Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Structural Engineering

Emil Stubsjøen

Deep Learning analysis of the LV myocardium in CCTA for identification of patients with significant coronary artery stenosis

Trondheim, June 2021

NTNU – Trondheim
Norwegian University of
Science and Technology

NTNU – Trondheim
Norwegian University of
Science and Technology

Department of Structural Engineering
Faculty of Engineering
NTNU – Norwegian University of Science and Technology

ACCESSIBILITY

MASTER THESIS 2021

SUBJECT AREA: Biomechanics	DATE: 10.06.2021	NO. OF PAGES: 98

TITLE:

Deep Learning analysis of the LV myocardium in CCTA for identification of patients with significant coronary artery stenosis

BY:

Emil Stubsjøen

SUMMARY:

For patients affected by coronary artery stenosis of intermediate severity, the significance of the stenosis has to be determined. In clinical practice, measuring the fractional flow reserve (FFR) is one of the most commonly utilized methods. In this procedure FFR measurements are conducted during invasive coronary angiography (ICA), which has a small health risk associated with it.

In order to automatically identify patients with functionally significant coronary artery stenosis, a pipeline composed of several stages has been developed. To begin with, the left ventricular (LV) myocardium is segmented using a Convolutional Neural Network (CNN). Subsequently, the LV myocardium is characterized from encodings of an unsupervised/semi-supervised Convolutional Autoencoder (CAE). As changes in the tissue of the LV myocardium are anticipated to occur locally, the obtained automatic segmentation is split into 500 spatially connected clusters. Two different methods were employed for computing the patients features using statistics of the encodings. At last, patients are classified based on the presence of functionally significant stenosis using a FFR cut-off value of 0.8 for separating the negative and positive samples.

RESPONSIBLE TEACHER: Leif Rune Hellevik

SUPERVISOR(S): Fredrik Eikeland Fossan and Jacob Sturdy

CARRIED OUT AT: Department of Structural Engineering

Abstract

For patients affected by coronary artery stenosis of intermediate severity, the significance of the stenosis has to be determined. In clinical practice, measuring the fractional flow reserve (FFR) is one of the most commonly utilized methods for evaluating the severity of the disease. In this procedure FFR measurements are conducted during invasive coronary angiography (ICA), which has a small health risk associated with it. In Zreik et al. (2017), a non-invasive method that uses Deep Learning to extract myocardial properties from coronary computed tomography angiography (CCTA) has shown that it may be possible to get accurate predictions on determining functionally significant stenosis [4]. This is accomplished by segmenting and extracting structural features from the left ventricle (LV) myocardium. This thesis aims to reproduce and extend the methods proposed in Zreik et al. In particular an automatic pipeline was implemented which consists of three main steps: 1) automatic segmentation, 2) characterization of the LV myocardium through clustering and autoencoding and 3) final classification of patients with functional significant CAD based on features extracted in step 2. Moreover, several aspects of all three steps were explored.

A dataset of 66 CCTAs from patients that underwent invasive FFR measurements was utilized, where manually performed segmentations of the LV myocardium was available for 28 of the patients. In step 1 we evaluated three different CNN architectures for automatic segmentation of the LV myocardium via 3-fold cross-validation experiments. The best results were obtained with the U-Net Standard and DSL which obtained an average DSC of 0.89 across all three folds, which is comparable with results reported in Zreik et al. (DSC: 0.91). Furthermore, increasing the complexity of the CNN did not yield improved results. In step 2 several sub-tasks were performed. First a K-means algorithm which divides the myocardium into 500 sub-regions was implemented. Further, the data within each cluster were compressed by application of an unsupervised/semi-supervised Convolutional Autoencoder (CAE) which was trained to reproduce CCTA patches (2D sub-region of an image) associated with the clusters. We found that the performance of the CAEs was best for smaller patch sizes, which also provided best results in the classification (step 3). Finally, the information from the encodings of all clusters were combined into a vector of features characterizing the myocardium. Here, we evaluated the approach suggested by Zreik et. al and propose an alternative approach. In step 3 the classification was performed with both K-Nearest Neighbors (KNN) and Gaussian Process Classifier (GPC). We were not able to reproduce the results for patient classification presented by Zreik et. al (AUC ~0.74) by applying (our understanding of) their method directly. The best results were obtained with the GPC classifier by applying our new approach for combining/extracting features and by including an additional layer of feature-selection, which achieved an AUC of ~ 0.70. However, a shortcoming of this approach is related to the random ordering of clusters/features and the inability to consistently select features with high amounts of information in unseen populations. As a proof of concept, we show that high classification (AUC ~ 0.90) is possible if feature selection is performed on the entire dataset.

Sammendrag

For pasienter med innsnevring av moderat alvorlighetsgrad i koronararteriene må betydningen av innsnevringen defineres. I klinisk praksis gjøres dette vanligvis gjennom å måle FFR (Fractional Flow Reserve). I denne prosedyren måles FFR under invasiv koronar angiografi (ICA), som har en liten helserisiko knyttet til seg. I Zreik et al. (2017) har en ikke-invasiv metode som bruker dyp læring på CCTA (Coronary Computed Tomography Angiography) bilder for å hente «features» fra venstre ventrikkels (VV) myokard vist seg å gi gode predikasjoner i sammenheng med å avgjøre alvorlighetsgraden til en innsnevring [4]. Dette er blitt gjort ved å først segmentere, for så å hente ut strukturelle «features» fra VV myokard. Denne masteroppgaven sikter seg inn på å reprodusere og videreutvikle metodene foreslått i Zreik et al. En automatisk «pipeline» har blitt implementert, som består av tre overordnede steg: 1) automatisk segmentering, 2) karakterisering av VV myokard gjennom clusteranalyse og autoencoding og 3) endeling klassifisering av pasienter med koronar hjertesykdom av funksjonell signifikans basert på «featurene» hentet ut i steg 2). Videre har flere aspekter vedrørende alle de tre stegene blitt utforsket.

Et datasett bestående av 66 CCTA bilder av pasienter som gjennomgikk invasive FFR-målinger har blitt brukt, hvor manuelle segmenteringer av VV myokard var tilgjengelig for 28 av pasientene. I steg 1 har tre ulike CNN arkitekturer for automatisk segmentering blitt evaluert gjennom 3-fold kryssvaliderings eksperimenter. Best resultater ble oppnådd for U-Net Standard med DSL, som resulterte i en gjennomsnittlig DSC på 0.89 på tvers av alle folds. Dette er sammenlignbart med resultatene rapportert i Zreik et al. (DSC: 0.91). Videre viste resultatene at introduksjon av en mer kompleks CNN arkitektur ikke ga forbedrede predikasjoner. I steg 2 ble flere underoppgaver gjennomført. Først ble en K-means algoritme som deler myokard inn i 500 del-regioner (clusters) implementert. Videre ble data innenfor hver «cluster» komprimert gjennom anvendelse av en konvensjonell auto-encoder (CAE) som var trent opp til å reprodusere CCTA «patches» (2D sub-region av et bilde) knyttet til «clusterene». De beste resultatene for auto-encoderen ble oppnådd for mindre patch-størrelser, som videre også ga de beste klassifiserings resultatene (steg 3). Til slutt ble informasjonen fra kodingene produsert av auto-encoderen og «clusterene» fra K-means algoritmen kombinert til en vektor av «features» som en karakterisering av myokard. Her evaluerte vi metoden foreslått i Zreik et al., tillegg til å foreslå en ny alternativ metode. I steg 3 ble klassifiseringen gjennomført ved å bruke både K-Nearest Neighbors (KNN) og Gaussian Process Classifierr (GPC). Vi greide ikke å reprodusere resultatene fra klassifiseringen av pasienter presentert i Zreik et al. (AUC ~ 0.74) ved å anvende (vår tolking av) metoden deres direkte. Best resultater ble oppnådd for GPC klassifisering gjennom å anvende den nye foreslåtte metoden for å kombinere/hente ut «features» og ved å inkludere et nytt ledd med «feature selection», noe som resulterte en AUC på ~ 0.70. En svakhet ved denne metoden er relatert til den tilfeldige rekkefølgen av clusters/features og metodens manglende evne til å konsistent hente ut «features» med en høy mengde informasjon for en usett populasjon. Som et bevis på dette, viser vi til at høy klassifiseringsytelse (AUC ~ 0.90) er mulig dersom «feature selection» blir gjennomført basert på hele datasettet.

Preface

This thesis is written as my master thesis for the Departments of Structural Engineering at the Norwegian University of Science and Technology (NTNU). It extends the work done in my specialization project, where most of the methods and theory are still relevant.

I would like to thank my two supervisors Fredrik Eikeland Fossan and Jacob Sturdy for the excellent gradience and good advice throughout the process. Furthermore, I want to my supervisors for the opportunity to work on this exciting field, and for the opportunity to learn all the new things which were necessary in order to write this thesis.

Table Of Contents

List of Figures

List of Tables

Abbreviations

FFR	=	Fractional Flow Reserve
ICA	=	Invasive Coronary Angiography
CAD	=	Coronary Artery Disease
LV	=	Left Ventricle
CCTA	=	Coronary Computed Tomography Angiography
CAE	=	Convolutional Autoencoder
CNN	=	Convolutional Neural Network
ANN	=	Artificial Neural Network
TL	=	Tversky Loss
DSL	=	Dice Similarity Loss
DSC	=	Dice Similarity Coefficient
GPC	=	Gaussian Process Classifier
KNN	=	K-Nearest Neighbors
SVM	=	Support Vector Machines
ROC	=	Receiver Operating Characteristics
AUC	=	Area Under Curve
TP	=	True Positive
FP	=	False Positive
TN	=	True Negative
FN	=	False Negative
CV	=	Cross Validation

Chapter 1 Introduction

1.1 Motivation

Measuring the fractional flow reserve (FFR) is a commonly used method for determining the functional significance of coronary artery stenosis of intermediate severity. This method involves an invasive surgical procedure (invasive coronary angiography - ICA), that has a small health risk associated with it. Coronary artery disease (CAD) is the most frequent type of heart disease [1]. When one or more of the coronary arteries that are responsible for supplying blood to the heart are narrowed causing stenosis, it is an obstructive CAD. The narrowing happens as a result of plaque buildup in the inner wall of the arteries. The stenosis is said to be a functionally significant if it significantly restricts the supply of blood to the LV myocardium to a level that causes myocardial ischemia.

To reduce CAD morbidity, it is necessary to treat a functionally significant stenosis [2]. However, treating stenosis that is not functionally significant has been shown to do more harm than benefit. For that reason, an estimation of the severity of coronary artery stenosis' influence on LV myocardium perfusion is required. This is typically done by measuring FFR during ICA. FFR works as a quantitative marker of the stenosis' significance and is defined as the pressure measured distal (after) the stenosis relative to the pressure measured before (proximal to) the stenosis [3]. The result is an absolute number that has an ideal value of 1.0 (corresponding to no obstruction). Even though FFR is currently the standard technique used to determine the significance of coronary stenosis, the FFR cut-off value is not completely standardized. The cut-off value is the value that separates the functionally significant from non-significant stenosis. In clinical settings, values ranging from 0.72 to 0.80 have been utilized [4]. If a FFR measurement performed over a stenosis, lies below the cut-off-value, the stenosis is defined as functionally significant. Coronary CT angiography is a frequently used method to identify suspected CAD with high sensitivity (i.e., true positive rate). Although this method detects CAD with high sensitivity, it has restricted specificity in determining the functional significance of the stenosis [5][6]. Because of the poor specificity related to CCTA results, many patients then have to undergo invasive coronary angiography (ICA). As a result, roughly 22 - 52 % of patients unnecessarily undergo ICA and the risk associated with it [5].

As an alternative method to the invasive FFR measurements, previous work has shown that it is possible to get accurate results on detecting functionally significant stenosis (i.e., stenosis where the FFR measurement lies below the cut-off value) in the coronary artery using quantitative coronary analysis (QCA). In QCA the focus is on the geometry of the stenosis and does not look at the myocardium in general. Computational FFR is a non-invasive method presented for detecting functionally significant stenosis, which uses simulations of the blood flow to predict the pressure drop [7]. This method uses computational fluid dynamics and requires an accurate segmentation of the arteries and determination of boundary conditions.

Furthermore, another non-invasive method that uses myocardial properties from CCTA has shown to achieve accurate predictions [4]. This is accomplished by segmenting and extracting geometric features from the left ventricle (LV) myocardium, and then subsequently predict the significance of the stenosis using a Convolutional Autoencoder (CAE). As this removes the necessity for an invasive procedure, it can reduce the costs and risks associated with ICA.

Based on prior work, it would be expected to get accurate automatic segmentations utilizing a CNN model, choosing the right hyperparameters and architecture [4]. In Zreik et al. an average dice coefficient of 0.91 was achieved for the predicted segmentations. The segmented myocardium was subsequently encoded utilizing a CAE. The encodings were then further used to classify the occurrence of functionally significant stenosis in the coronary arteries based on a reference obtained during invasive FFR measurements. An average accuracy, sensitivity and specificity of 0.71, 0.70 and 0.71 were achieved, respectively. These results indicate that it is possible to use extracted features of the LV myocardium from CCTA-scans and get promising values of predictability of functionally significant stenosis compared to FFR measurements done during ICA.

1.2 Project Goals

The main goal of this thesis was to reproduce the pipeline for identification of patients with significant disease proposed by Zreik et al. and evaluate the methods on a novel dataset. This consists of three separate tasks. To be able to make use of myocardial properties associated with functionally significant stenosis, a pipeline including the following steps have been developed:

(1) A CNN model for automatic segmentation of LV myocardium
(2) Characterization of the segmented LV myocardium via clustering and CAE
(3) Classifying the presence of functionally significant stenosis based on these features

Secondary goals were to evaluate the effect of different choices necessary within each of these tasks (see contribution). The CNN model for automatic was trained on a limited dataset consisting of 28 CCTA-scans with a belonging manual segmentation of the LV myocardium, which was utilized as a variation database. Extensive data augmentation was used to overcome the sparsity of the training data, as well as several preprocessing methods. The model was trained utilizing different versions of the 3D U-Net architecture, which include U-Net Standard, U-Net Residual, and U-Net Compact. All of these have shown to provide state-of-the-art results for automatic segmentation of 3D images over the last years [8][9][10]. Two different loss functions were tested to overcome the issue of class-imbalance, which include the Tversky Loss and Dice Similarity Loss. The evaluation of each experiment was performed via 3-fold cross-validation. The best CNN model was used to obtain automatic segmentations of the entire dataset of 66 CCTA-scans.

In step (2) of the pipeline, the obtained automatic segmentations were clustered by the K-means algorithm. A Convolutional Autoencoder (CAE) was trained on extracted myocardium

patches from the 28 CCTAs with manual segmentations. Multiple patch-sizes, architectures and preprocessing techniques were explored. The trained CAE model and the clustered LV myocardium was used to extract features from all the 66 patients. Finally, in step (3) the patients were classified based on a FFR cut-off-value of 0.8 using multiple machine learning methods.

1.3 Contribution

The pipeline proposed in Zreik et al. consists of several steps, where certain aspects regarding the different steps were not evaluated in detail. Consequently, this thesis aims to explore some of those aspects.

The first contribution of this thesis is to find the optimal CNN hyperparameters for developing a model for automatic segmentation of the LV-myocardium. This includes the architecture, loss function and preprocessing techniques.

Next, it explores the impact of the CAE hyperparameters and preprocessing techniques for encoding of myocardium patches. Two methods for building the final patient feature vector from the CAE encodings have been explored. The first method utilized is a new method proposed in this thesis, whereas the second method is an interpretation of the method proposed in Zreik et al. The last contribution consists of finding the optimal model for the final patient classification of the extracted features.

1.4 Outline

This section gives a brief overview of the structure of the thesis.

Introduction

This chapter begins with an explanation of the main motivation for developing a pipeline for predicting functionally significant coronary artery stenosis. Furthermore, it covers the main goals, research questions, and the contribution of the thesis. At last, the overall structure of the thesis is presented.

Background

The background chapter starts with a clarification of some basic medical terms used in this thesis, which include the structure and functionality of the LV myocardium, CCTA images and FFR. Subsequently, an overall introduction to the most important deep learning techniques is presented.

Methodology

In this chapter the methods used for performing the different steps of the total pipeline are presented. It starts with a detailed explanation of the properties of the dataset, which is taken into consideration when developing the different steps of the pipeline. Furthermore, the training strategies, architectures, evaluation metrics and preprocessing techniques are

presented for the CNN and the CAE models. An explanation of the k-means clustering algorithm is presented, as well as the methods used for the final patient classification.

Results

The result chapter contains results of the different experiments completed, which includes the CNN for automatic segmentation, k-means clustering, CAE, and finally patient classification. The evaluation metrics are presented in tables, and visualizations are included for the CNN and CAE models.

Discussion

In the discussion chapter the results obtained for the experiments of different steps of the pipeline are discussed.

Conclusion and Future work

The final chapter gives a conclusion of the results and discussion and finishes with suggestions for improvements and future work for the different steps of the pipeline.

Chapter 2 Basic Theory

This chapter firstly explains some important medical terms in section 2.1 and 2.2 in order to get a deeper comprehension of the motivation and the dataset. Next, important concepts regarding deep learning will be described in section 2.3, touching a large proportion of techniques implemented for the different steps of the pipeline.

2.1 LV Myocardium

The dataset in this thesis is composed of CCTA-scans of the LV myocardium. The overall anatomy and general functionality of the LV myocardium are described in subsection 2.1.1. Coronary artery disease is explained in subsection 2.1.2, and the definition of FFR is given in in subsection 2.1.3.

2.1.1 Structure and Functionality

The heart consists of three layers: Endocardium as the innermost layer, myocardium as the middle layer, and epicardium as the outmost layer. The myocardium is the strongest of the three layers and contributes to the shape and functionality of heart [11].

In a human heart there are a total of four chambers, two of them being the left ventricle and the right ventricle. They operate in a double circulatory system, where the right ventricle pumps blood into the pulmonary arteries of the lungs. The left ventricle is supplied with oxygen-rich blood from the lungs (via the left atrium), and its functionality is to pump this blood through the aorta further to all regions of the human body via the systemic circulation. The wall thickness of the myocardium varies between the different sections and depends on the specific function of the section. The pressure in the left ventricle is higher compared to the other chambers, which makes it thicker compared to other sections of the heart. The coronary arteries depart from the aorta directly after the aortic valve and they provide oxygen-rich blood to the muscle tissue of the myocardium surrounding the left ventricle.

2.1.2 Coronary Artery Disease

Coronary artery disease (CAD) is the most frequent type of heart disease. When one or more of the coronary arteries that are responsible for supplying blood to the heart are narrowed causing a stenosis, it is an obstructive CAD. As illustrated in **Figure 1**, the narrowing happens as a result of plaque buildup in the inner wall of the arteries. The stenosis is said to be functionally significant if it significantly restricts the supply of blood to LV myocardium causing myocardial ischemia.

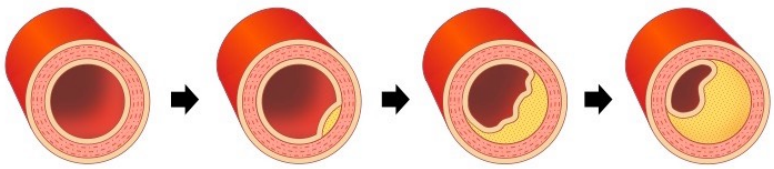


***Figure 1**: Illustration of artery narrowed by plaque [12].*

2.1.3 Fractional Flow Reserve (FFR)

FFR is a method used to determine the severity of a stenosis in the coronary arteries given by

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

where $\bar{P}_{distal}$ is the averaged pressure downstream the stenosis and $P_{proximal}$ is the averaged pressure in the Aorta. Both measurements are determined by averaging over a cardiac cycle. The pressure downstream can be quantified with a sensor-wire, while the pressure in the Aorta can be quantified by means of a catheter. The FFR measures the percentage of the remaining degree of blood flow that supplies areas of the myocardium downstream of the stenosis. If the FFR is measured to i.e., 0.8, the blood supply to the myocardium is 80% of what it would have been if there was no stenosis present. The cut-off value is the value that separates the functionally significant from non-significant stenosis. In clinical settings, values ranging from 0.72 to 0.80 have been utilized**.** If FFR measurements done over a stenosis lie below the cut off-value, the stenosis is defined as functionally significant, and an intervention/surgery is typically required.

2.2 Coronary Computed Tomography Angiography (CCTA)

Coronary Computed Tomography Angiography (CCTA) is a non-invasive method that uses a combination of X-rays and modern computer technology to obtain high-resolution images of the coronary arteries. The scans are stored in DICOM-format, but can be converted to different formats. In particular the NIfTI-format (Neuroimaging Informatics Technology Initiative)[13] stores data as a 3D image matrix with additional metadata, e.g., the thickness of CT slices.

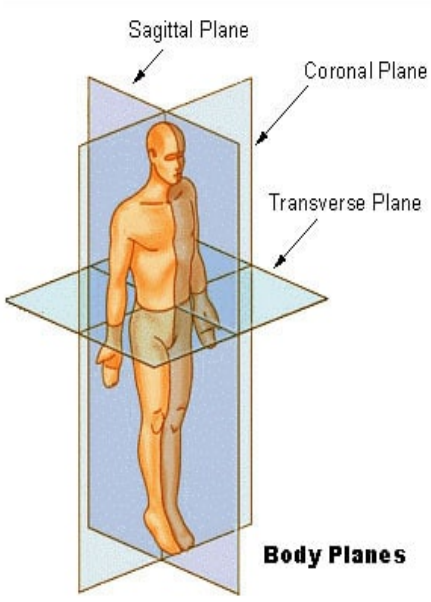


Figure 2*: Visualization of the three body planes [43].*

A CCTA-scan is performed by a radiographer, and it is used for visualization of organs, vessels or other tissues of the human body. The scan is performed on a patient lying on an X-ray sensitive plate, which is encircled by sensors. The X-ray source is also encircled by these sensors, and the scan is performed by rotating the source together with the sensors around the patient. Throughout this process of rotation, the X-ray goes through the patient and into the sensors on the opposite side of the source. A 2D slice of a delimited area is constructed when one rotation has been completed and the source is back at its starting point. A complete CCTA scan is created by stacking several 2D slices together[14]. This is done in one of the three body planes, illustrated in **Figure 2**.

The greyscale in medical CCTA imaging is recorded in Hounsfield Units. In order to increase the visibility of the arteries, an injection of a contrast medium is performed on the patient. The scale ranges from -1024 HU (black) to 3072 HU (white), where different tissue types have distinct values on the HU-scale. Water is centralized on the scale and is represented by 0 HU. Air has the lowest intensity of 1024 HU, while the maximum intensity of 3071 HU is produced by the densest tissue, i.e., bones or tooth enamel. Other tissue is somewhere between these two points on the scale. Fat is represented by approximately -100 HU, while muscles have an intensity of around 100 HU. From this point on, the term CCTA image is used when referring to a CCTA-scan.

2.3 Deep Learning

Deep Learning is a subfield of machine learning, which is an application of Artificial Intelligence (AI). In the field of machine learning, computers are able to learn patterns from data without being programmed explicitly. Artificial Neural Networks (ANNs) aim to mimic a biological brain, motivated by its structure and function. In this section, the fundamentals of ANNs are firstly presented in subsection 2.3.1, followed by an explanation of how the network learns patterns through optimization in subsection 2.3.2. Next, some common data processing techniques are described in subsection 2.3.3. Some common evaluation methods for machine learning models are presented in subsection 2.3.4. Concepts of regularization are explained in subsection 2.3.5. This chapter ends with a description of Convolutional Neural Networks (CNNs) and Convolutional Autoencoders (CAEs) in subsection 2.3.6.

2.3.1 Artificial Neural Networks (ANNs)

ANNs contain input and output layers, and normally also one or more hidden layers in between. The hidden layers are composed of what is known as artificial neurons, which aims to mimic the functionality of a human brain.

