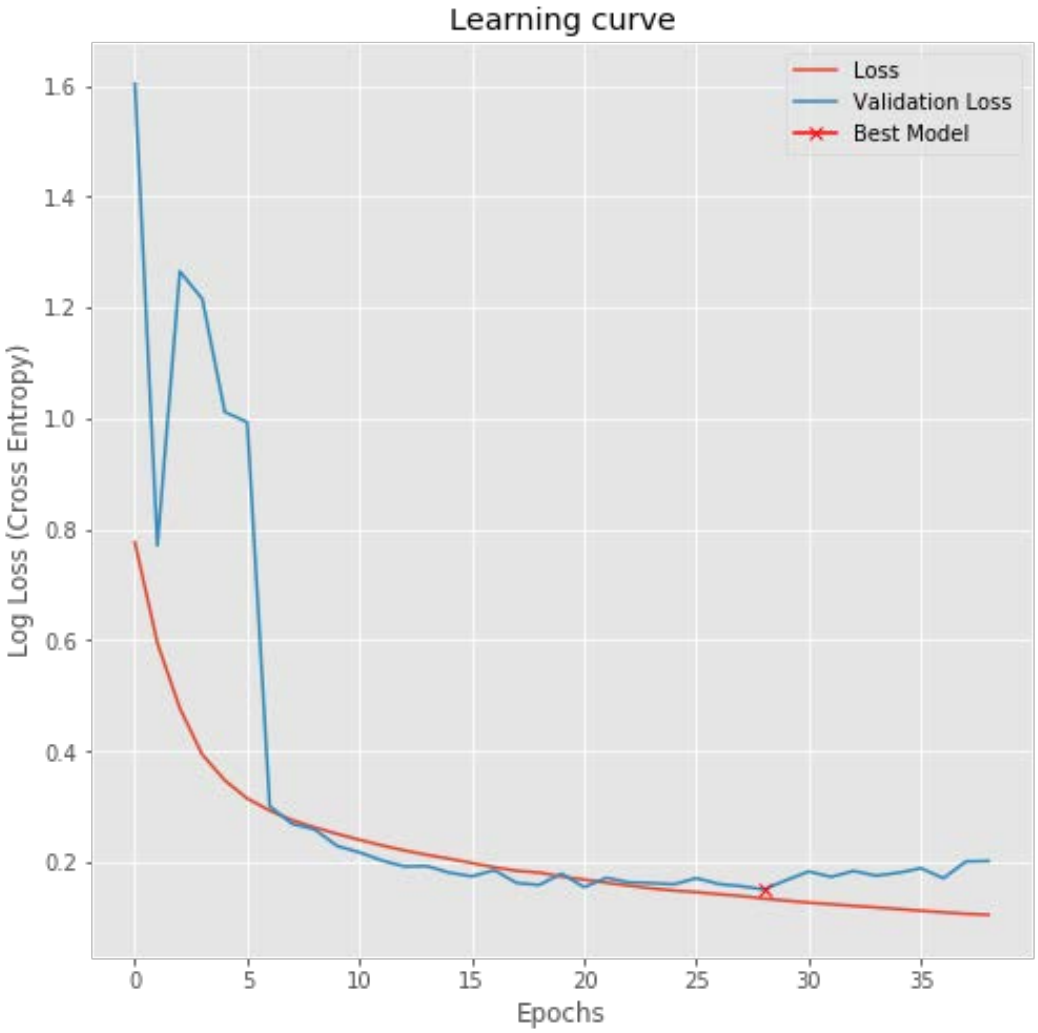


Figure A.3: Projection 4 - Loss Function

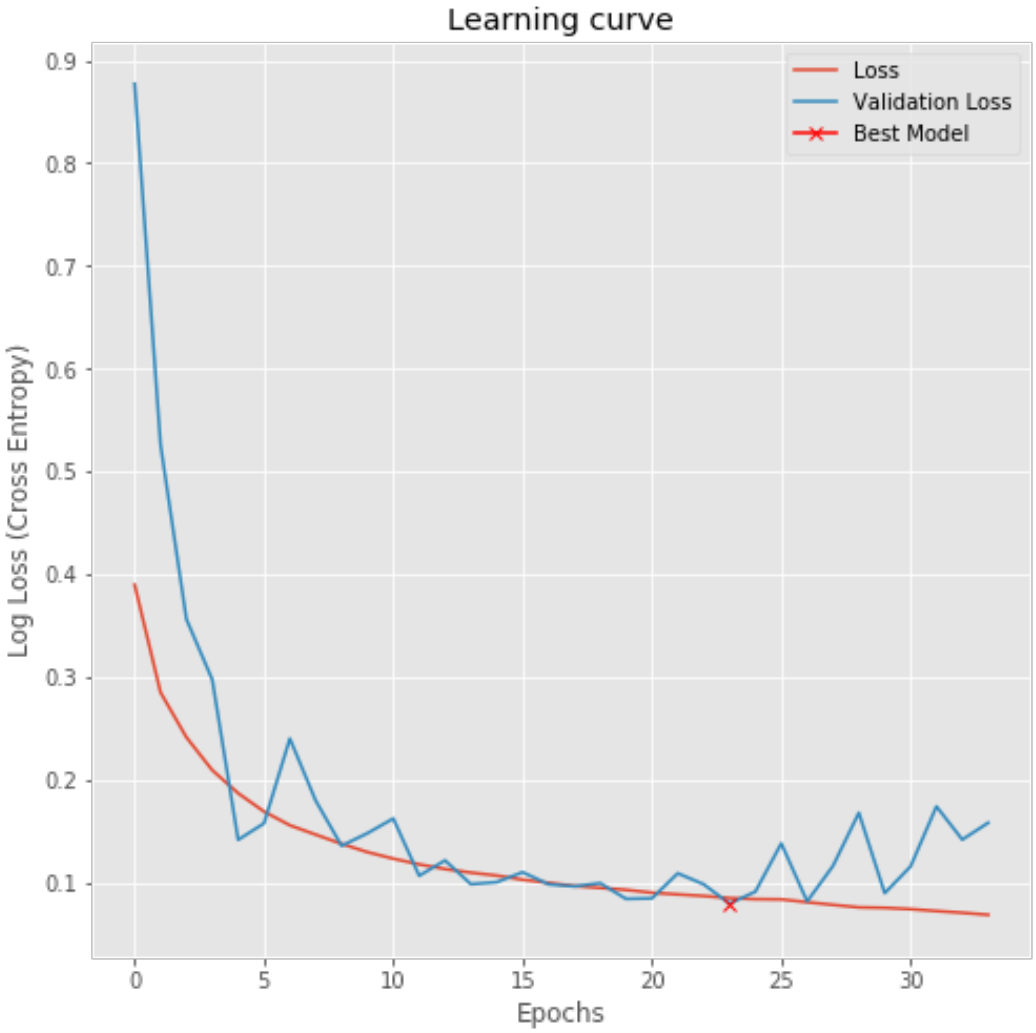


Figure A.4: Projection 7 - Loss Function

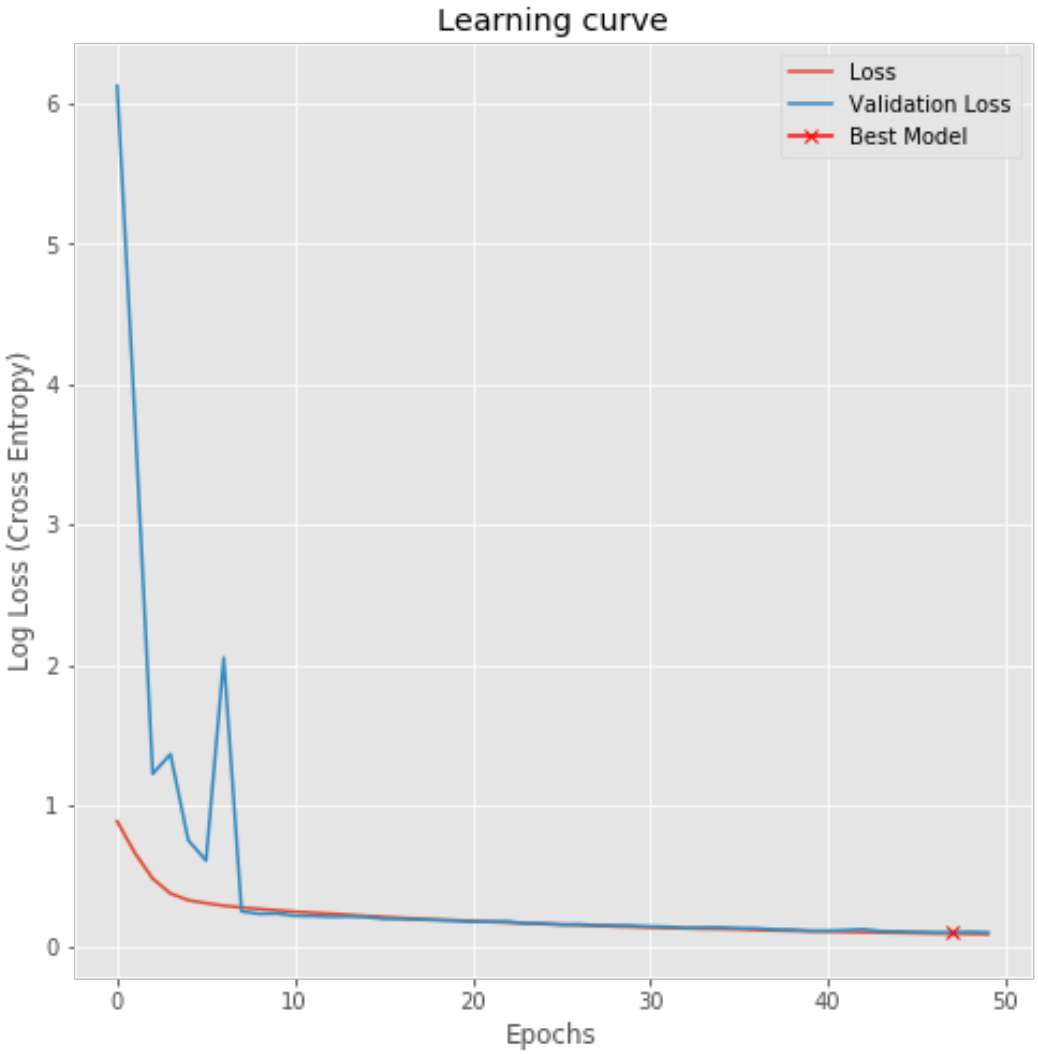


Figure A.5: Projection 9 - Loss Function

A.6.2 Image Resizing

Image resizing is required prior to feeding the data into the neural network. All source image projections are padded (width and height) so that their dimensions are a multiple of 64. This ensures that extraction and contraction paths of the U-Nets produce compatible image sizes for the input and output of files.

