Fully-Automatic Coronary Artery Segmentation using Deep Convolutional Autoencoders

Frode Jacobsen

Master of Science in Computer Science
Submission date: June 2018
Supervisor: Frank Lindseth, IDI

Norwegian University of Science and Technology
Department of Computer Science

Abstract

Automatic semantic segmentation of medical images is an important tool in aiding clinical experts in diagnosing diseases. The large volume of data produced in medical imaging modalities such as CT and MRI, and the inherent problem of working with volumetric data on a two-dimensional screen, makes manual or semi-automatic segmentation a time-consuming affair. This thesis explores the application of deep convolutional autoencoders to volumetric medical image segmentation. Three different convolutional autoencoder architectures are trained to perform binary and multi-class coronary artery segmentation, as well as liver and liver vessel segmentation.

Coronary artery disease is a leading cause of death in the world, and diagnosis requires invasive methods. Reasearch has been done on using computational fluid dynamics in order to simulate blood flow in the coronary arteries based on non-invasive medical imaging modalities. Fully-automatic extraction of precise geometry of the coronary arteries from coronary CT angiography could lead to computational fractional flow-reserve measurements becoming a viable alternative to the current gold standard of invasive FFR measurements. The neural networks are trained on a dataset provided by the St. Olavs Hospital FFR project. This dataset consists of manual segmentations of coronary CT angiography, and is still in development. The proposed methods are able to produce segmentations of coronary arteries of a high quality. Two of the models are also able to produce reasonable voxel-wise multi-class segmentations, correctly segmenting the right and left coronary branches, as well as the aorta in most of the validation images.

Additional validation is performed on the 3D-IRCADb-01 dataset from IRCAD. The dataset consists of CT images of the abdomen, with labels for most major organs and structures. The networks are able to learn to segment the complex vessel networks in the liver, as well as the liver itself, from a small dataset. The liver vessel segmentations contain some spurious responses, but still show that 2D convolutional autoencoders are viable for segmenting volumetric image data.

Sammendrag

Automatisk semantisk segmentering av medisinske bilder er et viktig verktøy for å bistå kliniske eksperter i diagnostisering av sykdommer. Medisinske avbildningsmetoder som CT og MRI produserer store mengder data, og det iboende problemet med arbeid med volumetriske bilder på en to-dimensjonal skjerm gjør manuell eller semi-automatisk segmentering svært tidkrevende. Denne masteroppgaven undersøker muligheten for å benytte dype konvolusjons-autoenkodere for å segmentere volumetrisk medisinsk billed-data uten menneskelig inngripen. Tre ulike konvolusjonsnettverk trenes for å segmentere koronararteriene, både for binær-segmentering og fler-klasse segmentering. De samme nettverkene trenes også til å utføre automatisk segmentering av lever og lever-arterier.

Iskemisk hjertesykdom er en sykdom som omfatter hjertekramper (angina pectoris) og hjerteinfarkt, og er en ledende dødsårsak i verden. Gull-standarden for diagnostisering av koronarsykdommer er invasiv måling av FFR–blodtrykksfall over en innsnevring i blodåren. Forskning gjøres på å benytte datamaskiner til å simulere blodstrømmen gjennom koronararteriene, for å på den måten kunne gjøre ikke-invasive FFR-målinger. Presis, helautomatisk uthentning av geometrien til koronararteriene fra CT bilder kan bidra til å gjøre simulert FFR til et mulig alternativ for diagnostisering av iskemisk hjertesykdom. I denne masteroppgaven vil kunstige nevrale nettverk bli trent på et datasett fra St. Olavs Universitetssykehus sitt FFR-prosjekt. Datasettet består av hånd-segmenterte CT-bilder av hjertet, og er fremdeles i utvikling. De foreslåtte metodene er i stand til å produsere segmenteringer av koronararteriene som holder høy kvalitet. To av modellene er også i stand til å segmentere og klassifisere høyre og venstre koronar-gren, samt aorta, i de fleste valideringsbildene.

Videre er modellene trent på 3D-IRCADb-01-datasettet fra IRCAD. Datasettet består av CT bilder av abdomen, hvor de fleste organer og større strukturer er håndsegmentert. Nettverkene er i stand til å lære hvordan å segmentere de komplekse blodåre-nettverkene i leveren, samt selve leveren, fra et lite datasett. Segmenteringene inneholder noen spuriøse responser, men viser hvordan 2D konvolusjonsautoenkodere er fullt brukbare for å segmentere volumetrisk bildedata.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Segmentation of an image is the process of assigning a label to every image point, associating it with a specific class or group such that all pixels with a given label share some common property [39]. Image segmentation is used in medical images for multiple purposes, such as diagnostics and surgery planning. Currently, the gold standard for medical image segmentation is hand-labeled segmentations performed by clinical experts. Semi-automatic methods for segmentation have been devised, and are useful tools for speeding up the process. Nonetheless, manual segmentation of medical images is a time-consuming process, that is also prone to human bias and error.

Fully-automatic methods for medical image segmentation would be of great use, as the volume of medical image data keeps increasing [36]. A satisfactory fully-automatic medical image segmentation tool could potentially reduce the time required to segment an image from hours to only a few minutes, or less. Such a tool would help free up valuable time spent by clinical experts, and allow for faster diagnostics in potentially time-critical situations.

This thesis explores the possibility of applying deep convolutional autoencoders to the task of fully-automatic medical image segmentation. Three different models, all variations of the same underlying concept, are tested on three different segmentation tasks: binary and multi-class coronary artery segmentation, and multi-class liver vessel segmentation. The gold standard for coronary artery disease (CAD) diagnosis is an invasive measurement of the fractional flow reserve (FFR) in the coronary arteries. Segmentation of the coronary arteries can be used for non-invasive diagnosis of CAD through the application of computational fluid dynamics (CFD) [43]. Liver and liver vessel segmentations are commonly used when planning liver transplant surgeries, as it helps visualize and guide how the procedure should be performed.

In this thesis improvements to a previously proposed method for fully-automatic segmentation [24] by using deep convolutional autoencoders based on the U-net architecture [34] is proposed. Furthermore, the possibility of multi-class segmentation of both coronary arteries and liver vessels is explored. Multi-class segmentation with neural networks allows for extraction of multiple features in a single pass using a single model, as opposed to multiple passes using several models trained to segment one structure each.

The networks are trained on two different datasets. The coronary artery dataset is provided by St. Olavs Universitetssykehus, developed as a part of their ongoing computational FFR project. The dataset currently consists of 51 hand-labeled coronary CT angiography images, and is still being developed. The images include segmentations for the right and left coronary artery (RCA and LCA), and ascending aorta (AA). The dataset used for training the networks on liver vessel segmentation was compiled by IRCAD - Institut de recherche contre les cancers de l'appareil digestif, and includes 20 CT images of the abdomen. Each image has several associated segmentation volumes, where most of the major organs and structures are labeled.

1.1 Research Questions

This thesis seeks to improve on previously developed methods for fully-automatic coronary artery segmentation using deep learning, and furthermore investigate the possibilities of multi-class segmentation of 3D medical images. This gives rise to the following research questions:

1. Is the deep convolutional autoencoder a viable neural network architecture for segmentation of the coronary arteries?
2. Are deep convolutional autoencoders able to perform satisfactory single-pass multi-class segmentation of medical images?
3. Is there any significant gain in using 3D convolutional autoencoders as opposed to 2D convolutional autoencoders when segmenting volumetric images?

1.2 Report Structure

This section presents the structure of the thesis, and gives a brief overview of the contents of each chapter.

Introduction

This chapter provides an introduction to the thesis, and describes the goals and motivation behind the research questions.

Background

The background chapter gives provides an overview of the central concepts used both as the motivation behind the thesis, as well as for developing the proposed method for fully-automatic segmentation. A brief overview of coronary arteries and coronary artery disease is given, as well as the key concepts behind computational FFR analysis. After this, the basic theory behind neural networks, as well as evaluation metrics and algorithms for data augmentation, is presented in order to give the reader an introduction to the challenges related to deep learning, and introduce concepts necessary for evaluating the results. Finally, the literature review provides an overview of related work done in the field of deep learning and image segmentation, as well as existing methods for assisting in the task of medical image segmentation.

Method

This chapter provides an in-depth description of the proposed method for fully-automatic image segmentation. Details surrounding the neural network architectures is presented first, followed by a description of the training setup, including a description of the different experiments that were carried out. A description of the different full-volume segmentation approaches is then given. Finally, the different datasets, and pre- and post-processing procedures are presented.

Results

The results chapter provide detailed results of the experiments. Quantitative results are given based on the Dice Similarity Coefficient (DSC)[7] between both raw/refined network outputs and the hand-labeled ground truth segmentations. Visualizations of network segmentation output are presented in order to facilitate a qualitative evaluation of the networks' performance.

Discussion

In this chapter the results of the experiments are discussed, and the proposed methods are evaluated based on the performance in the different tasks. Problems regarding quantitative evaluation of network segmentations in certain tasks are identified and discussed, and an attempt at qualitative assessment of performance through visual inspection of network segmentations is given.

Conclusion and Future Work

The final chapter provides a conclusion drawn from the results of the experiments in light of the research questions. Finally, possibilities for future work is proposed and briefly discussed.

Chapter 2

Background

2.1 Coronary Arteries and Coronary Artery Disease

The coronary arteries are the blood vessels providing the heart muscle with oxygenated blood. They consist of two main branches, the right and left coronary arteries (RCA and LCA). Coronary artery disease(CAD) is a term encompassing diseases such as stable and unstable angina, and myocardial infarction. CAD is one of the leading causes of death in the world [3], and is caused by build-up of plaque in the coronary arteries. The plaque causes a narrowing of the blood vessels (stenosis), resulting in decreased blood flow to the heart muscle. The reduced oxygen-levels damages the muscle tissue, and over time can lead to partial or complete failure of the heart muscle.

Diagnosis of coronary artery disease is today done through an invasive procedure, where a catheter is introduced into the coronary arteries in order to measure pressure drops across the stenosis and determine the fractional flow reserve (FFR). FFR is computed by measuring the blood flow pressure up- and downstream (proximal and distal) of the stensosis, and is averaged throughout a cardiac cycle. FFR is specified as

$$\text{FFR} = \frac{\overline{P}_{distal}}{\overline{P}_{proximal}} \tag{2.1}$$

and is used to determine whether a stenosis resists blood flow enough to cause reduced oxygen-levels in the heart muscle.

Non-invasive methods for determining the fractional flow reserve based on coronary computed tomography angiography (CCTA) have been developed [43], and are in continuous development. Computational estimation of FFR is based on extracting the geometry of the coronary arteries from a CCTA image. This geometry can be used to run computational fluid dynamics (CFD) simulations of the blood flow. The CFD methods provide pressure estimates which can then be used to determine the presence, and degree, of a stenosis.

In order to get the geometry for computing FFR, volumetric images have to be segmented by a clinical expert. This is a time-consuming task, and can take upwards of 2-3 hours per image. Segmentation of medical images is also used in other areas such as

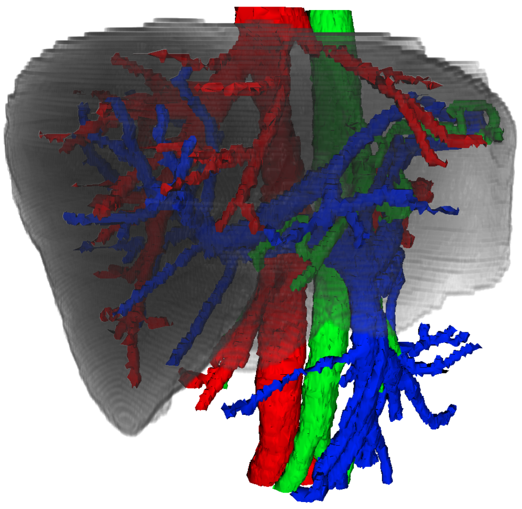


Figure 2.1: Rendering of labeled volume from the 3D-IRCADb-01 dataset showing liver and liver blood vessels. Hepatic vein shown in red, portal vein in blue, and hepatic arteries in green.

surgery planning. For liver transplants, having precise segmentations of the liver and liver blood vessels such as the hepatic vein (HV) and portal vein (PV) (see Fig. 2.1) helps guide surgery.

2.2 Neural Networks and Deep Learning

Neural networks was a very popular research topic in the early 1990s, before falling out of favor in the early 2000s. The last 10 years, as computing power has continued to increase, and the advent of powerful GPUs designed specifically for the purpose of doing the heavy lifting of matrix calculations, neural networks have again become the state-of-the-art method for many different machine learning tasks. There is a multitude of books on the topic, e.g. [14, 16], explaining in great detail the inner workings of artificial neural networks, but an introductory overview will be given below.

2.2.1 Artificial Neural Networks

Artificial neural networks consists of artificial neurons arranged in a network structure, with the artificial neurons feeding their output into the inputs of other artificial neurons. Figure 2.2 gives a visual representation of a simple feed-forward network, where data enters the network through the input layer and is passed through layer-by-layer to the output layer.

Artificial Neurons

Artificial neurons are mathematical functions inspired by biological neurons. They compute a weighted sum of its inputs (Eq. 2.2), and applies a non-linear transformation

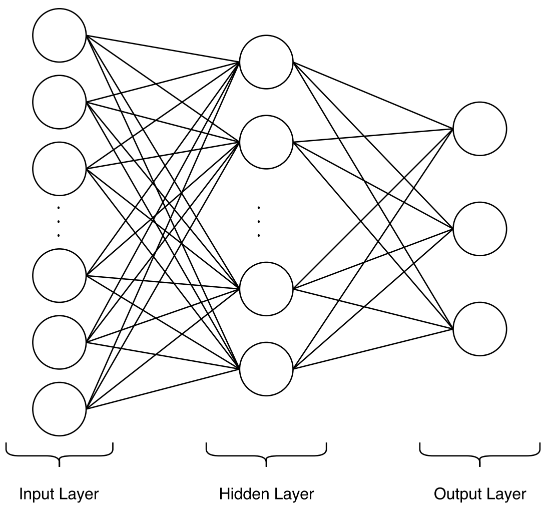


Figure 2.2: Visual representation of an artificial neural network, with an input layer feeding into a hidden layer, which again is feeding into the output layer.

of the weighted sum using an activation function (Eq. 2.3).

$$z = f(\boldsymbol{x}, \boldsymbol{w}, b) = \sum_{i=0}^{N} w_i x_i + b \tag{2.2}$$

$$y = \phi(z) \tag{2.3}$$

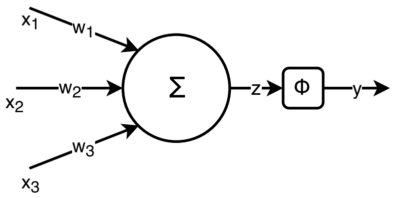


Figure 2.3: Diagram of artificial neuron computing a weighted sum of its input.

Activation Function

Activation functions are non-linear transformations applied to the result of the weighted sum. It has been shown that by using non-linear transformations neural networks are universal approximators [19], able to approximate any computable function.

Two widely used activation functions are the logistic function, also known as the sigmoid function, and the hyperbolic tangent:

$$\phi(z) = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.4}$$

$$\phi(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.5}$$

These functions transforms inputs from the $(-\infty,\infty)$ range into the $(0,1)$ and $(-1,1)$ range, respectively. They have somewhat fallen out of use as activation functions in hidden layers, as their derivatives has a tendency to produce poor gradients when a neuron approaches saturation (function output comes very close to the upper or lower bounds of the output range). An activation function that has been shown to provide good results for deep neural networks [12] is the rectified linear unit (ReLU) [15]. The ReLU is defined as:

$$\phi(z) = \max(0, z) \tag{2.6}$$

The ReLU is a closer approximation of the excitation response of biological neurons than the logistic function and the hyperbolic tangent [12], and its use drives networks to learn more sparse intermediate representations of data. The sparse representations are more easily linearly separable, and, during training, the sparse activations cause gradients to flow along the paths of non-zero activation, applying updates to the 're-sponsible' neurons. This has an effect similar to an ensemble model, causing a single network to consist of multiple, specialized sub-networks.

For classification tasks with multiple classes, where the goal is for the network to output a label classifying the input as belonging to some class $c \in C$, it is common to use the softmax activation function in the last layer of a network. The softmax activation is given by

$$\text{softmax}(\boldsymbol{z})_c = \frac{e^{z_c}}{\sum_n e^{z_n}} \tag{2.7}$$

and corresponds to the probability distribution of the input belonging to any class in C.