Artificial Neurons

The hidden layers are composed of components that transform the input into something the output layer can utilize. In a biological brain, neurons are cells that get signals from dendrites, which is a tree-like extension in the beginning of the neuron. The input values x_i of an artificial neuron can be viewed as features for which the neuron predicts an output value y. The strength of a connection between two neurons, i.e., how much a given input should count,

is controlled by the weight w. A summation function binds the inputs and weights and calculates the total sum. In addition, a bias is added to the result of the summation before it is sent to the activation function f, which calculates the activation of the neuron, y_{pred}. The bias is added to shift the activation function to the left or right, in order to get a better separation of the data. The overall structure of an artificial neuron is illustrated in **Figure 3**.

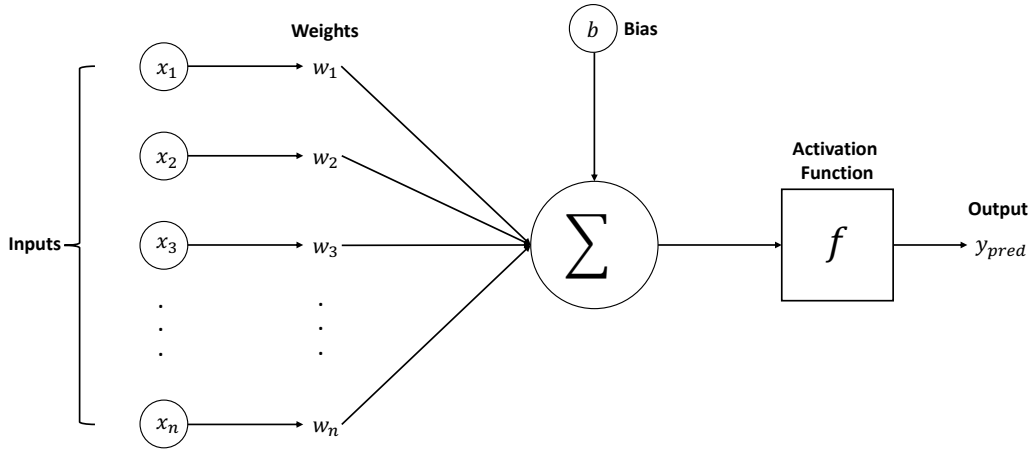


***Figure 3**: Structure of an artificial neuron with inputs ($x_1, x_2, \ldots x_n$), weights ($w_1, w_2, \ldots w_n$), bias b, activation function f, and the predicted output y_{pred}*

Neural Network Structure

ANNs are constructed of connected neurons in three different types of layers. The first layer is the input layer, which is the initial data for the network to predict. The intermediate layers are hidden layers, where all the computations take place. At last, the results from the hidden layers are sent to the output-layer. The network is called a fully connected network if all the neurons within a layer are connected to all the neurons in the next layer. An example of a fully connected ANN is given in **Figure 4** below.

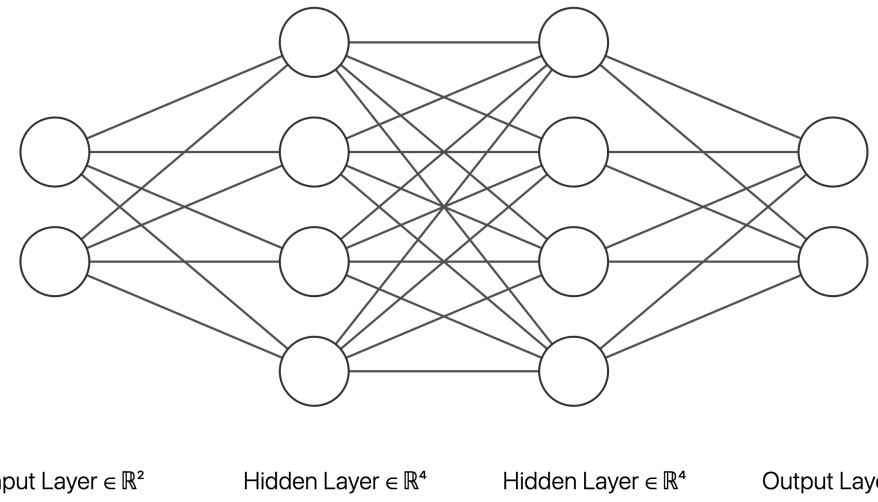


***Figure 4**: Illustration of a fully connected ANN with two hidden layers. The network has two neurons in the input and output layer, while the hidden layers contain four neurons each.*

A forward pass in the context of ANNs refers to the calculation process, i.e., when the input has traversed through all neurons in the network and a prediction is provided from the output layer. The network is learning by utilizing a loss function, which is calculated from the output

values. The loss function is a measurement of the model's ability to predict the expected outcome. The main goal in all machine learning problems is to minimize or maximize this function. The next step is to adjust the weights of the network through a backward pass, which is called optimization. This is done by utilizing an optimization algorithm, where computations are made by traversing from the last layer to the first layer. One forward- and backward pass together makes one iteration. An iteration typically consists of a pass of a mini-batch, which is a subset of the utilized dataset. An epoch is defined as one pass of the entire dataset.

Activation Function

The output of a neuron in a neural network is given by the activation function. At the neuron the weighted sum of all the inputs is calculated, and the activation function takes this result added with the bias of the neuron as input. Three common activation functions are $sigmoid$, ELU and $ReLU$, characterized by:

$$sigmoid(z) = \frac{1}{1 + e^{-z}} \tag{2.2}$$

$$ELU(z) = \begin{cases} z, & z > 0 \\ \alpha \cdot (e^{z} - 1), & z \leq 0 \end{cases} \tag{2.3}$$

$$ReLU(z) = \begin{cases} z, & z > 0 \\ 0, & z \leq 0 \end{cases} \tag{2.4}$$

Sigmoid takes a real value z as input and transforms it to a value between 1 and 0. The fixed output range makes it suitable for probability distribution in binary classification problems. A disadvantage of sigmoid is that the gradient becomes very small towards both ends of the function. This causes a very small response in output y to fluctuations in input z. This is known as the problem of vanishing gradients, which is further explained in subsection 2.3.2.

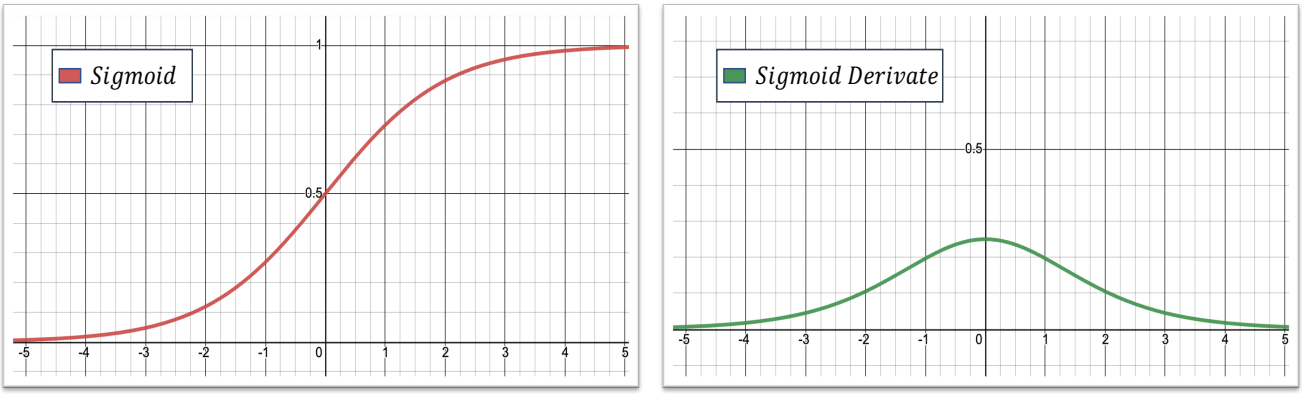


***Figure 5**: The Sigmoid activation function (left) and its derivative (right)*

ELU (Exponential Linear Unit) includes an extra non-negative constant α, which makes it different from other activation functions. Unlike sigmoid, ELU is non-saturating, which solves the problem of vanishing gradients. Previous research has revealed that it tends to converge cost to zero faster and produces more accurate results [18]. For positive input values, the function returns the same value. The function becomes smooth gradually until its output equals $-\alpha$.

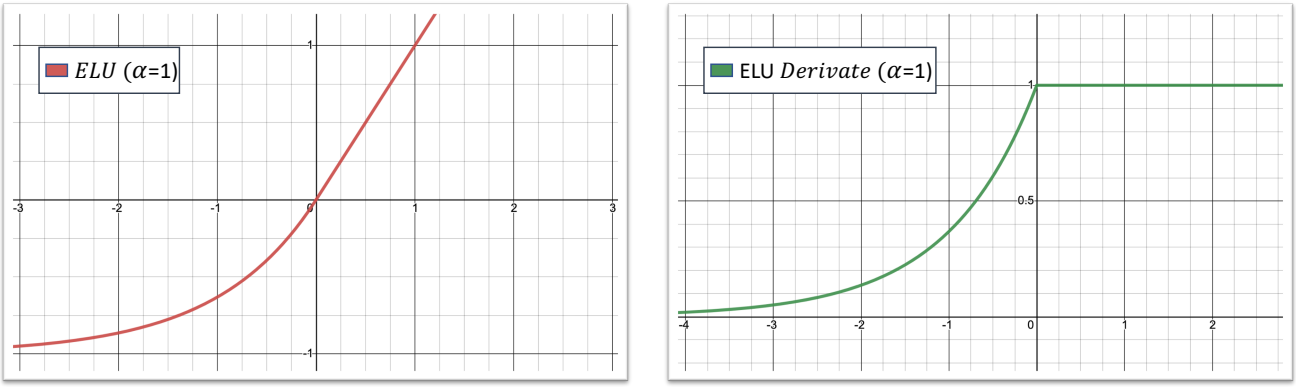


***Figure 6**: The ELU activation function (left) and its derivative (right)*

ReLU (Rectified Linear Unit) is identical to ELU for non-negative inputs. All negative inputs are eliminated by setting them to zero, and the function therefore smoothens sharply. ReLU also solves the problem of vanishing gradients [19]. Because ReLU can produce dead neurons (i.e., neurons that always output zero), it should only be used within the hidden layers of a Neural Network.

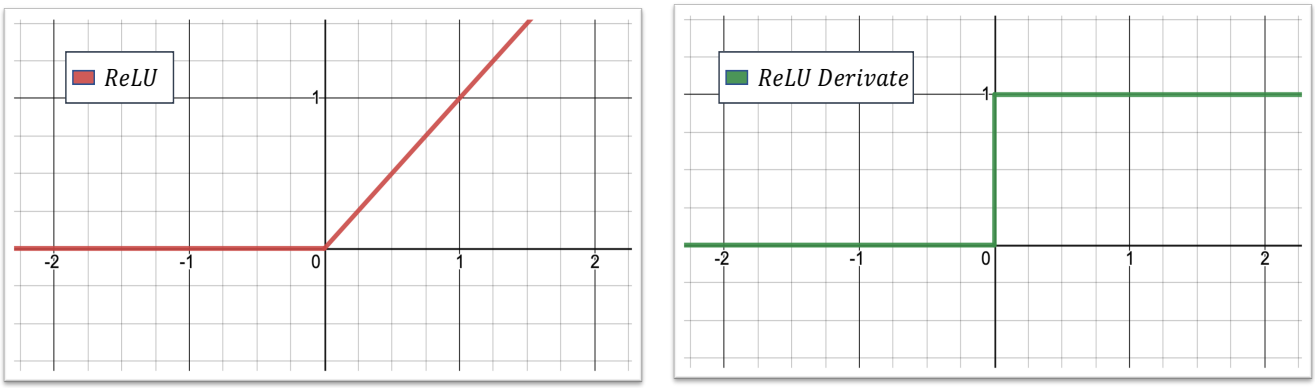


***Figure 7**: The ReLU activation function (left) and it derivative (right)*

2.3.2 Optimization

ANNs learn by utilizing an optimizer to update the weights and biases for the neurons through a backward pass. Depending on the loss function, the optimization process involves either finding the global maximum or global minimum. In the context of neural networks, the loss function is a non-convex function. In a non-convex optimization problem, there exists multiple local optima and only one global optimum. The aim is to minimize the prediction error of the dataset by finding this global optimum on the surface of the loss function. An

optimizer uses the delta rule to update the weights and biases in the neurons. The delta rule is characterized by

$$\theta = \theta + \Delta\theta \tag{2.5}$$

where θ represents the weights and the biases and $\Delta\theta$ is varying depending on the optimizer utilized. Two popular optimization techniques are SGD and Adam, which are further explained below.

Stochastic Gradient Descent (SGD) is an iterative technique for optimizing the objective function, where $\Delta\theta$ is given by,

$$\Delta\theta = -\eta g \tag{2.6}$$

In this equation η represents the learning rate, which decides how much the network should learn from the error of each batch that is processed. In machine learning problems, the training data is typically divided into batches. The mean gradient of the loss function g is calculated for each batch of size m, and is given by

$$g = \frac{1}{m}\nabla_\theta \sum_{i=0}^{m} L(z_i) \tag{2.7}$$

When all the samples (i.e., individual instances of the whole dataset) in a batch have traversed through the network in a forward pass, the next step is to calculate the gradients of the network. This is done through backpropagation, which is a backward pass where the gradient of the loss function is found for all weights and biases by utilizing the chain rule.

One of the challenges with SGD optimization is to find the best parameters that prevent the algorithm from getting stuck at a local optimum. Furthermore, the learning rate is a parameter which has a great impact on the convergence of the network. Setting this too high may result in the algorithm missing the global optima [16]. A momentum can be employed to accelerate the learning process of SGD optimization. This is done by including the gradient of the previous iteration

$$\Delta\theta_t = \alpha\Delta\theta_{t-1} - \eta g_t \tag{2.8}$$

In this equation $\Delta\theta_{t-1}$ represents the gradient from the previous iteration, $\Delta\theta_t$ is the new gradient, and α is the momentum constant determining how much the previous gradient should count in the current update.

Adam is an adaptive optimization technique designed to specifically suit the needs of deep neural networks [17]. The method is adaptive because it computes individual learning rates for the different parameters in the gradient update. The learning rate is adapted for each weight in the network by making use of estimations of the first moment and second moments of the gradient. These two momentums are initialized to zero and updated by

$$m_t = \eta_1 m_{t-1} + (1 - \eta_1) g_t \tag{2.9}$$

$$v_t = \eta_2 v_{t-1} + (1 - v_2) g_t^2 \tag{2.10}$$

In these equations m_t and v_t represent the first and second momentum for the current iteration, m_{t-1} and v_{t-1} represent the first and second momentums for the previous iteration, and η_1 and η_2 are the learning rate parameters. The gradient is the same as for SGD, given by equation (2.7). The next step is to make estimations of the two momentums, which is given by

$$\widehat{m_t} = \frac{m_t}{1 - {\eta_1}^t} \tag{2.11}$$

$$\widehat{v_t} = \frac{v_t}{1 - {\eta_1}^t} \tag{2.12}$$

where t represents the current epoch. Finally, the momentums are used to update the weight by scaling the learning rate individually for each parameter

$$\Delta\theta_t = \alpha \Delta\theta_{t-1} - \eta \frac{m_t}{\sqrt{\widehat{v_t} + \epsilon}} \tag{2.13}$$

The small constant ϵ is used to prevent division by zero. Adam has shown promising results compared to SGD in terms of speed of training and sensitivity to other chosen hyperparameters [17].

Problems with optimization

A common problem when developing deep learning models is the problem of vanishing gradients. This happens as the derivative of the of the gradient approximates to zero, and as a result the weights of the network are not being updated [15]. This problem is solved by avoiding the sigmoid activation function in the hidden layers. Both ReLU and ELU can be used instead, as the derivative of these function do suffer from the problem of vanishing gradients.

During training of deep learning models, the goal of the optimizer is to find the global minimum of the loss function. Sometimes however, it gets stuck at a local minimum. This can be solved by using an adaptive optimizer such as Adam, which is described above.

2.3.3 Data Processing

In this subsection, common data processing techniques in the context of deep learning are described. This includes preprocessing, data augmentation, full-image and patch-wise analysis, and batch management.

Preprocessing

Preprocessing describes each of the transformations applied to the raw data prior to the actual training of the model. This is a crucial step when developing machine learning and deep learning models, as it can accelerate the training process. Some common preprocessing techniques on medical imaging data are described in this subsubsection, which includes pixel intensity normalization, resampling and clipping.

Pixel Intensity Normalization

There are many factors that may influence the signal intensity ranges of biomedical imaging data. The use of different image formats, diverse instruments/hardware, and biological variation are some of the elements that make the signal intensity range highly heterogeneous across datasets [20]. This inconsistency can radically influence the performance of segmentation algorithms [21]. Moreover, machine learning methods used for image segmentation normally perform much better on detecting patterns for features that follow a normal distribution. In order to homogenize the dataset, the images must be scaled. A commonly utilized method is normalization, which is done by rescaling the data to a predefined range, usually between [0,1] or [0, 255]. The normalization of a pixel x in an image array to a range of [0,1] is given by,

$$x_{normalized} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.14}$$

Z-score standardization is another popular rescaling technique, which aims to rescale the data to a range which gives a mean of 0 and a standard deviation of 1. The standardized value for each pixel is given by

$$z = \frac{x - \mu}{\sigma} \tag{2.15}$$

where μ represents the mean value and σ is the standard deviation of the pixel values in the dataset.

Resampling

Resampling is a technique used to alter the width, height and depth of an image which results in a new image having a modified quantity of voxels. A voxel in a 3D image represents a compressed part of the raw image provided by the CCTA-scanner. The size of this compression ratio is defined as the voxel spacing. Because the raw images provided by the scanners are resized into a shape of (512, 512, z), the images might be somewhat stretched or compressed in the z-axis when they are compared with each other. This is the case because the original raw images have different sizes. Neural Networks have big problems with

recognizing patterns from images with a varying voxel spacing. Thus, a normalization of all samples to a common voxel spacing is required. This is accomplished by calculating a distinctive new shape for each image, which is done by first calculating the spacing ratio

$$ratio = \frac{current\ spacing}{new\ spacing} \tag{2.16}$$

In order to reshape the image, the ratio is applied to the current shape of the image through interpolation. The required GPU memory for both training and prediction can be reduced by downsampling the images to a smaller size.

Pixel Intensity Clipping

Clipping is another popular preprocessing technique used in medical image analysis. In computer tomography it is expected that pixel intensities for the same organs lie within a specific range, even when the images are derived from different scanners. Comparable to pixel intensity normalization, clipping is accomplished by clipping the intensities into a particular range. The pixel intensities outside this range are set to the maximum or minimum value of the specified range.

Data Augmentation

In machine learning, data augmentation is a technique to increase the amount of training data by adding modified copies of the already existing data. In medical imaging, this is normally applied when only a small number of samples are available. The images can be altered with numerous methods, and the intent is to produce variants of the pattern that is desired, which will help prevent overfitting [22]. Techniques that have been utilized by the winners of the most recent medical image processing challenges include spatial translation, brightness, contrast, elastic deformation, scaling, rotation and gamma & noise (Gaussian noise) [23][24] [25].

Patch-wise and Full Image Analysis

Based on the image resolution of medical images, accessible GPU-hardware plays an important part in 3D image segmentation. It is not feasible to fully fit CCTA images with a size of 512x512x347 (mean value from the dataset used in this thesis) into the CNN without applying any preprocessing techniques on the data. This is a consequence of very high GPU memory demands required for 3D images of such size. Because of this, the 3D images may be sliced into smaller cuboid patches or examined slice by slice. It is also possible to resample the images in order to fit the entire image in the CNN, i.e., full image analysis.

As to completely use the information of all three axes (as opposed to using 2D slices), a 3D image analysis approach was utilized for the CNN automatic segmentation in this thesis. Both a patch-wise and full image analysis were tested in preliminary experiments. Choice of the input size of the network depends on the requirements of utilized architecture and the available GPU. The U-Net Standard is a common CNN architecture utilized for medical image segmentation, which requires all axes of the input size to be divisible by 2^4.

Additionally, the images (or patches) also have to fit into the GPU. A common choice of input size is therefore e.g., 160x160x80.

Batch Management

When the preprocessing and data augmentation are finished, sets of patches are bundled into batches. A batch comprises several prepared patches that are processed in a single step by the CNN. The batches can then be processed in parallel by the GPU. For every batch that is processed through a single step, the internal weights of the neural network are updated sequentially based on a predefined learning rate. The batch size, i.e., the number of images/patches inside a single batch, is highly dependent on the GPU memory and needs to be configured properly.

In order to allow for continued access throughout the training process and dramatically reduce training time, the batches are saved and stored to disk. The time consumed for calculations is then reduced because of the avoidance of unnecessary recurrent batch processing. For extremely large datasets, this strategy is not ideal as it requires high demands for disk space. However, the dataset utilized in this thesis is relatively small, which makes this approach suitable.

Furthermore, the storage of prepared batches on disk also facilitates a parallelizable processing approach. To reduce the variance of the neural network during fitting, the sequence of batches is shuffled at the end of each epoch. If the order of the data within each epoch is the same, the model may use this information to reduce the training loss. This is a type of overfitting, which is solved by shuffling the data. This can either be done by shuffling the whole dataset and create new batches, or only shuffle the processing sequence of batches.

2.3.4 Evaluation

Evaluation is an important step when developing deep learning models. The two main evaluation techniques utilized in this thesis are described in this subsection, which includes Train/Test split and cross-validation.

Train/Test Split

In training and evaluation of machine learning models, the data is often split into subsets for training, testing and validating. For this approach, the data is first broken up into two parts, which typically consist of 90 % for training and 10 % for testing. The training set is then normally divided further into a split consisting of 90 % for training and 10 % for validating. The performance of the model on the validation set during training is used to monitor the generalization error. This is the error produced by the model when predicting unseen data. Validating the performance during training makes it much easier to detect occurrence of overfitting, which is further explained in subsection . The testing data from the first split is used to measure the final performance of the trained model. If the validation set is used to select the final model, it is not suited to evaluate the final performance of the model. This is the case because this data is no longer unseen by the model and will lead to an error rate estimation that has potential bias, i.e., an error rate that is smaller than the true error.

Cross-Validation

Cross-validation is a widely utilized statistical model for estimating the performance of deep learning and machine learning models. It provides the opportunity to compare trained models, and then pick the model with best performance. It is well suited for evaluation of machine learning models on a limited dataset, and it normally provides a lower bias than other methods.