After completion of the neural network segmentation process, the segmented images are cropped back to their original size. The code snippet for this process is provided below.

```
#-----------------------------------
# Author:       Rudiger von Hackewitz
# Date:         December 2018
#
# UNet Models that have been built for training and execution of the segmentation processes
# on the 9 different 2-dimensional image projections


import numpy as np
import math
from utils.support import create_black_image
import cv2
import os

from keras.models import Model, load_model
from keras.layers import Input, BatchNormalization, Activation, Dense, Dropout
from keras.layers.convolutional import Conv2D, Conv2DTranspose
from keras.layers.pooling import MaxPooling2D, GlobalMaxPool2D
from keras.layers.merge import concatenate, add
from keras.optimizers import Adam

class MRA_UNets():
    def __init__(self, unet_path,d,h,w,no_projections, threshold):
        self._unet_path = unet_path
        self._d = d
        self._h = h
        self._w = w
        self._no_projections = no_projections
        self._threshold = threshold

        self._models = dict()


```

```
    def load_model(self,p):
        # use the adjusted image size of the projection for the neural network
        h, w = self.get_image_projection_size (p)
        input_img = Input((h, w, 1), name='img')
        # now define the model
        model = self.get_unet(input_img, n_filters=16, dropout=0.05, batchnorm=True)
        model.compile(optimizer=Adam(), loss="binary_crossentropy", metrics=["binary_accuracy"])
        # If model had been trained before, load its parameters
        model_par_file = self._unet_path+'model-Projection-'+p
        if os.path.exists(model_par_file):
            model.load_weights(model_par_file)
        return model



    def get_models(self):
        for i in range(self._no_projections):
            projection = str(i+1)
            self._models[i] = self.load_model(projection)



    def segment_projections(self, mra_src_2Ds):
        lst = list()
        for i in range(self._no_projections):
            projection = str(i+1)
            h, w = self.get_image_projection_size (projection)
            ho, wo = self.get_image_size (projection)
            img = np.zeros((1, h, w, 1), dtype=np.float32)
            if (ho, wo) != (mra_src_2Ds[i].shape[0], mra_src_2Ds[i].shape[1]):
                raise Exception ("Wrong input format for projection "+ projection)
            img [0,:ho,:wo,0] = mra_src_2Ds [i] / 255.0
            seg_map = self._models[i].predict(img, verbose=0)
            seg_map = seg_map.squeeze() # squeeze away empty dimensions (just two dimensions: height/width remain)
            se = np.zeros((ho, wo), np.uint16)
            se = (seg_map > self._threshold).astype(np.uint16)
            se = se[:ho,:wo] # trim it to the original image dimensions
            lst.append(se)

        return lst


    # return the original pair of (height, width) of the projected image:
    def get_image_size (self, pr):

        if pr   == '1':
            return self._h, self._w
        elif pr == '2':
            return self._d, self._h
        elif pr == '3':
            return self._d, self._w
        elif pr == '4':
            return self._d, self._h+self._w-1
        elif pr == '5':
            return self._d, self._h+self._w-1
        elif pr == '6':
            return self._w, self._h+self._d-1
        elif pr == '7':
            return self._w, self._h+self._d-1
        elif pr == '8':
            return self._h, self._w+self._d-1
        elif pr == '9':
            return self._h, self._w+self._d-1
        else:
            raise Exception('Invalid projection number: '+ pr)



    # padding the image size before feeding it into the neural network
    def get_image_projection_size (self, pr):
        height, width = self.get_image_size (pr)
        base = 64
        return int (math.ceil(height / base) * base), int (math.ceil(width / base) * base)
```

A.7 Initialisation File

The initialisation File MRASegmentation.ini is used to set hyper parameters and configurations for the segmentation runs

```
#
# Author: Rudiger von Hackewitz
# Date:   December 2018
#
# content of file is read by MRASegmentation.py; these are the fundamental
# parameter settings that have been used for training and optimising
# the MRA segmentation process
#

[GLOBAL]
# Directory that contains log details for the process
LogPath = data/logs/logs.txt
# Parameter defines whether to use debug mode or not
Debug = Yes
# Directory specifies where to write debug information (if it is turned on)
DebugPath = data/debug/


# Source Format Settings
[SOURCE]
# directory for the source data
SourcePath = data/source/
# number of slices per MRA scan
NoSlices = 247
# Width of an image slice
ImgWidth = 320
# Height of an image slice
ImgHeight = 320
# Extension of source file slices in file system
ImgExt = .jpg
# Source Image Threshold for pixels (limit to be used for candidates of blood vessel pixels)
PxlThreshold = 100


# Neural Network UNET settings
[UNET]
# Directory with the model coefficients for each of the 9 U-Nets
ModelPath = data/models-3/
# Threshold used to define whether pixel is assigned blood vessel or not. Each pixel is assigned a value between
# 0 and 1 by the UNET. This value can be interpreted as probability that the pixel belongs to a blood vessel stream.
Threshold = 0.5


[POST_PROCESSING]
# Minimum number of hits required from projections for pixel to be classified as blood vessel
MinHits = 5
# minimum number of blood vessel pixels for 'bumping up' background pixel to blood vessel pixel
MinNeighbours = 9
# Final dilution threshold
DilutionThreshold = 241


#####################################################################
# OPTIONAL - only required for model training and evaluation / testing #
#####################################################################
# Settings for ground truth data (they are not required for the segmentation process; however they are used to calculate
performance metrics, in case that they are provided with the source data to assess the quality of the segmentation);
# ground truth was predominantly used for training, optimisation and evaluation of the neural networks and dilution optimisation
[GROUND_TRUTH]
# directory for the ground truth data
GTPath = data/ground_truth/
# Extension of ground truth file slices in file system
ImgExt = .jpg
# Ground Truth Image Threshold for pixels (limit to be used for candidates of blood vessel pixels)
PxlThreshold = 100


```

A.8 UNet Configuration Files

The nine neural networks are trained and the coefficient values for each network are stored in a sub folder. The location of this folder is specified in the initialisation file MRASegmentation.ini.

Consequently, the user can execute multiple UNET training runs, and each run stores the results in a separate folder. This helps analyse the performance of the final 3D MRA segmentation map for a range of UNet versions.

Furthermore, different UNET models can be trained for a range of MRI devices, to offer adapters to differing MRI machines. The models for each device are stored in separate sub-folders.

File *model-Projection-1* stores the coefficients of Model 1 (which is responsible for segmentation of image projection 1) It is important the model parameters are supplied for all 9 models, prior to starting the segmentation process.

The model parameter settings that have been used to produce the numbers in this thesis, are provided with the source code and can be reused to reproduce the numbers and graphical outputs that are recorded in this document.

Alternatively, the interested reader may use the included Jupyter Notebook file *1a - UNET Training.ipynb* to kick off the creation of 9 new UNet models.

```
Rudigers-MacBook-Air:models-3 rvh$ ls
model-Projection-1      model-Projection-4      model-Projection-7
model-Projection-2      model-Projection-5      model-Projection-8
model-Projection-3      model-Projection-6      model-Projection-9
Rudigers-MacBook-Air:models-3 rvh$
```

A.9 Support Functions

To follow are support functions that are utilised by the program.

```
#-----------------------------------
# Author:       Rudiger von Hackewitz
# Date:         December 2018
#-----------------------------------

import configparser
import os
import numpy as np
import cv2

def read_ini_section (section):
    config = configparser.ConfigParser()
    config.read('MRASegmentation.ini')
    return config[section] # read the section and return data


def read_ini_parameter (section, par):
    # return the parameters for further use
    return section[par]


def create_black_image (h, w):
    # initialise black image with zeros and return result
    return np.zeros((h,w), np.uint16)


def convert_image_to_grey(image,h,w):
    # grab the image dimensions
    if image.shape[0] != h:
        logging.info('Wrong height')
        raise ValueError
    if image.shape[1] != w:
        logging.info('Wrong width')
        raise ValueError
    grey_image = create_black_image(h,w)
    grey_image[:h,:w] = image[:,:,0]
    # return the grey image (with a single colour channel)
    return grey_image

```

```
42
43
44 def get_filenames (pathname, ext):
45     list_of_filenames = []
46     for (dirpath, dirnames, filenames) in os.walk(pathname):
47         for f in filenames:
48             if f.endswith(ext): # just pick images with the correct extension
49                 list_of_filenames.append(f)
50
51     # sort the files by the numbers in their filenames
52     list_of_filenames.sort()
53     return list_of_filenames
54
55
56
57 def load_images (pathname,ext,d,h,w):
58     lf   = get_filenames (pathname,ext)
59     limg = np.ndarray(shape=(d,h,w), dtype=np.uint16)
60     i=0
61     for f in lf:
62         img = cv2.imread(pathname+"/"+f)
63         # convert image to grey
64         img = convert_image_to_grey(img,h,w)
65         limg[i]=img
66         i=i+1
67     return limg
68
69
70
71
72 def get_scores (seg_map, gt_map):
73     diff = np.sign(gt_map.astype(int)) - (np.sign(seg_map.astype(int))*2)
74     fn = (diff==1).sum()
75     tp = (diff==-1).sum()
76     fp = (diff==-2).sum()
77     tn = (diff==0).sum()
78
79     return fn,tp,fp,tn
80
82 def get_metrics (seg_map, gt_map):
83     fn,tp,fp,tn = get_scores (seg_map, gt_map)
84     p  = tp / (tp + fp)  # precision
85     r  = tp / (tp + fn)  # recall
86     f1 = 2*p*r/(p+r)     # F1 score
87
88     #print('Precision:    ' + str(np.round(100*p,2))+'%')
89     #print('Recall:       ' + str(np.round(100*r,2))+'%')
90     #print('F1 Score:     ' + str(np.round(100*2*p*r/(p+r),2))+'%')
91     #print('    ')
92
93     return p, r, f1
```

A.10 Main Source Code File

Known as the 'main' file in some programming languages, *MRASegmentation.py* is the file, that starts the segmentation process and brings all the elements together that need to be executed for the successful completion of an MRA segmentation run. For those, who just want to get an overview about the program flow, it is sufficient to read the content in this source file.

```
#-----------------------------------
# Author:       Rudiger von Hackewitz
# Date:         December 2018
#
# Main process to run MRA image segmentation process
# Important: This process does NOT include training of
# the neural networks


from utils.support import read_ini_section, read_ini_parameter, get_metrics
import projections.ImageProjections as ip
import unets.UNetModels as un
import postprocessing.FinaliseSegmentationMap as pp
import sys
import logging
import os
import utils
import numpy as np
import cv2


class MRASegmentation():
    def __init__(self):
        # Read initialisation parameters from ini-file
        sc = read_ini_section('GLOBAL')
        self._log_path    = read_ini_parameter(sc,'LogPath').strip()
        self._debug = (read_ini_parameter(sc,'Debug').upper() == 'YES')
        self._debug_path  = read_ini_parameter(sc,'DebugPath').strip()


        # Source img sections, specifying characteristics of the source image slices
        sc = read_ini_section('SOURCE')
        self._source_path    = read_ini_parameter(sc,'SourcePath').strip()
        self._no_slices      = int(read_ini_parameter(sc,'NoSlices'))
        self._img_height     = int(read_ini_parameter(sc,'ImgHeight'))
        self._img_width      = int(read_ini_parameter(sc,'ImgWidth'))
        self._img_ext        = read_ini_parameter(sc,'ImgExt').strip()
        self._pxl_threshold  = int(read_ini_parameter(sc,'PxlThreshold'))


        # Neural Network (UNET) settings
        sc = read_ini_section('UNET')
        # path to the parameter model settings of the nine trained neural networks (U-Nets)
        self._model_path  = read_ini_parameter(sc,'ModelPath').strip()
        # Threshold used to define whether pixel is assigned blood vessel or not after UNET
        # assigns each pixel a value between 0 and 1. By default it is set to 0.5
        self._unet_threshold  = float(read_ini_parameter(sc,'Threshold'))


        # Parameters for post-processing
        sc = read_ini_section('POST_PROCESSING')
        self._min_hits           = int(read_ini_parameter(sc,'MinHits'))
        self._min_neighbours     = int(read_ini_parameter(sc,'MinNeighbours'))
        self._dilution_threshold = int(read_ini_parameter(sc,'DilutionThreshold'))


        # set up logging into a file for the process
        # create empty log file if it does not exist, otherwise leave it (and append log entries to it
        logf = self._log_path
        open(logf,"a").close()
        logging.basicConfig(format='%(asctime)s %(message)s', datefmt='%d-%b-%Y %I:%M:%S %p: ',filename=logf,level=logging.DEBUG)


        # OPTIONAL (only if GT available to calculate performance metrics)
        sc = read_ini_section('GROUND_TRUTH')
        self._gt_path           = read_ini_parameter(sc,'GTPath').strip()
        self._gt_img_ext        = read_ini_parameter(sc,'ImgExt').strip()
        self._gt_pxl_threshold  = int(read_ini_parameter(sc,'PxlThreshold'))

        # The process has been designed with a total of 9 projections (9 orthogonal projections into the
        # MRA cube and 6 edge views into the MRA cube)
        self.NO_PROJECTIONS = 9