The activation functions are commonly used in a vectorized form, where $\phi(\boldsymbol{z})$ applies the activation function on each element of $\boldsymbol{z}$:

$$\phi(\boldsymbol{z}) = [\phi(z_0), \phi(z_1), \ldots, \phi(z_n)] \tag{2.8}$$

Network Layers

As shown in Figure 2.2, artificial neurons can be structured in layers, where each neuron in a layer feeds its output into the input of the neurons in the next layer. Let $l : \mathbb{R}^N \to \mathbb{R}^M$ be a function representing a layer, where N is the number of elements in the input vector $\boldsymbol{x}$, M is the number of neurons in the layer, and $\boldsymbol{\theta}$ represents the model parameters:

$$l(\boldsymbol{x};\boldsymbol{\theta}) = [\phi(f(\boldsymbol{x}, \boldsymbol{w_0}, b_0)), \phi(f(\boldsymbol{x}, \boldsymbol{w_1}, b_1)), \ldots, \phi(f(\boldsymbol{x}, \boldsymbol{w_n}, b_n))]^\top \tag{2.9}$$

Each individual layer is given by:

$$l^{(L)}(\boldsymbol{x};\boldsymbol{\theta}) = [\phi(f(\boldsymbol{x}, \boldsymbol{w_0^L}, b_0^L)), \phi(f(\boldsymbol{x}, \boldsymbol{w_1^L}, b_1^L)), \ldots, \phi(f(\boldsymbol{x}, \boldsymbol{w_n^L}, b_n^L))]^\top \tag{2.10}$$

By chaining layers together, a feed-forward network can be represented as a function F given by

$$F(\boldsymbol{x};\boldsymbol{\theta}) = l^{(N)}(l^{(N-1)}(\ldots l^{(1)}(\boldsymbol{x}))) \tag{2.11}$$

Backpropagation and Gradient Descent

Training of neural networks is done through iterations of calculating network output for a given input, comparing given output to a ground truth using a *cost* or *loss* function, and updating the parameters of the network (weights and biases) in such a way that the output of the next iteration comes closer to the ground truth.

Backpropagation is an algorithm for efficiently calculating the derivative of the loss function with respect to every weight and bias in the network. Gradient descent is a method for updating model parameters in such a way that it will minimize the value of the loss function for the given input. Mini-batch stochastic gradient descent (SGD) is an extension of gradient descent, where updates are performed based on the average gradient $\hat{\boldsymbol{g}}$ of several randomly selected training samples. The average gradient of a batch of training samples is given by

$$\hat{\boldsymbol{g}}(\boldsymbol{x};\boldsymbol{\theta}) = \frac{1}{m}\nabla_{\boldsymbol{\theta}}\sum_{i=1}^{m} C(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_i) \tag{2.12}$$

where $\hat{\boldsymbol{y}} = F(\boldsymbol{x};\boldsymbol{\theta})$ is the network output given input $\boldsymbol{x}$ and model parameters $\boldsymbol{\theta}$, and $C(\hat{\boldsymbol{y}}, \boldsymbol{y})$ is some loss function calculating the error in network output with respect to ground truth $\boldsymbol{y}$. The gradient can then be used to update network parameters using an update rule. For mini-batch SGD, the simplest update rule is given as

$$\boldsymbol{\theta_{i+1}} = \boldsymbol{\theta_i} - \eta\hat{\boldsymbol{g}}(\boldsymbol{x};\boldsymbol{\theta_i}) \tag{2.13}$$

where η is a scaling factor known as the learning rate, governing the magnitude of the updates performed in each step. Determining an appropriate value for the learning rate can be difficult. If the learning rate is too high, the global minimum of the loss function might never be found, as the large updates 'skip' across it. If the learning rate is too low, training will take a very long time. Too low learning rates may also lead to the model never finding the global minimum, as the it gets 'stuck' in local minima. One way to adress this problem is by using a learning rate schedule, where the model starts out training using a given learning rate, and at specific times during training the learning rate is reduced to some lower value. This allows a network to quickly converge using larger updates, and at later stages fine-tune the weights by small updates. This is however not a perfect fix, as it introduces new problems by having to decide when, and by how much, the learning rate shoul be reduced. Another way to adress this issue is by changing the update rule. The update rule given in Eq. 2.13 can be modified by adding a momentum term. This means that previous updates are taken into account when calculating the current update. Rewriting Eq. 2.13 as

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \Delta\boldsymbol{\theta}_i \tag{2.14}$$

and letting $\Delta\boldsymbol{\theta}_i$ be defined as

$$\Delta\boldsymbol{\theta}_i = -\eta\hat{\boldsymbol{g}}(\boldsymbol{x}_i;\boldsymbol{\theta}_i) + \rho\Delta\boldsymbol{\theta}_{i-1} \tag{2.15}$$

we have included previous updates in the update rule, scaled by ρ. Another way to think of it is that $\Delta\boldsymbol{\theta}$ is the velocity of the network, and ρ is its mass, and changes to the velocity is resisted by its momentum [31]. Other update rules, such as AdaGrad [5], AdaDelta [46], or Adam [23] compute adaptive learning rates for each parameter by keeping track of previous gradients and scaling the learning rate individually for each parameter according to given rules.

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) use the same basic elements as ANNs, artificial neurons with weights and biases structured in layers, but differ in how these layers are connected. CNNs use localized connections, a way of routing outputs from one layer to the next in a manner equivalent to discrete convolution. Discrete convolution in two dimensions of a function I by a convolution kernel K is given by

$$(K * I)(x, y) = \sum_i \sum_j I(x - i, y - j) K(i, j) \tag{2.16}$$

and similarly for three dimensions

$$(K * I)(x, y, z) = \sum_i \sum_j \sum_k I(x - i, y - j, z - k) K(i, j, k) \tag{2.17}$$

Convolution layers of a CNN organize neurons into one or more *feature maps*. In a 2D CNN the feature maps will be a three-dimensional array with shape (width, height, channels). The channels are used to represent information about each spatial location (x, y) in the image. Each feature map is obtained by convolving the output of the previous layer with a distinct convolution kernel. Activations of feature map m in layer l is given by

$$\boldsymbol{y}_l^m = \phi(\sum_n (\boldsymbol{y}_{l-1}^n * \boldsymbol{K}_l^{m,n}) + b_l^m) \tag{2.18}$$

Convolutional layers utilize parameter sharing in order to drastically decrease the number of parameters needed for each layer. Each feature map has its own unique set of weights that is applied to every spatial location of the input. The number of weights for each feature map is thus only dependent on the kernel size and the number of channels in the input to the layer. Shared parameters also introduce shift-invariance, meaning that any given feature map will detect if a certain feature is present anywhere in the input.

Another difference between CNNs and ANNs is the use of pooling layers. Pooling layers are used to effectively downsample the input according to some rule. The max-pooling layer for example uses a window or receptive field of some size and a stride, and outputs the maximum value found in the receptive field. This window is moved across the input according to the stride, and individually for each feature map.

Two-dimensional CNNs can be extended to 3D by adding a third spatial dimension and modifying each operation accordingly. This means that the data flowing through the network will be four-dimensional with a shape of (width, height, depth, channels).

Convolutional neural networks have performed well in several image classification tasks. The ImageNet challenge (ILSVRC) [35] is a benchmark challenge for image classification models, and has been dominated by CNNs [38, 42, 18] since ILSVRC-12 was won by a deep convolutional neural network [25].

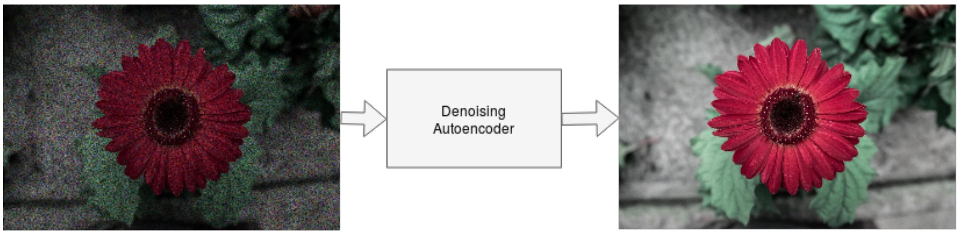


Figure 2.4: Diagram of a de-noising autoencoder

2.2.3 Convolutional Autoencoder

In principle, an autoencoder is a function that takes some data as input, encodes it into a representation (encoding) of the data, before decoding it to restore the original data. In its purest form, the autoencoder can be used to learn useful representations of data, allowing for dimensionality reduction and thus data compression. An autoencoder can also be trained to decode a representation into a modified version of the original. As an example an autoencoder can be trained to de-noise images by training it on noisy images and comparing the output to the original, non-noisy, images, as illustrated in figure 2.4.

Convolutional autoencoders use convolutional layers to encode the input. Dimensionality reduction can be achieved either through max-pooling layers or strided convolutional layers. In order to restore the encoding to its original, the encoding is scaled back up to its original size either through upsampling or deconvolution (transposed convolution). Convolutional autoencoders have been used for segmentation tasks [34, 28, 4], where precise spatial localization as well as classification is important.

2.3 Evaluation Metrics

Pixel- or voxel-wise segmentation tasks is the task of assign a class label $c_i \in C$ to each pixel or voxel in an image. Each image will have a corresponding ground-truth label image, where each pixel or voxel is assigned a 'true' class label. It is common to use a catch-all background label (normally given the value 0) for each pixel or voxel not belonging to a structure or semantic region of interest.

In binary classification tasks a pixel or voxel can either belong to the background, or to the structure of interest (foreground). A *positive* sample is a sample which the ground truth classifies as belonging to the foreground, and a *negative* sample belongs to the background. A prediction in any given binary classification task will have four possible outcomes when compared to the 'true' class:

TP: True Positive – a *positive* sample, predicted to be a *positive* sample

FP: False Positive – a *negative* sample, incorrectly predicted to be *positive*

TN: True Negative – a *negative* sample, predicted to be a *negative* sample

FN: False Negative – a *positive* sample, incorrectly predicted to be *negative*

For multi-class segmentation tasks, where the number of classes is greater than two, the different outcomes must be seen with regards to each individual foreground class in turn. In other words, the occurences of true positives, false positives, etc. must be counted individually for each foreground class.

Evaluation metrics based on the number of occurrences of the different outcomes have been devised, each giving different insights into the performance of the prediction model. If we let $\text{TP}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ be a function counting the number of true positives when comparing the prediction $\hat{\boldsymbol{y}}$ to the ground truth $\boldsymbol{y}$, and similarly for FP, TN, and FN, the *accuracy* of a prediction is determined by

$$\text{Accuracy}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.19}$$

Accuracy measures the ratio of correctly classified elements to the total number of elements, and for dense classification tasks, where the ratio of foreground to background elements is roughly even, gives a fairly precise evaluation of performance. However, for classification tasks where the number of foreground elements is heavily outweighed by the number of background elements, accuracy gives no meaningful performance evaluation; simply predicting everything as background will give a high ratio of correctly predicted elements.

Sensitivity is a measure of the ratio of elements classified as positive actually being true positives. Sensitivity is also known as *recall*, and is determined by

$$\text{Recall}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.20}$$

Recall can give insight into how good a model is at predicting that foreground elements belong to the foreground. However, it does not give any insight into the validity of such a model. A prediction model can achieve perfect recall simply by classifying all inputs as foreground elements, and as such high recall alone is not a good measure of performance. A negative prediction from a model with high recall, on the other hand, can be taken as a sign that the negative prediction has a high degree of certainty.

Specificity can be thought of as recall for the background labels, as it measures the proportion of correctly classified negatives. Specificity is defined as

$$\text{Specificity}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{2.21}$$

and gives the same insight as recall, only for negative samples. As with recall, specificity is not a good measure for the validity of negative predictions, but positive predictions from a model with high specificity has a high degree of certainty.

2.3.1 Dice Similarity Coefficient

All of the aforementioned evaluation metrics have issues when it comes to evaluating the performance of a prediction model on a sparse dataset. Specificity and recall cannot be used alone to determine performance, and accuracy will be heavily biased towards the background label giving no meaningful insights. For this reason, the Dice Similarity Coefficient (DSC) [7] is commonly used in segmentation tasks with heavy class-imbalance. With DSC, performance is determined by the ratio of correctly classified foreground elements to all positive predictions plus false negatives. It is defined as

$$\text{DSC}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{2.22}$$

and the disregarding of true negatives solves the issue of class-imbalance. Additionally, true positives are taken more in account than wrong predictions. When used as a loss function, this gives a prediction model incentive to attempt to predict at least some elements as positive, leading to clear gradients and increased performance.

2.4 Elastic Deformation

Elastic deformation [37] is a way of deforming an image by translating pixels or voxels from their original position to a new position by applying a displacement field on the image. The displacement field is created by, for each pixel or voxel, calculating a new target location using a displacement function Δ. For the two-dimensional case, we have displacement field $d : \mathbb{R}^2 \to \mathbb{R}^2$ given by

$$d(x, y) = [\Delta x(x, y), \Delta y(x, y)] \tag{2.23}$$

The displacement functions are given by convolving a random field $R_i(x, y) = \text{rand}(-1, 1)$ with a Gaussian convolution kernel, k, with standard deviation σ, and scaled by a scaling factor α:

$$\Delta x(x, y) = (R_x(x, y) * k(\sigma)) \cdot \alpha \tag{2.24}$$

$$\Delta y(x, y) = (R_y(x, y) * k(\sigma)) \cdot \alpha \tag{2.25}$$

where rand$(-1, 1)$ is a pseudo-random real-valued number between -1 and 1. The scaling factor is used to adjust the intensity of the transformation, with a scaling factor of N corresponding to shifts of at most N pixels in any direction. The standard deviation of the convolution kernel, σ, can be viewed as an elasticity coefficient, with lower values corresponding to more elasticity. For very low values of σ (extreme elasticity), the displacement functions closely resemble the random function R, and for very high values (no elasticity) the displacement functions become translations.

The new image $G(x, y)$ is created by shifting all voxels from the original image $I(x, y)$ by $d(x, y)$, such that

$$G(x, y) = I(x - \Delta x(x, y), y - \Delta y(x, y)) \tag{2.26}$$

All intensity values outside the domain of the original image I is treated as 0. For real-world applications, the image I is a discrete function only defined for $x, y \in \mathbb{Z}^+ \cup \{0\}$. For this reason it is necessary to use interpolation to find the intensity value when the displacement for a given pixel is a non-integer. Figure 2.5 shows a visualization of the random function R and the displacement functions.

Elastic deformation can be further parametrized by introducing a different scaling factor for each direction, giving a different range of deformation in each direction:

$$d(x, y, \boldsymbol{\alpha})[\Delta x(x, y, \alpha_x), \Delta y(x, y, \alpha_y)] \tag{2.27}$$

In the case of CT images, this allows for deformations more closely related to those observed in-vivo.

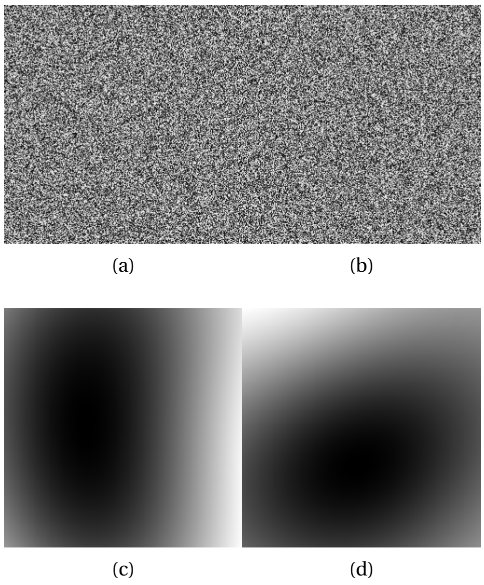


Figure 2.5: Visualization of displacement functions. (a) and (b) shows the random fields $R_x(x, y)$ and $R_y(x, y)$, (c) and (d) show displacement functions $\Delta x(x, y)$ and $\Delta y(x, y)$ obtained with $\sigma = 128$. Original image shape is 256x256 pixels. Black corresponds to displacements of $-\alpha$ and white to $+\alpha$.

2.5 Related Work

Long et al. [26] proposed a fully convolutional neural network for semantic segmentation, and laid the ground work for the use of CNNs in semantic segmentation tasks. He et al. [17] builds upon the steady improvement of the R-CNN model [11, 10], an approach to object detection that uses region-proposals and CNNs to evaluate and classify these proposals. The proposed method of He et al., dubbed Mask R-CNN, builds upon the work of Ren et al. [33] and their Faster R-CNN model by adding a parallel path that predicts a per-pixel semantic region for each class instance.

The deep convolutional autoencoder architecture originates from U-net, proposed by Ronneberger et al. [34]. It consists of a contracting and expanding path, with feature map forwarding between corresponding encoding and decoding steps. This network is able to produce voxel-wise classification of 2D input images of high quality. During training they employed a weighted loss function in order to incentivize the learning of correct border segmentations. The U-net architecture was extended to three dimensions by the introduction of 3D U-net [4] and V-net [28], by replacing the 2D convolutional layers and max-pooling layers by their 3D equivalent. Dou et al. [8] proposed a deeply supervised neural network for volumetric medical image segmentation. This network uses several deconvolutional pipelines in order to produce intermediate segmentations. The intermediate segmentations are then used during loss calculation in order to facilitate improved parameter updates. Ravi et al. [32] show that deep autoencoder architecture is slowly, but surely, gaining momentum, but is still a rather underused deep learning method in health informatics.