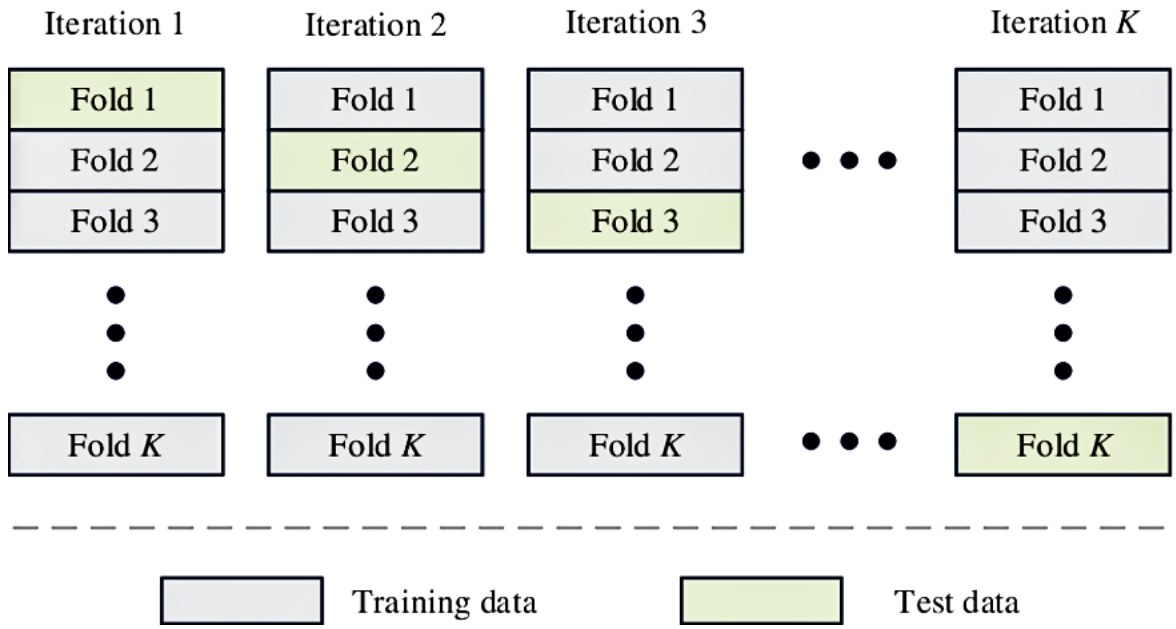


***Figure 8**: K-fold cross-validation method [26]*

In this thesis a K-fold cross-validation has been employed, which is illustrated in **Figure 8**. In this method the data is divided into k folds, and then the model is trained k-times with a different fold being the validation/test each iteration. This process is continuing until all folds have been used as validation/test set exactly once. There does not exist any formal rule for choosing the number of groups to split the data in, but usually k is set to 5 or 10. For larger values of k, the size difference between the training data and resampling subsets (i.e., number of folds) is decreasing. As this size difference gets smaller, the bias of the model is reduced along with it. The main problem with choosing a larger value for k is the computation requirements, as a k-fold cross-validation includes training k independent models [27].

2.3.5 Regularization

Overfitting is a common issue when training deep learning models. The model is said to be overfitted if the model is learning the characteristic patterns for the training set, rather than generalizing to unseen data. When modifications are made to the model with an intent to reduce the generalization error, and not the training error, it is known as regularization [16].

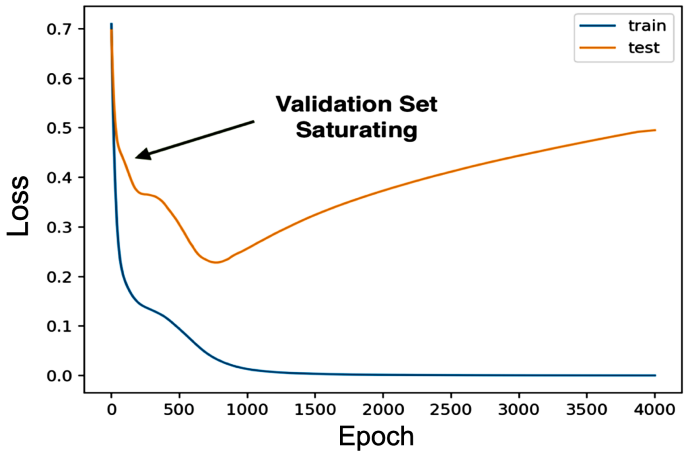


***Figure 9**: Overfitting – the training continuous while the model accuracy on the validation set decreases (i.e., the validation loss increases)*

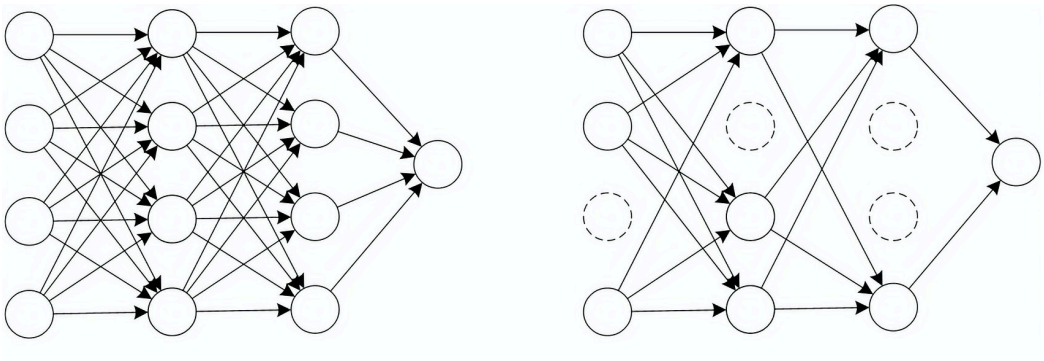
Early Stopping

Early stopping is used to prevent overfitting (see **Figure 9**) by monitoring the generalization error during training. The training is terminated if there is no decrease in the validation error for a x number of epochs. This parameter is called patience and has to be specified before starting the training [16].

Dropout

Another regularization technique that addresses the issue of overfitting is dropout, which is illustrated in **Figure 10**. Dropout regularization is employed by adding dropout layers to the model, which is initialized with a dropout probability. The neurons in these layers are randomly deactivated with a probability of p, while the weight of the active neurons is scaled down by multiplying it with p-1 [16].

When a node is dismissed, it is temporarily removed from the network, along with its ingoing and outgoing edges. As a result, the view of the configured layer is different for every update of the network. As a consequence of dropout regularization, the training procedure becomes noisy, and some nodes are forced to carry more or less responsibility for the inputs. As dropout layers mimic a sparse activation of the layer, it also encourages the network to learn sparse representation of the data [28].



(a) Standard Neural Network (b) Network after Dropout

***Figure 10**: Visualization of an ANN with three dropout layers. The network to the left represents normal state of the network, while the network to the right represents the state of the network after dropout is applied. The neurons without edges represent the deactivated neurons.*

Batch Normalization

Batch normalization is a technique that can accelerate and stabilize the training process of deep neural networks by adding additional layers. These layers perform a normalization of the output data from the previous layer, and output values between 0 and 1 [16]. As the name indicates, the normalization is done over a batch processed by the network.

2.3.6 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks is a subclass of ANNs, which is suited for capturing local information from images. A neural network is classified as a CNN if it has one or more convolutional layers. The mathematical formula for a 2D convolution is given by,

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.17}$$

where I represents the input matrix of size $m\ x\ n$, K represents the filter matrix of size $i\ x\ j$, and S is the feature map containing the detected features. The size of the filter is crucial for the properties of the features that are discovered. A larger filter has the ability to detect more general features, whereas a smaller filter is more suited to detect subtle features. These features can typically be corners or a variation in the pixel intensity strength. The filter matrix at a convolutional layer represents the weights assigned to the pixel values, which is updated for a batch during training.

One of the main benefits of CNNs is that it has the property of being invariant to affine transformation. This means that the neural network is able to recognize objects even if the location of the object changes. This characteristic is due to shared weights, spatial subsampling, and local receptive fields [29]. The neurons within a layer in a CNN receive its inputs from corresponding neurons in a related section from the previous layer, whereas the neurons within a layer in an ordinary fully connected neural network are connected to all the neurons in the previous layer. As a result, the neurons in a CNN get the responsibility for specific regions of the input image, which is called local receptive fields. The weights are shared across the local receptive fields of the neurons in a specific layer. The sharing of weights happens as the filter moves through the image. As a result, when the CNN detects an important feature to learn in a specific region of the image, it treats the feature as equally important to learn in other parts of the image. Shared weights also help reduce the computational cost when training the network [16].

The feature map that is produced by the convolutional layer will often contain subtle details which might not be useful for solving the task. A common method to solve this problem is to down sample the output from the convolutional layer, which is typically done by adding a pooling layer after a convolutional layer. This is a normal pattern for CNN architectures, and is often repeated several times. The pooling layer creates a lower resolution version of the input signal, with an objective to create new signals that still includes the most significant components, while removing the subtle details. A normal procedure is to reduce the feature

map by a factor of 2. The pooling operation is specified in the creation of the architecture, where two of the most common operations are maximum pooling and average pooling. These methods differ in how they replace a specified location with the summary of the nearby inputs. Maximum pooling uses maximum values, while average pooling uses the average values. The result is a summary of the features detected. In addition to local receptive fields and shared weights, the usage of pooling layers strengthens the characteristics of invariance to affine transformations [16].

Both upsampling and downsampling can be made learnable by introducing strided convolutions. This increases the expressiveness of the model, but also the number of trainable parameters. Previous work has shown that the usage of strided convolutions may improve the overall accuracy of the CNN model [30]. In that case, transposed convolution is used for upsampling. This is done by translating the convolutions into matrix operations between the flattened input I of size $Nx1$ and the filter K of size MxN [31].

Convolutional Autoencoder (CAE)

An autoencoder is a unique kind of neural network which aims to reproduce the input data in the output layer. Because the autoencoder uses the input values as the target value, it is classified as unsupervised/semi-supervised learning. An autoencoder is convolutional if it contains one or more convolutional layers.

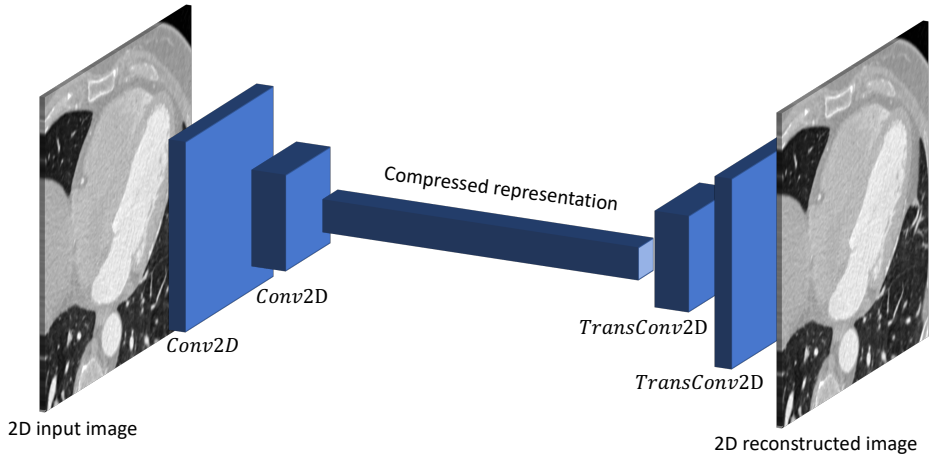


***Figure 11**: Illustration of a CAE using two convolutional layers in the encoder (left of the compressed rep.) and two transposed convolutional layers in the decoder (right of the compressed rep.).*

The CAE reduces the dimensionality of the input data in the hidden layer and tries to reproduce the input from this decomposed representation. The number of neurons in the hidden layer is therefore much smaller than the original size of the input data. A good reproducibility, i.e., a small difference between the original input and the reconstructed data, indicates that the compressed representation in the hidden layer contains essential features of the input values. The structure of a CAE forces the hidden layers to learn crucial patterns and dismiss information that is redundant. This is achieved by designing the CAE architecture using hidden layers with smaller dimensions compared to the input and output layers.

Chapter 3 Methodology

In this chapter, the methods used to accomplish the required steps of the pipeline are described. It consists of three main tasks, including:

(1) Automatic segmentation of LV myocardium from CNN
(2) Myocardial characterization by means of k-means clustering and feature extraction via CAE encodings of 2D patches
(3) Classification of features extracted from the LV myocardium using GPC (Gaussian Process Classifier) and KNN (K-Nearest Neighbors)

This chapter clarifies the details regarding the implementation of the various experiments performed for each of the steps listed above. The technical tools utilized in the experiments are given in section 3.1, and the dataset is described in section 3.2. The overall implementation details are presented in section 3.3. The automatic segmentation is presented in subsection 3.3.1, followed by the myocardial characterization in subsection 3.3.2. At last, the methods used for the final patient classification are presented in subsection 3.3.3.

3.1 Technical Tools

In this section the most important technical tools utilized for the required steps are listed.

GPU Cluster A cluster/collection of computational nodes that can have one or more Graphic Processing Units (GPUs) connected to it. In this thesis NVIDIA Tesla P100 GPUs with 16 GB were utilized in the training of the CNN and CAE. The training of deep learning models is a process that is highly parallelizable, which makes the usage of GPUs speed up the calculations and shorten the overall training time. All computations were performed on resources provided by the NTNU IDUN/EPIC computing cluster [44]

TensorFlow An end-to-end open-source platform for developing machine learning models.

Keras A high-level deep learning library that runs on top of either TensorFlow, CNTK, or Theano. In this thesis TensorFlow is utilized as backend for the various models. Most of the functionality provided by Keras is known as a black box, which means that how the network updates its parameters is not explicitly programmed, and only the hyperparameters and the training data has to be specified by the user.

MIScnn An API framework for instant setup of state-of-the-art deep learning models and CNNs for medical image segmentation with Keras and TensorFlow as backend. The framework offers a complete pipeline for

	preprocessing, data augmentation, patch cutting, and batch creation. MIScnn also gives the user the choice to switch between multiple modern CNN-models, as well as the opportunity to add custom version architectures. Furthermore, multiple different loss-functions and evaluation metrics are available.
sklearn	An open-source machine learning library that provides efficient tools for predictive data analysis. This includes methods such as features selection, data normalization and classification algorithms.

3.2 Dataset

The dataset used in the project consists of a total of 66 CCTA images, described in section 2.2. The images have a shape of (512, 512, z). The z-value is the number of slices in the transverse plane, which is varying between the different patients. The number of slices in the coronal- and sagittal plane are fixed to 512, which constitutes the height and width for an image slice from the transverse plane.

3.2.1 NIfTI Data I/O

The data that is used are in NIfTI-format (Neuroimaging Informatics Technology Initiative), which is supported by MIScnn API. The NIfTI-file stores information about the 3D image matrix and diverse metadata, e.g., the thickness of CT slices.

3.2.2 Data Exploration

From **Table 1** we can observe that the mean and median number of slices in the transverse plane for the entire dataset are 346.9 and 329.5, respectively. There is some variation in the voxel spacing for different images, and the mean/median value is approximately (0.42, 0.42, 0.35).Manually performed semantic segmentations of the LV myocardium were available for 28 of the CCTA images as an associated binary matrix. The data exploration for these images revealed almost identical results as for the entire dataset, which is given in **Table 1**. The aim of a sematic is to label each pixel in an image to a corresponding class, which in our case is background or myocardium [32]. The segmentation matrices have the same shape as the belonging image, where corresponding voxels are a part of myocardium if it has the value 1 in the segmentation matrix, and 0 if not. All the 28 images with an associated manual segmentation were used in the training and prediction of the CNN and CAE. All of the 66 patients in the dataset were utilized in the classification step of the pipeline, i.e., clustering, feature extraction and classification of automatic segmented images.

An unbalanced dataset is a common issue in medical image segmentation, where the semantic annotations include a strong bias in the class distribution towards the background class. This is also the case for our dataset, and a class distribution of 97.1% background and 2.89% myocardium was revealed in the data exploration.

Table 1: *Data Exploration for the entire dataset of 66 images and the 28 images with manual segmentation*

	Entire dataset	With manual seg.
X-Axes Mean	512.0	512.0
X-Axes Median	512.0	512.0
Y-Axes Mean	512.0	512.0
Y-Axes Median	512.0	512.0
Z-Axes Mean	346.5	347.9
Z-Axes Median	329.5	344.0
Voxel spacing	(0.42, 0.42, 0.35)	(0.38, 0.38, 0.3)
Background	0.971	0.971
Myocardium	0.029	0.029

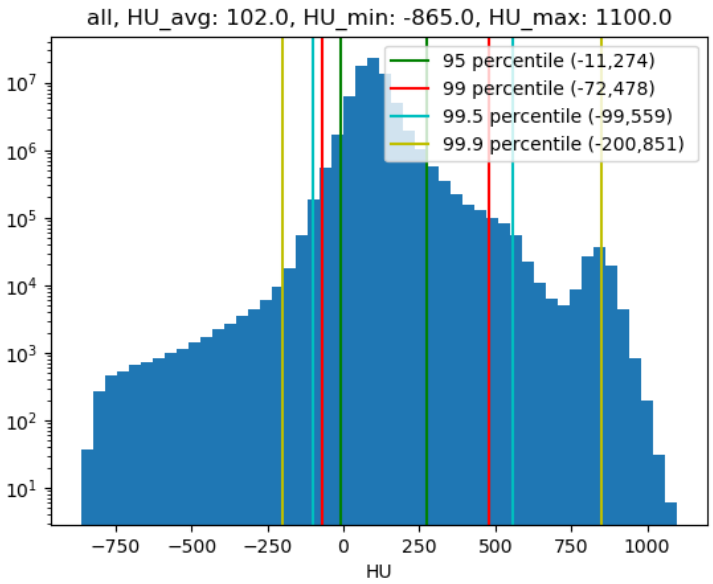


Figure 12*: Histogram of the average range of HU of myocardium voxels*

The histogram in **Figure 12** was obtained by analyzing the Hounsfield Units of pixels labeled as myocardium in the manual segmentations. From this one can observe the following results

⇒ 95 % of the pixels lies in the range of [-11, 274]
⇒ 99 % of the pixels lies in the range of [-72, 478]
⇒ 99.5 % of the pixels lies in the range of [-99, 559]
⇒ 99.9 % of the pixels lies in the range of [-200, 851]

When a CCTA-scan is performed, the patient is injected with a contrast medium in the blood flow. This gives a higher intensity for the blood flow, which usually lies in a range from 300 up to 800 depending on the scanner. The left LV myocardium is muscle mass, which means that it does not get any direct contrast from the medium. It is therefore reasonable to assume

that pixels with a higher intensity than around 300 are incorrectly labeled as myocardium. This is most likely the case on the borders between the blood and myocardium, such as in the chamber of the left ventricle.

3.3 Implementation

In this section, the implementation details for the methods used to solve the required steps are presented. This includes, among other things, the preprocessing approaches and training strategies for the deep learning and machine learning models. A method for automatic segmentation of the LV myocardium via 3D CNN using 28 CCTA images is first presented. Subsequently, the best of the trained models is used to obtain automatic segmentations of the LV myocardium for the entire dataset, i.e., 66 CCTA images. Each segmentation is split into regions via k-means clustering, and a trained autoencoder is subsequently used to extract features from the clusters. Finally, the patients are classified through comparison of two different classification algorithms, including GPC (Gaussian Process Classifier) and KNN (K-Nearest Neighbors). An illustration of the total pipeline implemented in this thesis is given in **Figure 13** below.

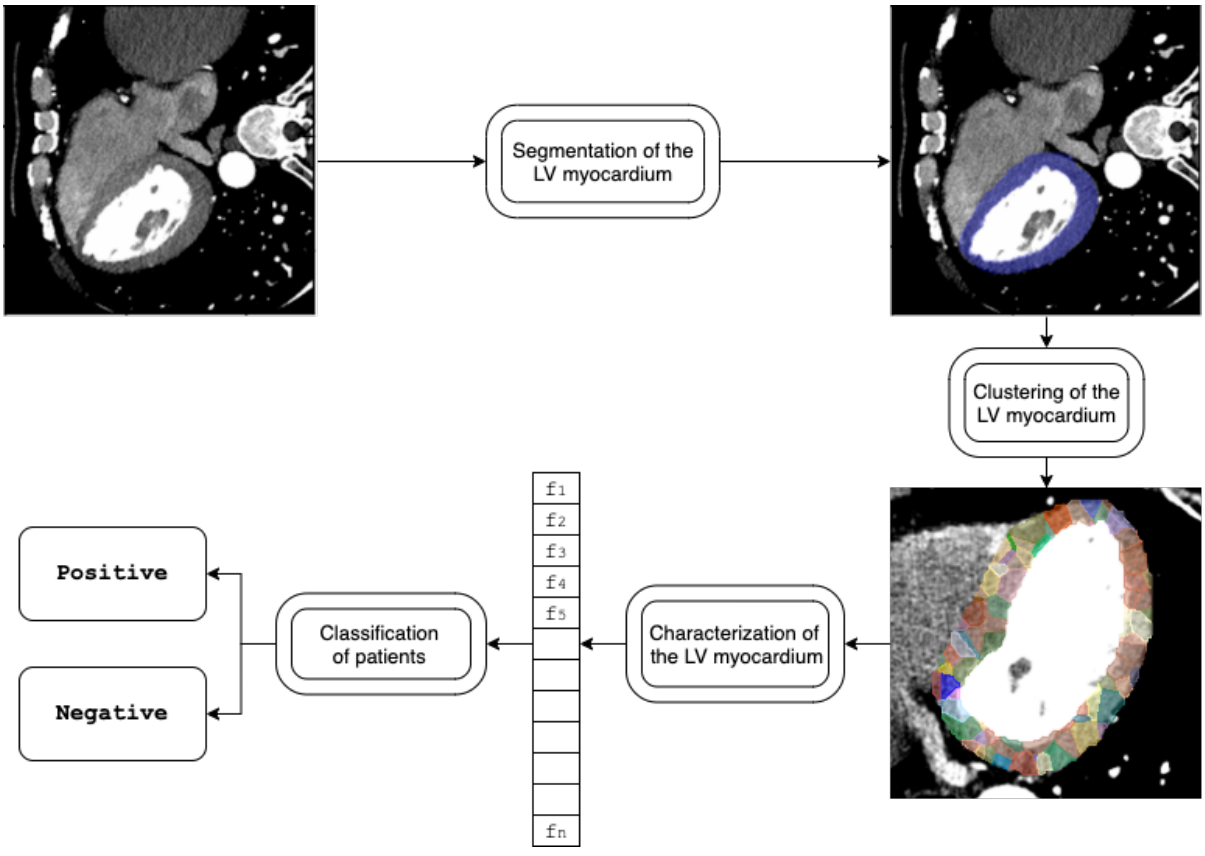


***Figure 13**: Overview of the proposed pipeline. The LV myocardium is first segmented using a 3D CNN and subsequently clustered via k-means. Encodings are extracted from the clustered LV myocardium using a CAE to compute the features $[f_1, f_2, f_3, \ldots, f_n]$. At last these features are used to classify the patients with functionally significant stenosis (positive) and those without (negative).*

3.3.1 Automatic Segmentation

The automatic segmentation was done utilizing a CNN through multiple 3-fold cross-validation experiments. Extensive data augmentation was employed to increase the amount of training data, which includes all the techniques described in subsection 2.3.3. Spatial augmentation was achieved by elastic deformations, rotations, mirroring and scaling. For color augmentations, brightness, contrast and gamma variation were utilized. At last, noise augmentation was obtained by adding Gaussian Noise. An overview of the pipeline for the automatic segmentation is presented in in **Figure 14**.

The images were saved and stored as batches in order to efficiently parallelize the training process by utilizing one or more GPUs. In this section the technical properties of the utilized CNN are discussed, including the architecture, preprocessing, training, loss function, evaluation metrics, post processing and architecture.