```

```
        # pointer to the 9 instances of the UNet models (models not yet loaded)
        self._unets = un.MRA_UNets(self._model_path,
                                   self._no_slices,
                                   self._img_height,
                                   self._img_width,
                                   self.NO_PROJECTIONS,
                                   self._unet_threshold)


    def set_up_debug_directories (self, lst):
        for n in lst:
            try:
                # Create target Directory
                os.mkdir(self._debug_path + n)
            except FileExistsError:
                pass


    def get_3d_data(self, path,ext):
        r = utils.support.load_images (path,ext,self._no_slices,self._img_height,self._img_width)

        if r.shape != (self._no_slices,self._img_height,self._img_width):
            logging.info('Input data are provided in the wrong format: ')
            raise ValueError

        return r

    # load the 3D source file images
    def get_3d_mra_source(self, mra_id):
        r = self.get_3d_data(self._source_path + mra_id,self._img_ext)

        if self._debug:
            r.tofile(self._debug_path + mra_id + '/Step1 Source-3D-Map')

        return r

    # Load the 3d ground truth images
    def get_3d_mra_gt (self, mra_id):
        gt = self.get_3d_data(self._gt_path + mra_id,self._gt_img_ext)

        # all data above a certain threshold in the GT are considered
        gt = (gt > self._gt_pxl_threshold).astype(int)
        gt = (gt * 255)

        if self._debug:
            gt.tofile(self._debug_path + mra_id + '/Step1 GT-3D-Map')

        return gt


    def get_2d_mra_src_projs(self, mra_src_3D):
        projs = ip.MRAProjection(mra_src_3D)
        projs.run_projections()
        p = projs._img_projs

        # Write the files into the debug directory (only if debug mode switched on)
        if self._debug:
            for i in range(self.NO_PROJECTIONS):   # work through the nine different 2D image perspectives
                cv2.imwrite(self._debug_path+self._mra_id+'/Step2 2D-Source-Projection-'+str(i+1)+'.jpg',p[i+1][0])
                cv2.imwrite(self._debug_path+self._mra_id+'/Step2 2D-Source-Projection-'+str(i+1)+'-CLAHE.jpg',p[i+1][1])

        # just return the list of CLAHE enhanced image projections, as they will be fed into the neural network
        return [p[i+1][1] for i in range(self.NO_PROJECTIONS)]

    # for the training of the neural networks we need to create the image projections of the 3D ground truth data
    def get_2d_mra_gt_projs(self, mra_gt_3D):
        projs = ip.MRAProjection(mra_gt_3D)
        projs.run_projections()
        p = projs._img_projs

        # Write the files into the debug directory (only if debug mode switched on)
        if self._debug:
            for i in range(self.NO_PROJECTIONS):   # work through the nine different 2D image perspectives
                cv2.imwrite(self._debug_path+self._mra_id+'/Step2 2D-GT-Projection-'+str(i+1)+'.jpg',255*p[i+1][0])

        # just return the raw image projections, CLAHE is not required for the ground truth data
        return [p[i+1][0] for i in range(self.NO_PROJECTIONS)]


    def get_2d_mra_seg_projs(self, mra_src_2Ds):
        r = self._unets.segment_projections(mra_src_2Ds)

        # Write the files into the debug directory (only if debug mode switched on)
        if self._debug:
            for i in range(self.NO_PROJECTIONS):   # work through the nine different 2D image perspectives
                cv2.imwrite(self._debug_path+self._mra_id+'/Step3 2D-Segmentation-'+str(i+1)+'.jpg', r[i]*255)

        return r


    def reconstruct_3d_segmentation(self, mra_seg_2Ds,mra_src_3D):
        projs = ip.MRAProjection(mra_src_3D)
        r = projs.reconstruct3DMRA(mra_seg_2Ds)

        # cross check that the segmentation map is in the right shape
        if r.shape != (self._no_slices,self._img_height,self._img_width):
            logging.info('3D raw segmentation map is in the wrong format: '+ str(shp))
            raise ValueError

        # include a cross validation / count check to verify sanity of the pixel values in segmentation map:
        hits = [ np.sum(r == i) for i in range(self.NO_PROJECTIONS + 1) ]
        s1 = sum([x for x in hits])
        s2 = self._no_slices * self._img_height * self._img_width
        if s1 != s2:
            logging.info('Pixel Count in Segmentation Map does not match with expected pixel count in Source')
            raise ValueError

        if self._debug:
            r.tofile(self._debug_path + self._mra_id + '/Step4 Raw-3D-Segmentation-Map')

        return r
```

```
187
188     def post_processing(self, mra_seg_3D_raw,mra_src_3D):
189         postp = pp.SegmentationMap(mra_seg_3D_raw, mra_src_3D, self._min_hits, self._min_neighbours,
190                                    self._pxl_threshold,self._dilution_threshold)
191         r = postp.finalise()
192
193         if self._debug:
194             r.tofile(self._debug_path + self._mra_id + '/Step5 Final-3D-Segmentation-Map')
195
196         return r
197
198
199     def peformance_metrics(self,mra_seg_3D):
200         if os.path.exists(self._gt_path + self._mra_id):
201             # Load the ground truth data
202             gt = self.get_3d_mra_gt (self._mra_id)
203
204             # Calculate the performance metrics of the segmentation
205             p, r, f1 = get_metrics(mra_seg_3D, gt)
206
207             logging.info('Precision:    ' + str(np.round(100*p,2))+'%')
208             logging.info('Recall:       ' + str(np.round(100*r,2))+'%')
209             logging.info('F1 Score:     ' + str(np.round(100*f1,2))+'%')
210             print('Precision:    ' + str(np.round(100*p,2))+'%')
211             print('Recall:       ' + str(np.round(100*r,2))+'%')
212             print('F1 Score:     ' + str(np.round(100*f1,2))+'%')
213
214
215     def get_segmentation_map(self, mra_id):
216
217         logging.info('************************************************************')
218         logging.info('Starting the blood vessel segmentation process for ID ' + mra_id)
219         logging.info('************************************************************')
220
221         self._mra_id = mra_id
222
223         # Read in 3D MRA source data
224         mra_src_3D = self.get_3d_mra_source(mra_id)
225         logging.info('Step 1 - 3D source data loaded')
226
227         # construct the nine 2D MRA projections
228         mra_src_2Ds = self.get_2d_mra_src_projs(mra_src_3D)
229         logging.info('Step 2 - 2D image projections created')
230
231         # run the nine unet processes across the image source projections
232         mra_seg_2Ds = self.get_2d_mra_seg_projs(mra_src_2Ds)
233         logging.info('Step 3 - 9 image segmentation processes completed')
234
235         # Reconstruct the 3D segmented raw image
236         mra_seg_3D_raw = self.reconstruct_3d_segmentation(mra_seg_2Ds,mra_src_3D)
237         logging.info('Step 4 - 3D segmentation map for blood vessels reconstructed')
238
239         # Postprocessing
240         mra_seg_3D = self.post_processing(mra_seg_3D_raw,mra_src_3D)
241         logging.info('Step 5 - Postprocessing for 3D segmentation map complete')
242
243         # Calculate performance metrics (if ground truth data is available)
244         self.peformance_metrics(mra_seg_3D)
245
246
247         return mra_seg_3D
248
249     # get all the source IDs (based on the directory names in the source folder)
250     def get_mra_source_ids(self):
251         # run through all the records in the source directory
252         return [f.name for f in os.scandir(self._source_path) if f.is_dir() and f.name[0] != '.']
253
254 # run the MRA segmentation process
255 def run_it ():
256     app = MRASegmentation()
257
258     # get all the source ids in the source folder
259     lst = app.get_mra_source_ids ()
260
261     # process all the ids that are provided via the command line and are available in source directory:
262     if len(sys.argv) > 1:
263         lst = list(set(lst) & set(sys.argv[1:]))
264
265     if len(lst) == 0:
266         print ('No MRA scans are available and nothing can be processed. Set up source data in directory '+app._source_path)
267
268     else:
269         logging.info('Initialising the process...')
270         # If required, set up debug subdirectories for each record id
271         if app._debug:
272             app.set_up_debug_directories(lst)
273
274         # Load the parameters of the nine UNet models
275         app._unets.get_models()
276         logging.info('Initialisation complete and UNET models loaded')
277
278         # Now segment the source data for each provided record id:
279         for nme in lst:
280             try:
281                 print('Processing Segmentation Map for Record '+nme+' ...')
282                 r = app.get_segmentation_map(nme)
283                 print('Record '+nme+' segmented successfully.')
284             except Exception as e:
285                 logging.fatal(e, exc_info=True)
286                 print('Record '+nme+' could not be segmented. Check the log/debug files for errors.')
287
288         logging.info('Overall process complete.')
289 run_it()
290
```

A.11 Starting the Program

The program can be started via the command line. The names of the MRA source records can be submitted with the execution of the command line. If no argument is supplied, all records in the source folder are processed by the program.

If corresponding ground truth data is provided with the source record (optional), then the program will calculate the performance metrics as they are achieved against the ground truth.

To follow is the command line, that is required to segment record MNI 0663. This is the record that has been held back and is just used in the final test run. Enter *python MRASegmentation.py MNI-0663* into the command line.

If the program should process all records in the source folder, then just enter *python MRASegmentation.py*

```
Rudigers-MacBook-Air:05-Final-Model rvh$ python MRASegmentation.py MNI_0663
/Users/rvh/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWarning:
 Conversion of the second argument of issubdtype from `float` to `np.floating` is de
precated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
2019-01-05 17:11:29.305051: I tensorflow/core/platform/cpu_feature_guard.cc:141] You
r CPU supports instructions that this TensorFlow binary was not compiled to use: AVX
2 FMA
Processing Segmentation Map for Record MNI_0663 ...
Precision:   94.25%
Recall:      82.16%
F1 Score:    87.79%
Record MNI_0663 segmented successfully.
Rudigers-MacBook-Air:05-Final-Model rvh$
```

A.11.1 Log File

Once started, the program will automatically append to the logfile. Its location and name is specified in the initialisation file MRASegmentation.ini.

The file includes the seconds in the timestamp, and this is useful information to monitor the overall performance of the process against individual MRA scans.

```
14-Dec-2018 07:49:40 PM:  Initialising the process...
14-Dec-2018 07:50:13 PM:  Initialisation complete and UNET models loaded
14-Dec-2018 07:50:13 PM:  ****************************************************************
14-Dec-2018 07:50:13 PM:  Starting the blood vessel segmentation process for ID MNI_0656
14-Dec-2018 07:50:13 PM:  ****************************************************************
14-Dec-2018 07:50:13 PM:  Step 1 - 3D source data loaded
14-Dec-2018 07:50:16 PM:  Step 2 - 2D image projections created
14-Dec-2018 07:50:30 PM:  Step 3 - 9 image segmentation processes completed
14-Dec-2018 07:50:37 PM:  Step 4 - 3D segmentation map for blood vessels reconstructed
14-Dec-2018 07:50:38 PM:  Step 5 - Postprocessing for 3D segmentation map complete
14-Dec-2018 07:50:39 PM:  Precision:   92.41%
14-Dec-2018 07:50:39 PM:  Recall:      76.38%
14-Dec-2018 07:50:39 PM:  F1 Score:    83.63%
14-Dec-2018 07:50:39 PM:  ****************************************************************
14-Dec-2018 07:50:39 PM:  Starting the blood vessel segmentation process for ID MNI_0657
14-Dec-2018 07:50:39 PM:  ****************************************************************
14-Dec-2018 07:50:40 PM:  Step 1 - 3D source data loaded
14-Dec-2018 07:50:42 PM:  Step 2 - 2D image projections created
14-Dec-2018 07:50:47 PM:  Step 3 - 9 image segmentation processes completed
14-Dec-2018 07:50:55 PM:  Step 4 - 3D segmentation map for blood vessels reconstructed
14-Dec-2018 07:50:57 PM:  Step 5 - Postprocessing for 3D segmentation map complete
14-Dec-2018 07:50:58 PM:  Precision:   37.88%
14-Dec-2018 07:50:58 PM:  Recall:      85.33%
14-Dec-2018 07:50:58 PM:  F1 Score:    52.47%
14-Dec-2018 07:50:58 PM:  ****************************************************************
14-Dec-2018 07:50:58 PM:  Starting the blood vessel segmentation process for ID MNI_0592
14-Dec-2018 07:50:58 PM:  ****************************************************************
14-Dec-2018 07:50:58 PM:  Step 1 - 3D source data loaded
14-Dec-2018 07:51:01 PM:  Step 2 - 2D image projections created
14-Dec-2018 07:51:05 PM:  Step 3 - 9 image segmentation processes completed
14-Dec-2018 07:51:13 PM:  Step 4 - 3D segmentation map for blood vessels reconstructed
14-Dec-2018 07:51:15 PM:  Step 5 - Postprocessing for 3D segmentation map complete
14-Dec-2018 07:51:16 PM:  Precision:   49.19%
14-Dec-2018 07:51:16 PM:  Recall:      82.84%
14-Dec-2018 07:51:16 PM:  F1 Score:    61.73%
14-Dec-2018 07:51:16 PM:  ****************************************************************
14-Dec-2018 07:51:16 PM:  Starting the blood vessel segmentation process for ID MNI_0643
14-Dec-2018 07:51:16 PM:  ****************************************************************
14-Dec-2018 07:51:16 PM:  Step 1 - 3D source data loaded
14-Dec-2018 07:51:19 PM:  Step 2 - 2D image projections created
14-Dec-2018 07:51:23 PM:  Step 3 - 9 image segmentation processes completed
14-Dec-2018 07:51:32 PM:  Step 4 - 3D segmentation map for blood vessels reconstructed
14-Dec-2018 07:51:34 PM:  Step 5 - Postprocessing for 3D segmentation map complete
14-Dec-2018 07:51:35 PM:  Precision:   41.61%
14-Dec-2018 07:51:35 PM:  Recall:      85.67%
14-Dec-2018 07:51:35 PM:  F1 Score:    56.02%
14-Dec-2018 07:51:35 PM:  ****************************************************************
14-Dec-2018 07:51:35 PM:  Starting the blood vessel segmentation process for ID MNI_0663
14-Dec-2018 07:51:35 PM:  ****************************************************************
14-Dec-2018 07:51:35 PM:  Step 1 - 3D source data loaded
14-Dec-2018 07:51:38 PM:  Step 2 - 2D image projections created
14-Dec-2018 07:51:42 PM:  Step 3 - 9 image segmentation processes completed
```

A.11.2 Debug Folder

Only if Debug is enabled (in the initialisation file), content will be written into this directory.