2.5.1 Coronary Artery Segmentation using Deep Learning

Moeskops et al. [29] performs multi-class voxel-wise segmentation of several different different structures in the human body using a deep convolutional neural network. The CNN is trained on MRI images of the brain in order to distinguish between six types of brain matter, as well as the pectoral muscle from breast MRI, and coronary arteries from coronary CTA. The networks was trained to perform several different combinations of these tasks, as well as all three tasks simultaneously. The proposed model works by doing voxel-wise segmentation using three 51x51 patches from the axial, sagittal and coronal directions, centered on the target voxel. The model was made up of 25 convolutional layers, transforming the 51x51x3 input to a 32x3 feature vector. The feature vector is then passed to a fully connected classification network with a softmax output layer.

Kjerland [24] proposed a method for automatic semantic segmentation of the coronary arteries using the DeepMedic [21] framework. The method is a dual-pipeline process, where one CNN is trained to segment the aorta, and another is trained to segment the coronary arteries. The reason for this dual-CNN pipeline, is that the raw output from the coronary artery CNN contains a lot of spurious responses. Segmentation refinement is performed by selecting the largest connected component from the output of the coronary CNN that overlaps with the output of the aorta CNN.

Nasr-Esfahani et al. [30] uses CNNs to segment coronary arteries from CT angiograms. Their method is a multi-stage process where the initial image is contrast enhanced before being fed into a CNN which produces an initial probability map of 'vesselness'. An edge-map is also created from the contrast-enhanced image. The edge-map and intitial prediction is fed into a second CNN that produce an improved probability map. Finally, this probability map is binarized and the final segmentation is produced by selecting the largest connected component in the binarized image.

Zeng et al. [47] propose an automated method for liver vessel segmentation, using 3D region growing and hybrid active contour model. Yang et al. [45] proposes a method for automatic seed point identification, and automated thresholding in order to extract segmentations of hepatic artery, hepatic vein (HV), portal vein (PV) and liver.

Chapter 3

Method

This chapter describes the experiments that were performed, and the methods used when obtaining the results. The neural network architectures used in the experiments will be presented along with a description of how they were trained and the hyperparameters used during training. Secondly, the datasets the neural networks were trained on will be presented, along with pre- and post-processing steps used when obtaining the final segmentations.

3.1 U-net Architecture

The original U-net architecture proposed by Ronneberger et al. [34] is a convolutional autoencoder consisting of an encoding path and a decoding path. The encoding path is made up by blocks consisting of two convolutional layers followed by a max-pooling layer. The decoding path consists of decoding blocks; an up-sampling layer followed by two convolutional layers. The feature maps from the corresponding encoding block is forwarded to and concatenated with the result of the up-sampling layer. This results in an architecture able to create high-level abstractions of features in the original image while retaining information on their specific location.

Experiments with the original U-net architecture on the St. Olavs dataset has shown that the deepest encoding block brings very little benefit when segmenting coronary arteries. This is not surprising, as the size of the coronary arteries has an upper diameter of approximately $\sim 4.5\,\text{mm}$ [9], or, with a voxel resolution of $0.4\,\text{mm}^3$, a maximum diameter of around 11 voxels. With 4 max-pooling layers, as used in the original architecture, each pixel in the deepest encoding block encodes information about a 16x16 area in the original image. This means that precise localization information about structures smaller than 16x16 pixels is lost.

3.1.1 2D U-net Architecture

Here, a modified version of the U-net architecture is proposed. A diagram of this architecture can be seen in Figure 3.1. Compared to the original U-net, the deepest encoding block is removed from the network and a third convolutional layer is added to each encoding and decoding block. The additional convolutional layers result in the network having roughly the same number of trainable parameters as the original U-net proposed by Ronneberger et al., but organized in such a way that they contribute more to the final prediction. Each convolutional layer is also followed by a batch-normalization layer. Additionally, dimensions are preserved during convolutions, resulting in an output image with the same shape as the input image. This means that the edge pixels in the feature maps and the final segmentations are not truly valid, as the previous feature maps must be padded in order to keep the dimensions. This is only done for convenience, and should not affect the final segmentation as the coronary arteries are generally somewhat centered in the input image, with the edge pixels containing no structures of interest.

The network is structured in an encoding pipeline and a decoding pipeline. The encoding pipeline consists of four encoding blocks, where each encoding block passes a 2x2 max-pooled feature map down to the next encoding block, and forwards a non-max-pooled feature map to the corresponding decoding block in the decoding pipeline. The encoding blocks are made up of three convolutional layers. The convolutional layers consists of a 3x3 convolution, followed by batch normalization before ReLU activation. The last convolutional layer in each block doubles the number of feature map channels. The first encoding block starts out with 64 channels, and ends up with 128. This continues down to the fourth and deepest encoding block, resulting in a feature map with 1024 channels.

The decoding pipeline is made up by decoding blocks. The decoding blocks have an initial 2x2 'de-convolutional' layer, a transposed convolution effectively up-sampling the feature map input to the layer, doubling the size of the feature map. The de-convolutional layer also halves the number of feature map channels. The up-sampled feature map is then concatenated with the corresponding feature map forwarded from the encoding pipeline. The resulting feature map has the same number of channels as the feature map before deconvolution. The concatenated feature map is then fed through a convolutional layer that halves the number of feature map channels again, before passing it through two more convolutional layers. Batch-normalization is applied to each convolution and deconvolution, before ReLU activation. The last decoding block is slightly different, as it successively halves the feature map channels until reaching 64 channels. These 64 channels are then run through a 1x1 convolutional classification layer, reducing the number of channels down to N classes. Total parameter count for the BVNet architecture is $24,357,057$, with $24,342,593$ being trainable parameters.

3.1.2 3D U-net Architecture

The 3D U-net architecture was proposed by Çiçek et al. [4], and is a 3D adaption of the U-net architecture proposed by Ronnerberger et al. As with BVNet, the architecture consists of encoding and decoding blocks, with a total of four max-pooling (and corresponding deconvolution) layers. However, each block consists of two, not three, convolutions. The architecture uses three-dimensional (de-)convolution and pooling in order to work on volumetric data. Çiçek et al. showed that coupled with a weighted softmax loss layer, the architecture was able to quickly learn dense segmentations from sparsely annotated training data.

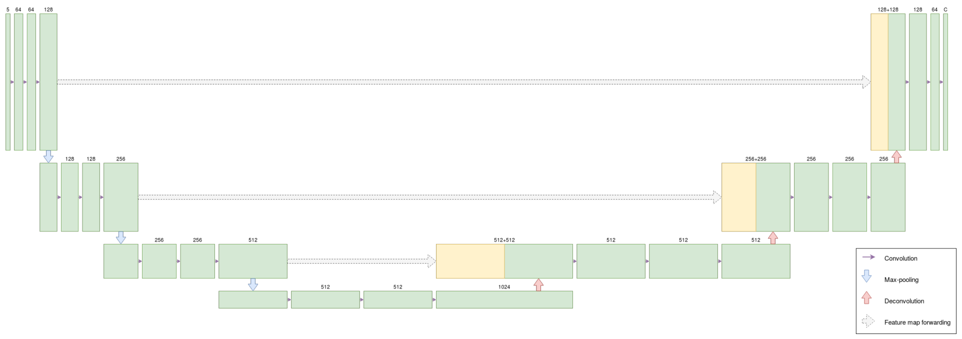


Figure 3.1: Diagram of BVNet architecture. Green rectangles represent feature maps, with number of channels noted above. Yellow rectangles represent the forwarded feature maps. For a high-resolution diagram, the reader is referred to the digital version of this report.

The additional dimension in the data means that each voxel in the network output has access to a lot of spatial contextual information, and may therefore be able to learn rich features describing how structures in the input volume relate to eachother.

3.1.3 Implementation Details

The neural networks were implemented using the Keras [6] framework, an open-source high-level API to multiple tensor computation libraries such as Tensorflow [1] or Theano [44]. For this implementation, the Tensorflow backend was used. The networks were trained on NVIDIA Tesla P100 GPUs.

3.2 Loss function

The datasets the networks are trained have a very low foreground-to-background voxel ratio. This means that many of the commonly used loss functions like mean-squared-error or cross-entropy would report very low loss as long as most of the background labels are correctly predicted. Taking the coronary artery dataset as an example, where the foreground voxels make up $< 0.1\%$ of the total volume of voxels, simply predicting every voxel as background will yield an accuracy (Eq. 2.19) of $> 99.9\%$. Class-imbalance in datasets can be resolved in several ways, e.g. under- or over-sampling certain classes to restore class balance. Another approach is to use a weighted loss function, where the loss for one or more underrepresented classes is taken more into account when calculating the total loss.

The loss function used during training is the Dice loss function, defined as $1-\mathrm{DSC}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ (see section 2.3.1 for details on Dice Similarity Coefficient). This resolves the class-imbalance by completely ignoring true negative predictions, and has been shown to give good results when used with imbalanced segmentation problems [28]. For the multi-class segmentation tasks this is extended by calculating Dice loss for each class individually, such that

$$\text{DSC}_{\text{loss}}(\hat{\boldsymbol{y}}, \boldsymbol{y}; N) = -\sum_{i}^{N} 1 - \text{DSC}(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_i) \tag{3.1}$$

where N is the number of classes for the giving multi-class problem. The reason for negating the sum is so that the multi-class Dice loss is a monotonically decreasing function with regards to performance. This allows for gradient calculation to be done in the standard way, i.e. letting the gradient point in the direction of lowest loss.

3.3 Training

The networks were trained to do several different tasks, as listed in table 3.1. For the binary coronary artery segmentation task, BVNet and Unet2D were trained on 46102 slices of size 256x256x5, extracted from 41 CT volumes of the heart. Unet3D was trained on 11204 patches of size 64x64x64, extracted from the same 41 CT volumes. For multi-class coronary artery segmentation training was performed on 60610 slices and 14045 patches, respectively.

Task	Label 0	Label 1	Label 2	Label 3
Coronary Artery Segmentation	RCA+LM	N/A	N/A	N/A
Multi-class CA Segmentation	BG	RCA	LM	Aorta
Liver Vessel Segmentation	BG	Liver	PV	HV

Table 3.1: Semantic regions of each task the networks were trained to do.

The 256x256x5 slices used when training BVNet and Unet2D were extracted by selecting a depth, z, and a (x, y)-position for the upper left corner. Using ':' as a slicing operator, a slice, S, was then extracted that $S = I(x : x+256, y : y+256, z-2 : z+3)$. This results in a a 2D image with 5 channels, where the channels are neighbouring depth slices from the original volume. The networks were trained to segment the 'center' channel in this image, with the other channels providing some spatial context. This results in a sort of 2.5D architecture; a planar slice with information about its neighbouring planes.

3.3.1 Hyperparameters

A series of experiments were ran in order to determine values for the hyperparameters to use during training. A small dataset consisting of 1000 samples was constructed, and a series of networks were trained to over-fitting using different combinations of hyperparameters for each network. The two hyperparameters varied were learning rate and number of channels in input data. Every network used dice loss as loss function, as this was experimentally determined to work well. The learning rates tested were fixed learning rates of 10^{-3}, 10^{-4}, 10^{-5}, and 10^{-6}.

The experiments showed that increasing the number of channels beyond 5 gave no significant performance improvements (see Fig. 4.1), and only increased data pre-processing and -fetching times. The learning rate used during training was $\eta = 10^{-4}$. As the number of parameter updates increased a lot when using a full dataset compared to the hyperparamter experiments, exponential decay was applied for each epoch using Eq. 3.2, with $k = 0.02$, and t being the current epoch. This was done in an attempt to help the network gradually improve performance after the initial convergence. Another approach would be to increase the batch size over time as suggested by Smith et al. [41], but this was not practically feasible as the batch size used during training was just about the maximum possible with the given hardware resources.

$$\eta_t = \eta_0 * e^{-k*t} \tag{3.2}$$

3.3.2 Dataset under-sampling

The training data was extracted from the original volumes using very slight under-sampling of the background class. Each extracted slice or patch was required to contain at least 1 foreground voxel. This was done in order to ensure that none of the training batches contained no foreground voxels, as this might cause problems with the network to favouring setting all outputs to 0. For slices of shape (256,256) and a foreground voxel threshold of 1, this means that each slice contained at least 0.0015% foreground voxels. For the coronary artery dataset, this is a foreground voxel ratio significantly lower than the average foreground voxel ratio of the dataset as a whole ($\approx 0.06\%$).

3.4 Full Volume Segmentation

Segmentation of the full CT volumes can be performed in two ways. A slice-/patch-wise segmentation approach is performed by extracting overlapping slices or patches of the same size the network was trained on. The predicted slices/patches are then stitched back together in a new volume in the same position it was extracted from. The degree of overlap of the extracted slices can be varied, where the fastest way is selecting slices with no overlap. This may however lead to poor results, as the edge pixels or voxels in the resulting predictions have access to less valid context (as the input to each convolution layer is 0-padded in order to keep dimensions the same). Choosing a greater overlap means slower, but more precise predictions. Overlapping predictions also makes it possible to accumulate predictions for each voxel, resulting in a heatmap-like volume like the one seen in Fig. 3.2. Voxels that are predicted as to belong to the foreground in several slices/patches have a higher degree of certainty to actually belong to the foreground.

The other approach is performing prediction on larger inputs than the networks were trained on. This is possible, as all three network architectures are fully convolutional, meaning that the parameters are independent of input size. The convolutional filters learned by the networks are applied to every position in the input, no matter the size or shape. The only dimension that cannot be varied is the number of channels in the input. By exploiting the fact that the networks are fully convolutional, prediction can be done on non-cropped slices from the full CT volumes. This has one drawback, however, as the networks learn some global features during training that will no longer apply to the larger images (such as general spatial position of coronary arteries). For the 3D architecture, predicting full volumes in a single pass is not an option, as the memory requirements to hold all the activations are simply too great.

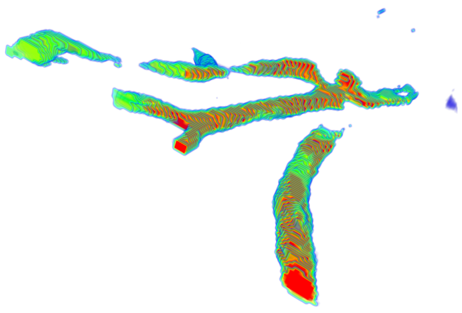


Figure 3.2: Clipped heat-map of a patch-wise prediction. Blue voxels indicate low certainty, green medium certainty, and red voxels were predicted in every overlapping slice. The clipping reveals a gradient of certainty towards the center of the blood vessels.

Slice-wise prediction has a drawback of using substantially longer time than full-size prediction, as the number of predictions that have to be done increases. Number of predictions can be adjusted by increasing or decreasing the stride used when extracting slices, with larger strides resulting in lower overlapping of patches and potentially lower accuracy of final prediction. For a volume with shape $(512, 512, 300)$ the number of predictions needed with the patch-wise approach using a stride of 64 is over 7000, while the full-size prediction only requires 300 predictions.

3.5 Datasets

The coronary artery dataset consists of 51 hand-labeled CT images from St. Olavs Universitetssykehus. The dataset is developed as a part of their ongoing FFR project, and is still in developement. The 3D-IRCADb-01 dataset is a dataset compiled by IRCAD - Institut de recherche contre les cancers de l'appareil digestif, consisting of 20 CT scans of the abdomen, where all major organs and vessel structures are labeled.

3.5.1 Preprocessing

The images in both datasets are preprocessed before training and prediction. The images are resampled to a fixed voxel-spacing, as both datasets includes volumes with a varying resolution. When considering the images as points in world-space, this is not an issue, as the voxel spacing determines where each point (i, j, k) lies in relation to the origin. However, when treating the images as arrays, discrete data structures, each voxel is a point on a fixed-size grid. If the original resolution in the volumes differ, this will lead to unintentional deformations of the geometry of the structures of interest. For the St. Olavs dataset, the images have a resolution ranging from $\approx (0.35\,\text{mm}, 0.35\,\text{mm}, 0.35\,\text{mm})$ to $\approx (0.65\,\text{mm}, 0.65\,\text{mm}, 0.65\,\text{mm})$, with most images having a voxel-spacing close to $(0.4\,\text{mm}, 0.4\,\text{mm}, 0.4\,\text{mm})$. Before training and prediction, all images are therefore resampled to a fixed voxel-size of $0.4\,\text{mm}^3$, using tri-linear interpolation. The images in the 3D-IRCADb-01 dataset have a lower resolution than the St. Olavs dataset, and is therefore resampled to a fixed $(0.8\,\text{mm}, 0.8\,\text{mm}, 0.8\,\text{mm})$ voxel-spacing in order to not introduce an abundance of interpolated data points. After resampling, each volume is normalized, the mean subtracted, and divided by the standard deviation of the normalized image.

3.5.2 Data Augmentation

During training, random elastic deformation (section 2.4) was applied on-the-fly in order to increase the volume of training data. The value for the elasticity parameter was found experimentally through visual inspection of deformed images. The value used during training was $\sigma = \text{width}(I) \cdot 0.08$, meaning that for an image I of size $(256, 256)$, $\sigma = 256 \cdot 2 = 20.48$. The values for the scaling factor used when training on coronary arteries was determined in an attempt to get deformation similar to coronary artery motion seen in-vivo. Right coronary artery (RCA) motion during cardiac cycles is greater in the x and z-directions than in the y-direction, and during respiratory cycles the motion is greatest in the y-direction, however this was not modeled directly as the random deformation fields yield varying deformations in each axis anyways. The scaling factor, α, was set such that the magnitude of the deflection in millimeters would approximate the range observed in-vivo, a maximum of around $8.0\,\text{mm}$ [22]. Given a dataset voxel size of $0.4\,\text{mm}^3$, the scaling factor was set such that the maximum possible deflection in any given direction would be approx. 20 voxels.