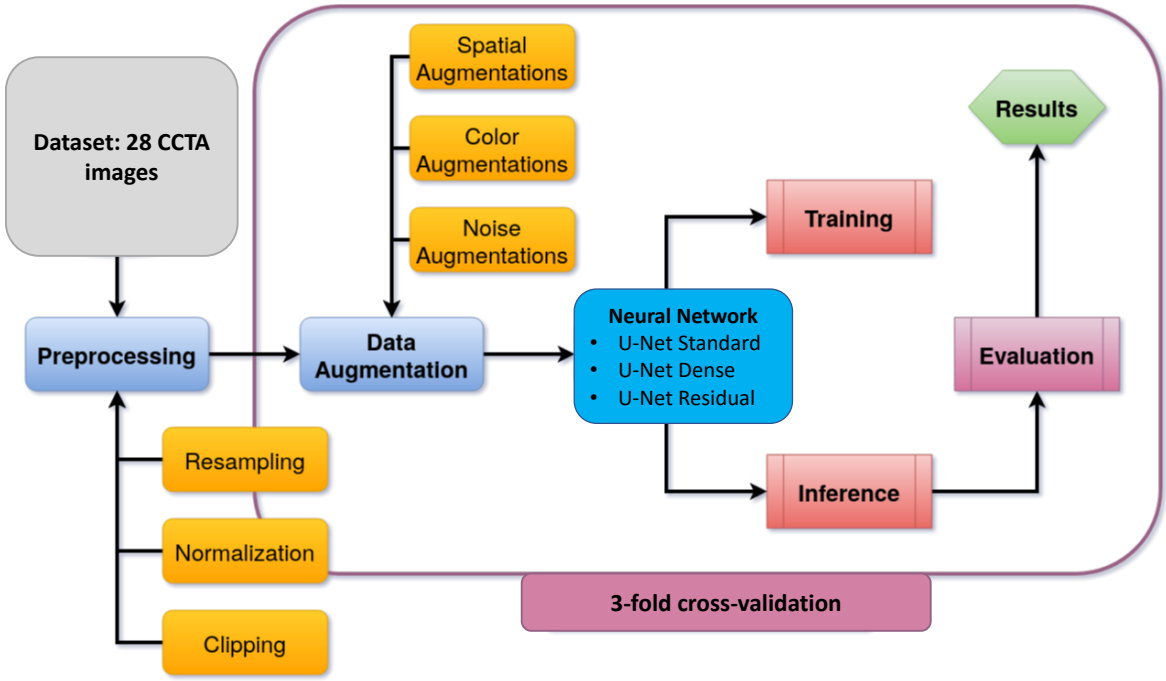


***Figure 14**: Overview of the pipeline utilized for automatic segmentation of the LV myocardium. The workflow starts with the datasets and describes the order of each step, ending with an evaluation for each fold in the cross-validation. Figure adapted from [8].*

Architecture

The selection of CNN architecture and its hyperparameters is a critical step in the pipeline of medical image segmentation. There are many different architectures to choose from and each has different advantages and disadvantages. Three different architectures were tested, which include 3D-UNet Standard, 3D-Unet Dense and 3D-UNet Residual. A view of the U-Net Standard is shown in **Figure 15**.

For all architectures upsampling was attained by using transposed convolution, whereas downsampling was attained using maximum pooling. At its highest resolution, the architectures used 32x32-feature maps, while 512x512 feature maps (i.e., the whole image in

the coronal/sagittal plane) were used at its lowest resolution. The U-Net architecture consists of two paths; the analysis path (left part of the network model illustration) and the synthesist path (right part of the network model illustration). In the analysis path, the convolutions for the upsampling and downsampling were applied with a kernel size of 2x2x2 in a stride of 2x2x2. For the rest of the layers, convolutions were applied with a kernel size of 3x3x3 in a stride of 1x1x1.

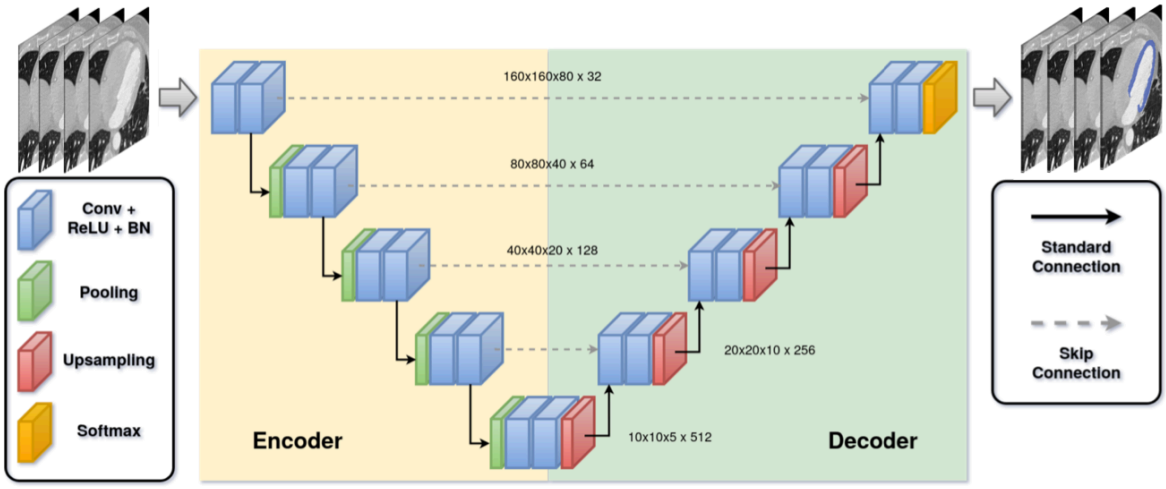


***Figure 15**: Architecture of standard 3D U-Net. The network input is 3D patches (cuboids), and the output is the segmentation of myocardium. Conv: Convolutional layer; ReLU: Rectified linear unit; BN: Batch normalization. Figure adapted from*

3D-UNet Standard

Compared to more complex architectures, e.g., U-Net Residual and the U-Net Dense, the U-Net Standard has fewer parameters. Therefore, by utilizing a U-Net Standard, we will prevent a significant increase in parameters that comes with utilizing a more complex architecture [9][33][34]. This makes the training faster and requires less GPU RAM. Another view of the 3D U-Net Standard is shown in **Figure 16**.

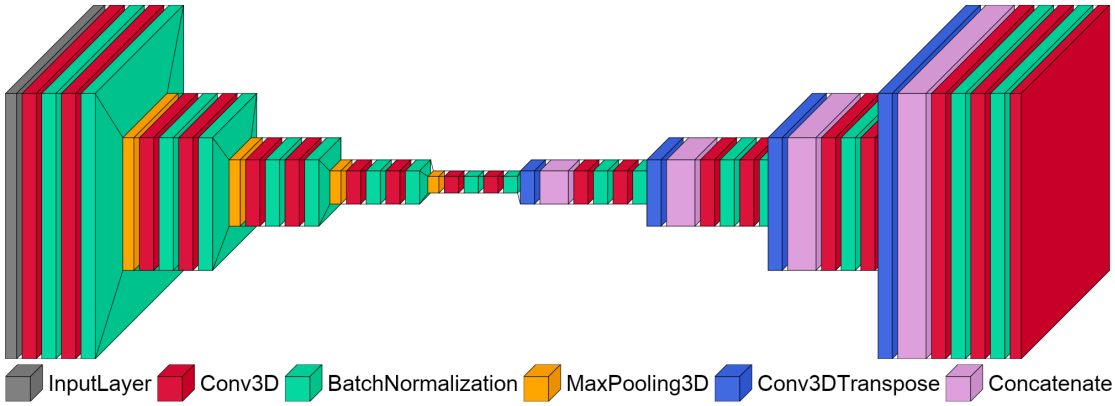


***Figure 16:** Another view of the 3D-UNet Standard Architecture*

3D-UNet Residual

Previous work has shown that it may be possible to improve the segmentation results by utilizing this residual variant of the U-Net standard [9][35]. It uses an additional add layer

after each convolutional block (2x convolutional layers). The 3D-Unet Residual has a significant increase in trainable parameters compared to the 3D U-Net Standard, and therefore requires more GPU RAM and increases the total training time. The view of the 3D U-Net Residual is shown in **Figure 17**.

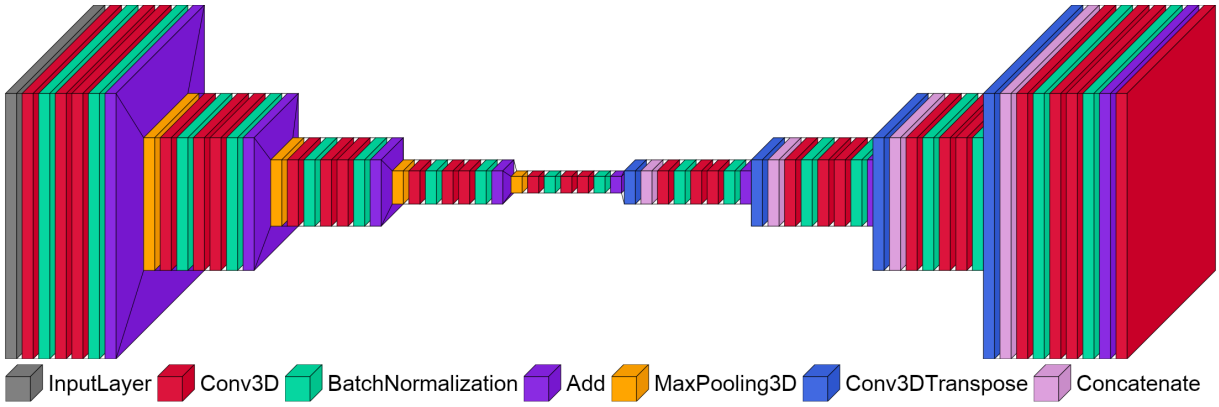


Figure 17: *View of the 3D U-Net Residual architecture*

3D-UNet Dense

The dense version of the U-Net architecture has also shown improved results compared to U-Net Standard [10]. The difference is that it uses multiple concatenate layers in each convolutional block of two convolutional layers. The concatenate layers are placed after each convolutional layer in each block of convolutions. The extra concatenate layers give a significant increase in trainable parameters, and therefore requires more GPU RAM and increases the training time. The view of the 3D U-Net Dense is shown in **Figure 18**.

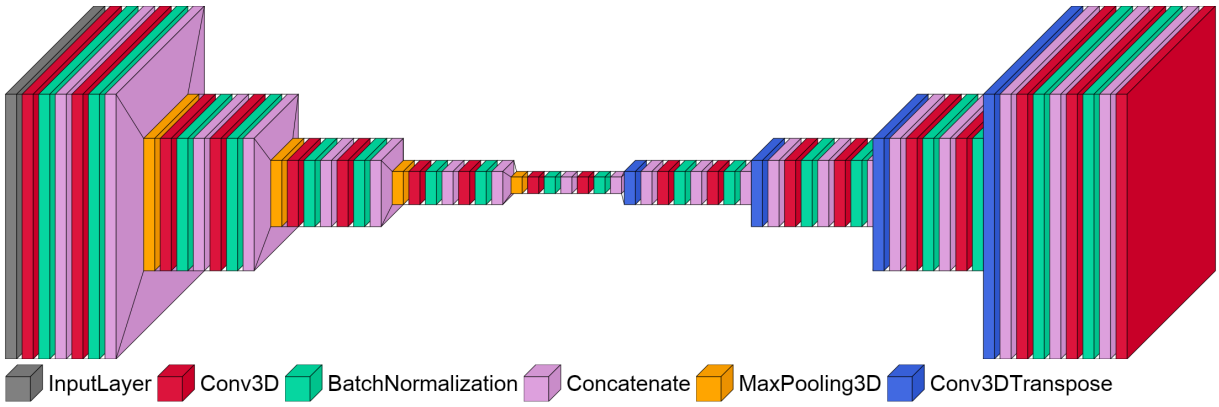


Figure 18*: View of the 3D U-Net Dense architecture*

Preprocessing

We analyzed the effect of various preprocessing techniques on the performance of automatic segmentation on the left ventricle myocardium. Intensity clipping was decided based on percentiles of intensities of myocardium voxels from manual segmentations. As mentioned in subsection 3.2.2, it may be reasonable to assume that pixels with a higher intensity than around 300 most likely are labeled wrong. Based on this assumption, a clipping range between the 95 % case to the 99 % case from **Figure 12** was used in the various experiments.

Furthermore, pixel intensity scaling and resampling were applied to the imaging data in all experiments. These methods are explained in detail in subsection 2.3.3. The data was scaled by means of z-score standardization, as this outperformed scaling the data in the range [0,1], independent of the other parameters and methods utilized.

The common voxel spacing used for resampling was found through initial exploration. Further, both a patch-wise approach and a full-image approach was tested in preliminary experiments. Data exploration revealed a relatively low mean voxel spacing of (0.42, 0.42, 0.35). As a consequence, it was possible to fit whole images in the CNN, which also proved to outperform a patch-wise analysis approach in preliminary experiments. As a result, the full-image approach was used for all the CNN experiments presented in this thesis.

A suitable voxel-spacing of (1.24, 1.24, 1.42) was found via experimenting, which gave a median image size close to the desired input size of 160x160x80. The desired input size was chosen with respect to the GPU limitations and CNN architectures which all require the input size to be divisible by 2^4. Finally, all the images were resized by interpolation to the desired input size of 160x160x80.

Training

K-fold cross-validation was used to evaluate the performance of the CNN models, where a k-value of 3 was used for all the experiments. The number of epochs was set to 200 for all the models. The number of iterations per epoch was set to 150, and the batch size for each iteration was set to 1. One epoch is then defined as an iteration over 150 batches. This allowed for an improved fitting process for randomly generated batches in which the dataset acts like a variation database [8]. One GPU was utilized for training each model.

An adaptive learning rate optimization was utilized during model fitting by applying Adam optimization with initial weight decay of 1e-3. The dynamic learning rate was reduced by a factor of 0.1 if no improvement in the validation loss was achieved over 15 subsequent epochs. To prevent the learning rate from becoming too small, a lower limit of 1e-5 was set. To further reduce the risk of overfitting, an early stopping technique with a patience of 30 epochs monitored on the validation loss was employed. Because of the limited size of the dataset with only 28 CCTA images, the validation data was also used to evaluate the final performance of the model. This could lead to overfitting on the test set, as this also was used to monitor the early stopping condition. However, this should not be a big issue as it does not affect the final classification results directly.

Evaluation metrics

The predicted outcome of the trained model was evaluated using multiple evaluation metrics. The output of the model is binary, where the predicted value for a voxel can either be true positive (TP), false positive (FP), true negative (TN), or false negative (FN). These terms are explained in **Figure 19**. As our segmentation problem is binary, it consists of labeling the voxels as either background class or myocardium class.

True Negative (TN) Predicts that the voxel does not belong to the given class, when it does not belong to the given class	**False Positive (FP)** Predicts that the voxel belongs to the given class, when it does not belong to the given class
False Negative (FN) Predicts that the voxel does not belong to the given class, when it does belong to the given class	**True Positive (TP)** Predicts that the voxel does belong to the given class, when it does belong to the given class

***Figure 19**: Confusion matrix for evaluation*

Data exploration revealed a class distribution of 97% for background and 3% for LV myocardium. To address the issue of class imbalance when evaluating the performance of the trained model, the Dice Similarity Coefficient (DSC), Sensitivity, and Specificity were utilized as evaluation metrics. These are based on the confusion matrix from **Figure 19**, and measure the segmentation overlap between prediction and ground truth.

DSC The DSC is a metric that takes the issue of class imbalance into consideration by measuring the relative overlap between prediction and ground truth. This is done by calculating the intersection over union, which makes it more suited for imbalanced data compared to metrics that evaluate on individual pixels.

$$DSC = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{3.1}$$

Sensitivity The sensitivity measures the proportion of actual positive pixels (i.e., pixels that are classified as a particular class) that were predicted as positive.

$$Sensitivity = \frac{TP}{TP + FN} \tag{3.2}$$

Specificity The specificity is similar to the sensitivity, but instead of measuring the proportion of positive samples, it measures the proportion of actual negative pixels that got predicted as negative.

$$Specificity = \frac{TN}{TN + FP} \tag{3.3}$$

Loss Function

In order to solve the problem of class imbalance during training, the Tversky Loss (TL) and the Dice Similarity Loss (DSL) was utilized as loss functions [36].

DSL The Dice loss is a popular choice of loss function for training deep learning model for binary segmentation of medical images. It is given by

$$L_{DSL} = 1 - DSC(T, P) \tag{3.4}$$

where $DSC(T, P)$ is given in equation (3.1). Compared to other loss functions, like the cross-entropy, the Dice loss uses the predicted probabilities directly rather than thresholding and transforming them into a binary mask.

TL The Tversky loss is a multi-class adaption for the Tversky index, which is an asymmetric indicator used to quantify the overlap of the segmented area with the ground truth. It is a generalization of the Dice coefficient and Jaccard index, and is given by

$$L_{Tversky} = N - \sum_{c=1}^{N} \frac{TP_c}{TP_c + \alpha \cdot FN_c + \beta \cdot FP_c} \tag{3.5}$$

where TP_c, FN_c, and FP_c represent the true positive, false negative and false positive rate, respectively. The sum of parameters α and β is 1. By setting $\alpha > \beta$, the false negatives are panelized more.

Postprocessing

As the myocardium is a compact structure, it should be possible to improve the results of the obtained automatic segmentations by removing components that are lying outside the geometric boundaries of myocardium. Such types of components are illustrated in **Figure 20**. This is a crucial step in the total pipeline presented in this thesis, as the CNN model is used to obtain automatic segmentations for images that should be clustered in the next step of the pipeline. It is reasonable to assume that segmentation containing components that do not belong to myocardium will result in inaccurate clustering results across the different segmentations.

The postprocessing step consists of removing small components from the obtained automatic segmentations that are not attached to myocardium. This is done by removing all except the biggest of the connected components in the binary segmentation matrix. In the process it is assumed that the myocardium is the biggest component, which was verified by inspecting all the obtained automatic segmentation visually before they are clustered.

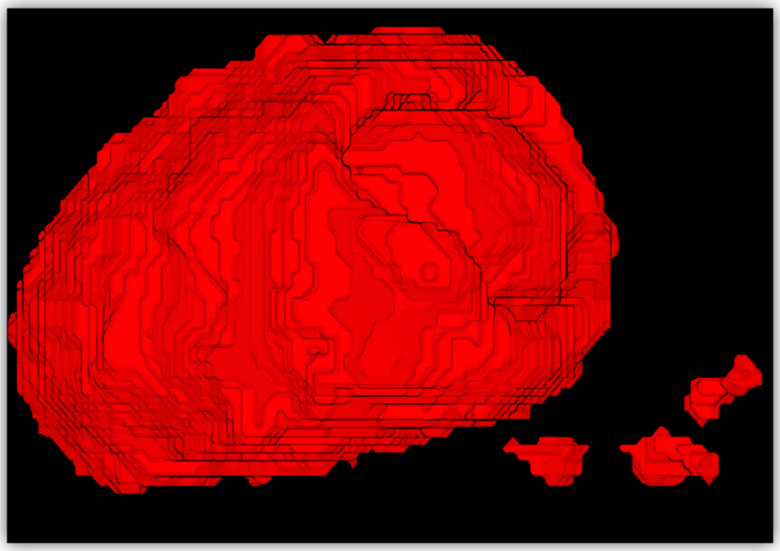


***Figure 20**: Segmentation with small components outside the geometric boundaries of the LV myocardium*

3.3.2 Myocardial Characterization

Once an automatic segmentation is obtained the next step is to extract relevant information from the myocardium. In this section the steps and methods used for the feature extraction part of the pipeline are explained in detail. This includes clustering of LV myocardium, training and validation of CAE, and implementation details on how the features were extracted from the clusters utilizing the trained CAE.

Clustering

The trained CNN was utilized to obtain automatic segmentations of all the 66 available CCTA images. In order to detect presence of inhomogeneity in the LV myocardium tissue, the segmentations were first divided into connected clusters using a vectorized implementation of k-means.

K-means is an unsupervised learning algorithm for dividing data into clusters. It follows a number of simple steps which aims to classify the data into a specified k number of clusters. The first step of the algorithm is to initialize k centers in 3D space, i.e., one for each cluster. In the next step, all the datapoints are assigned to the nearest center based on the spatial location. When this is completed, k new centroids are calculated from the new datapoints that are assigned to the clusters. A new binding is then performed, and the datapoints are assigned to clusters based on the newly calculated centroids. This is the loop of the k-means algorithm, which is repeated until the location of the centroids stays the same from one iteration to the next. The main aim of the algorithm is to minimize the sum of squared distances between the datapoints in each cluster. This is done by minimizing the objective function, which is given by,

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n_j} \left\| x_i^{(j)} - c_j \right\|^2 \tag{3.6}$$

where k is the number of clusters, n_j is the number of datapoints in the j^{th} cluster, $x_i^{(j)}$ is the i^{th} datapoint of cluster j, and c_j is the center of cluster j. The difference that is summed over is known as the Euclidean distance, which in our case is a span in 3D space.

In this process, the metadata of the CCTA images stored in NIfTI-files was used to transform voxel-indices to spatial coordinates, i.e., to a cartesian coordinate system. The Euclidean distance was then calculated from these transformed coordinates of myocardium voxels.

CAE

Different preprocessing techniques were utilized prior to the training of the CAE. The best results were obtained using pixel intensity normalization in the range [0,1], which was used for all experiments. Clipping of pixel intensities was performed for all the trained models. Multiple clipping ranges were explored, which includes the range between the 95 % case to the 99 % case from **Figure 12**. The effect of resampling was also explored to detect if it could have any impact on the final classification results.

Training

The CAE was trained and validated using the same data as for the automatic segmentation, which included 28 CCTA images with a corresponding manual annotation of the LV myocardium. Patches used for training were extracted from the LV-myocardium based on two main requirements, which is defined by,

Center Voxel	The center voxel of the patch must be a part of myocardium, i.e., the segmentation mask must cover this voxel.
Min Labeled Voxels	The minimum share of labeled voxels must be higher than a given percentage. If this is set to 0.6, the patch has to consist of a minimum of 60% labeled voxel to be included in the training.

The extracted patches were first divided into a train/test split, where 90 % were randomly selected for training the CAE, and the remaining 10 % was used for testing. The training set was then further divided into training and validation sets, where 90 % was used for training and the remaining 10 % was used for validation.

The total number of patches used for training each CAE model was varying based on the specific patch size, minimum labeled voxels and a specified patch overlap. The number of patches was aimed to be in a range of 200 000 to 400 000 in order to reduce the time spent on training. All the models were trained for 700 epochs, where each epoch consisted of an iteration of a minibatch of size 500. The batches were prepared prior to training, which drastically decreased the total training time of each model. Random shuffling was performed on the processing sequence of batches each epoch, which is further explained in subsection 2.3.3.

An adaptive learning rate optimization was utilized during model fitting by applying Adam optimization with initial weight decay of 1e-3. The dynamic learning rate was reduced by a factor of 0.1 if no improvement in the loss value was achieved over 25 subsequent epochs. To prevent the learning rate from getting too small, a lower limit of 1e-6 was set. To further reduce the risk of overfitting, an early stopping technique with a patience of 50 epochs monitoring the loss on the validation data was applied. The test set was set aside during training and was used to evaluate the final performance of the model.

Loss Function

The mean squared error (MSE) was utilized as loss function for all the CAE experiments. It measures the mean of the squared error between the input patches and the reconstructed patches, and is defined by,

$$MSE = \frac{1}{NM} \sum_{i}^{N} \sum_{j}^{M} (\bar{I}_{ij} - I_{ij})^2 \tag{3.7}$$

where $\bar{I}_{ij}$ and I_{ij} represent the reconstructed patch and the input patch, respectively. The error is calculated for each pixel in the processed image, i.e., the scaled and/or resampled image. The MSE was also used as metric for evaluation of the final performance of the model on the test set.

Architecture

Multiple architectures were tested in each CAE patch reconstruction experiment. This includes varying the number of convolutional layers, different methods for upsampling and downsampling, and varying number of filters for each convolutional layer. All convolutions were performed with a kernel size of 2x2. Furthermore, upsampling and downsampling was performed with a size of 2x2 for all the utilized architectures. The number of dense layers was consistent for all the architectures, which was set to two. The first consisted of 512 units, while the second was dependent on patch size for the particular experiment. For all experiments, batch normalization was performed after each convolutional layer except for the output layer. ELU was used as activation function for the hidden layers (i.e., for the dense layers and the hidden convolutional layers). For the output layer multiple activation functions were tested, with the best results obtained using sigmoid.

The simplest architecture utilized is shown in **Figure 21**. It consists of one convolutional layer in the encoder, and one convolutional layer in the decoder. Downsampling is performed via max-pooling, while in the decoder an upsampling layer was used to increase the size of the compressed representation.