A separate subfolder will be created for the processing of an individual MRA scan. The folder includes intermediary steps, such as the image projections, CLAHE enhancements, UNET segmentation results, raw 3D maps and final 3D segmentation results. It is useful to better understand how the final segmentation result was reached or to debug the code.

In production, debug mode should be disabled and the folder will remain empty.

```
Rudigers-MacBook-Air:debug rvh$ ls
MNI_0590        MNI_0592        MNI_0643        MNI_0656        MNI_0663
MNI_0591        MNI_0640        MNI_0648        MNI_0657        MNI_0664
Rudigers-MacBook-Air:debug rvh$ ls MNI_0590
Step1 GT-3D-Map                         Step2 2D-Source-Projection-5.jpg
Step1 Source-3D-Map                     Step2 2D-Source-Projection-6-CLAHE.jpg
Step2 2D-GT-Projection-1.jpg            Step2 2D-Source-Projection-6.jpg
Step2 2D-GT-Projection-2.jpg            Step2 2D-Source-Projection-7-CLAHE.jpg
Step2 2D-GT-Projection-3.jpg            Step2 2D-Source-Projection-7.jpg
Step2 2D-GT-Projection-4.jpg            Step2 2D-Source-Projection-8-CLAHE.jpg
Step2 2D-GT-Projection-5.jpg            Step2 2D-Source-Projection-8.jpg
Step2 2D-GT-Projection-6.jpg            Step2 2D-Source-Projection-9-CLAHE.jpg
Step2 2D-GT-Projection-7.jpg            Step2 2D-Source-Projection-9.jpg
Step2 2D-GT-Projection-8.jpg            Step3 2D-Segmentation-1.jpg
Step2 2D-GT-Projection-9.jpg            Step3 2D-Segmentation-2.jpg
Step2 2D-Source-Projection-1-CLAHE.jpg  Step3 2D-Segmentation-3.jpg
Step2 2D-Source-Projection-1.jpg        Step3 2D-Segmentation-4.jpg
Step2 2D-Source-Projection-2-CLAHE.jpg  Step3 2D-Segmentation-5.jpg
Step2 2D-Source-Projection-2.jpg        Step3 2D-Segmentation-6.jpg
Step2 2D-Source-Projection-3-CLAHE.jpg  Step3 2D-Segmentation-7.jpg
Step2 2D-Source-Projection-3.jpg        Step3 2D-Segmentation-8.jpg
Step2 2D-Source-Projection-4-CLAHE.jpg  Step3 2D-Segmentation-9.jpg
Step2 2D-Source-Projection-4.jpg        Step4 Raw-3D-Segmentation-Map
Step2 2D-Source-Projection-5-CLAHE.jpg
Rudigers-MacBook-Air:debug rvh$
```

Appendix B

2D and 3D Image Artefacts

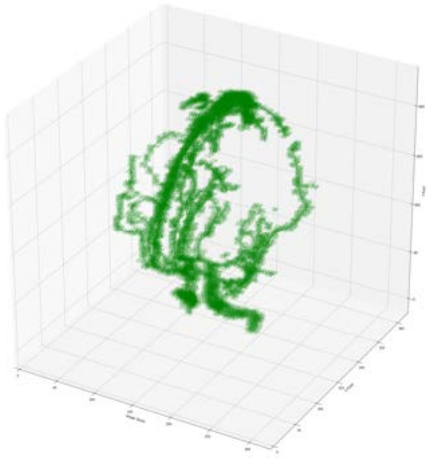
To follow is a small extract for some of the many graphical artefacts that have been produced and analysed during the work on this project. All artefacts are created using the Jupyter Notebook IDE. The Python libraries matplotlib and mpl-toolkits with its 3D extension mplot3d are used to graph the three-dimensional scatterplots.

Library CV2 is used for processing of 2D images (reading, writing and grey-scaling).

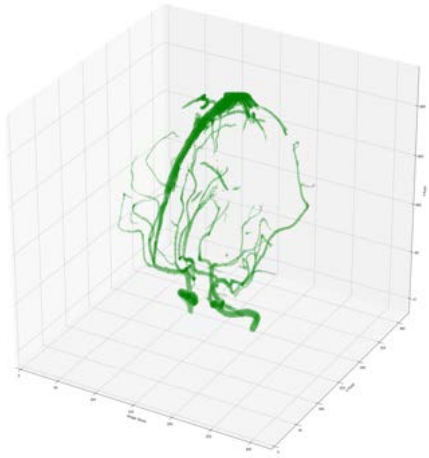
Just by scrolling through the Appendix B, the reader will get a clear picture about the steps and tuning methods, that are applied, to build an optimised segmentation process with the outcomes presented in the final Figure B.16.

B.1 3D Ground Truth - Signal Strength Threshold

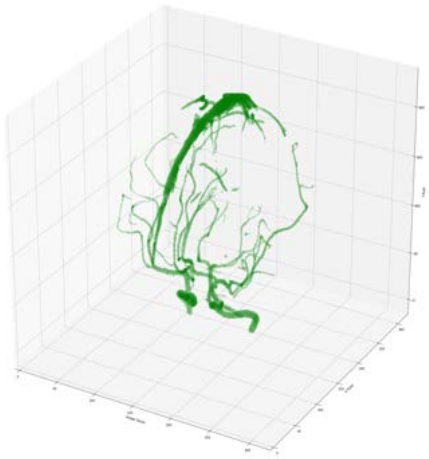
The ground truth data are not provided as binary maps, but its voxels are still in the range between 0 and 255. Therefore, it is necessary to introduce a threshold value T, that discriminates between background and blood vessel. Only voxels with values greater than T are classified as blood vessel. The resultant 3D ground truth maps for different values of T are depicted in Figure B.1. After investigating the 3D ground truth maps for various values of T, it has been decided to choose T = 100 as threshold for the creation of the binary ground truth segmentation maps.



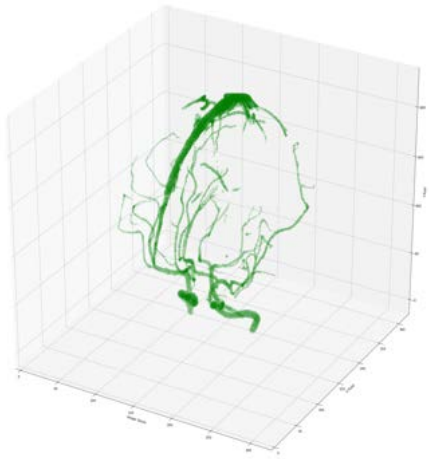
(a) Signals Strength Threshold: 1



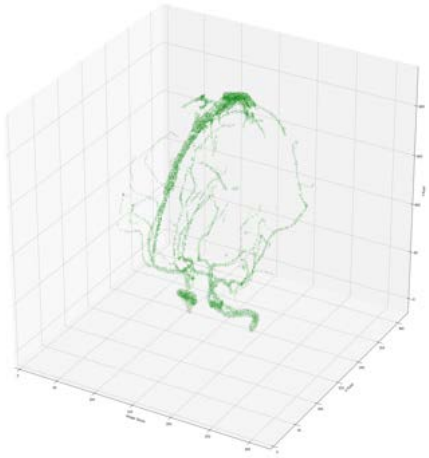
(b) Signals Strength Threshold: 50



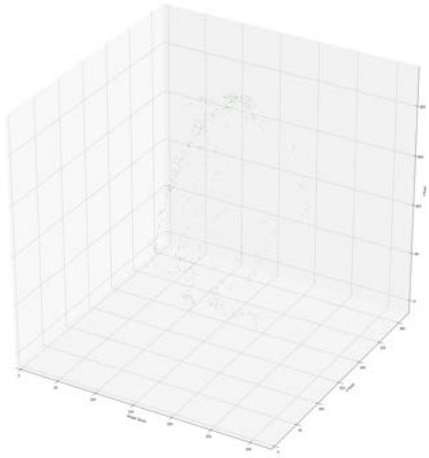
(c) Signals Strength Threshold: 100



(d) Signals Strength Threshold: 120



(e) Signals Strength Threshold: 130



(f) Signals Strength Threshold: 140

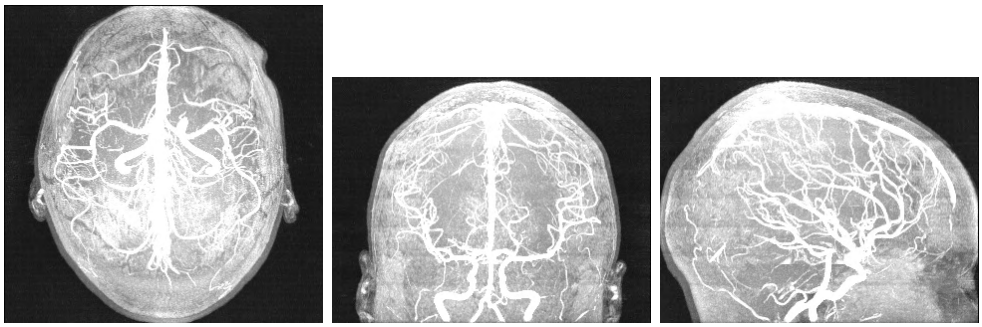
Figure B.1: Ground Truth Maps for MNI-0656 with various values for the Signal Strength Threshold

B.2 2D Image Projections

The following section includes the nine image projections for MRA scan record MNI-0656.

B.2.1 MNI-0656: Front Projections

The front projection look from a 90 degree angle at the 3D MRA scan. The rectangular MRA cuboid has 6 faces, and projections into the opposing faces are excluded, as they do not add additional information. They produce mirrored 2D image projections.



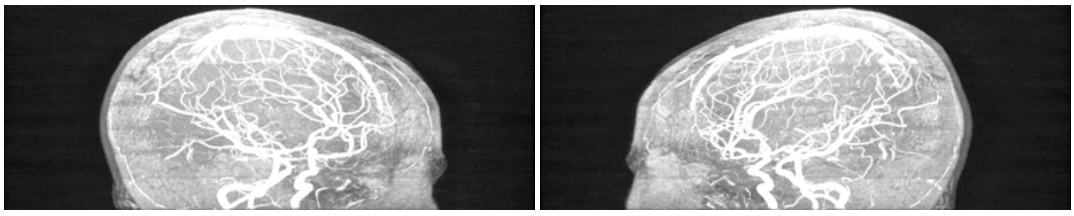
(a) Top of the Head (b) Person's Face (c) Person's Ears

Figure B.2: Three Front Views for Record MNI-0656

B.2.2 MNI-0656: Edge Projections

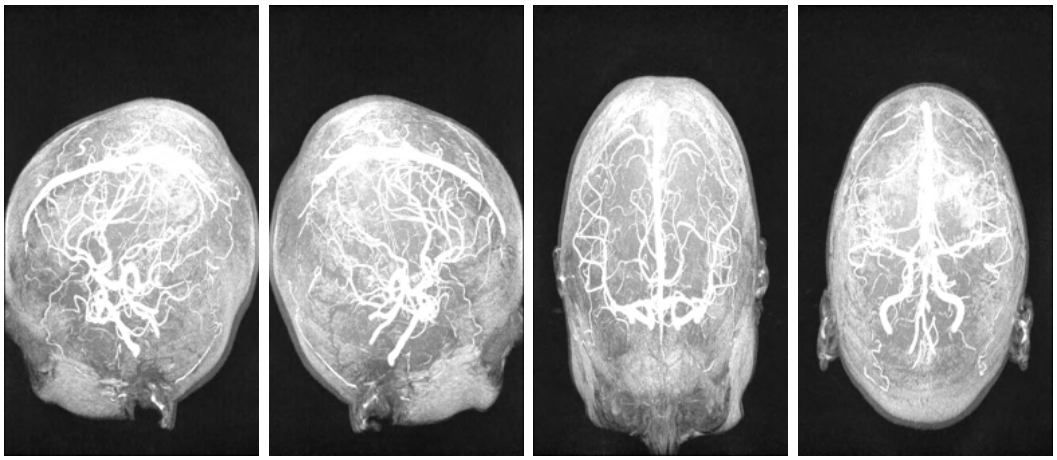
The six edge views below are created, by looking at the edges of the 3D cuboid at 45 degree angles, relative to the cuboid faces. The cuboid has a total of 12 edges, and 6 edge views are processed by the algorithm, excluding opposing edges.