It is worth noting that this is not an attempt to create perfectly realistic deformations, which include among other things rotations and non-random deformations not emulated by elastic deformation. This method for data augmentation is applied in order to increase the network's robustness and ability to generalize, but done in a manner that is not entirely inconsistent with what might be seen in the true function's domain. Figure 3.3 illustrates the effect of applying elastic deformation on a sample from the St. Olavs dataset, with exaggerated parameters for better visualization.

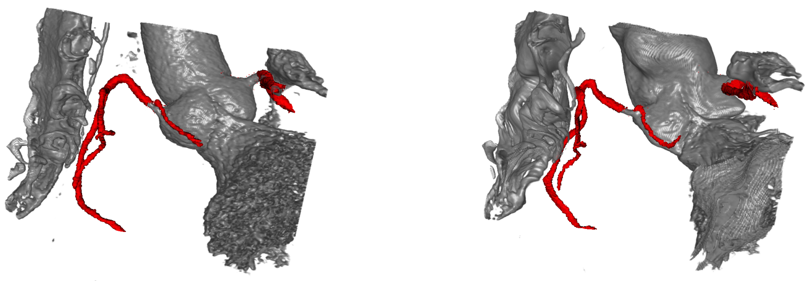

(a) (b)

Figure 3.3: Volume before and after application of elastic deformation. Values used were $\alpha = 4000$, $\sigma = 14$. Coronary arteries marked in red.

3.6 Refining segmentations

The raw segmentation outputs from the networks contain varying degrees of spurious responses. To remedy this, the segmentations are refined through a post-processing step. Due to differences in network output in the binary and multi-class segmentation tasks, two different refinement procedures are used.

Refining binary segmentations

For the binary segmentation tasks, the output is fairly clean, and refinement can be done with connected components analysis. However, in the 2D network outputs there are some gaps in the segmentations, where the networks fail to correctly predict certain slices. In order to combat this, the connected components analysis is performed in two stages. First, the binary closing (see [13, p. 657–661] for details on morphological opening and closing) of the segmentation is computed with a NxNxN structuring element. This has the effect of filling in gaps smaller than the structuring element, and thus allowing (some) disconnected parts to be 'reconnected' to the main branch. Connected components analysis is then performed on the closed volume, giving each connected component a unique label. The segmentations are clean enough that the coronary artery branches can be selected as the two largest connected components. Secondly, connected components analysis is performed on the original, raw segmentation. The final refined segmentation is obtained by selecting from the raw segmentation every connected component overlapping the two largest components from the closed volume.

Refining multi-class segmentations

The network outputs in the multi-class segmentation tasks have some distinct errors, as can be seen in Fig. 4.10. For this reason, more complex analysis of the structures is necessary in order to get decent refined segmentations. The refinement process is different for the aorta labels and the coronary artery labels, as the raw aorta segmentation requires less post-processing. Refining the aorta segmentations is done by simply doing connected components analysis and selecting the largest connected component.

Refining the coronary artery segmentations is a multi-stage process, as illustrated by Fig. 3.4. First connected components analysis is used to label all connected components in the volume. From this, all components with a volume (number of voxels in component) larger than a given threshold, t, is selected as candidate components. The threshold used in refinement was determined by looking at the volume of all coronary branches in the training dataset and setting it to be slightly lower than the smallest branch. For this dataset, the smallest volume of any coronary branch is 5036 voxels, significantly lower than the average volume of ≈ 28500 voxels. In order not to exclude any such small branches, the threshold was set to $t = 5000$.

When a set of candidate components is found, the solidity of each component is measured in order to determine which component is most likely to a coronary artery. The solidity of a component, C, is determined by

$$\text{Solidity}(C) = \frac{\text{Volume}(C)}{\text{SurfaceArea(C)}} \tag{3.3}$$

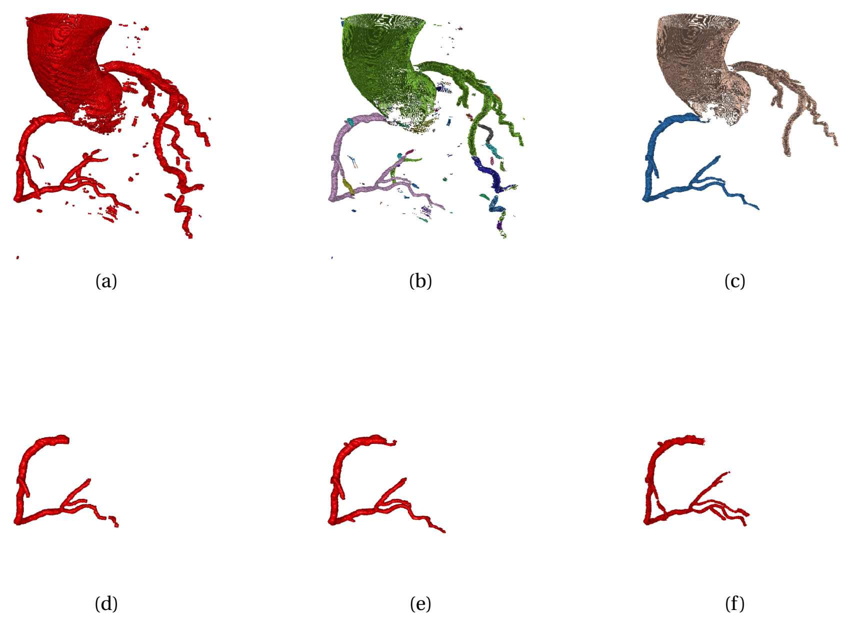


Figure 3.4: Multi-class segmentation refining workflow. (a) Raw RCA segmentation output from BVNet. (b) Connected components of raw segmentation. (c) Candidate components based on volume threshold. (d) Morphological opening of selected component based on solidity. (e) Final refined segmentation. (f) Hand-labeled ground truth.

and is maximized by a solid sphere. Surface area is determined as the sum of voxel faces in a structure facing a background voxel. The shell-like structures found in the multi-class network outputs have a very low solidity compared to the coronary artery structures. The coronary artery component is therefore determined to be the candidate component with the highest solidity.

After finding the coronary artery component, in many volumes there still some remaining spurious responses. In order to prune these, the binary opening of the selected component is calculated, removing all structures smaller than the structuring element. The final refined segmentation is obtained by performing connected components analysis on the raw segmentation, and including in the final output all components overlapping the opened component.

Chapter 4

Results

This chapter presents the results of the experiments. Results of hyperparameter experiments and training performance is presented first, followed by the results of the different segmentation tasks. As CT images are three-dimensional images, visualization of segmentations with 2D images has some limitations. Several videos[1] showing the raw and refined segmentations for different volumes have been created in order to better visualize the three-dimensional images.

4.1 Hyperparameters

The results of the hyperparameter experiments can be seen in Figure 4.1. Each plot shows loss and validation loss for each epoch for networks trained with different number of input channels using learning rates of $\eta = 10^{-3}$, $\eta = 10^{-4}$, $\eta = 10^{-5}$, and $\eta = 10^{-6}$.

4.2 Training performance

Network training performance in the binary coronary artery segmentation task can be seen in Fig. 4.2. Training time was 3 days for the 2D models, and 4 days for the 3D model. The final 2D models were trained for 70 epochs. The dataset for training the 3D model is smaller than the 2D dataset, and the final 3D model was therefore trained for 150 epochs.

Loss over time in the multi-class coronary artery segmentation task can be seen in Fig. 4.3. Multi-class training took approx. 10% longer time for the 2D networks, and approx. 25% longer for the 3D network. In order to compensate for higher training time, the final multi-class 3D model was trained for 70 epochs, using a higher initial learning rate of 0.001 (compared to 0.0001 for the other models) in the hopes of converging faster.

[1] `https://www.youtube.com/playlist?list=PLwj8-ca_B8YOCxHVYvi9f7iaPUSB1P4ox`

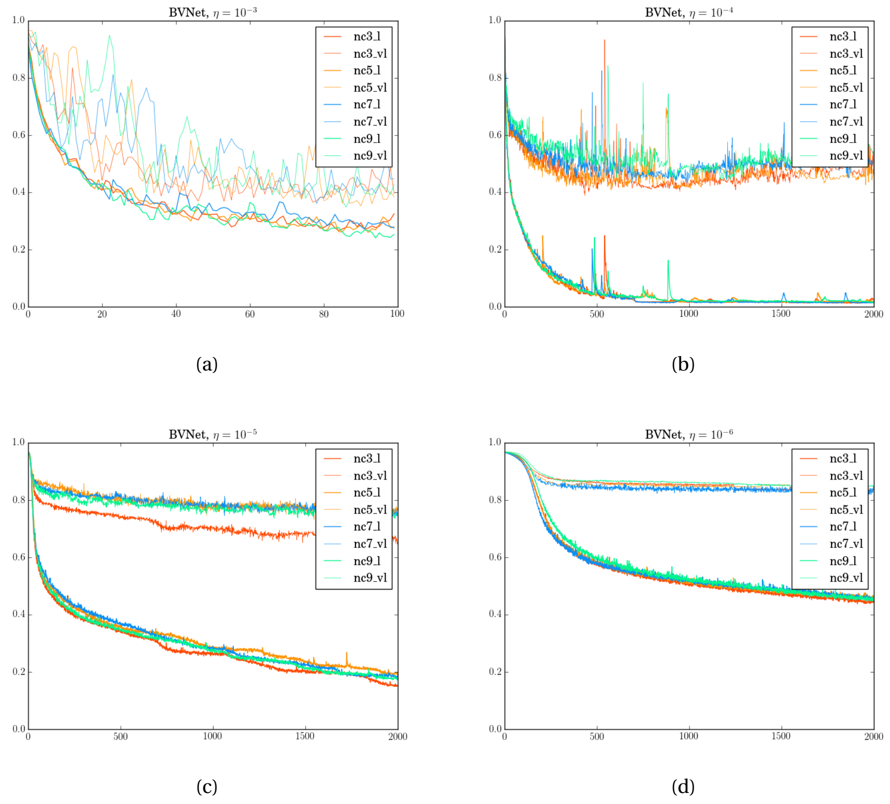


Figure 4.1: Training and validation loss over time for BVNet trained on a small dataset using different learning rates. Colors indicate number of channels used in input data.

4.3 Coronary Artery Segmentation

In this section results of the coronary artery segmentations is presented. Table 4.2 shows average predicition timings of the different network architectures and segmentation approaches.

4.3.1 Binary Segmentation

Tables 4.3 and 4.4 present Dice similarity coefficient results of the binary segmentation task using slice-/patchwise segmentation and full-size segmentation approaches, respectively. Figure 4.4 and 4.5 demonstrates the differences in slice-wise and full-size segmentations using BVNet. The slice-wise segmentation was performed with an input size of (256, 256), and a stride of (128, 128). For the sake of completeness, slice-wise segmentation by the 2D networks was also performed using the same stride as was used for UNet3D, (32, 32). The results of these segmentations can be seen in Table 4.5 with Figure 4.8 and 4.9 providing visualizations of the segmentations using BVNet and UNet2D,

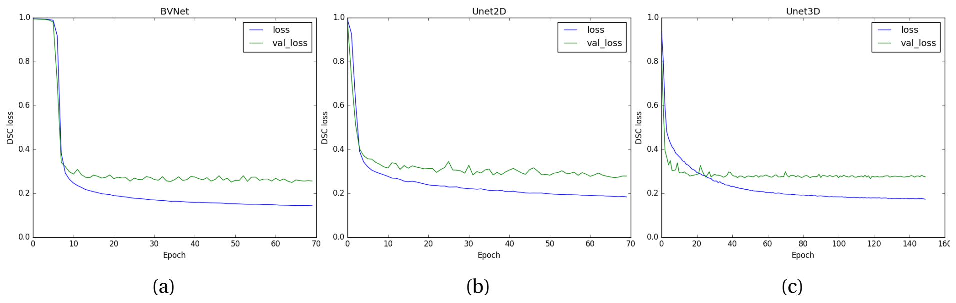


Figure 4.2: Training performance on binary coronary artery segmentation for each of the network architectures.

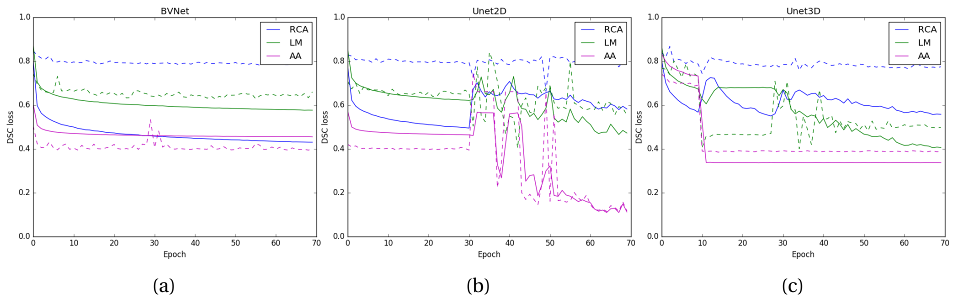


Figure 4.3: Training (solid) and validation (dashed) loss on multi-class coronary artery segmentation for each of the network architectures.

Model	BVNet	UNet2D	UNet3D
DSC	0.856	0.816	0.827

Table 4.1: Final training DSC of BVNet, UNet2D, and UNet3D in the binary segmentation task.

	Binary		Multi-class	
Model	Slice-/patchwise	Full-size	Slice-/patchwise	Full-size
BVNet	86.7s	45.3s	95.9s	45.6s
Unet2D	52.5s	22.2s	62.1s	23.6s
Unet3D	187.8	N/A	206.0	N/A

Table 4.2: Mean prediction time for each network architecture for the different tasks and full volume segmentation approaches.

respectively. A visual comparison of raw and refined binary segmentations using the three different network architectures is given in Figures 4.6 and 4.7.

	BVNet		Unet2D		Unet3D	
Volume	DSC_{RAW}	$DSC_{refined}$	DSC_{RAW}	$DSC_{refined}$	DSC_{RAW}	$DSC_{refined}$
V8	.847	.864	.818	.833	.767	.886
V16	.744	.793	.751	.802	.518	.862
V27	.687	.786	.654	.754	.445	.746
VP5	.792	.841	.807	.842	.479	.865
VP8	.720	.811	.673	.779	.387	.759
VP9	.792	.826	.746	.794	.637	.831
VR20	.741	.810	.728	.788	.601	.785
VR55	.700	.741	.721	.695	.492	.774
VR89	.601	.611	.587	.594	.424	.664
VR98	.815	.875	.788	.864	.750	.905
Mean	0.744	0.796	0.727	0.774	0.550	0.808

Table 4.3: DSC for raw and refined predictions for each network architecture in the binary segmentation task using slice-/patchwise full volume segmentation approach.

4.3.2 Multi-class Segmentation

Mean raw and refined DSC for each network in the multi-class coronary artery segmentation task can be seen in Table 4.6. Figure 4.10 visualizes effects of segmentation refinement on the output of BVNet when segmenting validation sample V16. A comparison of the refined segmentation outputs of each of the three models can be seen in Figure 4.12.