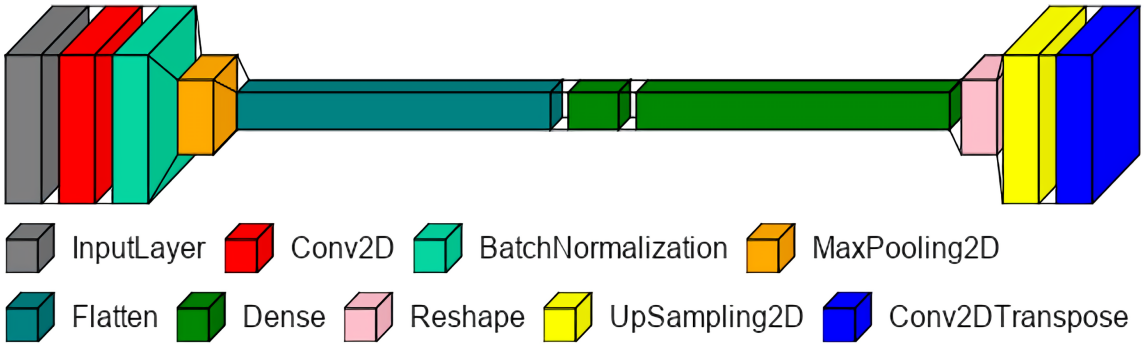


***Figure 21**: One-Layer 2D-CAE with max-pooling and upsampling layers*

Two convolutional layers in the encoder and decoder were also tested. For this approach two max-pooling layers were used in the encoder, while two upsampling layers was used in the decoder. The structure of the two-layer architecture is shown in **Figure 22**.

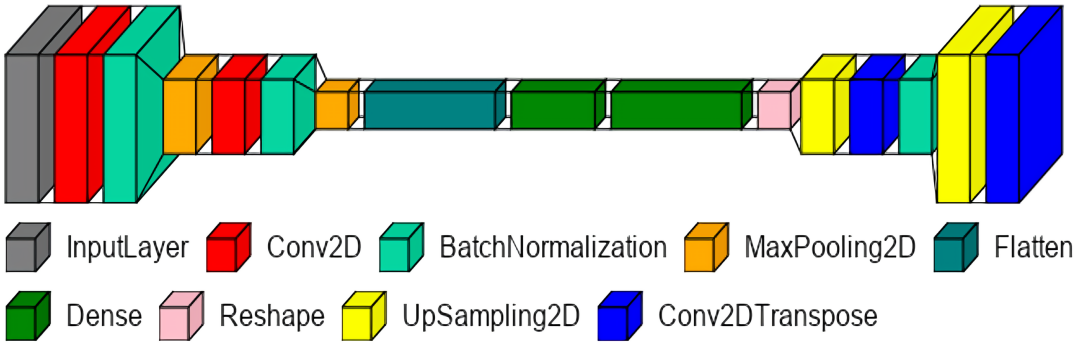


***Figure 22**: Two-Layer 2D-CAE with max-pooling and upsampling layers*

For the third approach, strided convolutions were utilized. This has the same effect as the max-pooling and upsampling layers used in the models above, which is to compress and decompress the data. While max-pooling and upsampling-layers are fixed operations, strided convolutions on the other hand are learned during training. The advantage of strided convolutions is that the model may learn certain properties in the process of compressing/decompressing. Even though the number of layers in the network are reduced with strided convolutions, the total amount of trainable parameters increases and therefore also the computational cost. The architecture of the CAE with one strided convolution in both the encoder and decoder are shown in **Figure 23**.

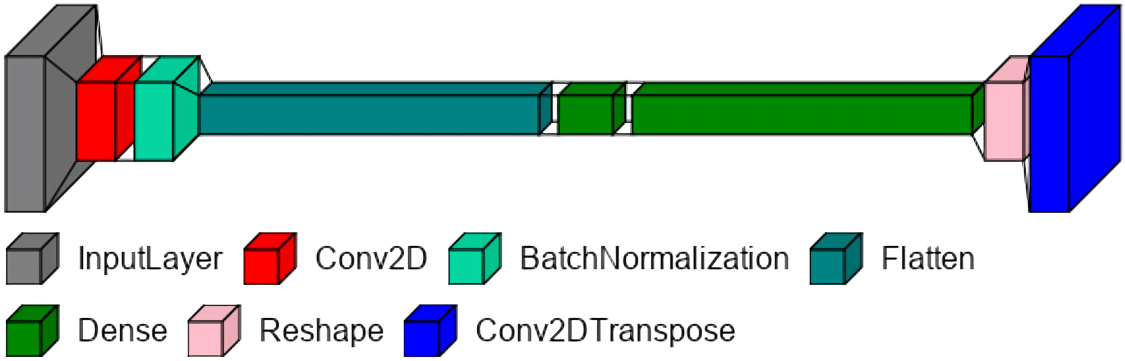


***Figure 23**: One-Layer 2D-CAE with strided convolutions*

Feature Extraction

Features were extracted from all of the 66 CCTA images using the clusters of the LV myocardium obtained via k-means in the previous step. To get the compressed representation of extracted image patches the decoder is removed from the CAE, which is illustrated in **Figure 24** below.

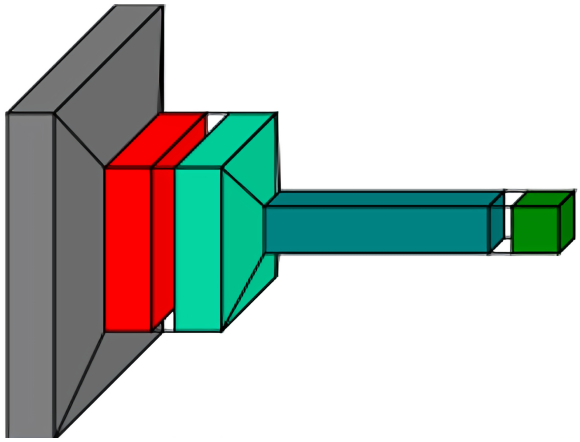


***Figure 24**: Encoder used to create a dense representation of myocardium patches*

The patches were preprocessed with the same techniques and parameters as were used to train the particular CAE model. This includes the minimum number of labeled voxels (requirement used to extract patches), clipping range, normalization approach, and voxel spacing used for resampling. For each patient, the image patches were assigned to a specific cluster through two different methods given by,

Center The patch is assigned to the cluster which is labeled at the center of the patch.

Highest Share The patch is assigned to the cluster which has the highest share of labeled voxels.

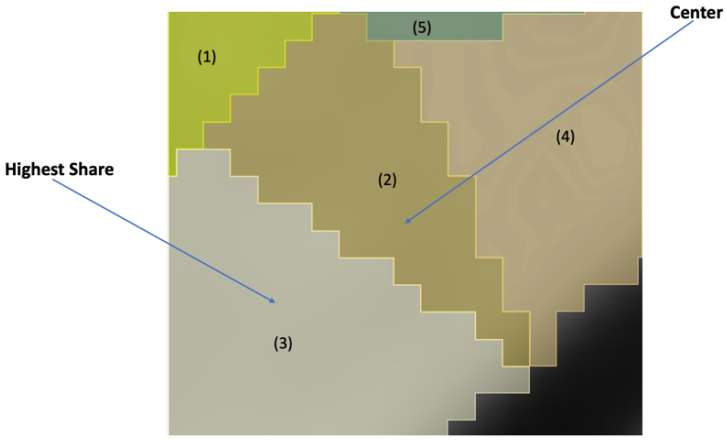


***Figure 25**: Extracted patch and its overlapping cluster-labels. The patch is assigned to cluster (2) for center-selection, while it is assigned to cluster (3) when the highest share method is used.*

After a particular patch has been assigned to the belonging cluster, the next step is to encode it. This is done by sending it through the encoder, which outputs a compressed vector of 512 units. Two methods were utilized to obtain the final feature vector for each patient, which are explained in detail below.

Methods for building the patient feature vector

After performing the previous step (encoding of myocardium patches), two different approaches for building the final patient feature vector from the clustered encodings were implemented. The first method is an interpretation of the method used in Zreik et al., while the second method is a new method introduced in this thesis.

Let x denote the coordinates/indices of a specific voxel in 2D space. Let $P(x)$ denote the patch centered at x. Let Enc denote the encoder and Dec denote the decoder. Let N_e denote the number of units/encodings in the compressed output vector from the encoder and let N_k denote the number of clusters. The vector of encodings for a particular patch at location x can then be expressed as $y(x) = Enc\big(P(x)\big)$.

Method 1 In the first method the sets $Y_i^k\{y_i(x) \mid x \in cluster\ k\}$ are collected for $k = 1,2,3 \dots N_k$, where $y_i(x)$ then represents the i^{th} encoding of the vector of encodings $y(x)$ and $i = 1,2,3 \dots N_e$. The standard deviation $\sigma_i^k = std(Y_i^k)$ is then computed and subsequently $f_e = max_i \sigma_i$ is extracted to build a vector

$$f = (f_1 \quad f_2 \quad f_3 \dots f_{N_e}) \tag{3.9}$$

Let $y_{ij}(x)$ denote the i^{th} encoding of the j^{th} vector of encodings at location $x \in cluster\ k$. First the standard deviation is computed for each encoding in cluster k by the following matrix operations

$$\begin{pmatrix} y_{11} & \cdots & y_{1j} \\ \vdots & \ddots & \vdots \\ y_{i1} & \cdots & y_{ij} \end{pmatrix} \rightarrow \begin{pmatrix} \sigma(y_{11}, \dots, y_{1j}) \\ \vdots \\ \sigma(y_{i1}, \dots, y_{ij}) \end{pmatrix} \rightarrow \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_i \end{pmatrix}$$

Subsequently the maximum standard deviation for each encoding y_i over all clusters is extracted

$$\begin{pmatrix} \sigma_1^1 & \cdots & \sigma_1^k \\ \vdots & \ddots & \vdots \\ \sigma_i^1 & \cdots & \sigma_i^k \end{pmatrix} \rightarrow \begin{pmatrix} max(\sigma_1^1, \dots, \sigma_1^k) \\ \vdots \\ max(\sigma_i^1, \dots, \sigma_i^k) \end{pmatrix} \rightarrow \begin{pmatrix} f_1 \\ \vdots \\ f_{N_e} \end{pmatrix}$$

This results in a feature vector with the same size as the original vector of encodings extracted from the patches.

Method 2 In the second method the sets $Y_j^k\{y_j(x) \mid x \in cluster\ k\}$ are collected for $k = 1,2,3 \ldots N_k$, where $y_j(x)$ represents the j^{th} vector of encodings for a particular cluster. The standard deviation $\sigma_j^k = std(Y_j^k)$ is then computed and subsequently $f_k = max_j \sigma_j^k$ is extracted to build a vector

$$f = (f_1 \quad f_2 \quad f_3 \ldots f_{N_k}) \tag{3.8}$$

Let $y_{ij}(x)$ denote the i^{th} encoding of the j^{th} vector of encodings at location $x \in cluster\ k$. The computation of the final feature value f_k for a particular cluster k is then given by the matrix operations

$$\begin{pmatrix} y_{11}^k & \cdots & y_{i1}^k \\ \vdots & \ddots & \vdots \\ y_{1j}^k & \cdots & y_{ij}^k \end{pmatrix} \rightarrow \begin{pmatrix} \sigma(y_{11}^k, \ldots, y_{i1}^k) \\ \vdots \\ y_{1j}^k, \ldots, y_{ij}^k) \end{pmatrix} \rightarrow max \begin{pmatrix} \sigma_1^k \\ \vdots \\ \sigma_j^k \end{pmatrix} \rightarrow f_k$$

This method computes the maximum standard deviation of any vector of encodings within a cluster, and results in a feature vector with the same size as the number of clusters.

3.3.3 Classification

The final vector of extracted features was used to classify the patients with functional significant stenosis. Two methods for classification were utilized, which includes Gaussian Process Classifier (GPC) and K-Neighbors Classifier (KNN). Classification was performed for all the CAE models presented in section 0. For each of the models, experiments were done for both *method 1* and *method 2* for defining the final vector of features. For both these methods, feature selection was tested on the train set and for the entire dataset. All the experiments were performed via 10-fold cross-validation. In this section the preprocessing approaches of the extracted features are described, as well as the details concerning the two different classification algorithms. Finally, the evaluation metrics used to evaluate the final performance of the model is presented.

Feature Selection

Both methods used to define the final vector of features provided a vector of a size around 500 features. The first method, i.e., *method 1*, results in a feature vector of size 512, which is the same as the method used in Zreik et al. However, this is a relatively large size for a vector that is meant to be used for classification purposes. In preliminary experiments low predictability was revealed from using the whole vector of features as input to the classification algorithms. This may be a result of the fact that the feature vector is containing irrelevant features, i.e., features that do not contain any information about the “unhealthiness” of the LV myocardium tissue. To solve this problem, feature selection can be applied as a preprocessing step, which compresses the vector of features before it is sent to the classification algorithm.

Univariate feature selection was used to extract the most relevant features from each CAE model. Multiple univariate statistical tests were performed on the vector of features, which works by comparing each feature to the target value in order to detect whether there is any statistically significant relationship between them. The selection is univariate as each feature is isolated, i.e., the other features are ignored in the analysis of the relationship. A test score is computed for each feature and the features with the highest scores are selected according to a specified threshold. Best results were obtained using a combination of chi-squared statistics (chi2) and mutual information statistics (MI). For non-negative features, the chi2-score is given by,

$$\mathcal{X}_c^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}\frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{3.10}$$

where O represents the observed value, E represents the expected value, and c is the degree of freedom. As our features are computed using standard deviation, the features are continuous and non-negative. Therefore, a chi2 statistic adapted for continuous features provided by sklearn was utilized [37]. This method computes the group sum of each non-negative feature value f given each possible target value y. In our classification problem the target is binary, i.e., a patient does either have a significant stenosis (1) or not (0). The algorithm therefore computes two sums for each feature value, which is the observed value. The observed values are then compared to the probability-weight grand total of the feature f given the target value y, which is the expected value. The observed and expected values are then compared by computing a p-value using the chi2 test with a degree of freedom equal to $(k-1)$, where k is the number of possible values for the target y. A chi-score and a p-value is obtained for each feature from this comparison.

The mutual information (MI) statistic is a non-negative value and estimates the mutual information for a discrete target variable and each individual feature. It measures the amount of information one can obtain from one random variable given another [39]. It uses the entropy, which is a quantitative estimate of how much information there is in a variable. For a discrete variable X and a continuous variable Y, the mutual information is given by,

$$I(X,Y) = H(X) + H(Y) - H(X,Y) \tag{3.11}$$

where $H(X)$ represents the entropy for X, $H(Y)$ is the entropy for Y, and $H(X,Y)$ is the entropy for X and Y. It relies on entropy estimation from k-nearest neighbors' distance [40]. Higher MI-values indicates higher dependencies between the variables, whereas a MI equal to zero implies that the two variables are independent.

In this thesis feature selection was performed in two stages, using a combination of MI-statistics and chi2-statistics. Firstly, the K-best MI features were selected, and subsequently the K-best chi2 features were selected based on the subset defined by the MI feature selection. The K-value for both methods was found through experimenting and is listed in the beginning

of section 4.4. The best K-value was found for both methods used to build the feature vector, i.e., *method 1* and *method 2*. Additionally, feature selection was performed on both the entire dataset and the train set. This was done in order to detect the amount of information that might be stored in the features. Varying K-values for these two approaches was also found through experimenting. A combination of MI-statistics and chi2-statistics were only used to select features for most of the experiments in classification of features extracted via *method 2*. In classification of features extracted via *method 1*, only chi2-statistics were utilized to extract the most relevant features.

Normalization

Multiple normalization approaches were tested on the features in preliminary experiments. Best results were obtained utilizing a robust scaler provided by *sklearn*, which uses statistics of the features which are robust to outliers [41]. It eliminates the median and scales the features in line with the interquartile range (IQR). The IQR ranges from the 1st quartile (Q1) to the 3rd quartile (Q3) which is illustrated in **Figure 26** below.

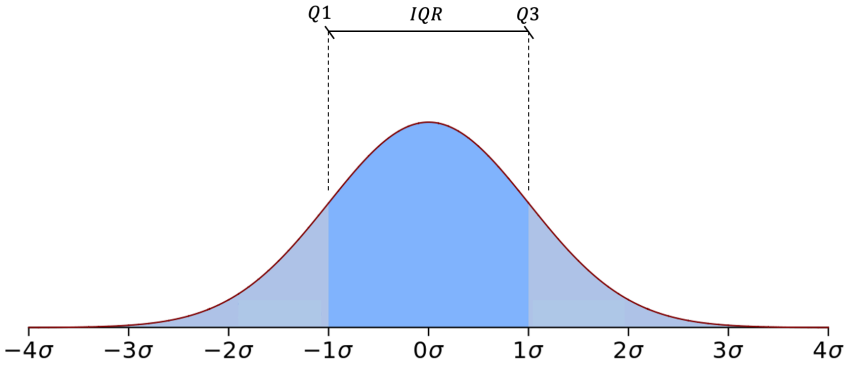


***Figure 26**:* The IQR illustrated for a probability density function of a normal distribution

Using this method, the features are centered and scaled separately by calculating the relevant statistics. These statistics were obtained by fitting the scaler on the training data in each classification experiment. Subsequently, the testing data was scaled based on the median and interquartile saved in the fitted scaler.

Classification Methods

Multiple classification methods were tested in preliminary experiments. Best results were obtained using Gaussian Process Classifier (GPC) and K-Neighbors Classifier (KNN) provided by *sklearn*. Other methods such as SVM and Deep Learning were tested, but provided lower classification results and were slower compared to GPC and KNN. GPC and KNN were utilized for all classification experiments. Both methods are considered to be non-parametric, which means that there are no underlying assumptions of the distribution of the data. Details regarding the two methods are given below.

Gaussian Process Classifier

A Gaussian Process Classifier (GPC) uses a Gaussian Process (GP) in order to perform probabilistic classification of the features. This means that the final predictions on the test data is obtained by means of class probabilities, which in our case is the probability of a binary target value.

A GP is a generalization of the Gaussian probability distribution. While the Gaussian probability distribution functions encapsulates the distribution of random variables, a GP on the other hand encapsulates the properties of the functions [42]. It requires a specification of a kernel, which regulates how samples relate to one another. This is known as the latent function f and characterizes the covariance function of the features and the target values [42]. This function holds the properties of a nuisance function, which means that we do not observe the values of f directly, but only the inputs X and target values y [42]. As the kernel regulates how the model distinguishes the samples, it is a crucial step for a GP model. The latent function f is removed via integration during prediction of the test-data. Multiple kernels were tested in preliminary experiments, where the best results were obtained using the Radial basis function (RBF) and Dot-Product (DP). The RBF is a stationary kernel defined by,

$$k(x_i, x_j) = \exp\left(\frac{d(x_i, x_j)^2}{2l^2}\right) \tag{3.12}$$

where l represents the length scale of the kernel and $d(x_i, x_j)^2$ is the Euclidean distance between x_i and x_j. For all experiments l was set to 1.0. The DP kernel on the other hand, is a non-stationary kernel. It is defined by,

$$k(x_i, x_j) = \sigma_0^2 + x_i \cdot x_j \tag{3.13}$$

where σ_0 is used for parameterization and controls the inhomogeneity of the kernel. The value of σ_0 was set to 1.0 for all experiments. As the Gaussian Process originally is used for regression, some important adjustments are made for it to work as a classifier. For binary classification, the model uses a function that interprets the internal representation and predicts the class probability of the features. This function is also known as a link function, which "squashes" its input into a range of [0,1] in order to obtain the probabilistic classification [42]. The Sigmoid function is a function that holds this property and is used in the GPC implantation provided by *sklearn*.

K-Neighbors Classifier

Neighbor-based classification is simpler and easy to understand compared to GPC. It is within the scope of instance-based learning algorithms, which means that it is non-generalizing. In a non-generalizing learning approach, the goal is to obtain classification of unseen data by storing instances of the training data. This is different from the GPC where a general internal model is constructed. In the KNN algorithm features are classified through a straightforward computation of the majority vote of the k-nearest neighbors of each datapoint. This happens as each feature of the k-nearest neighbors "votes" for its class and the class with the most votes are used as the prediction. All the training data was used to obtain classification for a new data point, where the Euclidean distance was utilized as the distance measure.

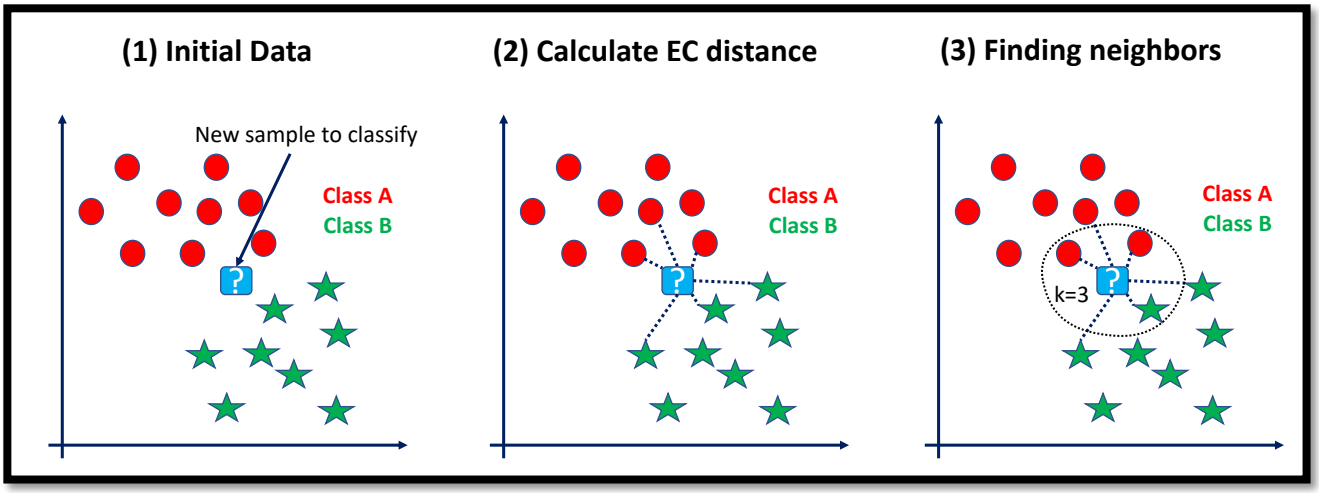


***Figure 27**: Visualization of the three steps of the KNN algorithm for classification of 2D samples using the Euclidean distance and the number of neighbors k=3*

The choice of k-value is highly data dependent. A lower value of k will make the predictions more sensitive to noise. A higher k-value on the other hand reduces this sensitivity but can make the classification boundaries less distinct. The k-value was found through experimenting, where a value between 3 and 5 was chosen as this provided the best classification results.

Evaluation metrics

Three evaluation metrics were used to evaluate the performance of the classification models, which includes Sensitivity, Specificity, ROC, and AUC. The definitions of Sensitivity and Specificity can be found in subsection 3.3.1, given in equations (3.2) and (3.3), respectively.

ROC (receiver operating characteristic) curve is a graph that visualizes the performance of a classification model at all thresholds. It plots two parameters, including the True Positive Rate (Sensitivity) and the False Positive Rate (1 – Specificity). AUC is strongly linked to the ROC and stands for "Area Under Curve". It measures the two-dimensional area under the ROC curve, which is an integral from (0,0) to (1,1). It summarizes the model performance across all the classification thresholds. It has an ideal value of 1, which is obtained for a model that classifies all the positive and negative samples correctly.

Sensitivity, Specificity and AUC is used for evaluation metrics of all the classification experiments. This includes all models used to classify features obtained via both *method 1* and *method 2* for defining the feature vector. The ROC curve is only visualized for *method 2* classification, as this method provided the best results. All curves are visualized with an 95 % asymptotic confidence interval of the average sensitivity given by,

$$CI = \mu \pm 1.96\frac{\sigma}{\sqrt{n}} \tag{3.14}$$

where μ represents the average sensitivity, σ is the standard deviation of the average sensitivity, and n is the number of samples.