The images are stretched along the axis that runs through the edge of the 3D cube. For calcuation of the stretching factor see Section 4.3.2. As a consequence, image width and height are not presented in a 1:1 ratio, and the six images of the edge view projections appear to be distorted.



(a) Right Ear and Eye (b) Left Ear and Eye

Figure B.3: Two (of the Six) Edge Views for Record MNI-0656

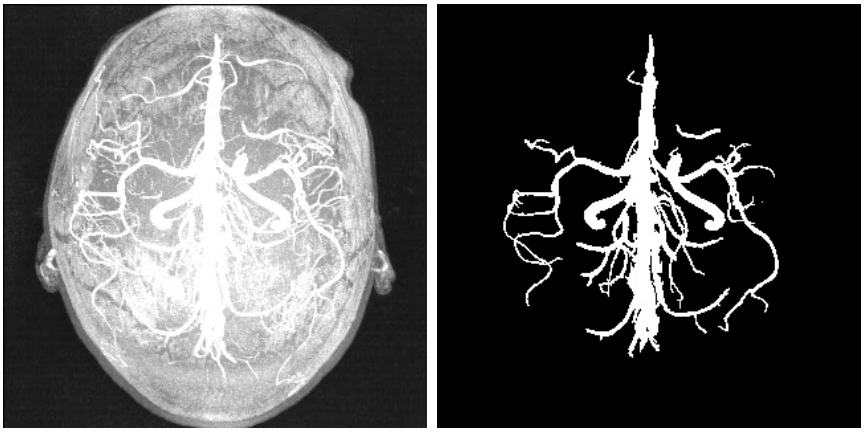


(a) From Top - Left Ear Angle (b) From Top - Right Ear Angle (c) From Top - Face Angle (d) From Top - Back Angle

Figure B.4: Four (of the Six) Edge View Projections for Record MNI-0656

B.2.3 MNI-0656: Source Projection vs Ground Truth Projection

Figure B.5 compares the top view projection of record MNI-0656 against its manually segmented ground truth projection. The ground truth version B.5b needs to be mimicked by the U-Net model.

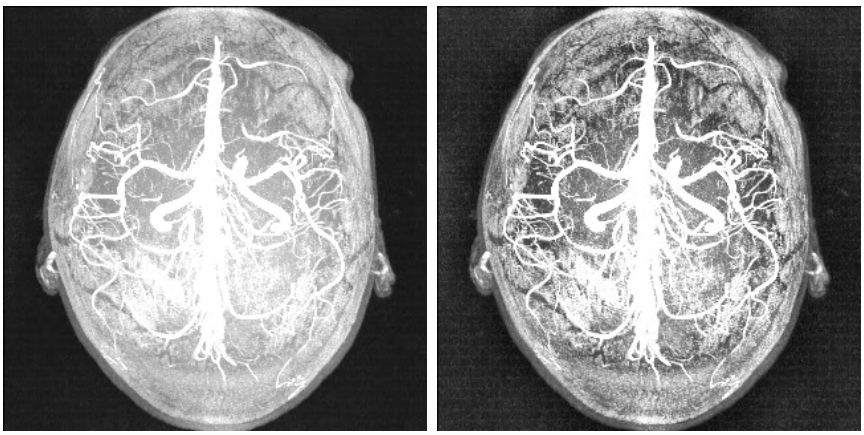


(a) Top View Source Projection (b) Top View GT Projection

Figure B.5: Source vs GT Projection for Record MNI-0656

B.2.4 MNI-0656: CLAHE Enhancements

Image projections are CLAHE-enhanced, prior to feeding the images into the U-Net segmentation process. This helps increase contrast between blood vessel regions and background, as illustrated in Figure B.6.



(a) Front Projection (b) CLAHE Enhancement

Figure B.6: Record MNI-0565: Projection 1 before and after CLAHE Enhancement

B.3 Blood Vessel Segmentations in 2D Image Projections

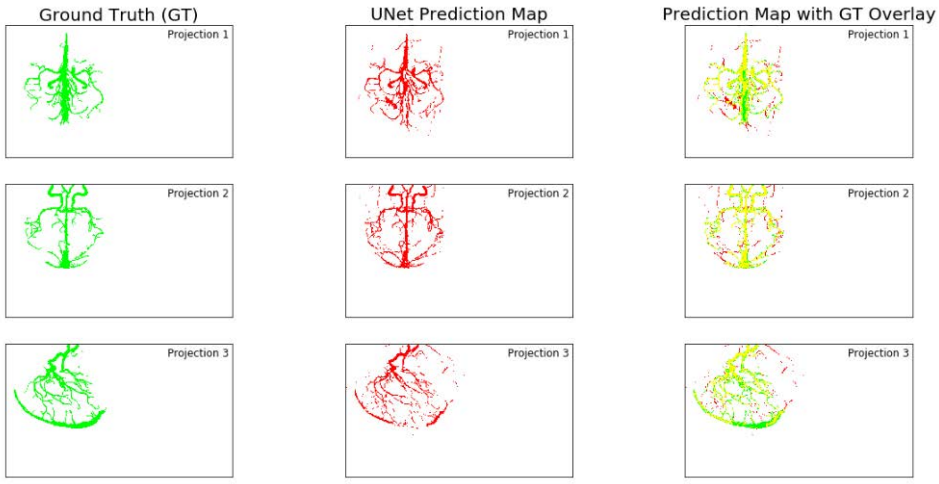
This section visualises the 2D segmentation results against the corresponding ground truth data. Separate U-Net versions are trained for each of the nine projection views.

B.3.1 2D Unet Predictions with GT Overlays for MNI-0656 and MNI-0663

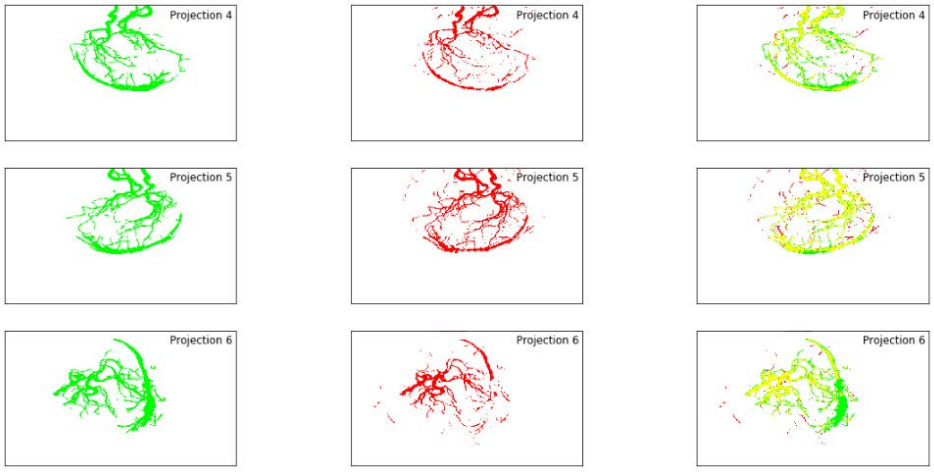
In Figure B.7, the green-coloured image to the left is the respective 2D projection of the ground truth, the red-coloured image in the middle is the segmentation map, predicted by U-Nets 1 to 9, and the image to the right is the overlay of 2D ground truth projection and predicted segmentation map.

The results for the two manually segmented records MNI-0656 and MNI-0663 are provided in Figures B.7 and B.8, respectively.

(a) Front View Projections (1 to 3)



(b) Edge Projections (4 to 6)



(c) Edge Projections (7 to 9)

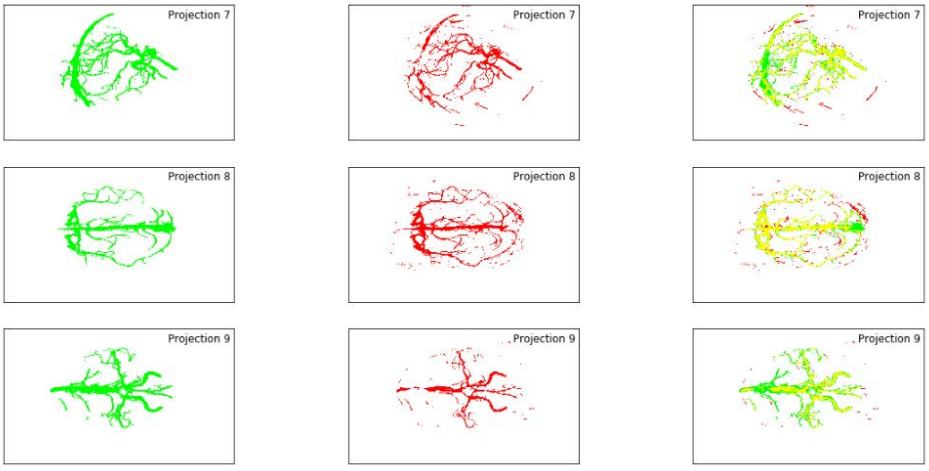
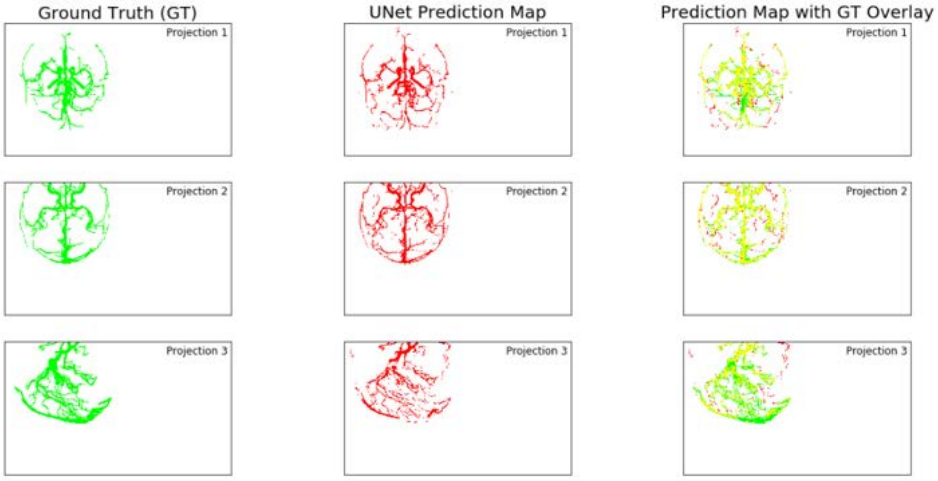
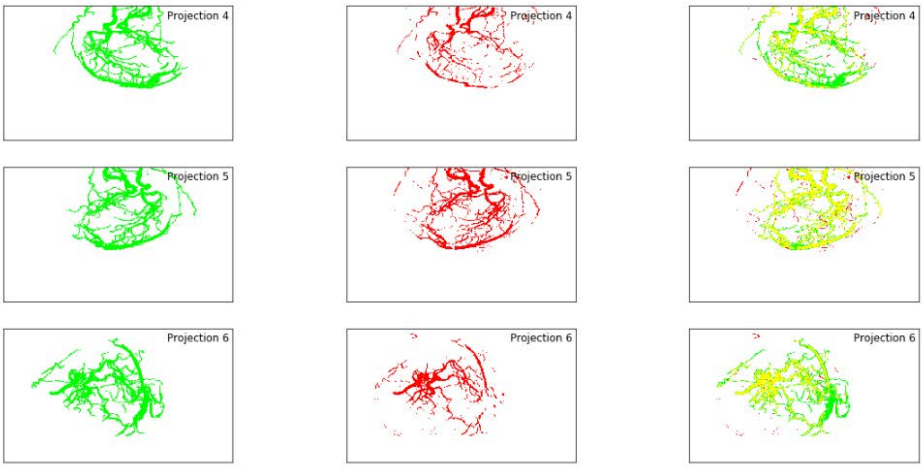


Figure B.7: MNI-0656 UNet 2D Segmentation Results with Ground Truth Overlays for Projections 1 through to 9

(a) Front View Projections (1 to 3)



(b) Edge Projections (4 to 6)



(c) Edge Projections (7 to 9)