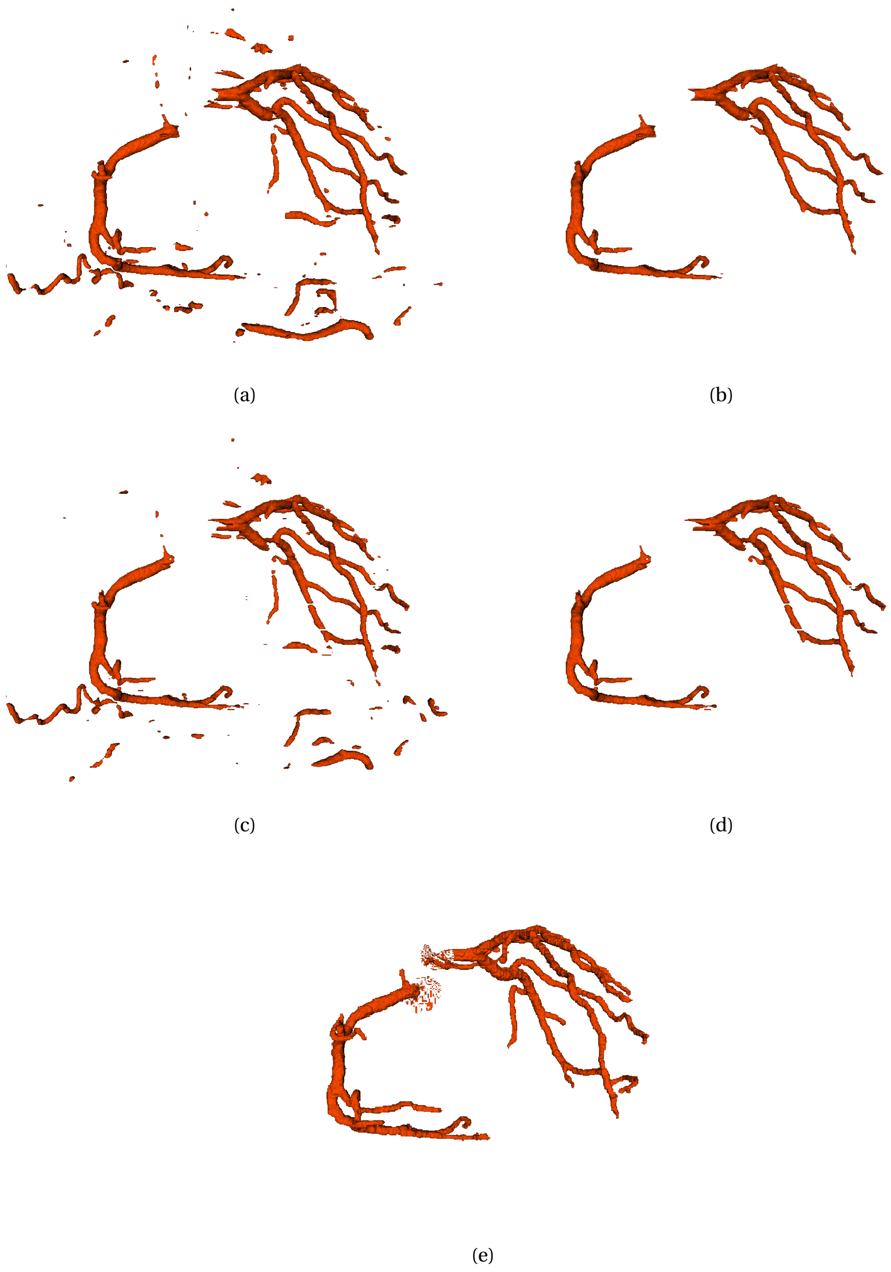
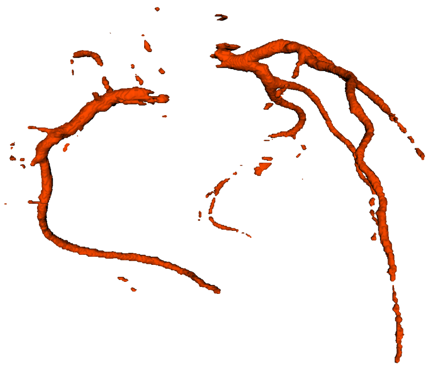
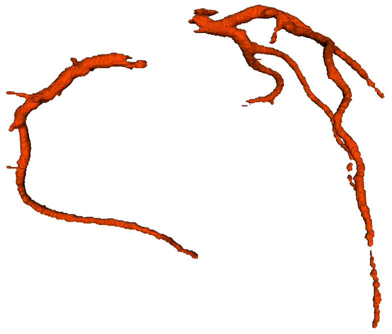


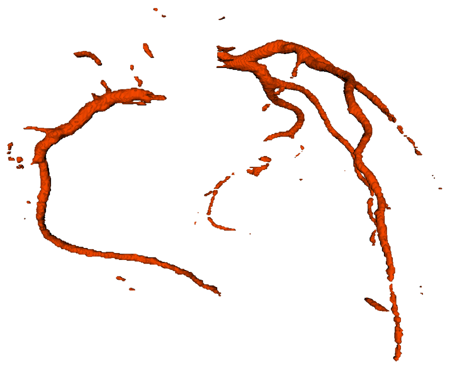
Figure 4.4: Comparison of slice-wise segmentation and full-size segmentation of V16 using BVNet. (a) Raw slice-wise segmentation, (b) refined slice-wise segmenetation (c) raw full-size segmentation (d) refined full-size segmentation (e) hand-labeled ground truth.



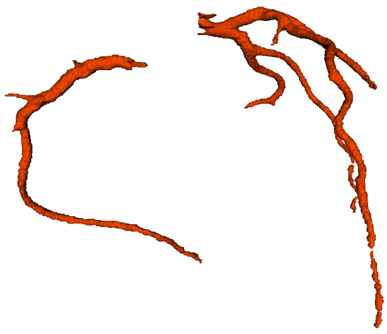
(a)



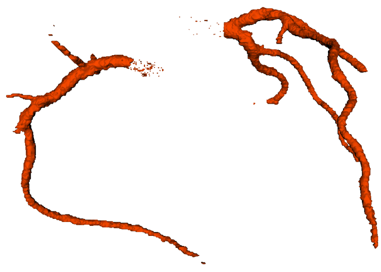
(b)



(c)

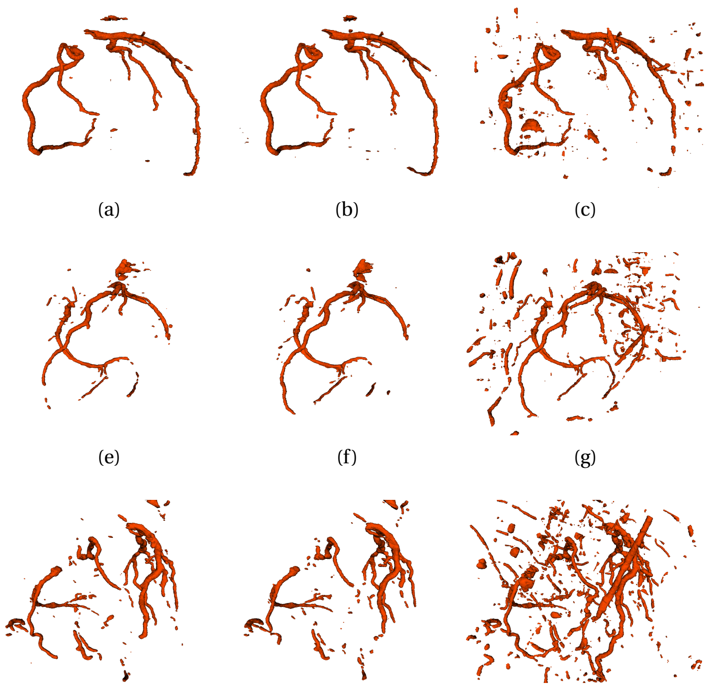


(d)



(e)

Figure 4.5: Comparison of slice-wise segmentation and full-size segmentation of VP9 using BVNet. (a) Raw slice-wise segmentation, (b) refined slice-wise segmenetation (c) raw full-size segmentation (d) refined full-size segmentation (e) hand-labeled ground truth.

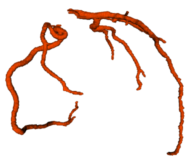


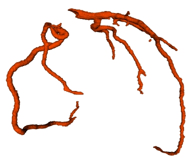
(a) (b) (c) (d)

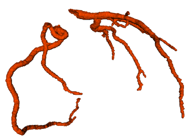
(e) (f) (g) (h)

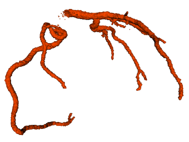
(i) (j) (k) (l)

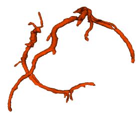
Figure 4.6: Raw binary coronary artery segmentations. From left to right: BVNet, UNet2D, UNet3D, and hand-labeled ground truth.

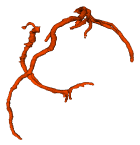

(a)

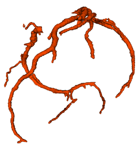

(b)

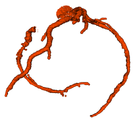

(c)

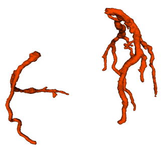

(d)

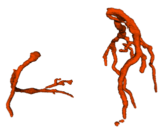

(e)

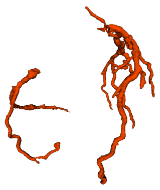

(f)


(g)


(h)


(i)


(j)


(k)

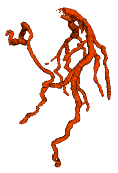

(l)

Figure 4.7: Refined binary coronary artery segmentations. From left to right: BVNet, UNet2D, UNet3D, and hand-labeled ground truth.

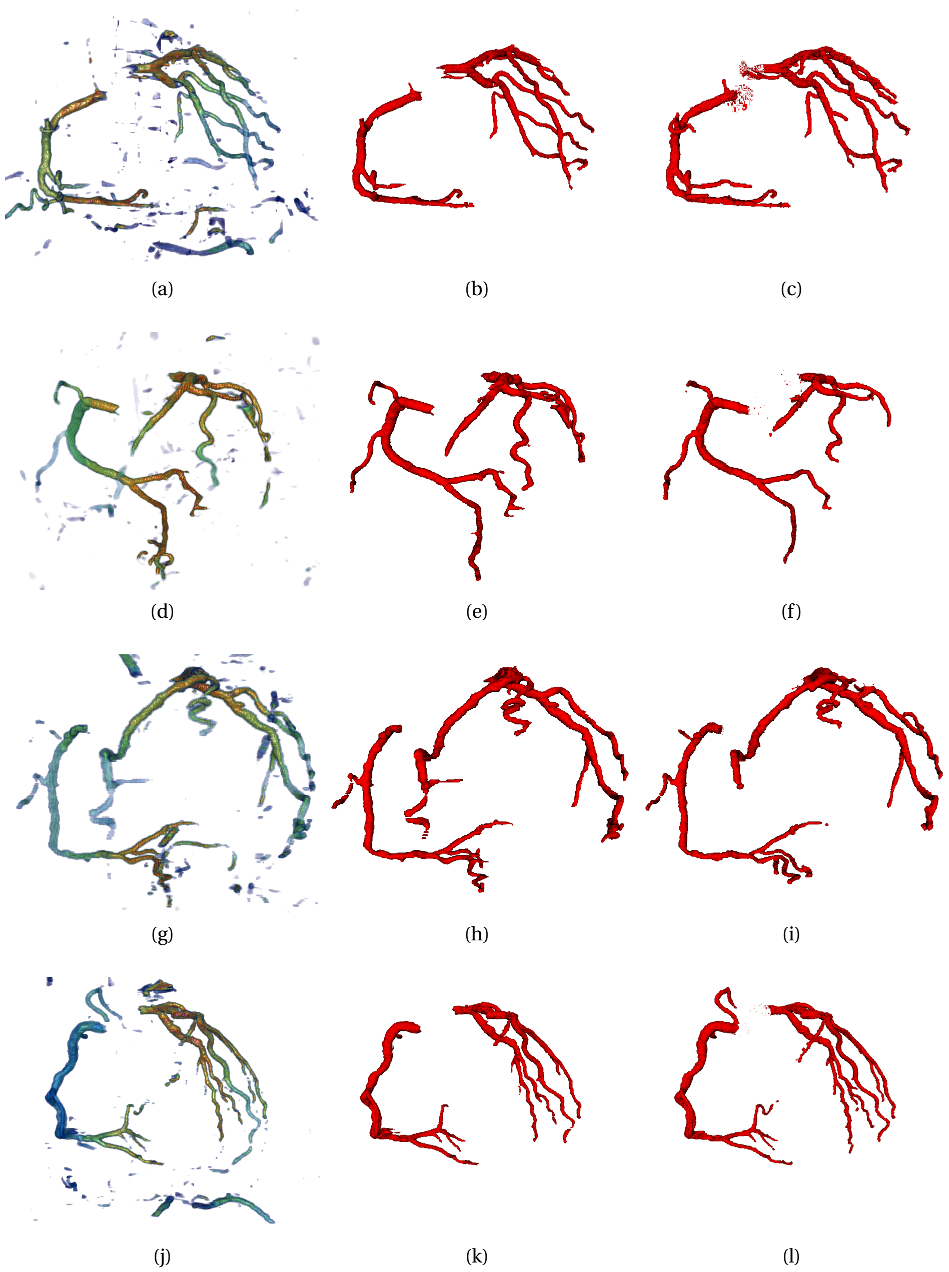


Figure 4.8: Binary slice-wise segmentation of coronary arteries using BVNet. Slice shape (256, 256) and stride (32, 32). Left: Heatmap of raw segmentation (blue represents low positive prediction rate and red represents high positive prediction rate). Middle: Refined segmentation. Right: Hand-labeled ground truth.

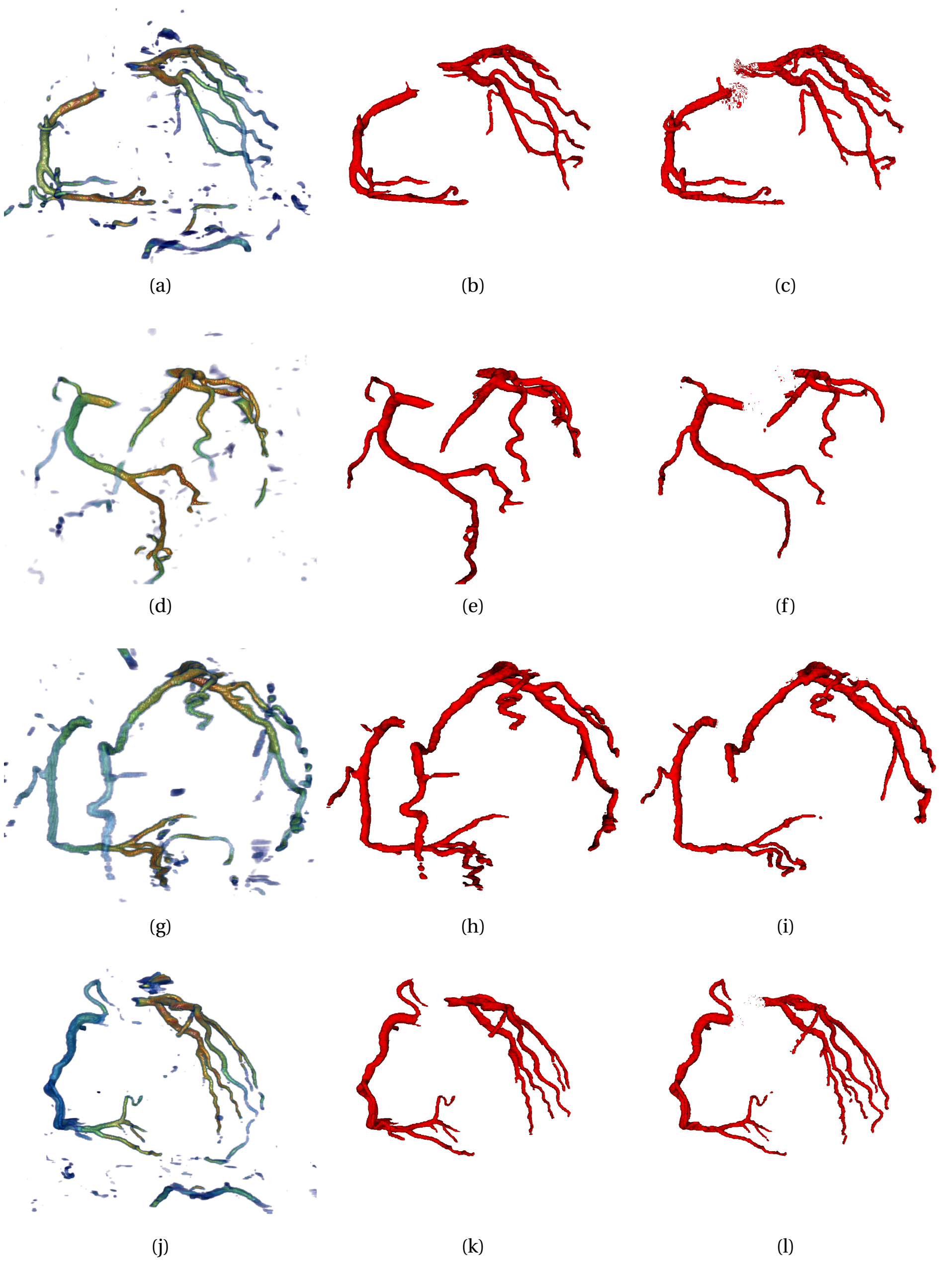
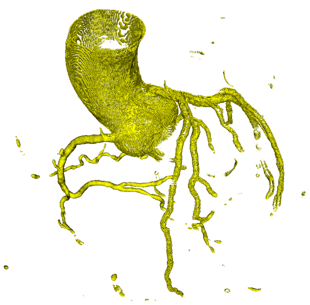
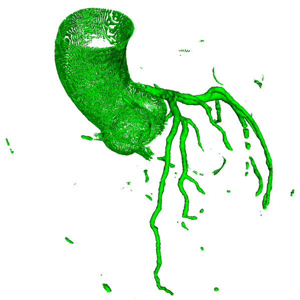
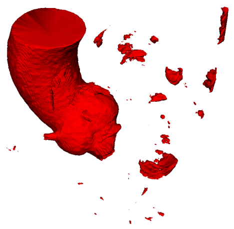
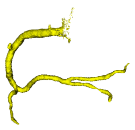


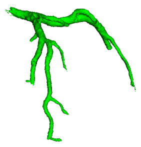
Figure 4.9: Binary slice-wise segmentation of coronary arteries using UNet2D. Slice shape (256, 256) and stride (32, 32). Left: Heatmap of raw segmentation (blue represents low positive prediction rate and red represents high positive prediction rate). Middle: Refined segmentation. Right: Hand-labeled ground truth.