Chapter 4 Results

4.1 Automatic Segmentation

In this section results from the automatic segmentation experiments are presented. All the models were evaluated using 3-fold cross-validation. Full-image analysis was chosen over a patch-wise crop as the resolution of the images were low. To be able to fit whole images into the network, the images were first resized by means of resampling to a common voxel-spacing that gave a median image size near to the desired input size, i.e., 160x160x80. A suitable voxel spacing of 1.24x1.24x1.42 for this purpose was found in preliminary experiments. Early stopping with a patience of 30 epoch monitoring the validation loss was used for all experiments. All models were scaled through z-score normalization and were set to train for 200 epochs.

Four different experiments were performed, where **Table 2** provides an overview of the parameters for each experiment, and **Table 3** shows the summary statistics for each experiment. The different experiments are presented in more detail below.

***Table 2**: Overview of parameters for the CNN automatic segmentations experiments.*

	Analysis	U-Net	CV	Input size	Norm.	Clipping	Resampling	Loss	Epoch
EX1	Full-image 3D	Standard	3-fold	160x160x80	z-score	(0, 275)	1.24x1.24x1.42	DSL	200
EX2	Full-image 3D	Standard	3-fold	160x160x80	z-score	(0, 275)	1.24x1.24x1.42	TL	200
EX3	Full Image 3D	Dense	3-fold	160x160x80	z-score	(0, 275)	1.24x1.24x1.42	TL	200
EX4	Full Image 3D	Residual	3-fold	160x160x80	z-score	(0, 275)	124x1.24x1.42	TL	200

***Table 3**: Overview of average results across all three folds for each CNN experiment. The model from the second fold of EX1 was utilized to obtain automatic segmentations for the entire dataset used for clustering and feature extraction in the next step of the total pipeline.*

	Background				Myocardium			
	DSC	Sens.	Spec.	Acc.	DSC	Sens.	Spec.	Acc.
EX1	0.996	0.996	0.892	0.993	0.887	0.892	0.997	0.993
EX2	0.997	0.997	0.866	0.993	0.881	0.868	0.997	0.994
EX3	0.996	0.996	0.868	0.993	0.872	0.868	0.996	0.993
EX4	0.996	0.996	0.868	0.993	0.872	0.868	0.996	0.993

4.1.1 Experiment 1: 3D U-Net Standard and DSL

A single GPU was used for this experiment. The models in each CV-split were set to train for 200 epochs. Overfitting started to occur at approximately 30 epochs for all the models, which resulted in termination of the training due to the early stopping condition. The model with the lowest validation loss was used to obtain the predictions which were used to evaluate the final model performance. All additional information about the specific parameters used in the experiment can be found in **Table 2**.

The predictions revealed a strong performance for background, as well as reasonably good results for myocardium. From **Table 4** we can observe that the average DSC of 0.887 was achieved for myocardium, while an average DSC of 0.996 was achieved for the background. Furthermore, an average sensitivity of 0.892 and average specificity of 0.997 was obtained for myocardium voxels. Additional information about the inference performance and predictions are recorded in **Table 4**.

***Table 4**: Average results from 3-fold cross-validation represented by the Dice similarity coefficient (DSC), Sensitivity, Specificity, and Accuracy for segmentation of LV myocardium using 28 CCTAs.*

	Background				Myocardium			
Fold	DSC	Sens.	Spec.	Acc.	DSC	Sens.	Spec.	Acc.
1	0.996	0.997	0.871	0.994	0.889	0.871	0.997	0.994
2	0.997	0.997	0.901	0.993	0.891	0.901	0.997	0.993
3	0.996	0.996	0.905	0.992	0.882	0.905	0.996	0.992
AVG	**0.996**	**0.996**	**0.892**	**0.993**	**0.887**	**0.892**	**0.997**	**0.993**

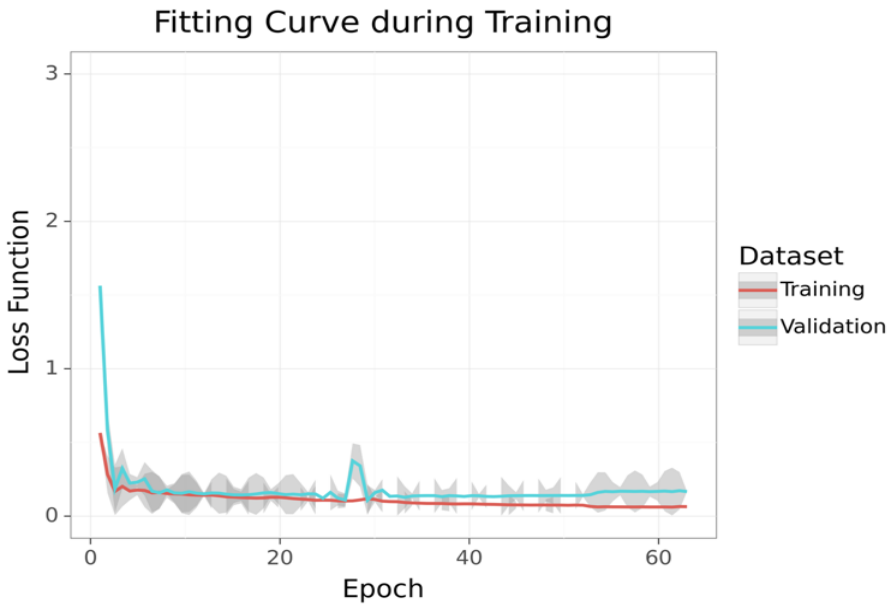


***Figure 28**: Fitting curve during training representing the average loss across all folds computed by Gaussian Process Regression. The red and cyan lines represent the training and validation data, respectively. The gray areas around represent the confidence interval.*

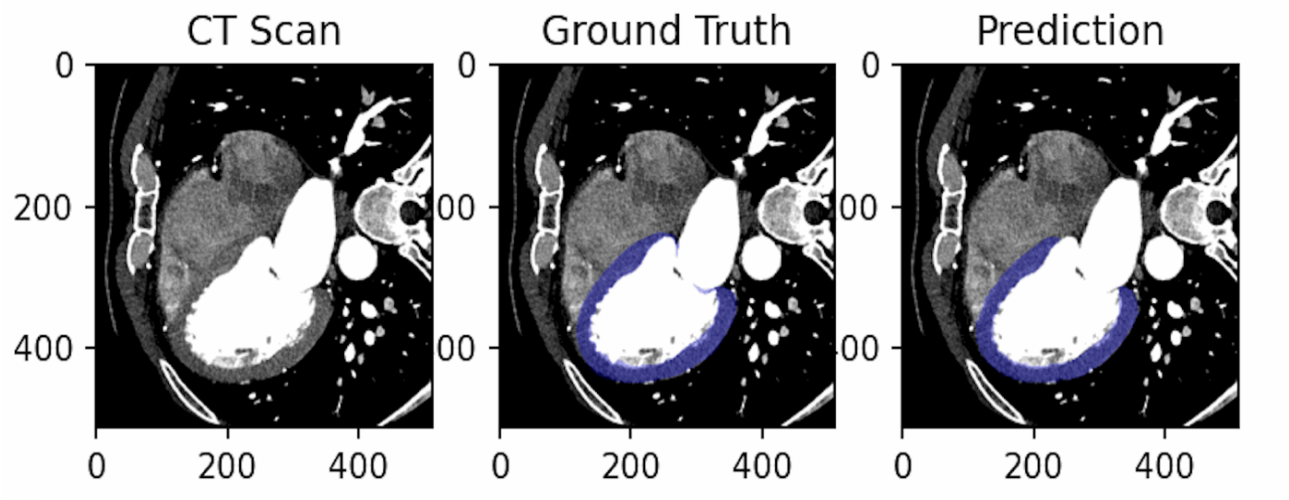


Figure 29: *Random slice from CT_FFR_29 with the highest DSC of 0.904.*

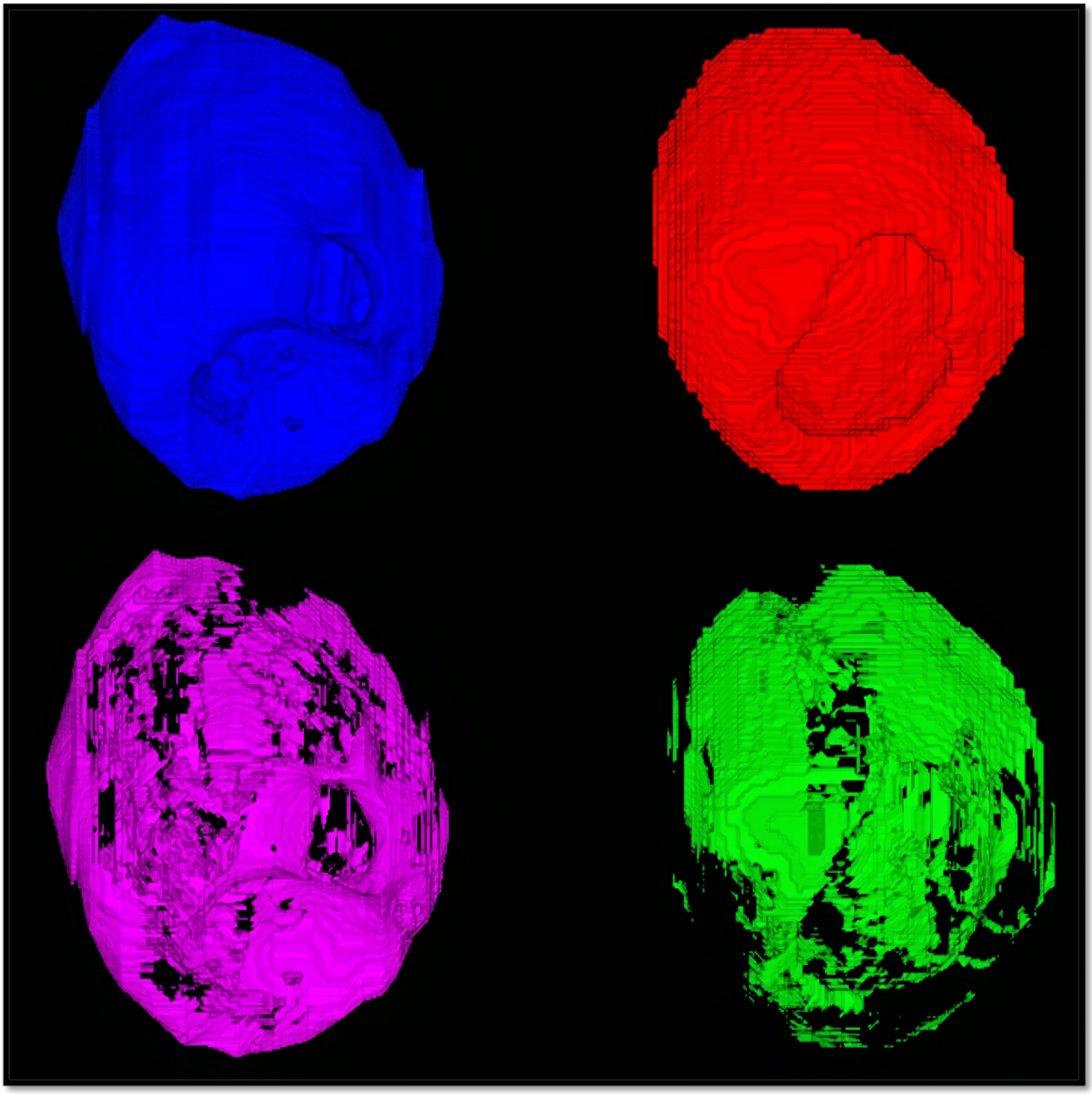


Figure 30*: Visualization of ground truth (blue), prediction (red), false negatives (pink), and false positives (green) from CT_FFR_29 with the highest DSC of 0.904.*

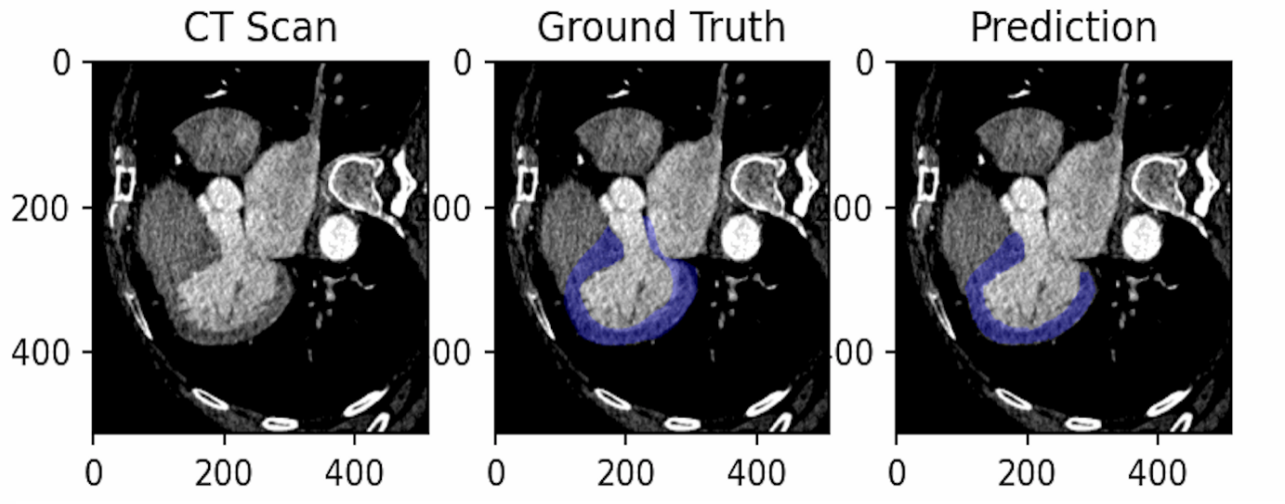


***Figure 31**: Random slice from CT_FFR_25 with the lowest DSC of 0.763.*

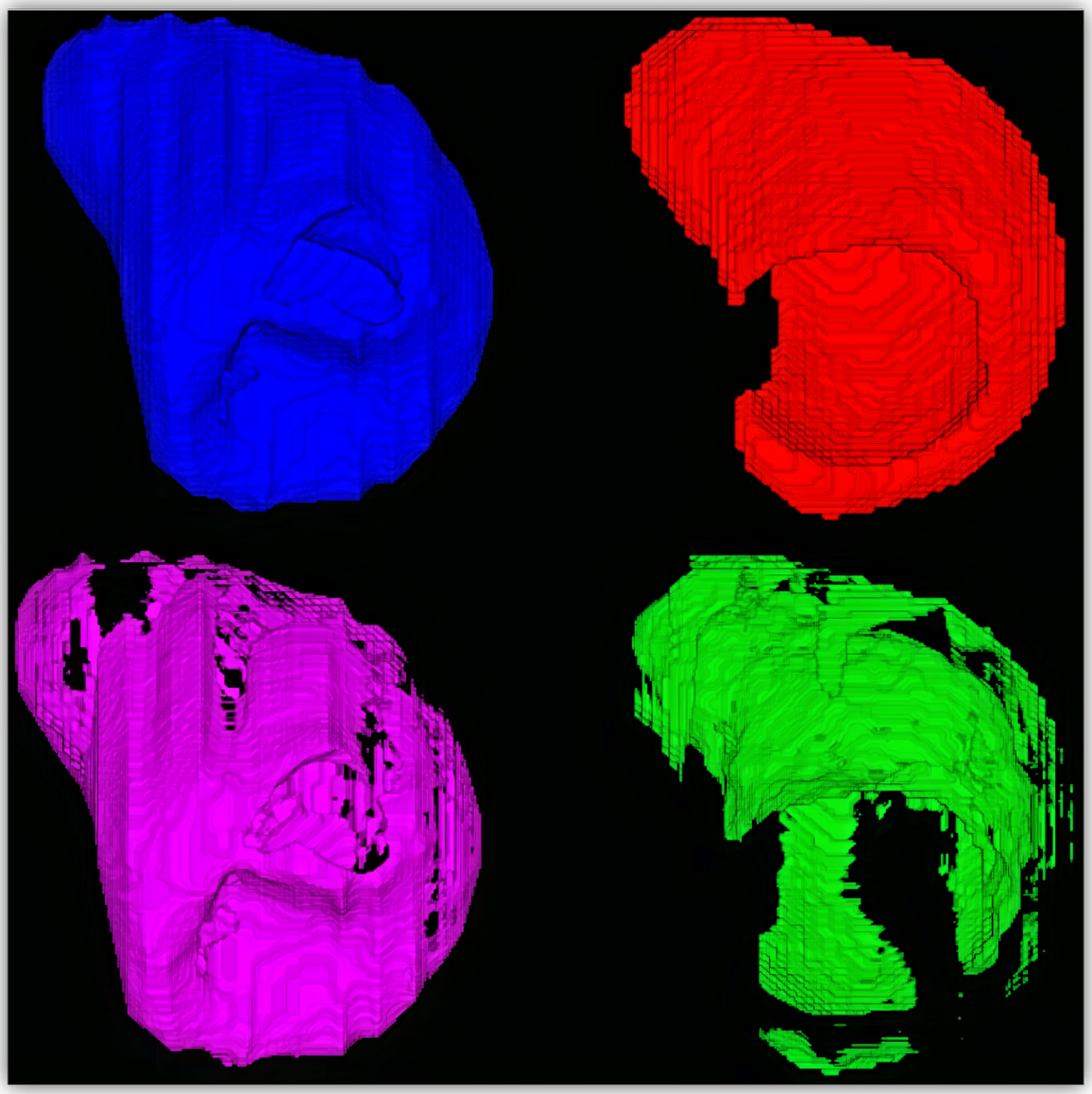


***Figure 32**: Visualization of ground truth (blue), prediction (red), false negatives (pink), and false positives (green) from CT_FFR_25 with the lowest DSC of 0.763.*

4.1.2 Experiment 2: 3D U-Net Standard and TL

A single GPU was used for this experiment. The models in each CV-split were set to train for 200 epochs. All models terminated at about 80 epochs due to the early stopping condition. No signs of overfitting on the training data were detected. The model with the lowest validation loss was used to obtain the predictions which were used to evaluate the final model performance. All additional information about the specific parameters used in the experiment can be found in **Table 2**.

The predictions revealed very similar results compared to the first experiment. An average DSC of 0.881 was achieved for myocardium, which is slightly lower compared to the experiment with DSL. Furthermore, an average sensitivity of 0.868 was achieved for myocardium voxels, which also is lower compared to using the DSL. Additional information about the inference performance and predictions are recorded in **Table 5.**

***Table 5**: Average results from 3-fold cross-validation represented by the Dice similarity coefficient (DSC), Sensitivity, Specificity, and Accuracy for segmentation of LV myocardium using 28 CCTAs.*

	Background				Myocardium			
Fold	DSC	Sens.	Spec.	Acc.	DSC	Sens.	Spec.	Acc.
1	0.997	0.998	0.860	0.993	0.881	0.858	0.998	0.994
2	0.997	0.997	0.890	0.994	0.893	0.899	0.997	0.994
3	0.996	0.997	0.848	0.993	0.870	0.848	0.997	0.993
AVG	**0.997**	**0.997**	**0.866**	**0.993**	**0.881**	**0.868**	**0.997**	**0.994**

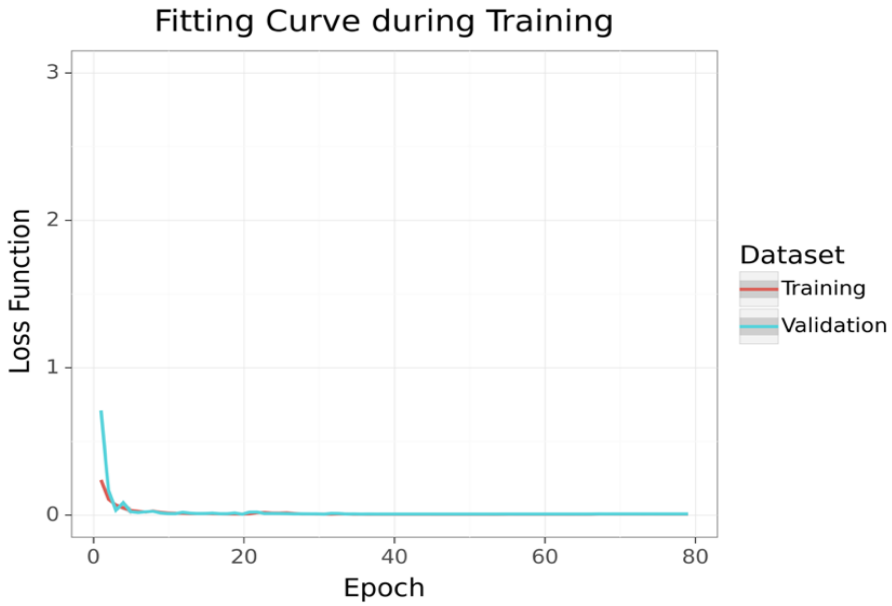


***Figure 33**: Fitting curve during training representing the average loss across all folds computed by Gaussian Process Regression. The red and cyan lines represent the training and validation data, respectively. The grey areas around represent the confidence interval.*