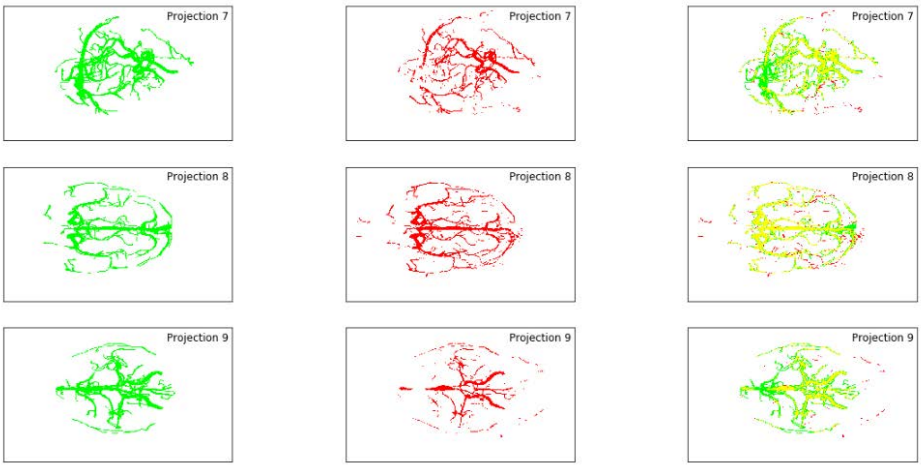
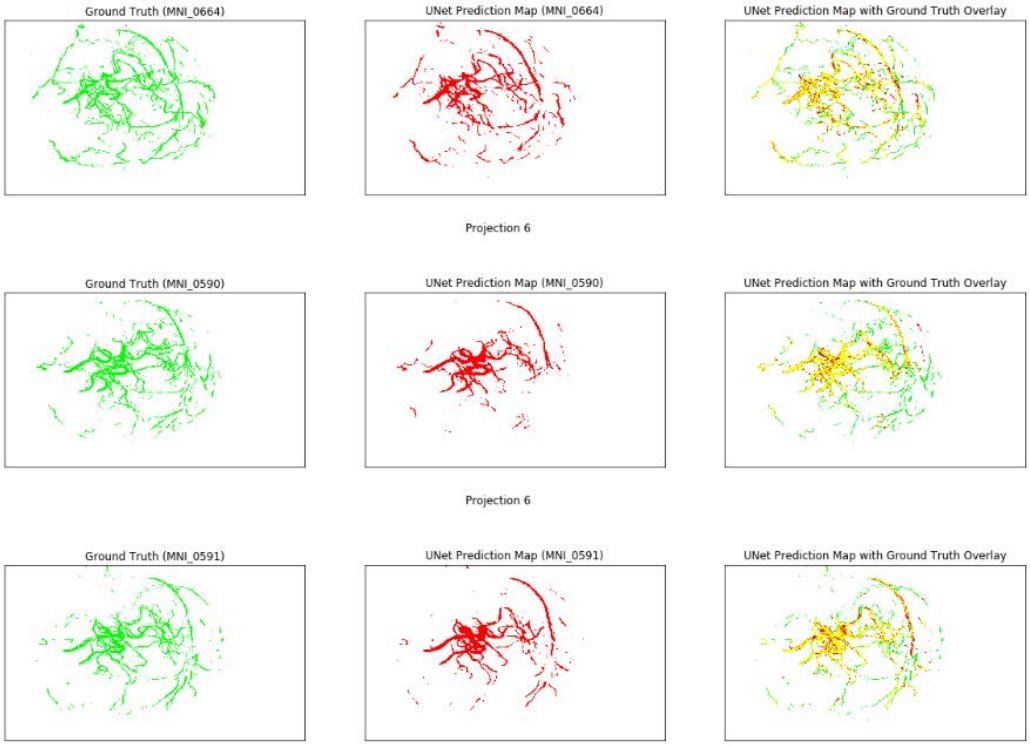


Figure B.8: MNI-0663 UNet 2D Segmentation Results with Ground Truth Overlays for Projections 1 through to 9

B.3.2 Frangi vs Manual: Ground Truth Data

(a) Frangi - GT, Prediction and Overlays



(b) Manual - GT, Prediction and Overlays

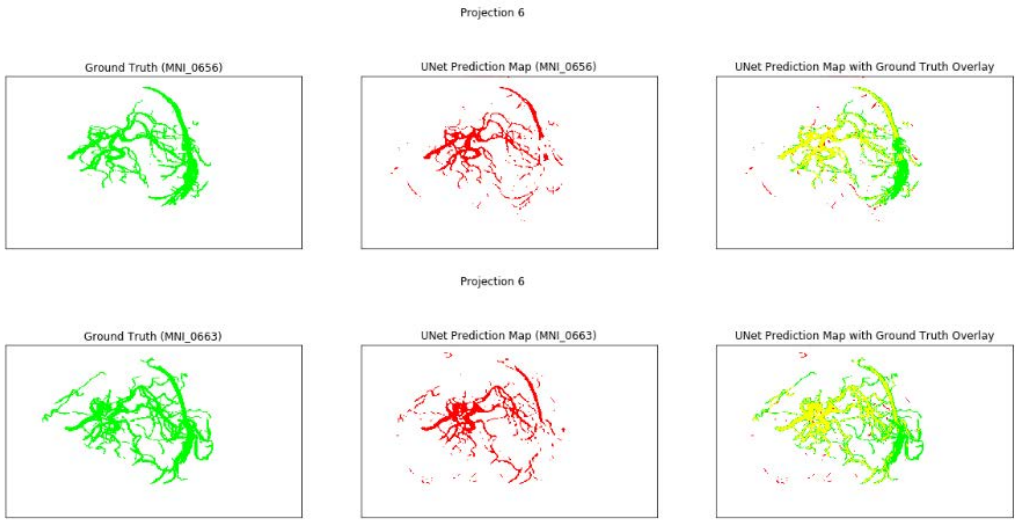


Figure B.9: Comparing Segmentation Maps for Projection 6: Frangi vs Manually Ground Truth

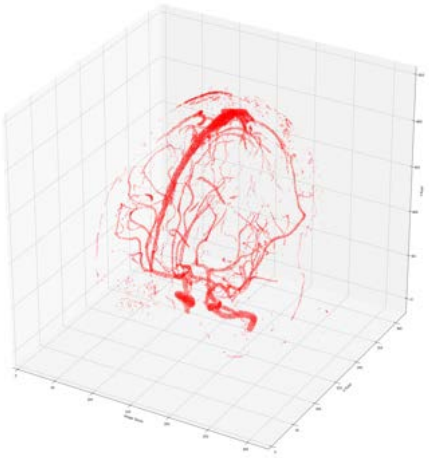
The segmentation results in Figure B.9 highlight, that the ground truth data, produced by the Frangi method, is not consistent with the manually segmented data sets. The results for several 'Projection 6' segmentations are compared in Figure B.9 and show that the manually segmented MRA scans have generally blood vessel segments with higher thickness.

B.4 3D Segmentation Maps

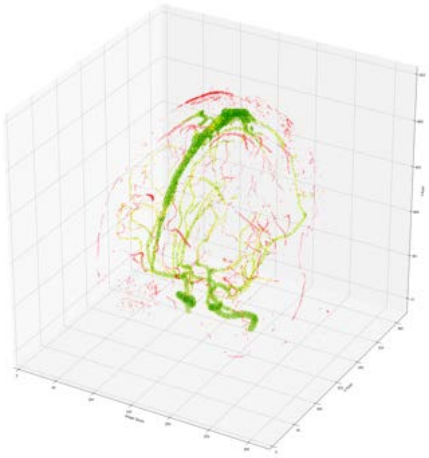
3D segmentation maps are attached for record MNI-0656 and are referred to repeatedly in Chapter 8 as parameter values are optimised in post-processing steps.

B.4.1 Parameter N-HITS

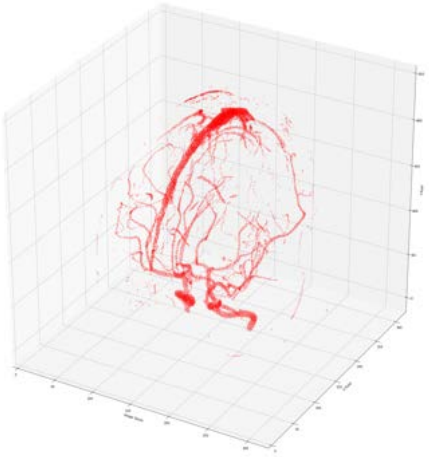
Figures B.10 to B.12 illustrate how accuracy decreases and recall increases as parameter N-HITS is moved up from 1 to 9. A relatively good balance between red and green misclassifications is achieved with the value N-HITS = 5.



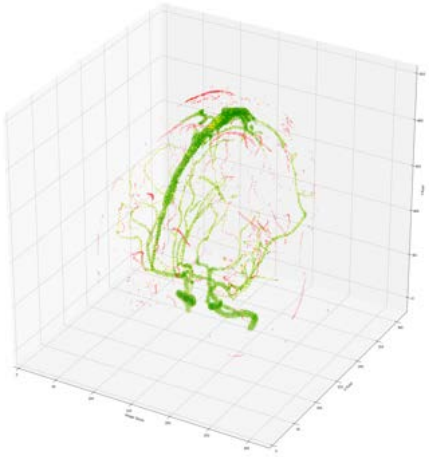
(a) N-HITS=1, 3D Prediction



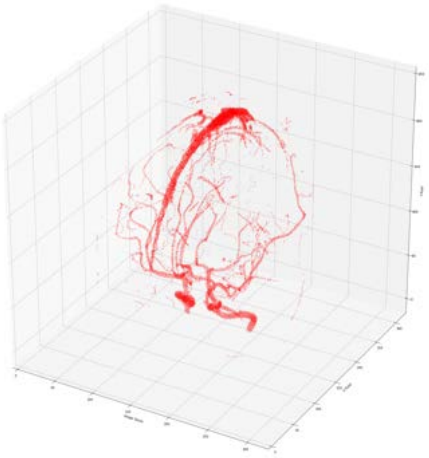
(b) N-HITS=1, 3D Pred. with Overlay



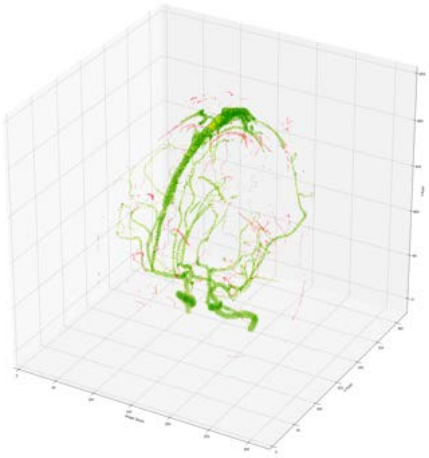
(c) N-HITS=2, 3D Prediction



(d) N-HITS=2, 3D Pred. with Overlay

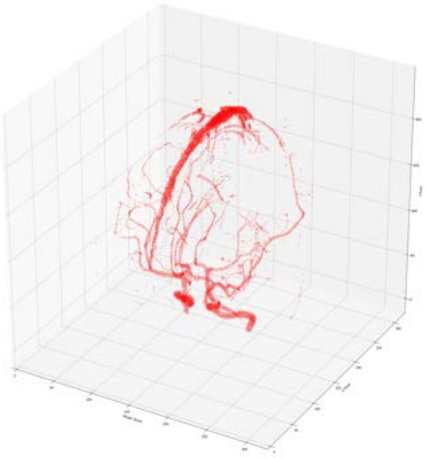


(e) N-HITS=3, 3D Prediction

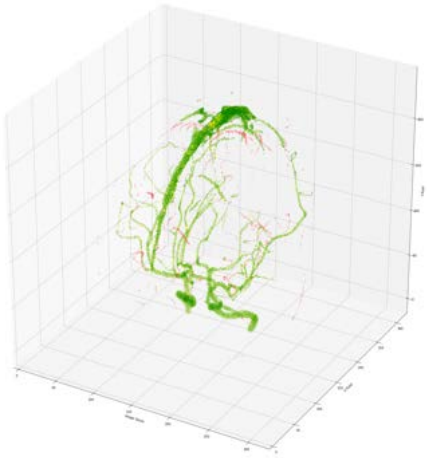


(f) N-HITS=3, 3D Pred. with Overlay

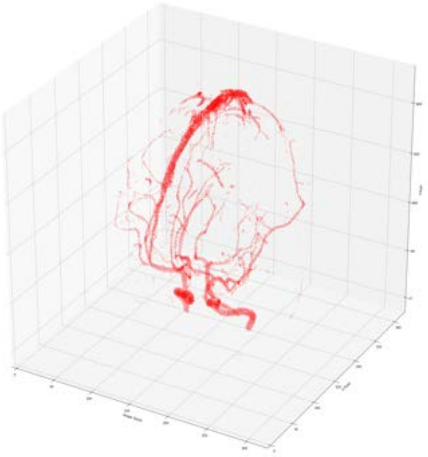
Figure B.10: MNI-0656: Running N-HITS from 1 through to 3 for 3D Segmentation Map