(a)


(b)


(c)


(d)


(e)

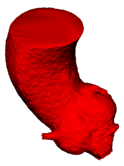

(f)

Figure 4.10: Raw and refined multi-class segmentation output from BVNet. Top: Raw RCA, LM and AA segmentation. Bottom: Refined RCA, LM, and AA segmentation.

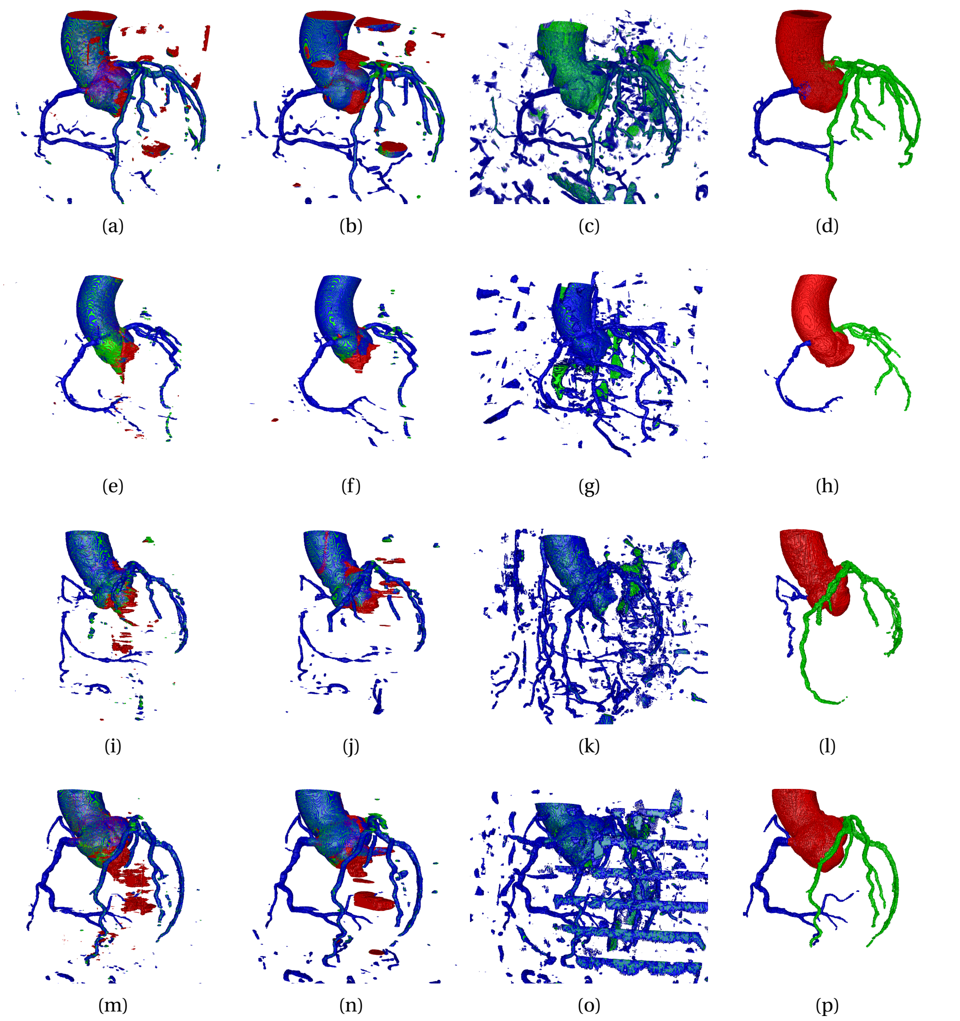


Figure 4.11: Visualizations of raw multi-class segmentations. Left to right: BVNet, UNet2D, UNet3D, hand-labeled ground truth.

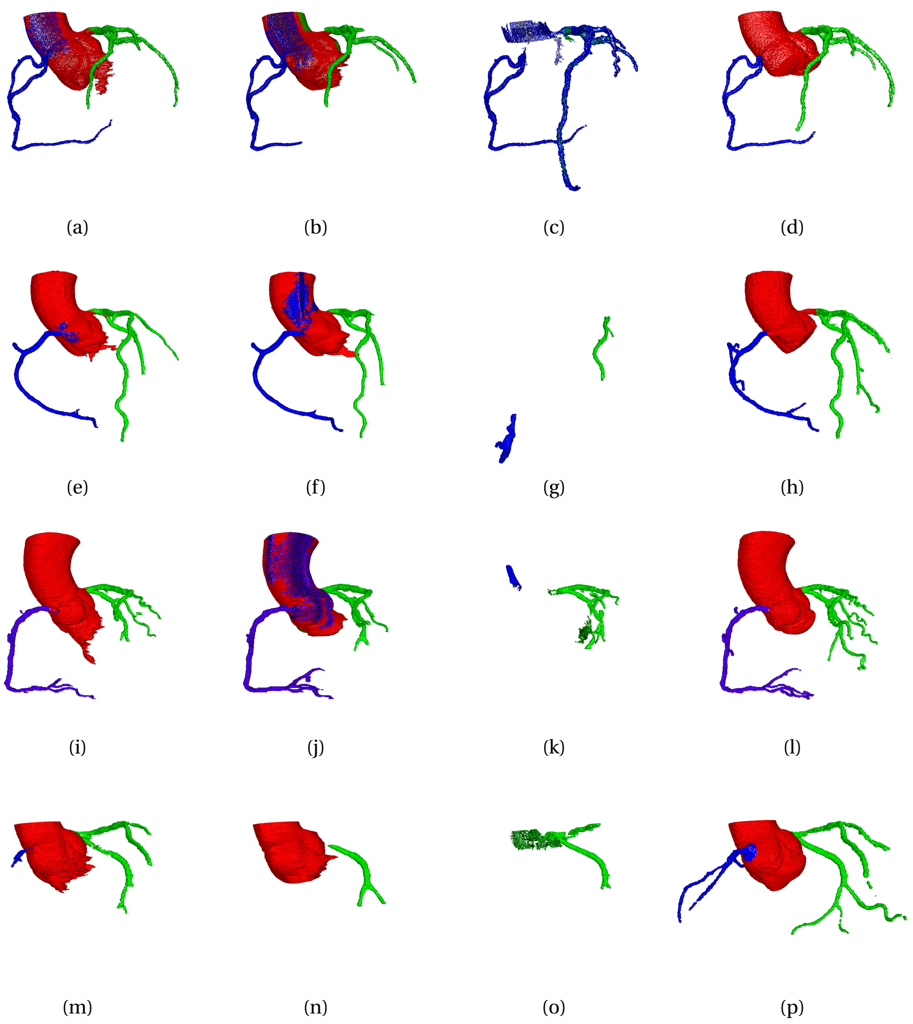


Figure 4.12: Visualizations of refined multi-class segmentations. Left to right: BVNet, UNet2D, UNet3D, hand-labeled ground truth.

	BVNet		Unet2D	
Volume	DSC_{RAW}	$DSC_{refined}$	DSC_{RAW}	$DSC_{refined}$
V8	.860	.873	.836	.846
V16	.730	.769	.739	.777
V27	.705	.806	.687	.786
VP5	.796	.840	.830	.833
VP8	.725	.818	.667	.800
VP9	.799	.840	.762	.803
VR20	.745	.807	.741	.795
VR55	.708	.728	.720	.708
VR89	.614	.612	.568	.598
VR98	.824	.872	.805	.858
Mean	0.751	0.797	0.736	0.780

Table 4.4: DSC for raw and refined predictions for each network architecture in the binary segmentation task using full-size slice full volume segmentation approach.

4.4 Multi-class Liver Segmentation

Dice similarity coefficients for segmentation of Patients 10, 11, and 12 from 3D-IRCADb-01 using BVNet and UNet2D is presented in Table 4.8 and Table 4.9. Figure 4.13 shows 3D renders of the raw and refined segmentations compared to ground truth.

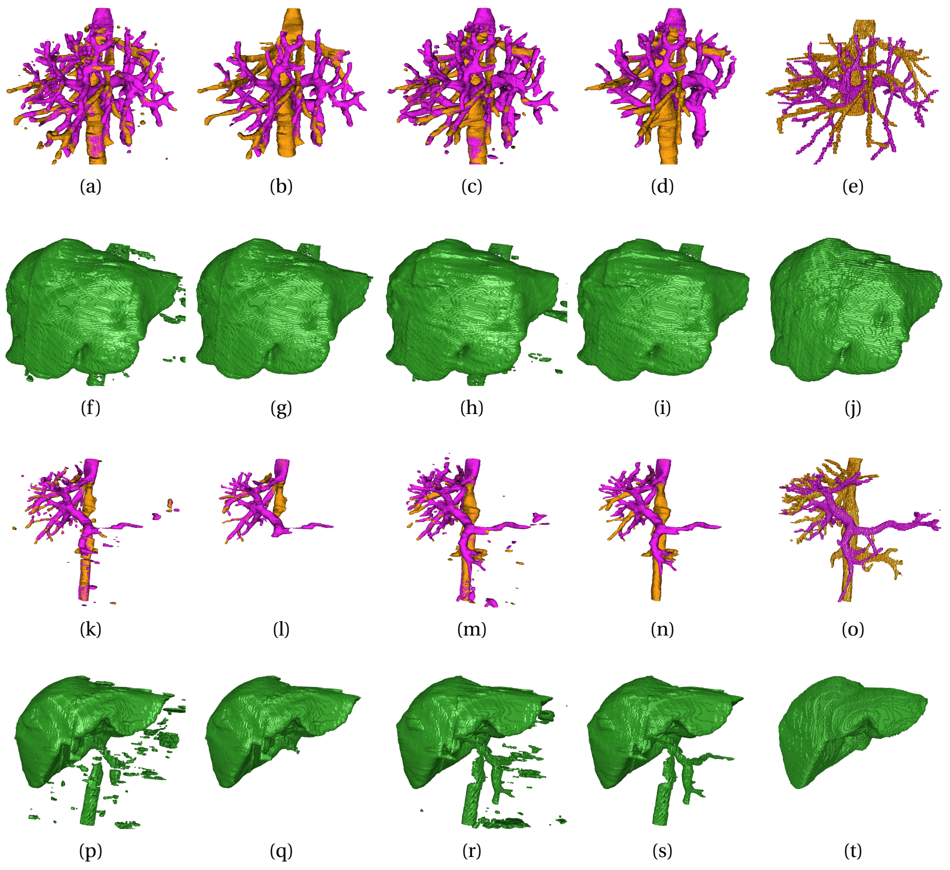


Figure 4.13: Multi-class segmentation of Patients 10 and 11 from 3D-IRCADb-01 using BVNet and UNet2D. Left to right: Raw and refined segmentation from BVNet, raw and refined segmentation from UNet2D, ground truth.

	BVNet		UNet2D	
Volume	DSC_{RAW}	$DSC_{refined}$	DSC_{RAW}	$DSC_{refined}$
V8	0.801	0.843	0.765	0.810
V16	0.700	0.829	0.698	0.826
V27	0.626	0.798	0.604	0.759
VP5	0.681	0.842	0.751	0.842
VP8	0.609	0.767	0.534	0.719
VP9	0.738	0.815	0.664	0.766
VR20	0.683	0.810	0.649	0.770
VR55	0.608	0.685	0.700	0.730
VR89	0.578	0.641	0.553	0.624
VR98	0.779	0.876	0.738	0.857
Mean	0.680	0.791	0.666	0.770

Table 4.5: DSC of binary slice-wise segmentation using BVNet and UNet2D with high degree of overlap. Slice shape (256, 256) and stride (32, 32).

	RCA		LM		AA	
Model	DSC_{RAW}	$DSC_{refined}$	DSC_{RAW}	$DSC_{refined}$	DSC_{RAW}	$DSC_{refined}$
BVNet	0.362	0.649	0.535	0.694	0.932	0.950
UNet2D	0.280	0.392	0.473	0.607	0.920	0.943
UNet3D	0.137	0.053	0.377	0.555	0.0	0.0

Table 4.6: Mean raw and refined DSC for each model in the multi-class coronary artery segmentation task.

	RCA		LM		AA	
Volume	DSC_{RAW}	$DSC_{refined}$	DSC_{RAW}	$DSC_{refined}$	DSC_{RAW}	$DSC_{refined}$
V8	.394	.812	.542	.585	.956	.967
V16	.339	.593	.452	.700	.973	.973
V27	.436	.818	.640	.833	.900	.905
VP5	.422	.815	.558	.733	.933	.946
VP8	.483	.823	.384	.738	.871	.949
VP9	.379	.807	.559	.760	.954	.957
VR20	.377	.786	.513	.759	.909	.960
VR55	.165	.287	.614	.699	.945	.946
VR89	.121	.000	.457	.449	.935	.940
VR98	.504	.752	.640	.682	.947	.958
Mean	0.362	0.649	0.535	0.694	0.932	0.950

Table 4.7: Raw and refined DSC for each class in multi-class coronary artery segmentation using BVNet

	BVNet			UNet2D		
Patient	Liver	HV	PV	Liver	HV	PV
10	0.954	0.464	0.628	0.958	0.432	0.601
11	0.940	0.558	0.681	0.941	0.587	0.698
12	0.910	0.433	0.610	0.924	0.401	0.666
Mean	0.934	0.485	0.640	0.941	0.473	0.655

Table 4.8: DSC of raw segmentation output of BVNet and UNet2D on the validation volumes from 3D-IRCADb-01.

	BVNet			UNet2D		
Patient	Liver	HV	PV	Liver	HV	PV
10	0.963	0.551	0.676	0.963	0.501	0.615
11	0.950	0.537	0.447	0.950	0.633	0.698
12	0.884	0.555	0.623	0.929	0.582	0.649
Mean	0.932	0.547	0.581	0.947	0.572	0.654

Table 4.9: DSC of refined segmentation output of BVNet and UNet2D on the validation volumes from 3D-IRCADb-01.

Chapter 5

Discussion

In this chapter the results of the binary and multi-class coronary artery segmentation experiments and the multi-class liver segmentation experiment, will be discussed. Training performance on the different tasks, and quality of segmentations will be evaluated. Problems relating to quantitative evaluation of network performance will be discussed, and possible solutions set forth. First, the binary and multi-class coronary artery segementation results will be discussed, followed by the results on the liver segmentation task. Finally, the work that has been done, and the chosen and devised methods will be evaluated, and possible shortcomings identified.

5.1 Coronary Artery Segmentation

In this section, the results of the coronary artery segmentation task will be discussed. Results of the binary segmentation task will be discussed first, followed by the multi-class segmentation task.

5.1.1 Binary Coronary Artery Segmentation

As can be seen by comparing Table 4.1 and mean $\mathrm{DSC}_{\mathrm{RAW}}$ from Table 4.3, the final training performance of the three models is slightly higher than the mean validation DSC. The obvious reason for this discrepancy is that the networks are somewhat overfitted; model parameters are fine-tuned to the training dataset and generalization suffers as a consequence. This is also apparent in Figure 4.2, where the difference in training and validation DSC shows how DSC on the validation slices stops improving significantly after the initial rapid convergence.

The results show a difference in training and validation loss of around 10-15% for the 2D models, and around 30% for the 3D architecture. It can be argued that this is in fact not so much a result of overfitting as the models being unable to generalize beyond a certain point. Loss and validation loss for extremely overfitted models are usually similar to what can be seen in Figure 4.1b: After a certain point the validation score actually starts worsen as the networks fine-tune parameters to the point of storing direct mappings of input to ground truth for each training sample. The application of random elastic deformation to training slices/patches makes it more difficult for the networks to overfit, as it essentially creates infinite variations of the training images. This makes it impossible for the networks to store any direct mappings between input and output, and can also explain why near-perfect DSC is never achieved on the training images.

Another reason for the discrepancy between training DSC and DSC of the full-volume segmentations of the validation images could be that the networks are only trained on slices/patches containing at least one foreground pixel/voxel. This results in the networks never having seen parts of the volumes where no coronary arteries are located, and therefore might reduce their ability to discern between coronary arteries and other, similar blood vessels located further from the heart. The raw segmentations from UNet3D seems to support this. The smaller receptive field in the (x, y)-directions (patch size of [64, 64, 64] vs. slice size of [256, 256]) for UNet3D results in each slice in the extracted patches having a higher foreground voxel density than slices extracted from the same area for training the 2D models. The smaller receptive field in the (x,y) direction results in fewer patches containing tubular structures and blood vessels from other parts of the chest cavity, and as such the models never learns how to filter these out.

Comparing training DSC and validation DSC in this manner might not be completely fair, as the training DSC is calculated per slice/patch, while validation DSC is calculated on whole volumes. If slices were extracted from the validation images requiring at least one positive voxel, the validation DSC might have shifted slightly upwards. However, for real-world applications this is not a valid strategy, as it requires that a segmentation of the coronary arteries already exists. For this reason this method for evaluating performance has been omitted from the results.