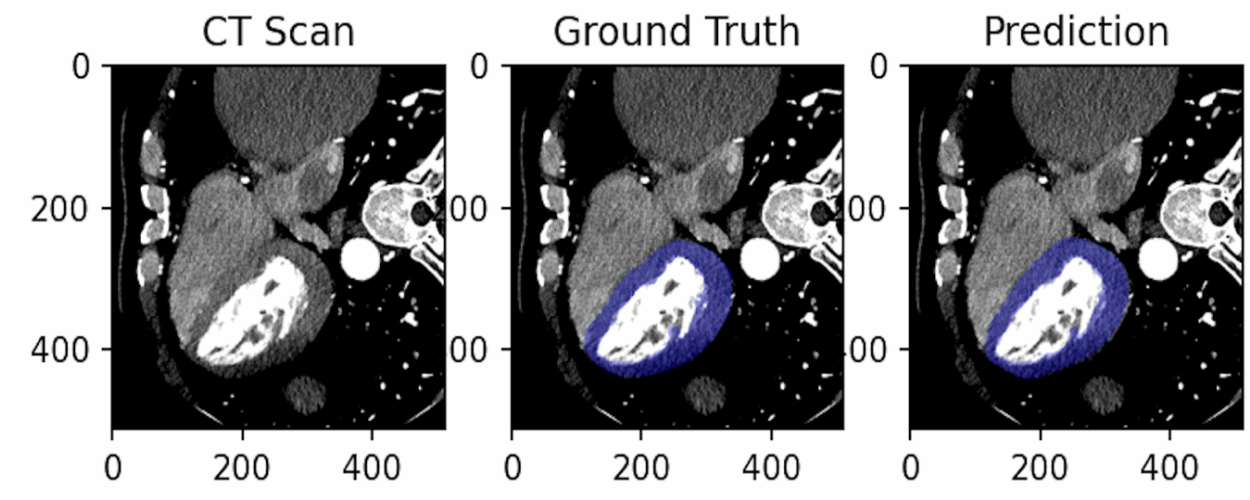


***Figure 34**: Random slice from CT_FFR_29 with the highest DSC of 0.912.*

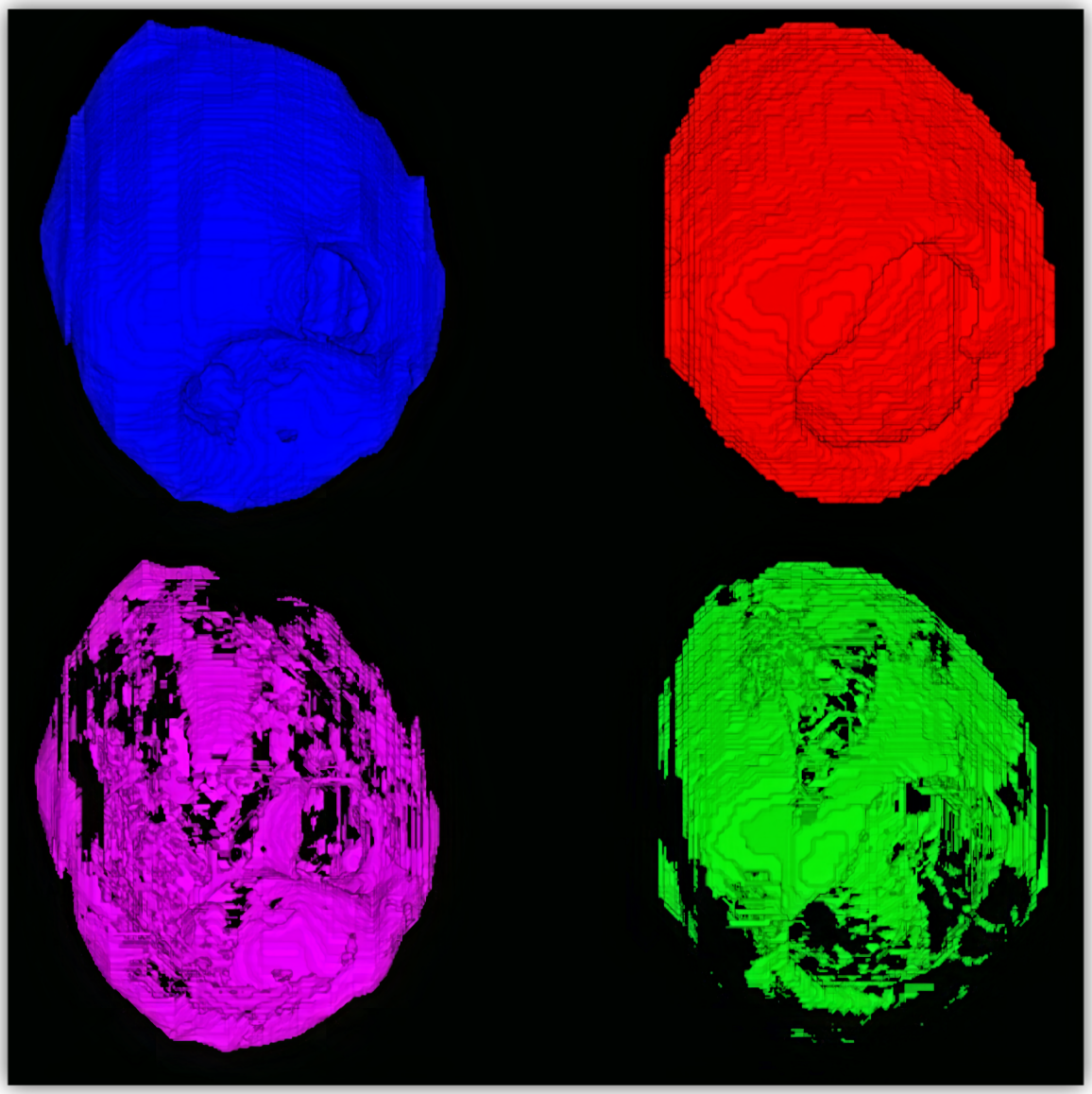


***Figure 35**: Visualization of ground truth (blue), prediction (red), false negatives (pink), and false positives (green) from CT_FFR_29 with the highest DSC of 0.912.*

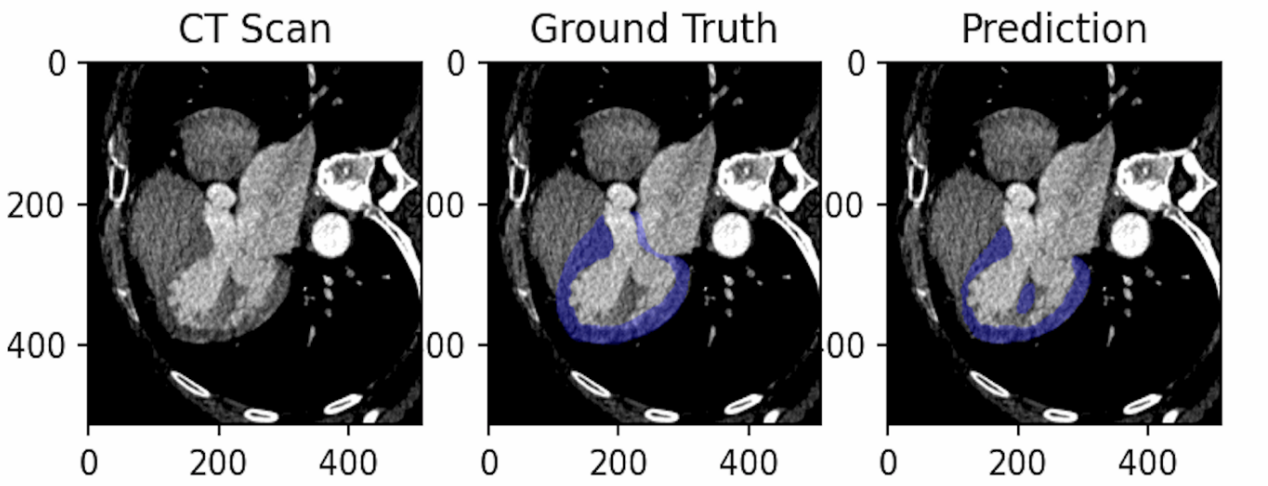


***Figure 36**: Random slice from CT_FFR_25 with the lowest DSC of 0.788.*

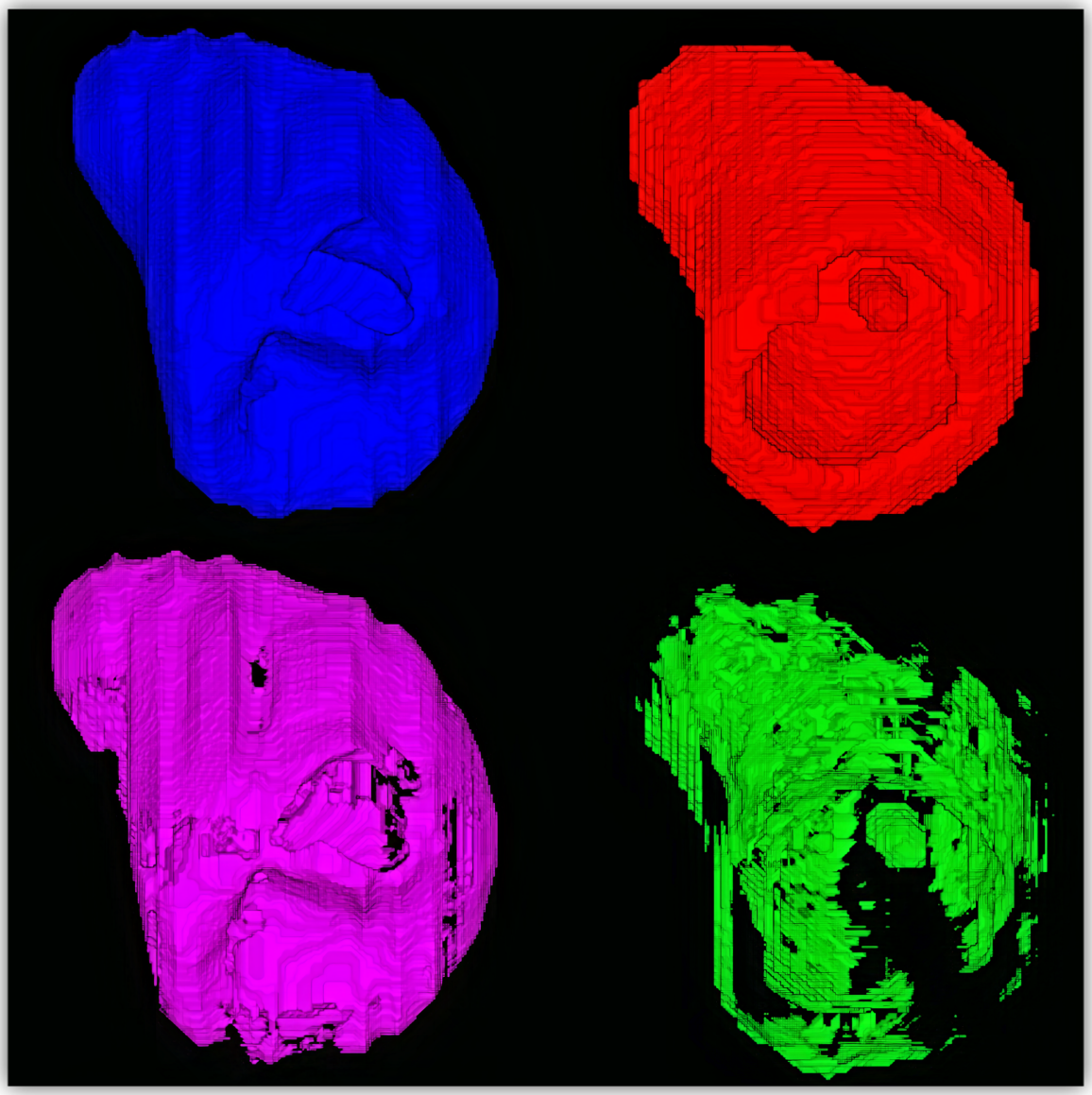


***Figure 37**: Visualization of ground truth (blue), prediction (red), false negatives (pink), and false positives (green) from CT_FFR_29 with the lowest DSC of 0.788.*

4.1.3 Experiment 3: 3D U-Net Dense and TL

A single GPU was used for this experiment. The models in each CV-split were set to train for 200 epochs. All models terminated at about 70 epochs due to the early stopping condition. No signs of overfitting on the training data were detected. The model with the lowest validation loss was used to obtain the predictions which were used to evaluate the final model performance. All additional information about the specific parameters used in the experiment can be found in **Table 2**.

An average DSC of 0.869 was achieved for myocardium, which is slightly lower compared to both the U-Net Standard experiments. Furthermore, an average sensitivity of 0.881 was obtained for myocardium voxels, which is higher than what was achieved for the U-Net Standard with Tversky Loss, but lower compared to the U-Net Standard with DSL. Additional information about the inference performance and predictions are recorded in **Table 6**.

***Table 6**: Average results from 3-fold cross-validation represented by the Dice similarity coefficient (DSC), Sensitivity, Specificity, and Accuracy for segmentation of LV myocardium using 28 CCTAs.*

	Background				Myocardium			
Fold	DSC	Sens.	Spec.	Acc.	DSC	Sens.	Spec.	Acc.
1	0.996	0.996	0.895	0.992	0.862	0.895	0.995	0.993
2	0.997	0.998	0.879	0.993	0.874	0.879	0.998	0.993
3	0.995	0.996	0.868	0.991	0.870	0.870	0.996	0.991
AVG	**0.996**	**0.997**	**0.881**	**0.992**	**0.869**	**0.881**	**0.997**	**0.992**

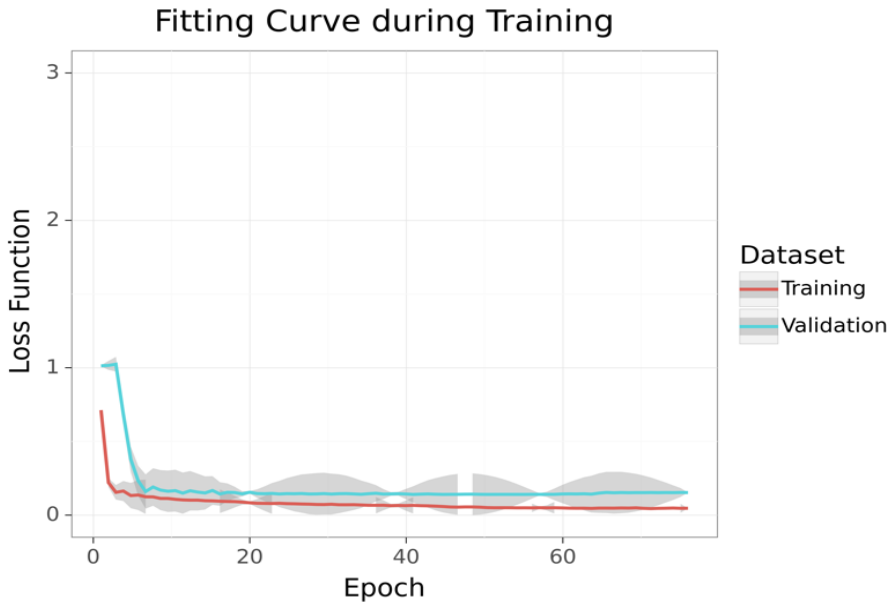


***Figure 38**: Fitting curve during training representing the average loss across all folds computed by Gaussian Process Regression. The red and cyan lines represent the training and validation data, respectively. The grey areas around represent the confidence interval.*

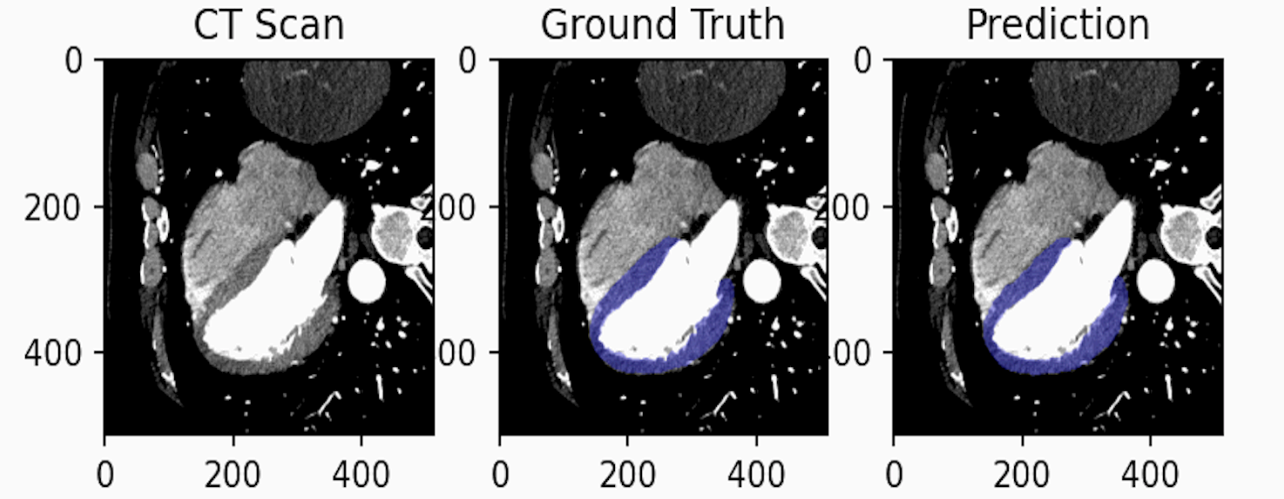


***Figure 39**: Random slice from CT_FFR_16 with the highest DSC of 0.914.*

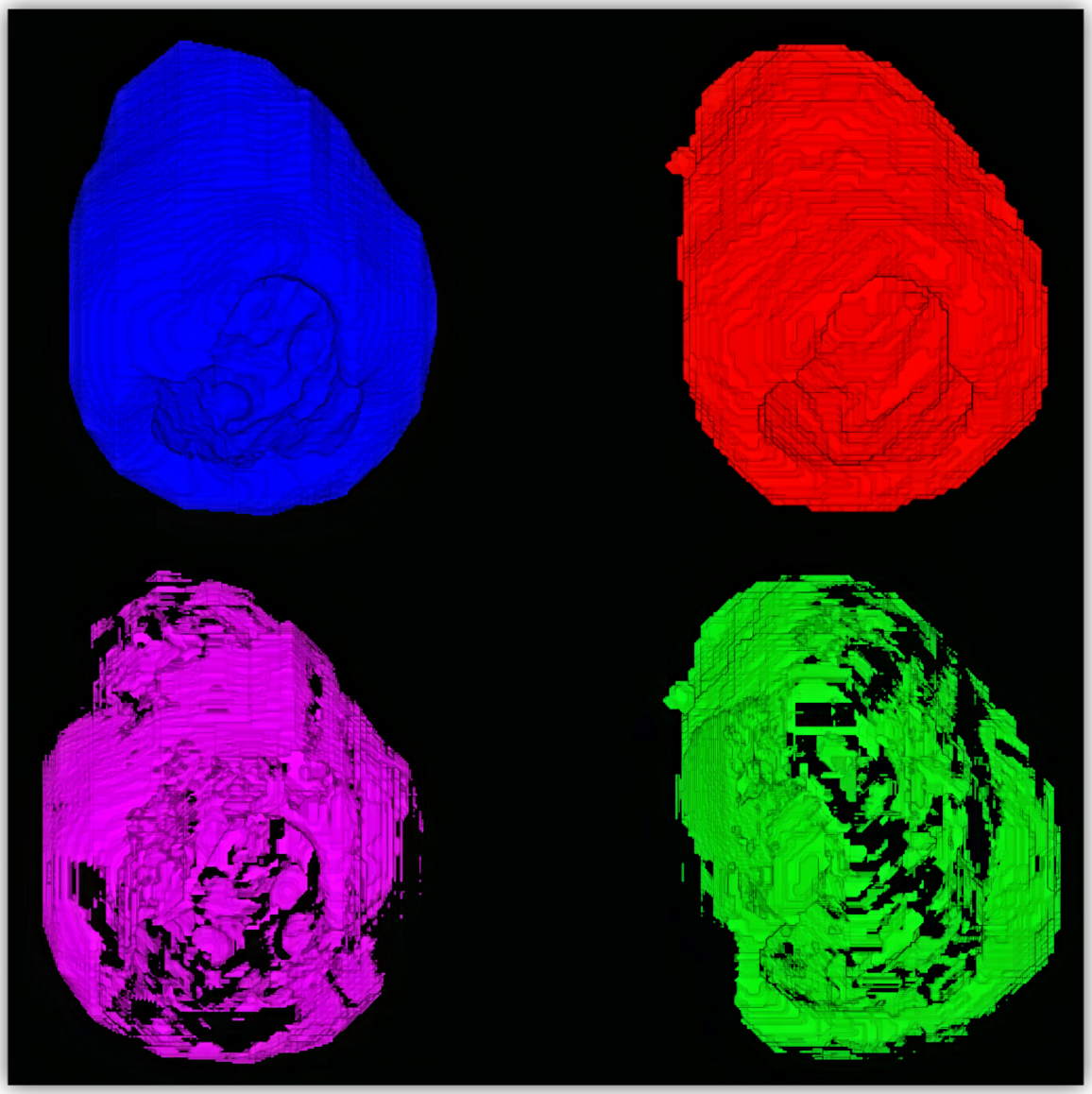


***Figure 40**: Visualization of ground truth (blue), prediction (red), false negatives (pink), and false positives (green) from CT_FFR_16 with the highest DSC of 0.914.*

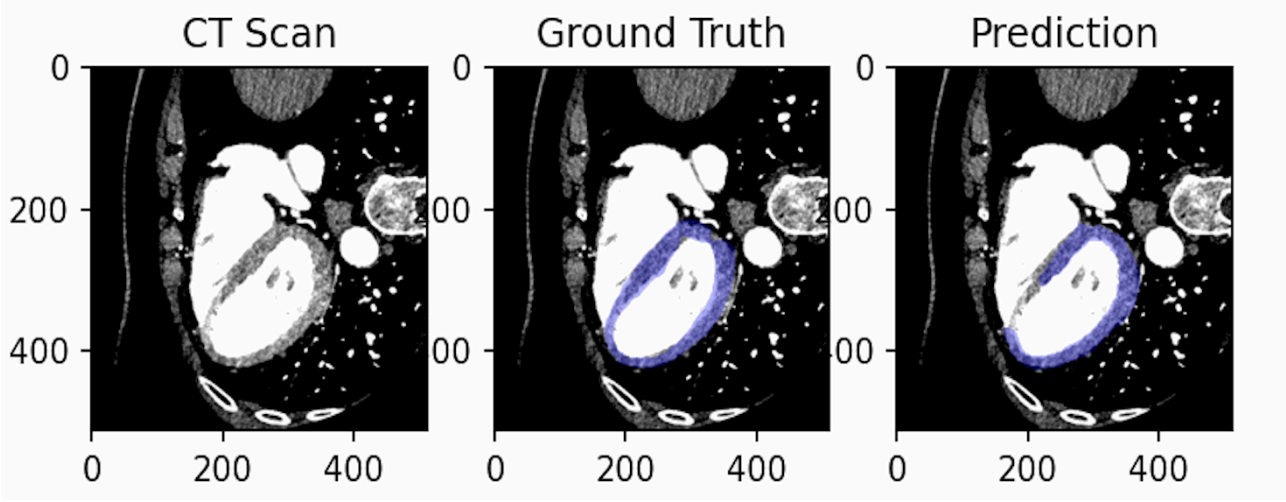


***Figure 41**: Random slice from CT_FFR_26 with the lowest DSC of 0.779.*

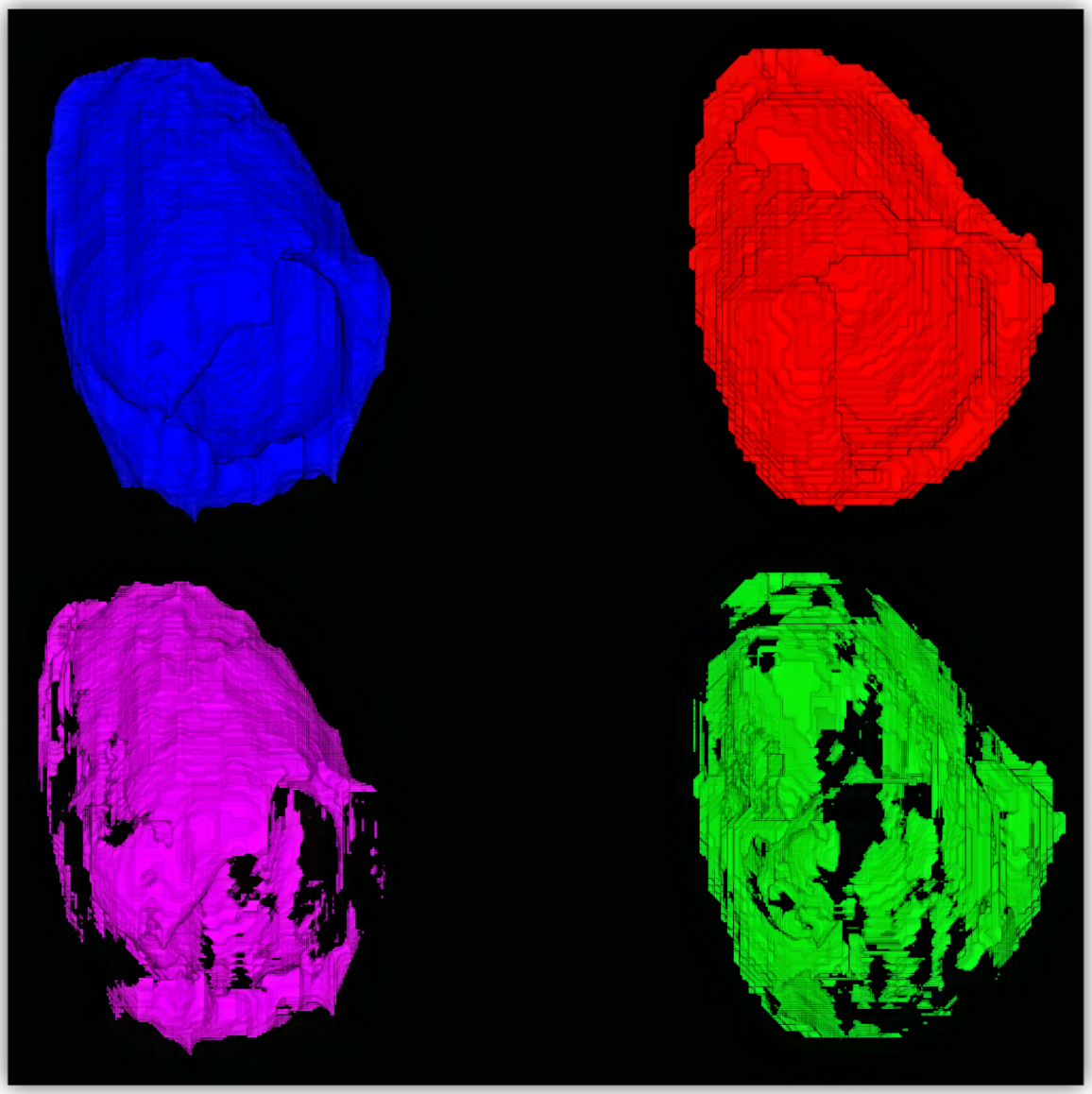


***Figure 42**: Visualization of ground truth (blue), prediction (red), false negatives (pink), and false positives (green) from CT_FFR_26 with the lowest DSC of 0.779.*

4.1.4 Experiment 4: 3D U-Net Residual and TL

A single GPU was used for this experiment. The models in each CV-split were set to train for 200 epochs. All models terminated at about 50 epochs due to the early stopping condition. Some signs of overfitting on the training data were detected. The model with the lowest validation loss was used to obtain the predictions which were used for evaluation of the final model performance. All additional information about the specific parameters used in the experiment can be found in **Table 2**.