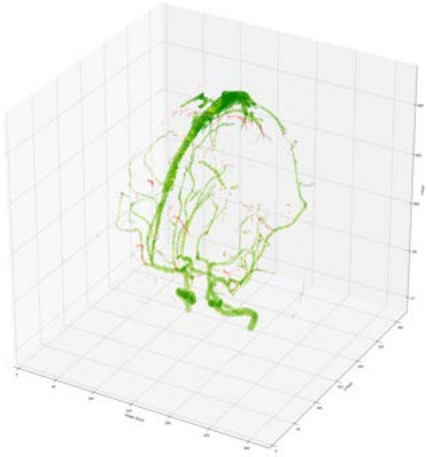
(a) N-HITS=4, 3D Prediction



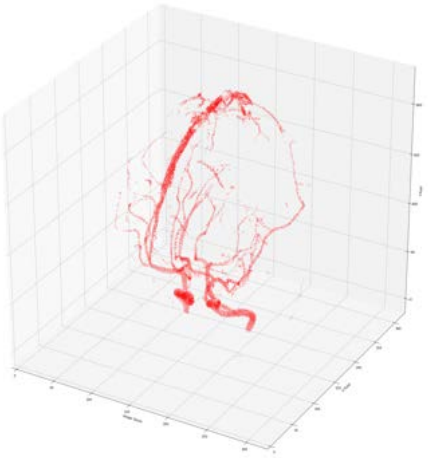
(b) N-HITS=4, 3D Pred. with Overlay



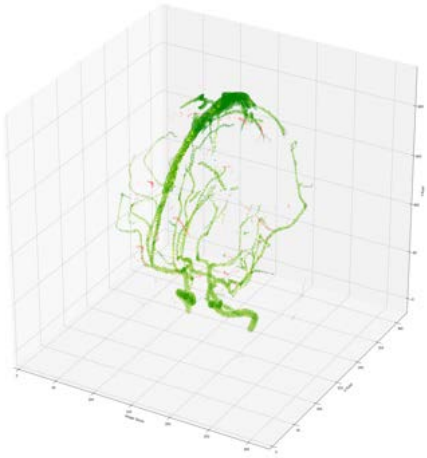
(c) N-HITS=5, 3D Prediction



(d) N-HITS=5, 3D Pred. with Overlay

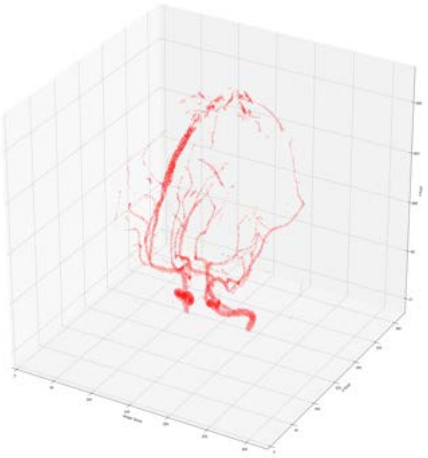


(e) N-HITS=6, 3D Prediction

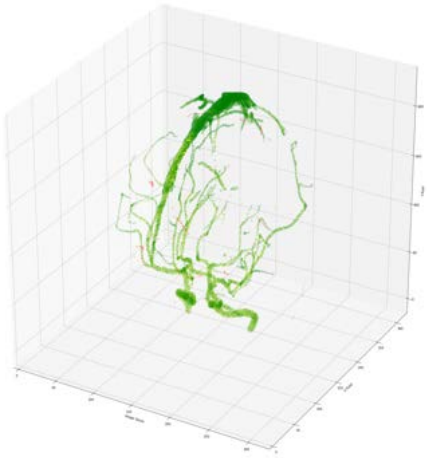


(f) N-HITS=6, 3D Pred. with Overlay

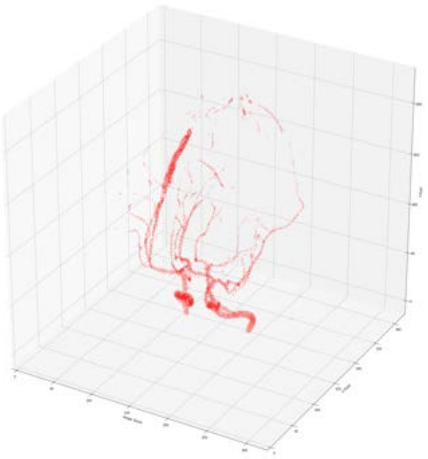
Figure B.11: MNI-0656: Running N-HITS from 4 through to 6 for 3D Segmentation Map



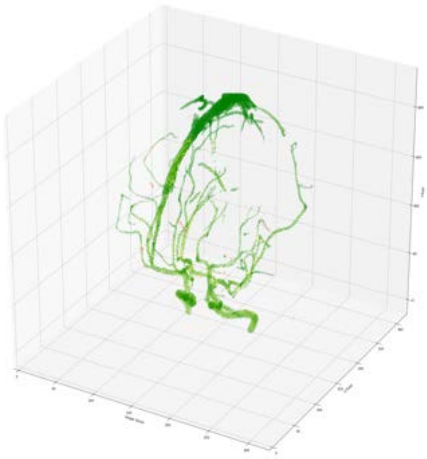
(a) N-HITS=7, 3D Prediction



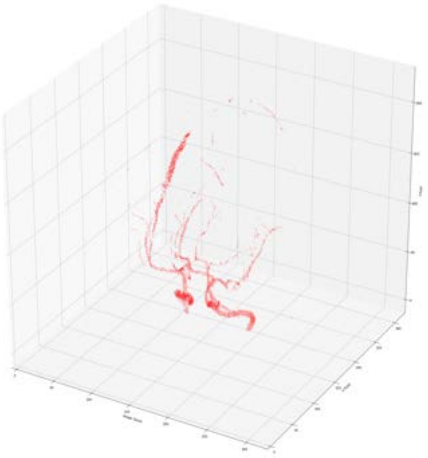
(b) N-HITS=7, 3D Pred. with Overlay



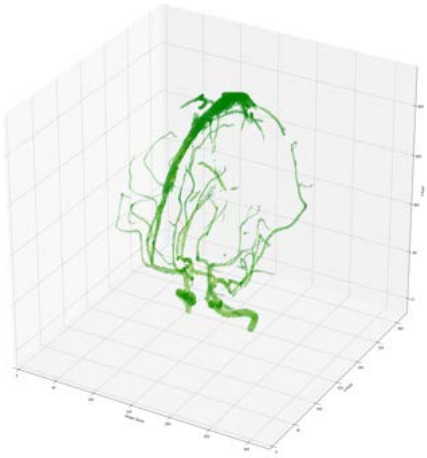
(c) N-HITS=8, 3D Prediction



(d) N-HITS=8, 3D Pred. with Overlay



(e) N-HITS=9, 3D Prediction



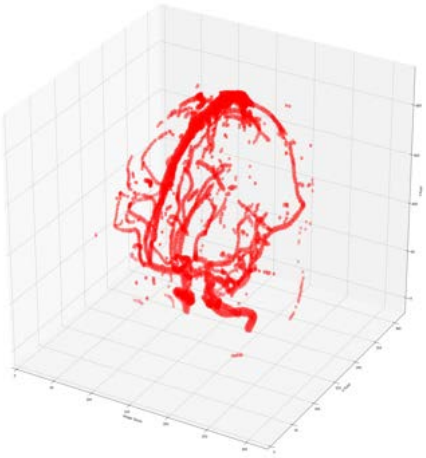
(f) N-HITS=9, 3D Pred. with Overlay

Figure B.12: MNI-0656: Running N-HITS from 7 through to 9 for 3D Segmentation Map

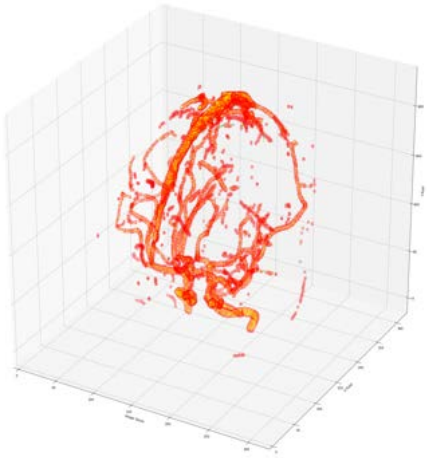
B.4.2 Parameter K-Neighbours

The first post-processing step fills up holes in the segmentation map with value N-HITS = 5. In this section, different values for K, and their impact on the 3D prediction map are visualised.

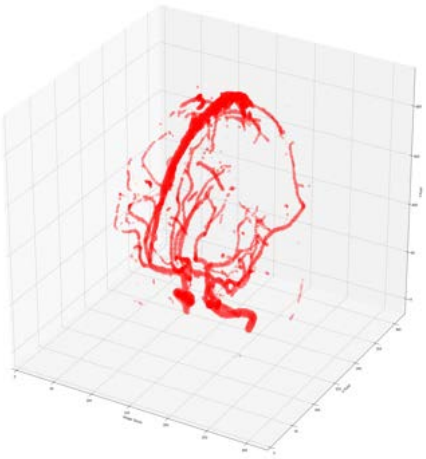
Figures B.13 and B.14 illustrate the results for the performance metrics in Figure 8.2: best segmentation results are achieved for parameter k with a value around 9.



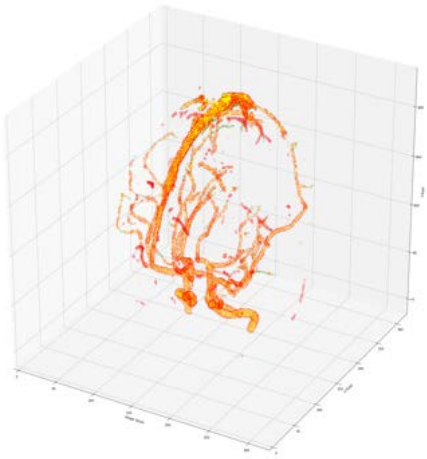
(a) K=1, 3D Prediction



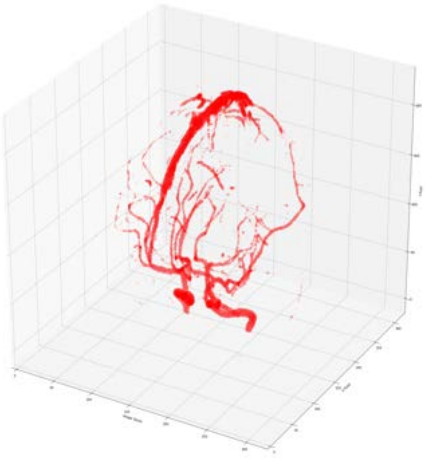
(b) K=1, 3D Pred. with Overlay



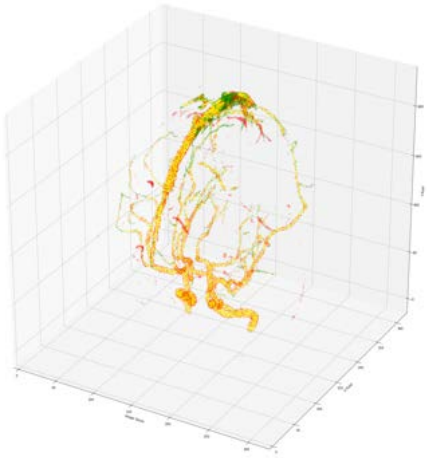
(c) K=3, 3D Prediction



(d) K=3, 3D Pred. with Overlay

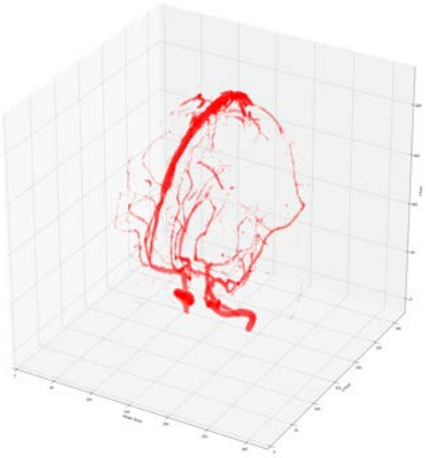


(e) K=5, 3D Prediction

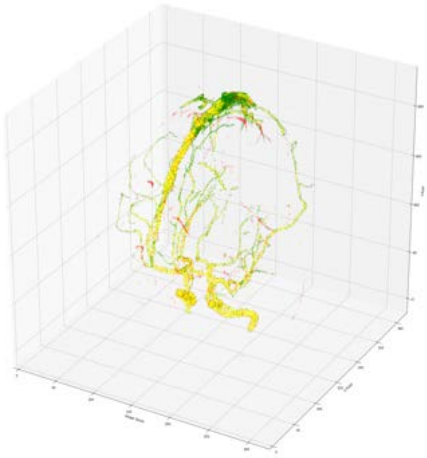


(f) K=5, 3D Pred. with Overlay

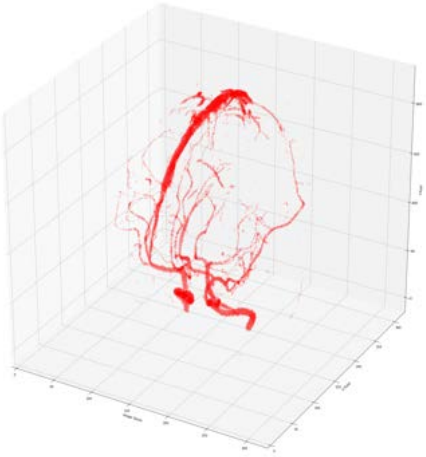
Figure B.13: MNI-0656: Running with K = 1, 3 and 5 for 3D Segmentation Map