The performance of these three models, both in terms of raw and refined segmentations, is significantly better than previously reported results [20, 24]. A direct comparison is difficult to make, as the dataset used in these experiments is larger than the one used in previous experiments. In addition, there is no overlap between the validation sets used in [20, 24], and so a side-by-side comparison of segmentations is not possible. Generally speaking, however, the method proposed in this report has a simpler workflow and a significant reduction in prediction time. Inference time is reduced from roughly five minutes to a minute and a half or less (depending on segmentation approach) using the 2D models, and around three minutes using UNet3D. The resulting raw segmentations also require simpler post-processing in order to obtain the final refined segmentation, as the raw network output is much more precise.

Binary Segmentation Refinement

Raw segmentation output from all three models contain various degrees of spurious responses, as demonstrated by Figure 4.6. These erroneously segmented blob-like structures are generally parts of other blood vessels than the coronary arteries. For the 2D models, the wrongly segmented structures are usually completely disconnected from the coronary artery branches, and are positioned above or below the heart in areas of the chest cavity that does not appear in the under-sampled training dataset. Of the disconnected structures appearing closer to the main coronary artery branches, many are in fact parts of the coronary arteries. The gaps in the segmentations leading to these parts becoming disconnected may be caused by several things. The blood vessels, at the position of the gaps, can in fact be so small that the original resolution of the CT input image is too low to determine whether there is a vessel at that point or not. Certain depth slices in the CT images can also be affected by artifacts, originating from the physical properties of the subject, movements during imaging, or other sources. Without clinical expertise it is difficult to precisely determine the true source of these errors.

In order to clean up the segmentations and remove spurious responses, post-processing as described in Section 3.6 was applied to the volumes. The refinement process for binary segmentations is made up by two well-known image processing procedures; connected components-analysis and binary closing. Existing efficient open-source implementations of these algorithms were used, and for binary segmentations the post-processing time is under 10 seconds per volume when run on a laptop with limited resources. The refinement process was devised under the assumption that the coronary arteries consists of two main structures; the right and left coronary arteries.

The refinement has a strictly positive impact on the DSC of the segmentations, as can be seen in Table 4.3, 4.4, and 4.5. The only exception is the full-size slice segmentation of VR89 using BVNet, where DSC decreases from .614 to .612. The reason for this can be seen in the bottom row of Figure 4.7. For this specific volume, if the network segmentations are to be trusted, the coronary arteries seems to consist of three branches. The refinement process assumes only two branches, and the RCA that can be seen in the hand-labeled volume is smaller than the connected component selected by the refinement process. This is the case for the segmentation output of all three models. The assumption that the coronary arteries consist of two, and exactly two, branches is not valid in all cases. Coronary artery anomalies include among other things the RCA and LM originating from the same ostium, resulting in only a single branch [2]. This assumption can therefore be considered a weakness of the method. A possible solution could be to instead select connected components based on a volume threshold, similar to the method used for finding candidate components in the multi-class refining process.

Qualitative Evaluation of Segmentations

Table 4.3 shows that the mean DSC achieved by BVNet is higher than that of UNet2D, and only slightly lower than what is achieved by UNet3D. As demonstrated by Figure 4.7, this result might be somewhat misleading. Visually inspecting the segmentations reveal that BVNet and UNet2D achieves similar segmentation performance, but with slight differences. BVNet generally includes more smaller vessels than UNet2D, but the latter is able to segment larger parts of the main vessel structure. As can be seen in Figure 4.7e-h, UNet3D includes significantly more of both the main vessels and smaller branches than the 2D models, but this is not reflected in the DSC. For this specific volume, V27, UNet3D achieves the lowest DSC of the three models, with $\text{DSC}_{\text{refined}} = 0.746$ (versus .786, and .754 for the 2D models).

This specific example illuminates a problem with evaluating the performance of the three networks; when segmentation outputs include more vessels than the hand-labeled ground truth, DSC will decrease as a number of voxels are regarded as false positives. This also creates further problems when it comes to determining which of the network architectures is best suited to the task of coronary artery segmentation. If one of the architectures achieve qualitatively better, but quantitatively worse performance than the others, is it because it is able to better model the concept behind the task, or is it because the other models are better at mimicking the segmentations seen in the training data? ANNs work by trying to minimize error of output when compared to a ground truth. The theoretically optimal performance will therefore be limited by the quality of the hand-labeled segmentations. For this task, it is likely that the larger receptive field in the z-direction of UNet3D gives it a significant advantage when segmenting tubular structures. However, both of the 2D architectures have significantly more parameters than the 3D model, and as such may be able to approximate more complex functions.

As it stands, there are two possibilities for extensive evaluation of network performance. One would be a qualitative evaluation performed by a clinical expert, with a blinded side-by-side comparison of automatic and hand-labeled segmentations. This would provide insight into which of the models would perform best in a clinical setting, and give indications of which areas the models would need to improve on. Another option is to perform computational FFR measurements using the automatic segmentations, and compare the results to both computational FFR done on the hand-labeled volumes, and in-vivo measurements. This would give a quantitave performance measure where it would be possible to determine if the segmentations are truly better than the hand-labeled volumes for the computational FFR use-case.

Segmentation approaches

Two different approaches to full-volume segmentation was evaluated on the validation set. Table 4.3 and 4.4 show that the mean DSC of both raw and refined segmentations is in fact slightly higher when using full-size slice segmentation, as opposed to overlapping slices of the same shape as the training input. This is somewhat surprising, as one generally expect poorer performance when predicting inputs of a different shape than the model was trained on due the learned global features no longer applies to the new crop size.

Table 4.2 show that there is a significant reduction in inference time when predicting full-size, non-overlapping slices. The time gain comes mainly from eliminating overhead in both slice extraction and fewer memory transfers onto the GPU, but is not directly proportional to the reduction in number of segmentations. This is because running inference on larger slices requires more calculations per slice.

When comparing the two approaches, the same problem as for the general evaluation of segmentations is present. The higher DSC of the full-size segmentations may in fact be due to lower-quality segmentations, and the actual performance is hard to determine without knowing the true quality of the hand-labeled ground truths. Taking the pragmatic view of 'more is more', i.e. assuming that all vessel segmentations, even when not present in the ground truths, are correct, full-size slice segmentation seems to perform slightly worse than the overlapping slice approach. This can be seen in Figure 4.4 and 4.5.

The slice-wise full-volume segmentations using a lower stride of (32, 32) (Figure 4.8 and 4.9) seem to support that in fact, more *is* more. The mean DSC_{RAW} and $DSC_{refined}$ for these segmentations were slightly lower than for the less-overlapping segmentations, as shown by Table 4.5, but visual inspections of the segmentations indicate that

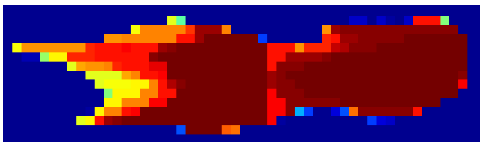


Figure 5.1: Close-up view of a slice from binary coronary artery segmentation of VP8 using a highly overlapping segmentation approach. Blue represents low positive prediction rate, dark red represents high positive prediction rate.

running inference on each part of the volume multiple times results in more complete segmentations with fewer gaps and better extraction of small vessel structures. A major drawback of selecting a smaller stride is that the inference time is vastly increased, as the number of predictions needed to segment the full volume increases quadratically. For a CT volume with shape $(512, 512, N)$, a slice shape of (256, 256) and a stride of (128, 128) requires 3*3*N predictions for full-volume segmentation. Decreasing stride to (32, 32) increases number of predictions to 9*9*N. This results in the inference time increasing from less than a minute to more than 15 minutes per volume.

A lower stride, such as (32, 32), means that voxels in the input image are segmented several times. Accumulating positive predictions in a final output volume makes it possible to present a positive prediction rate for each voxel in the volume. Due to the large input slice size of (256, 256), which for certain volumes is more than half of the full volume (x, y) size, the overlapping segmentations result in voxels close to the center of each slice receiving more passes than the voxels closer to the edge. This can be accounted for by dividing the positive prediction rate of each voxel by the number of times it has been segmented, effectively normalizing the PPR. Given such a normalized PPR segmentation, it could be possible to, instead of using connected components analysis to refine segmentations, use a simple threshold for pruning spurious responses. As it stands, the overlapping segmentation method does not account for the bias towards center voxels, and segmentation refinement through thresholding could result in the removal of correctly predicted branches.

The multiple segmentation of voxels allows for a visualization of the positive prediction rate (number of times a voxel has been predicted as a foreground voxel), as shown in the left columns of Figure 4.8 and 4.9. Examining a cross-sectional view of a coronary artery segmented using cumulative overlapping segmentations, such as can be seen in Figure 5.1, reveals how the certainty of a voxel belonging to the foreground becomes higher closer to the center of the vessels, despite the fact that this cross-sectional view is extracted from close to the center of the volume. It is unclear whether including the vessel boundaries with lower positive prediction rate in the final segmentation affects the segmentation quality in a positive or negative manner.

5.1.2 Multi-class Coronary Artery Segmentation

In the multi-class coronary artery segmentation task, the differences in performance between the three models were much greater than for binary segmentation. Table 4.6 shows how none of the models achieved a very high performance on the segmentation of the right and left coronary branch, but both BVNet and UNet2D was able to segment the aorta with a high degree of accuracy. There is a stark difference in the performance of the 2D models and UNet3D, as the latter has very poor results on RCA segmentation, and completely fails to segment the aorta in the validation volumes.

Figure 4.3c may help give a clue as to why the 3D model performed so poorly; It may seem that during training, after the first 10 epochs, there is a drastic shift in 'focus', as the loss for the AA label rapidly drops, while the loss for the two other labels increases. After this sharp drop in loss, the aorta segmentation stops improving completely. This may be the result of the model landing in a steep local minima, and gradient descent being unable to determine which direction to go in later steps. The momentum term in the update rule (Eq. 2.15) should help the model move across such local minima, and it is therefore not clear that this is the case. The Dice loss used during training calculates DSC based on real-valued numbers in the $(0,1)$-range. With the aorta taking up a significant volume in the input images it is possible that the model has learned to output some constant in the $(0,1)$-range for each voxel in the segmentation of the aorta, as this will receive a better score than other attempts. This argument is opposed by the fact that the network uses softmax activation in the final layer, giving a probability of each voxel belonging to a given class. When using softmax, the final output for each voxel sums to 1 across the output channels. As the loss for the coronary artery labels keeps decreasing, this should affect the DSC of the aorta label, as the output for that channel should change accordingly. However, as Figure 4.3c shows, only minor perturbations in the loss for the aorta label can be observed after epoch 10. It may also be the case that the loss for the aorta label stops improving as a result of floating point errors, or some other numerical error, but this is difficult to verify.

Looking at Fig. 4.3b, it is clear that Unet2D reached a point of numeric instability around epoch 30, similar to what was observed with the training of UNet3D. Up until this point the loss for the RCA label has been steadily decreasing, and is lower than the loss for LM. At epoch 30 this changes, and the network switches 'focus' from RCA to LM. In epochs 30 to 45, stretches of near-unchanging loss can be observed in the Dice loss for the aorta label. This is reminiscent of what happened during training of UNet3D, but the model is able to overcome the apparent local minimas at these points. After epoch 45 the loss for the aorta label starts to drastically decrease, and improves way beyond what is seen in the training performance of BVNet. It is quite possible that with longer training, this network might continue to increase perfomance. It is worth noting however, that the drastic changes around epoch 30 only results in improved aorta segmentation, and actually decreases overall performance of RCA and LM segmentations compared to BVNet. It could be the case that further training only would lead to the RCA and LM loss plateauing at a similar point as with BVNet. Seeing that UNet2D waas able to overcome periods of poor gradients for the aorta label, it is possible that UNet3D could also continue to improve the aorta segmentations if the gradient descent updates sufficiently stir the parameters in the earlier layers.

Looking at the training performance of BVNet and the first 30 epochs of UNet2D it seems that the learning rate used in the training is not optimal. The chosen learning rate was based on experiments done with binary classification, but the parameter space in the multi-class task likely has a very different topography than in the binary task. It would be interesting to see different approaches to learning rate schedules, such as cyclical learning rate schedules [40] or warm restarts [27] applied to these networks.

Multi-class Segmentation Refinement

As demonstrated by Table 4.6, the raw segmentation outputs of the networks are not very precise. Figure 4.10 shows how the segmentation output of BVNet contains a lot of errors in the coronary artery segmentations. This is also the case for the output of UNet2D, as can be seen in even in the refined outputs shown in Figure 4.12. These visualizations suggest that the networks are largely unable to tell which class a voxel at the border of a blood vessel belongs to. The outputs of all three networks show a tendency to segment a porous, shell-like structure around the aorta, and for the RCA label this is also present along the left coronary branch. The refinement process used for binary segmentation output has a tendency to select the wrong connected component in the multi-class segmentations, as there is a significant volume of erroneous classifications. For this reason, the solidity property (Eq. 3.3) was used to determine the connected component most likely to be a coronary artery branch.

The refinement process appears to produce the best results on the segmentations made by BVNet. The raw segmentations, seen in Figure 4.11, show how the RCA an LM segmentations from UNet2D contains more, and denser, erroneous segmentations than the BVNet segmentaitons. It is therefore more often strongly connected to the coronary artery branches, and connected components analysis group more of the erroneously classified voxels together with the correctly classified voxels. Attempts were made to apply morphological operations to the final segmentations in order to further remove the wrongly classified voxels surrounding the aorta, but the coronary artery structures are so small and intricate that the final result becomes too distorted to have any practical value. As Figure 4.12 shows, the results of segmentation refinement on the output of UNet3D is very poor. The raw segmentations are very noisy, and consists of multiple disjoint segments, and the refinement process will therefore at times select components not part of the coronary arteries at all. This results in the mean DSC for the refined UNet3D RCA segmentations actually being lower the raw DSC, going from 0.137 to 0.053.

All three models are able to segment the left coronary branch better than the right. This is inconsistent with what is seen in the training loss, where the loss for the RCA label is in fact lower than for the LM label. This discrepancy might be the results of the left coronary branch generally being larger than the right. When calculating per-label DSC, the larger volume of the LM label will result in a harder punishment if only a few voxels are correctly classified. The LM will also be present in more slices than the RCA, and so the average loss over multiple slices might suffer more from missing segmentations. However, as the validation loss in Figure 4.3 shows, the validation loss for the LM is generally much lower than for RCA. Seeing how the RCA is present in fewer slices than LM, it might indeed be the case that all three networks were able to overfit on the RCA prediction, but not the LM prediction, leading to the differences seen in training and validation loss.

The proposed method for refining multi-class segmentations is very domain-specific, and as such might not be very useful for other tasks. It was devised specifically for the refinement of the output from the specific models trained on the multi-class St. Olavs FFR dataset. If the same architectures were trained on a different dataset, it might not be the case that the segmentation output would have properties similar enough for the multi-class refinement procedure to work properly. However, the concept of refinement using the solidity of connected components could still be useful to apply in other contexts, as the desired components might have characteristic solidity properties differing from potential spurious responses.

5.2 Liver Vessel Segmentation

The results on the 3D-IRCADb-01, seen in Table 4.8 and 4.9, show that the networks performed worse in this task than for the coronary artery segmentation tasks. This was to be expected, as the 3D-IRCADb-01 dataset only consists of 20 images, of which 3 were reserved for validation. This results in only 17 volumes available for training, significantly less than the 41 volumes used for training in the coronary artery tasks. Fewer training samples will necessarily lead to a less robust model, as the training data will only contain a small subset of the variations seen in the full population.

To further complicate matters, the dataset is very inconsistent in how much of the vena cava and lower parts of the portal vein is in included in the hand-labeled segmentations, as can be seen by comparing Figure 4.13e and Figure 4.13o. This inconsistency is problematic when using gradient descent to learn segmentations, as very similar inputs will have very different ground truth labels. If one training batch only contain samples which consistently segments a large part of the vena cava and/or the portal vein, and the following batch contain samples which consistently do *not* segment the same parts, these updates will more or less undo each other. Larger batch sizes and the use of the momentum term in the update rule should help reduce the impact of this issue (as opposed to say training with a batch size of 1 and no momemtum), but the resulting model will most likely still end up with inconsitent segmentations.

As can be seen in the top row of Figure 4.13, the inconsistency in ground truth labels have caused both BVNet and UNet2D to learn to segment more of the larger veins than what is in the ground truth label. In this instance, the larger segmentation ends up having a negative effect on the quantitative validation score. For another volume, where the ground truth also has the larger segmentation, the effect would be opposite; increasing the DSC of the segmentation. This makes it hard to go by DSC alone when evaluating segmentations.

It is worth noting that the resolution of the 3D-IRCADb-01 dataset is much lower than the resolution of the St. Olavs FFR dataset. The mean resolution of 3D-IRCADb-01 is $(0.73\,\text{mm}, 0.73\,\text{mm}, 1.79\,\text{mm})$. This leads to the diameter of the blood vessels in terms of voxels being a lot lower than in the coronary artery segmentation task. Fewer positive voxels in essence leads to fewer samples to learn from. Resampling the dataset to a fixed voxel size of $(0.8\,\text{mm}, 0.8\,\text{mm}, 0.8\,\text{mm})$ will also have introduced errors in both the input images and the ground truth labels. This especially affects the ground truth volumes, as the labels are integers. When resampling, nearest-neighbor interpolation was used in order to preserve the integer labels and not end up with real-valued labels. As can be seen in the ground truth volumes in Figure 4.13, this results in the ground truth labels of the HV and PV having uneven surfaces, and might also introduce wrongly classified voxels in the hand-labeled segmentation volumes.

Despite the lack of training data, and the inconsistent and possibly imprecise ground truth labels, both BVNet and UNet2D were able to learn to create reasonable approximations of the vessel structures in the liver. The network of vessels in the liver is highly complex and intertwined, and the fact that neural networks only observing a 2D cross-section of the volume at a time is still able to differentiate between the two vessel networks is somewhat surprising. The results on the 3D-IRCADb-01 dataset were truncated by the fact that UNet3D completely failed to train due to implementation errors, and a comparison between the 2D and 3D models are therefore not possible to do on this dataset. Extrapolating from the results on the coronary artery segmentation task is also difficult, as UNet3D outperformed the 2D models in the binary segmentation task, but severely under-performed in the multi-class task.

Segmentation Refinement

Refinement of the segmentations from the 3D-IRCADb-01 dataset was done by greedily extracting the largest connected component from the raw network outputs. This works well enough for removing small groups of disconnected, erroneously classified voxels, but is perhaps a little too simple, as it also removes a lot of correctly classified voxels. This can be seen by comparing the mean DSC of the raw and refined segmentations (Table 4.8 and 4.9), where the refinement only has a positive effect on the liver and hepatic vein segmentations from UNet2D. In the other cases ut actually results in a decrease in DSC.

It is interesting to see that the raw liver segmentation output of the networks display some of the same properties as is seen in the multi-class coronary artery segmentations. Parts of the vessel walls of the hepatic and portal vein is wrongly classified as part of the liver, similar to how the corornary artery segmentations also segmented parts of the aorta. In these segmentations, however, the wrongly classified voxels are strongly connected to the correctly segmented liver, and cannot be pruned using the solidity property. It is however possible to remove some of the erroneous classifications by using binary opening. The downside of applying binary opening to the final segmentation is that it results in a loss of precision at the edges of the organ, and as such was not included as a step in the refinement process.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis details the implementation and evaluation of a method for fully-automatic segmentation of blood vessels from volumetric medical images using deep convolutional autoencoders. The neural networks were trained and evaluated on two datasets; one dataset consisting of 51 hand-labeled coronary CT angiograms from St. Olavs Hospital, and the 3D-IRCADb-01 dataset from IRCAD, consisting of 20 CT images of the abdomen, where all major structures are labeled. The results show that all three models–BVNet, UNet2D, and UNet3D–are able to extract very reasonable binary segmentations of the coronary arteries, with UNet3D having a slight advantage over the 2D models. In some cases it can even be argued that the network output has super-human performance. However, this is difficult to verify without clinical expertise. The raw binary coronary artery segmentations from the networks require some post-processing in order to clean up spurious responses. The proposed method for refinement uses connected components analysis, but is still able to include components disconnected from the main branch, as long as the gap is not too large. The refinement method has room for improvement, as in certain cases the wrong vessels will be selected as part of the coronary arteries. This is due to assumptions made not being entirely valid. The answer to the first research question is then that yes, deep convolutional autoencoders are indeed viable for coronary artery segmentation.

The second research question asks if deep convolutional autoencoders can perform satisfactory multi-class segmentations of medical images in a single pass. The results of multi-class segmentation of the coronary arteries show that at least the 2D models are capable of learning the difference between the right and left coronary arteries, as well as the aorta. The raw segmentations have some characteristic errors that can to some degree be removed by using a novel method for segmentation. The proposed method for refinement uses a measure of the solidity of each connected component larger than a given threshold in order to determine which of the connected components are most likely to be a coronary artery branch. This greatly improves the quality of segmentations using BVNet. However, it does not have the same positive effect on the segmentations from UNet2D and UNet3D. Using BVNet, it is possible to produce reasonable multi-class segmentations of the coronary arteries in under two minutes. The method was also validated on liver and liver vessel segmentation using the 3D-IRCADb-01 dataset. The results on this dataset were not as good as with the coronary artery dataset, but still show that 2D models are able to differentiate between complex, intertwined three-dimensional structures even after training on a small dataset with sub-optimal ground truth labels. With more training data, these architectures should be able to produce segmentations of a higher quality. This answers the second research question.

The final research question asks if there is a significant gain in the performance of volumetric medical image segmentations when using 3D convolutional autoencoders as opposed to 2D convolutional autoencoders. The results on binary coronary artery segmentation show that the 3D model has marginally better performance when using quantitative evaluation metrics. Qualitative evaluation through visual inspection reveal that the 3D model segments significantly more vessels than the 2D models. This is not reflected in the Dice similarity coefficient as many of those vessels are not included in the ground truths. In the multi-class coronary artery segmentation tasks, the 3D model failed to achieve any satisfactory performance, possibly due to the smaller receptive field in the (x, y)-direction making it more difficult for the model to learn good reference points. Going by the results in the binary segmentation task, it is reasonable to say that yes, there is a significant gain in segmentation quality when using a 3D convolutional autoencoder. However, the gain comes at a cost of greatly increased inference time. Further work is needed in order to answer whether the 3D model can outperform the 2D models in multi-class segmentation.

6.2 Future Work

The results show that all the models used in the experiments are able to create satisfactory segmentations of the main branches of the coronary arteries. However, some of the smaller vessel structures are often missing from the segmentation output. One possibility for improving network performance in these areas would be to use a weighted loss, where the output error is weighted proportionally to the diameter of the blood vessel. This would provide an incentive for the networks to learn more precise segmentations of these small structures.

The performance of the 2D models on all of the segmentation tasks also suggest that other network architectures, such as Mask R-CNN[17] could possibly be applied to volumetric medical image segmentation. This could provide even further improvements in inference time.

The training of the models has room for improvement, especially in the case of multi-class segmentation. As discussed in the previous chapter, the learning rates used in the training of these models are most likely not optimal. Extensive hyperparameter experiments for determining better initial learning rates could result in better performance across the board. Applying learning rate schedules with occasional increases in learning rate instead of the monotonically decreasing learning rate schedule used in these experiments could also be of benefit.

Appendix A

Complete DSC of Multi-class Coronary Artery Segmentation Task

The following pages contains tables detailing the complete results of raw and refined DSC scores for each volume in the validation dataset, using the three different models; BVNet, UNet2D, and UNet3d.

	Slice-wise						Full-size					
	RCA		LM		AA		RCA		LM		AA	
Volume	DSC_{RAW}	DSC_{ref}	DSC_{RAW}	DSC_{ref}	DSC_{RAW}	DSC_{ref}	DSC_{RAW}	DSC_{ref}	DSC_{RAW}	DSC_{ref}	DSC_{RAW}	DSC_{ref}
V8	0.436	0.818	0.640	0.833	0.900	0.905	0.492	0.501	0.666	0.847	0.910	0.910
V16	0.394	0.812	0.542	0.585	0.956	0.967	0.431	0.479	0.559	0.651	0.961	0.965
V27	0.339	0.594	0.452	0.700	0.973	0.974	0.345	0.363	0.513	0.765	0.974	0.974
VP5	0.422	0.815	0.558	0.733	0.933	0.946	0.470	0.490	0.601	0.766	0.947	0.947
VP8	0.484	0.823	0.384	0.738	0.871	0.949	0.515	0.600	0.422	0.781	0.907	0.962
VP9	0.379	0.807	0.559	0.760	0.954	0.957	0.400	0.878	0.600	0.701	0.955	0.956
VR20	0.377	0.786	0.513	0.759	0.909	0.960	0.413	0.814	0.562	0.760	0.941	0.963
VR55	0.165	0.287	0.614	0.699	0.945	0.946	0.185	0.000	0.619	0.402	0.945	0.945
VR89	0.121	0.000	0.457	0.449	0.935	0.940	0.131	0.000	0.469	0.547	0.940	0.940
VR98	0.504	0.752	0.630	0.682	0.947	0.958	0.549	0.665	0.644	0.723	0.965	0.965
Mean	0.362	0.649	0.535	0.694	0.932	0.950	0.393	0.479	0.566	0.694	0.945	0.953

Table A.1: Raw and refined DSC for each class in multi-class coronary artery segmentation using BVNet

	Slice-wise						Full-size					
	RCA		LM		AA		RCA		LM		AA	
Volume	DSC_{RAW}	DSC_{ref}	DSC_{RAW}	DSC_{ref}	DSC_{RAW}	DSC_{ref}	DSC_{RAW}	DSC_{ref}	DSC_{RAW}	DSC_{ref}	DSC_{RAW}	DSC_{ref}
V8	0.302	0.716	0.481	0.719	0.928	0.885	0.404	0.424	0.608	0.604	0.887	0.887
V16	0.287	0.000	0.407	0.416	0.972	0.968	0.338	0.000	0.497	0.375	0.951	0.966
V27	0.351	0.586	0.577	0.674	0.883	0.973	0.305	0.333	0.428	0.456	0.972	0.973
VP5	.0.332	0.753	0.465	0.682	0.912	0.936	0.372	0.443	0.519	0.660	0.936	0.936
VP8	0.392	0.000	0.333	0.629	0.852	0.944	0.429	0.000	0.380	0.665	0.895	0.946
VP9	0.281	0.460	0.458	0.623	0.936	0.949	0.312	0.339	0.496	0.531	0.952	0.952
VR20	0.308	0.606	0.477	0.660	0.949	0.954	0.327	0.386	0.501	0.527	0.913	0.954
VR55	0.068	0.000	0.515	0.479	0.937	0.944	0.065	0.000	0.476	0.000	0.939	0.939
VR89	0.075	0.000	0.456	0.582	0.911	0.927	0.079	0.000	0.471	0.468	0.933	0.933
VR98	0.408	0.799	0.562	0.615	0.922	0.953	0.458	0.628	0.585	0.210	0.961	0.961
Mean	0.280	0.392	0.473	0.608	0.920	0.943	0.309	0.255	0.496	0.449	0.934	0.945

Table A.2: Raw and refined DSC for each class in multi-class coronary artery segmentation using UNet2D

Bibliography

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[2] Angelini, P. (2007). Coronary artery anomalies. *Circulation*, 115(10):1296–1305.

[3] Benjamin, E. J., Virani, S. S., Callaway, C. W., Chamberlain, A. M., Chang, A. R., Cheng, S., Chiuve, S. E., Cushman, M., Delling, F. N., Deo, R., de Ferranti, S. D., Ferguson, J. F., Fornage, M., Gillespie, C., Isasi, C. R., Jimenez, M. C., Jordan, L. C., Judd, S. E., Lackland, D., Lichtman, J. H., Lisabeth, L., Liu, S., Longenecker, C. T., Lutsey, P. L., Mackey, J. S., Matchar, D. B., Matsushita, K., Mussolino, M. E., Nasir, K., O'Flaherty, M., Palaniappan, L. P., Pandey, A., Pandey, D. K., Reeves, M. J., Ritchey, M. D., Rodriguez, C. J., Roth, G. A., Rosamond, W. D., Sampson, U. K. A., Satou, G. M., Shah, S. H., Spartano, N. L., Tirschwell, D. L., Tsao, C. W., Voeks, J. H., Willey, J. Z., Wilkins, J. T., Wu, J. H., Alger, H. M., Wong, S. S., and Muntner, P. (2018). Heart Disease and Stroke Statistics-2018 Update: A Report From the American Heart Association. *Circulation*, 137(12):e67–e492.

[4] Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., and Ronneberger, O. (2016). 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. *ArXiv e-prints*.

[5] C. Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

[6] Chollet, F. et al. (2015). Keras. `https://keras.io`.

[7] Dice, L. R. (1945). Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302.

[8] Dou, Q., Yu, L., Chen, H., Jin, Y., Yang, X., Qin, J., and Heng, P.-A. (2017). 3d deeply supervised network for automated segmentation of volumetric medical images. *Medical Image Analysis*, 41:40 – 54. Special Issue on the 2016 Conference on Medical Image Computing and Computer Assisted Intervention (Analog to MICCAI 2015).

[9] Douglas, P. S., Fiolkoski, J., Berko, B., and Reichek, N. (1988). Echocardiographic visualization of coronary artery anatomy in the adult. *Journal of the American College of Cardiology*, 11(3):565–571.

[10] Girshick, R. (2015). Fast R-CNN. *ArXiv e-prints*.

[11] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *ArXiv e-prints*.

[12] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR.

[13] Gonzalez, R. C. and Woods, R. E. (2006). *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[14] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

[15] Hahnloser, R. H. R., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., and Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405:947 EP –.

[16] Hassoun, M. H. (1995). *Fundamentals of Artificial Neural Networks*. MIT Press, Cambridge, MA, USA, 1st edition.

[17] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. *ArXiv e-prints*.

[18] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. *ArXiv e-prints*.

[19] Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251 – 257.

[20] Jacobsen, F. (2017). Coronary Artery Lumen Segmentation using Deep Learning. Unpublished.

[21] Kamnitsas, K., Ledig, C., Newcombe, V. F., Simpson, J. P., Kane, A. D., Menon, D. K., Rueckert, D., and Glocker, B. (2017). Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. *Medical Image Analysis*, 36(Supplement C):61 – 78.

[22] Kataria, T., Bisht, S. S., Gupta, D., Abhishek, A., Basu, T., Narang, K., Goyal, S., Shukla, P., Bansal, M., Grewal, H., Ahlawat, K., Banarjee, S., and Tayal, M. (2016). Quantification of coronary artery motion and internal risk volume from ECG gated radiotherapy planning scans. *Radiotherapy and Oncology*, 121(1):59 – 63.

[23] Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *ArXiv e-prints.*

[24] Kjerland, Ø. (2017). Segmentation of coronary arteries from CT-scans of the heart using deep learning. Master's thesis, NTNU.

[25] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

[26] Long, J., Shelhamer, E., and Darrell, T. (2014). Fully Convolutional Networks for Semantic Segmentation. *ArXiv e-prints.*

[27] Loshchilov, I. and Hutter, F. (2016). SGDR: Stochastic Gradient Descent with Warm Restarts. *ArXiv e-prints.*

[28] Milletari, F., Navab, N., and Ahmadi, S.-A. (2016). V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. *ArXiv e-prints.*

[29] Moeskops, P., Wolterink, J. M., van der Velden, B. H. M., Gilhuijs, K. G. A., Leiner, T., Viergever, M. A., and Išgum, I. (2017). Deep Learning for Multi-Task Medical Image Segmentation in Multiple Modalities. *ArXiv e-prints.*

[30] Nasr-Esfahani, E., Karimi, N., Jafari, M., Soroushmehr, S., Samavi, S., Nallamothu, B., and Najarian, K. (2018). Segmentation of vessels in angiograms using convolutional neural networks. *Biomedical Signal Processing and Control*, 40:240 – 251.

[31] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145 – 151.

[32] Ravì, D., Wong, C., Deligianni, F., Berthelot, M., Andreu-Perez, J., Lo, B., and Yang, G. Z. (2017). Deep learning for health informatics. *IEEE Journal of Biomedical and Health Informatics*, 21(1):4–21.

[33] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *ArXiv e-prints.*

[34] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *ArXiv e-prints.*

[35] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2014). ImageNet Large Scale Visual Recognition Challenge. *ArXiv e-prints.*

[36] Scholl, I., Aach, T., Deserno, T. M., and Kuhlen, T. (2011). Challenges of medical image processing. *Comput. Sci.*, 26(1-2):5–13.

[37] Simard, P. Y., Steinkraus, D., and Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*, ICDAR '03, pages 958–, Washington, DC, USA. IEEE Computer Society.

[38] Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv e-prints*.

[39] Smistad, E., Falch, T. L., Bozorgi, M., Elster, A. C., and Lindseth, F. (2015). Medical image segmentation on GPUs – A comprehensive review. *Medical Image Analysis*, 20(1):1–18.

[40] Smith, L. N. (2015). Cyclical Learning Rates for Training Neural Networks. *ArXiv e-prints*.

[41] Smith, S. L., Kindermans, P.-J., Ying, C., and Le, Q. V. (2017). Don't Decay the Learning Rate, Increase the Batch Size. *ArXiv e-prints*.

[42] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going Deeper with Convolutions. *ArXiv e-prints*.

[43] Taylor, C. A., Fonte, T. A., and Min, J. K. (2013). Computational fluid dynamics applied to cardiac computed tomography for noninvasive quantification of fractional flow reserve: Scientific basis. *Journal of the American College of Cardiology*, 61(22):2233 – 2241.

[44] Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

[45] Yang, X., Yang, J. D., Hwang, H. P., Yu, H. C., Ahn, S., Kim, B.-W., and You, H. (2018). Segmentation of liver and vessels from CT images and classification of liver segments for preoperative liver surgical planning in living donor liver transplantation. *Computer Methods and Programs in Biomedicine*, 158:41 – 52.

[46] Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. *ArXiv e-prints*.

[47] zhan Zeng, Y., hui Liao, S., Tang, P., qian Zhao, Y., Liao, M., Chen, Y., and xiong Liang, Y. (2018). Automatic liver vessel segmentation using 3d region growing and hybrid active contour model. *Computers in Biology and Medicine*, 97:63 – 73.