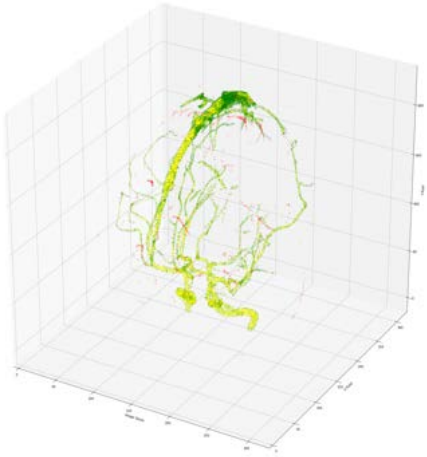
(a) K=7, 3D Prediction



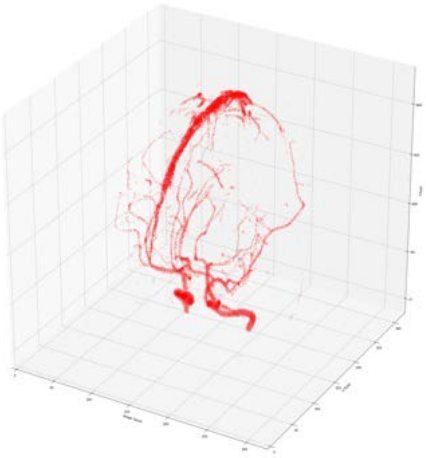
(b) K=7, 3D Pred. with Overlay



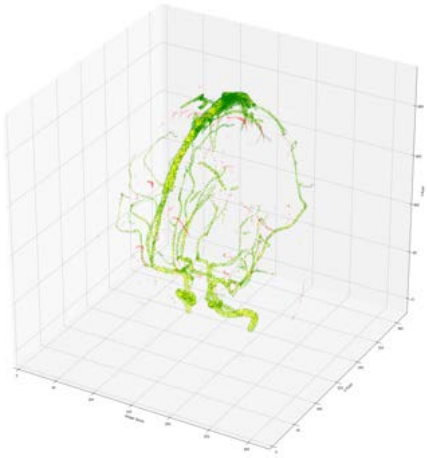
(c) K=9, 3D Prediction



(d) K=9, 3D Pred. with Overlay



(e) K=12, 3D Prediction

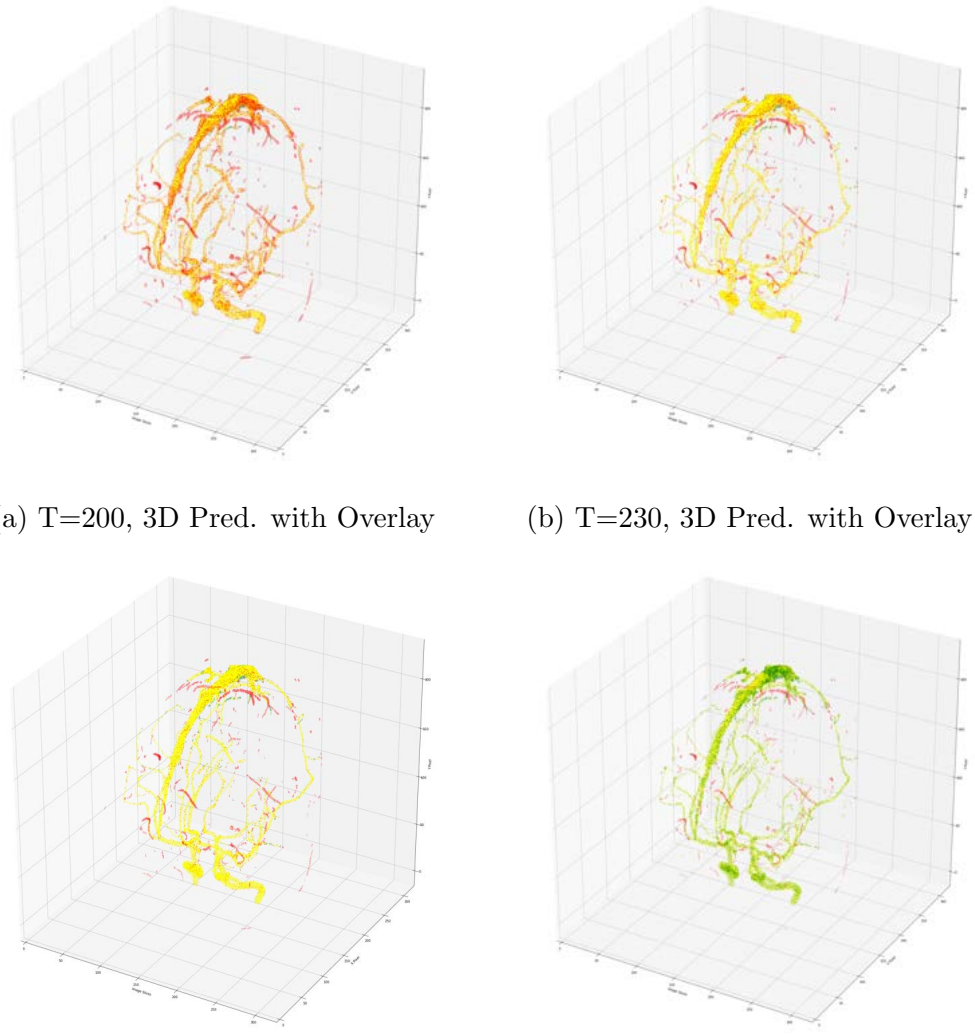


(f) K=12, 3D Pred. with Overlay

Figure B.14: MNI-0656: Running with K = 7, 9 and 12 for 3D Segmentation Map

B.4.3 Parameter Signal Strength Threshold T

Figure B.15 presents the overlays of ground truth with prediction map for various values of hyper parameter T. It can be seen, large yellow regions in the overlay maps are achieved with the value T = 241.



(a) T=200, 3D Pred. with Overlay

(b) T=230, 3D Pred. with Overlay

(c) T=241, 3D Pred. with Overlay

(d) T=250, 3D Pred. with Overlay

Figure B.15: MNI-0656: Running with K = 5 and T = 200, 230, 241 and 250 for 3D Segmentation Map

B.4.4 Final Segmentation Maps - MNI-0656 vs MNI-0663

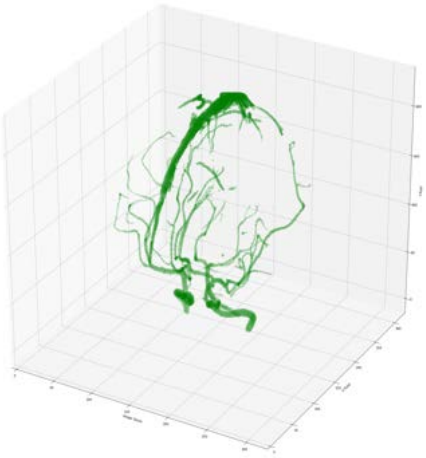
Figure B.16 compares the final segmentation results as they are achieved for the two manually segmented records MNI-0656 and MNI-0663.

The comparison identifies lower recall for record MNI-0663 than for MNI-0656, as there are larger green patches in the overlay of the former record.

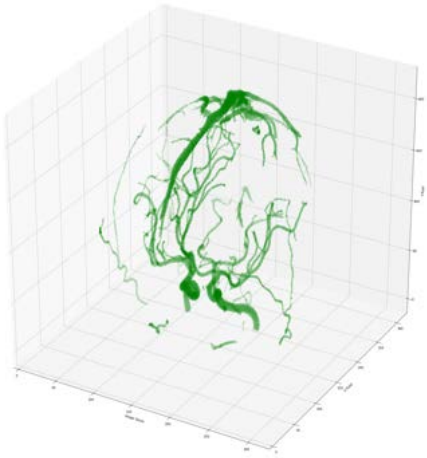
Record MNI-0656 to the left of Figure B.16 is used for validation and optimisation of hyper parameters.

Record MNI-0663 to the right of Figure B.16 is only used to test the final segmentation process against a formerly unseen record.

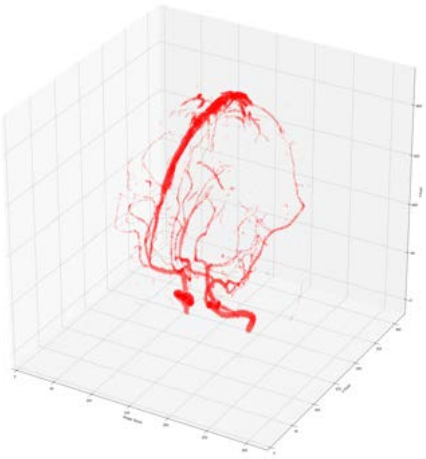
The segmentation maps for both records are of comparable quality, which suggests that the overall segmentation process does not overfit.



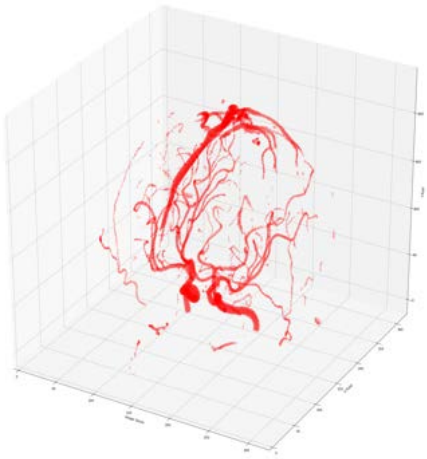
(a) Ground Truth - MNI-0656



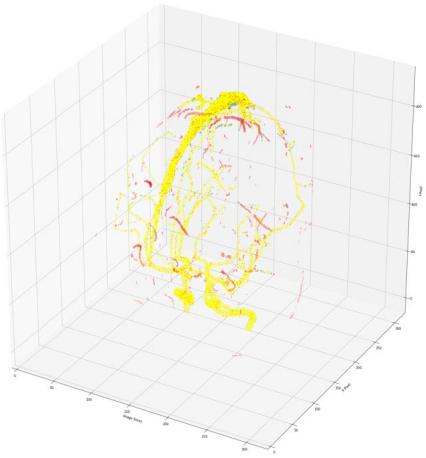
(b) Ground Truth - MNI-0663



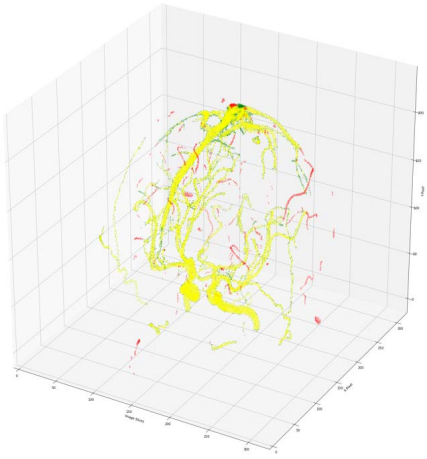
(c) Prediction - MNI-0656



(d) Prediction - MNI-0663



(e) Overlay - MNI-0656



(f) Overlay - MNI-0663

Figure B.16: Final Segmentation Results - Comparing Predicted Segmentation Maps for MNI-0656 and MNI-0663