An average DSC of 0.872 was achieved for myocardium, which is slightly higher compared to the U-Net Dense experiment, but lower than what was achieved in both the U-Net Standard experiments. Furthermore, an average sensitivity of 0.868 was obtained for myocardium voxels, which is lower than what was achieved for all the other experiments. Additional information about the inference performance and predictions are recorded in **Table 7**.

***Table 7**: Average results from 3-fold cross-validation for represented by the Dice similarity coefficient (DSC), Sensitivity, Specificity, and Accuracy for segmentation of LV myocardium using 28 CCTAs*

	Background				Myocardium			
Fold	DSC	Sens.	Spec.	Acc.	DSC	Sens.	Spec.	Acc.
1	0.996	0.996	0.877	0.993	0.867	0.877	0.996	0.993
2	0.996	0.996	0.861	0.993	0.869	0.861	0.996	0.993
3	0.996	0.997	0.866	0.992	0.879	0.866	0.997	0.992
AVG	**0.996**	**0.996**	**0.868**	**0.993**	**0.872**	**0.868**	**0.996**	**0.993**

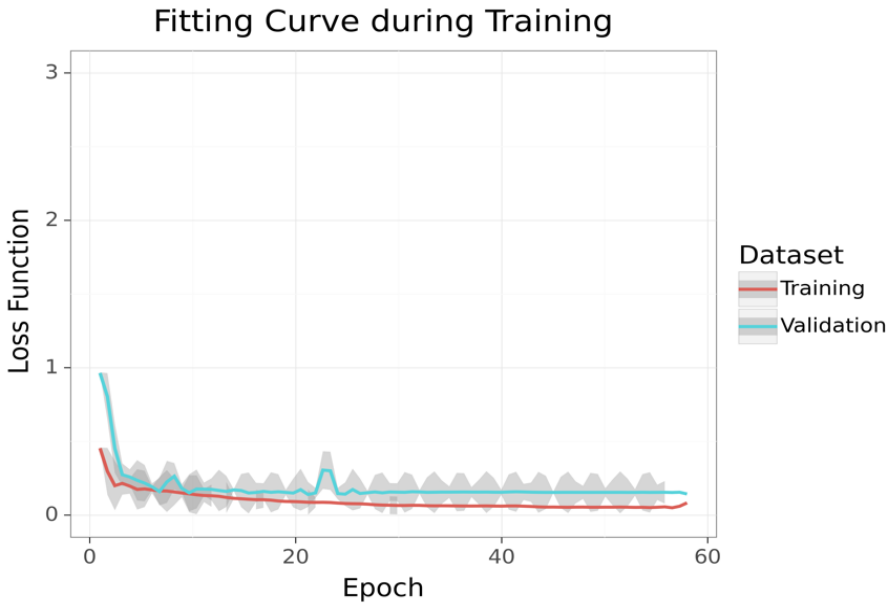


***Figure 43**: Fitting curve during training representing the average loss across all folds computed by Gaussian Process Regression. The red and cyan lines represent the training and validation data, respectively. The grey areas around represent the confidence interval.*

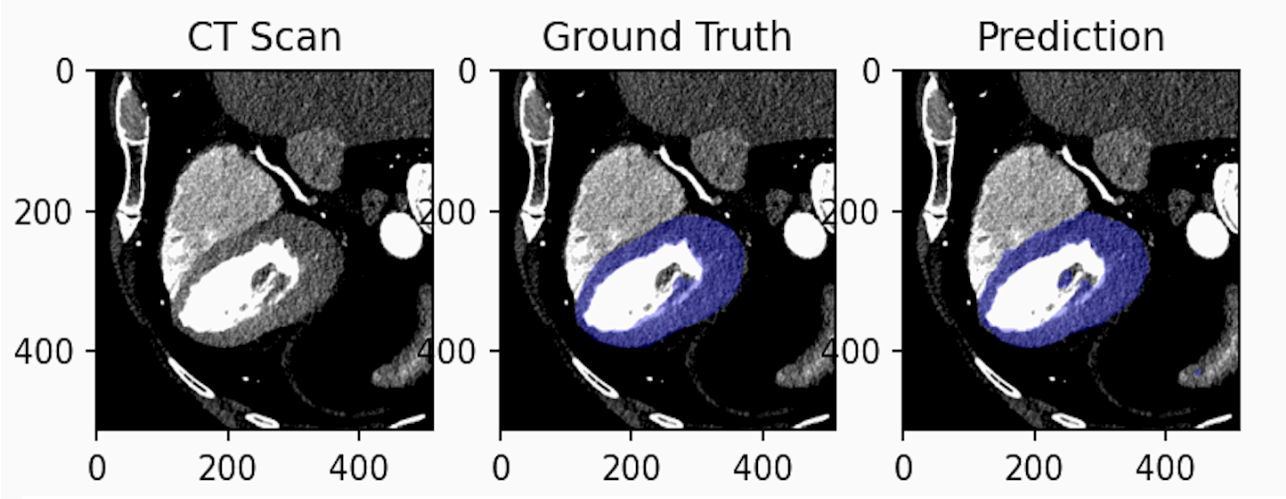


***Figure 44**: Random slice from CT_FFR_7 with the highest DSC of 0.896.*

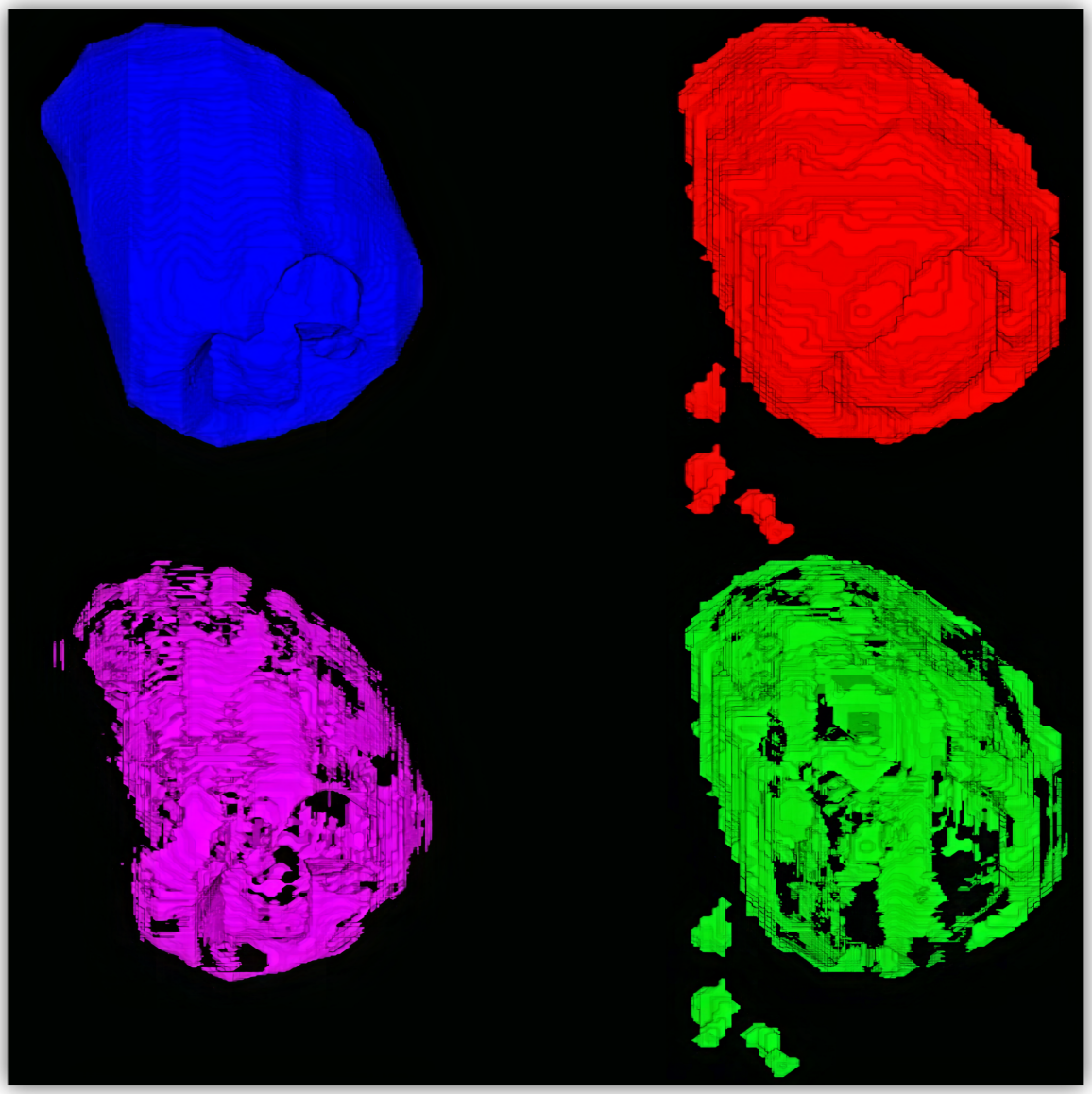


***Figure 45**: Visualization of ground truth (blue), prediction (red), false negatives (pink), and false positives (green) from CT_FFR_7 with the highest DSC of 0.896.*

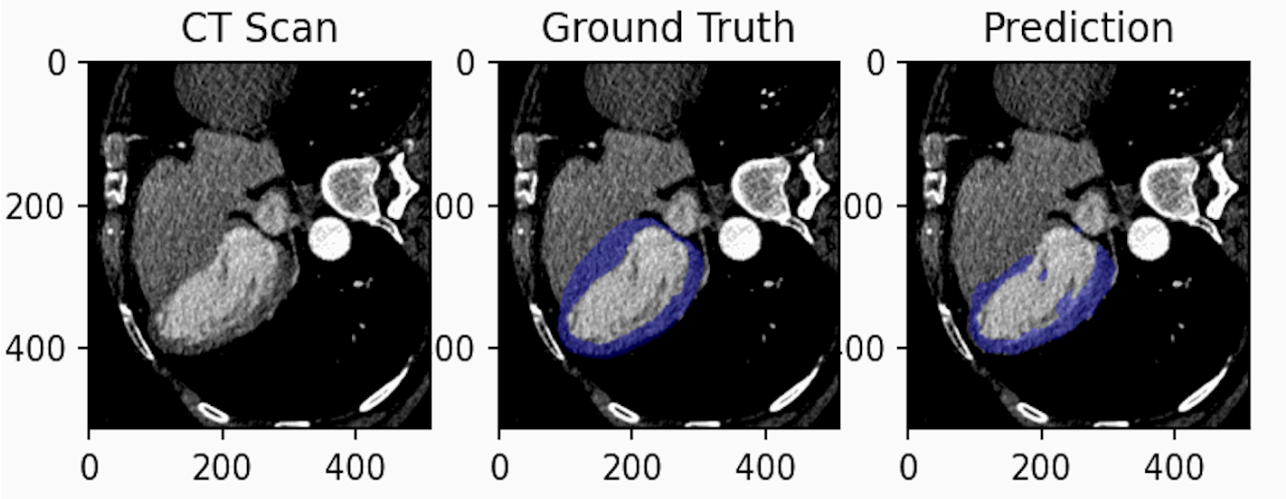


***Figure 46**: Random slice from CT_FFR_25 with the lowest DSC of 0.679.*

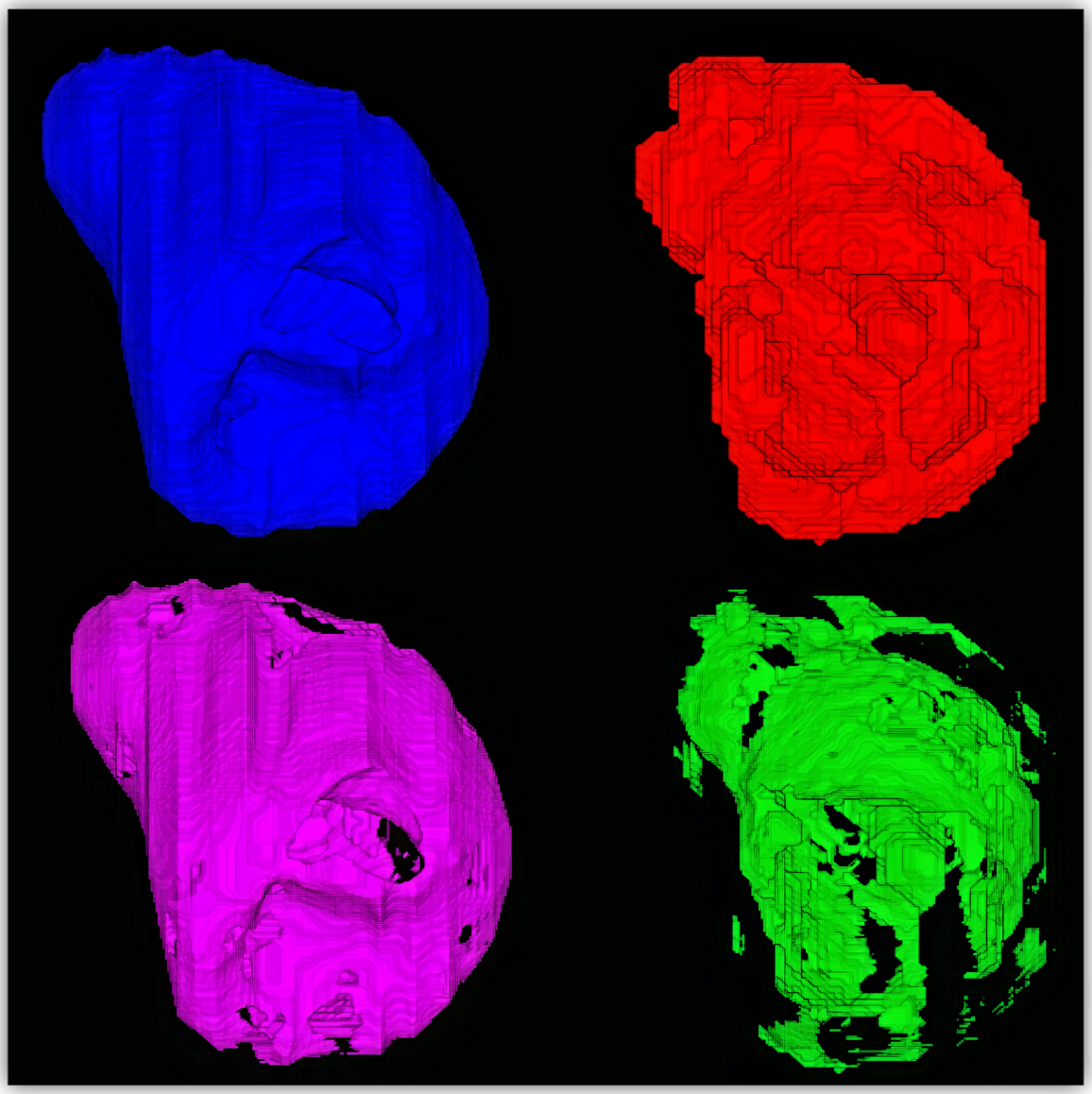


***Figure 47**: Visualization of ground truth (blue), prediction (red), false negatives (pink), and false positives (green) from CT_FFR_25 with the lowest DSC of 0.679.*

4.2 Clustering

In this section the results from the k-means clustering of the automatic segmented images are presented.

4.2.1 Experiment 1: K-means

The algorithm was set to run for 100 iterations. The number of clusters was set to 500, and random seed initialization was used to initialize the centroids prior to the first iteration. The utilized random seeds were different for each patient.

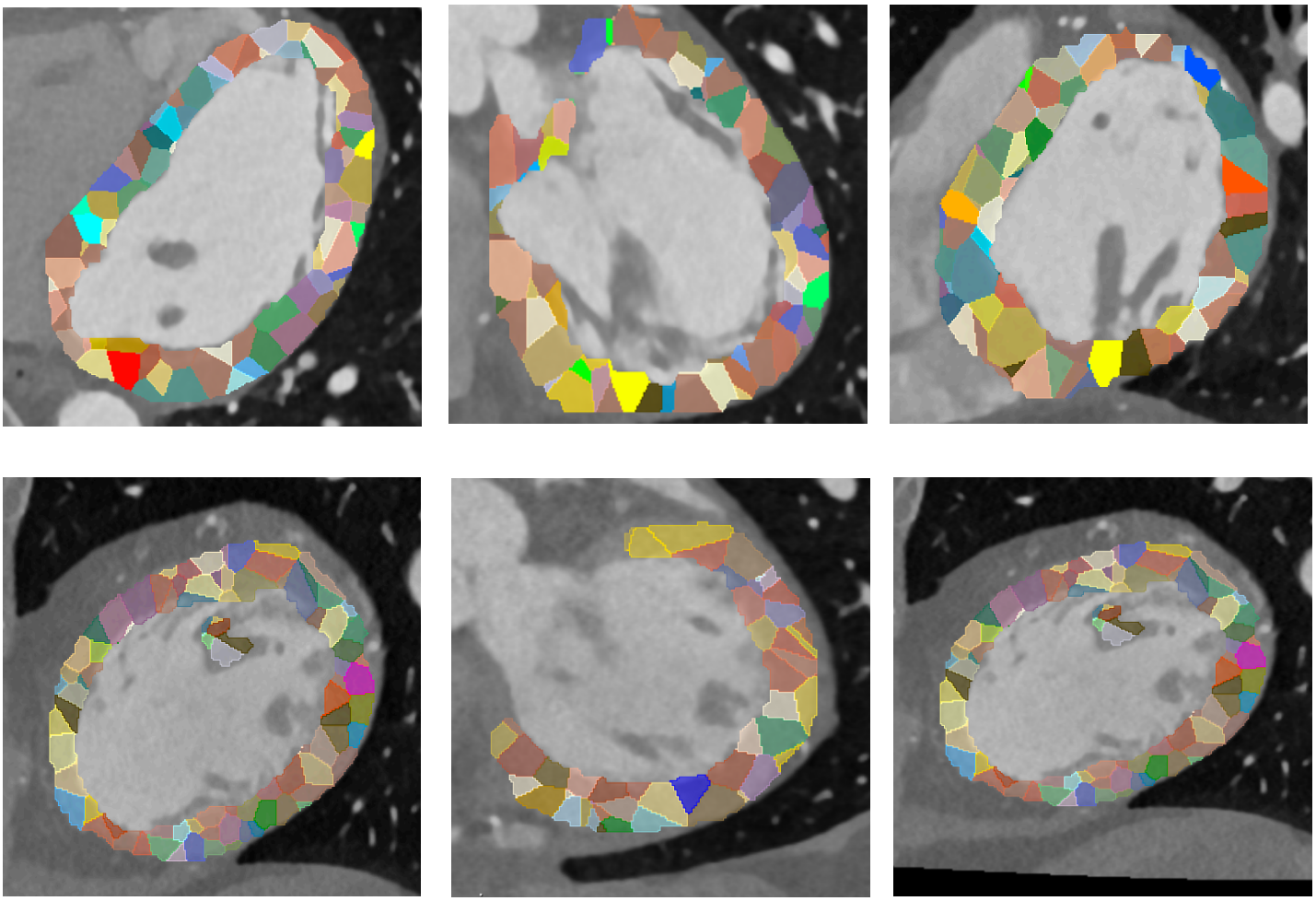


***Figure 48**: Results of k-means clustering for two different patients presented by random slices from sagittal plane (left), coronal plane (middle), and transverse plane (right).*

4.3 CAE

In this section the CAE experiments are presented. Different patch sizes were explored in relation to the other CAE parameters, which include the CAE architecture, filter size, patch-overlap, normalization approach, resampling, and the minimum number of labeled voxels. Upsampling via both max-pooling and strided convolutions was tested. Downsampling was either performed by means of an upsampling layer or strided convolutions. The number of convolutional layers were explored, where either one or two layers for encoder/decoder were employed. Resampling to a common voxel-spacing was tested on some of the models. Clipping was performed for most of the models where a varying threshold was explored.

A number of experiments were performed in preliminary experiments, where the one for each patch size that provided the best classification results are presented in this section. All the specific parameters for each model are listed in **Table 8**.

***Table 8**: Overview of the parameters used in each of the CAE experiments. Abbreviations: CNV = number of convolutional layers in enc/dec, DS = downsampling, US = upsampling, S = strides, MP = max-pooling, USL = upsampling layer, FS = filter size, PO = patch overlap, NR = normalization range, CL = clipping, RE = resampling, MLV = minimum labeled voxels, NP = total number of patches (training + validation + testing).*

	CNV	DS	US	FS	PO	NR	CL	RE	MLV	NP
P16	1	S	S	16	12	[0,1]	(0,275)	(0.8, 0.8, 0.8)	0.5	450 000
P20	1	S	S	32	8	[0,1]	(0, 275)	None	0.7	272 000
P24	1	S	S	16	12	[0,1]	(0, 275)	None	0.5	475 000
P28	2	2xMP	2xUSL	64→32	14	[0,1]	(0, 275)	None	0.7	215 000
P36	1	MP	S	32	18	[0,1]	(-1000, 400)	None	0.7	250 000
P48	1	MP	S	32	18	[0,1]	(-1000, 400)	None	0.7	250 000

4.3.1 Experiment 1: Patch Size 16x16

	MSE
Training	1.589-4
Testing	1.691-4

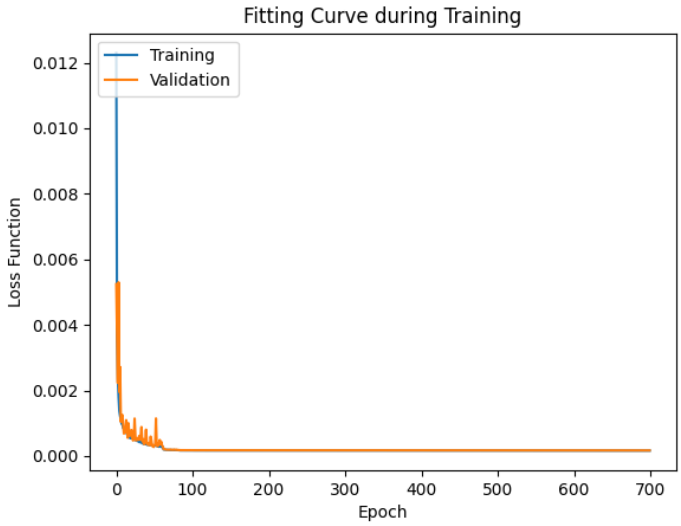


***Figure 49**: Fitting curve for the experiment with a patch size of 16x16.*

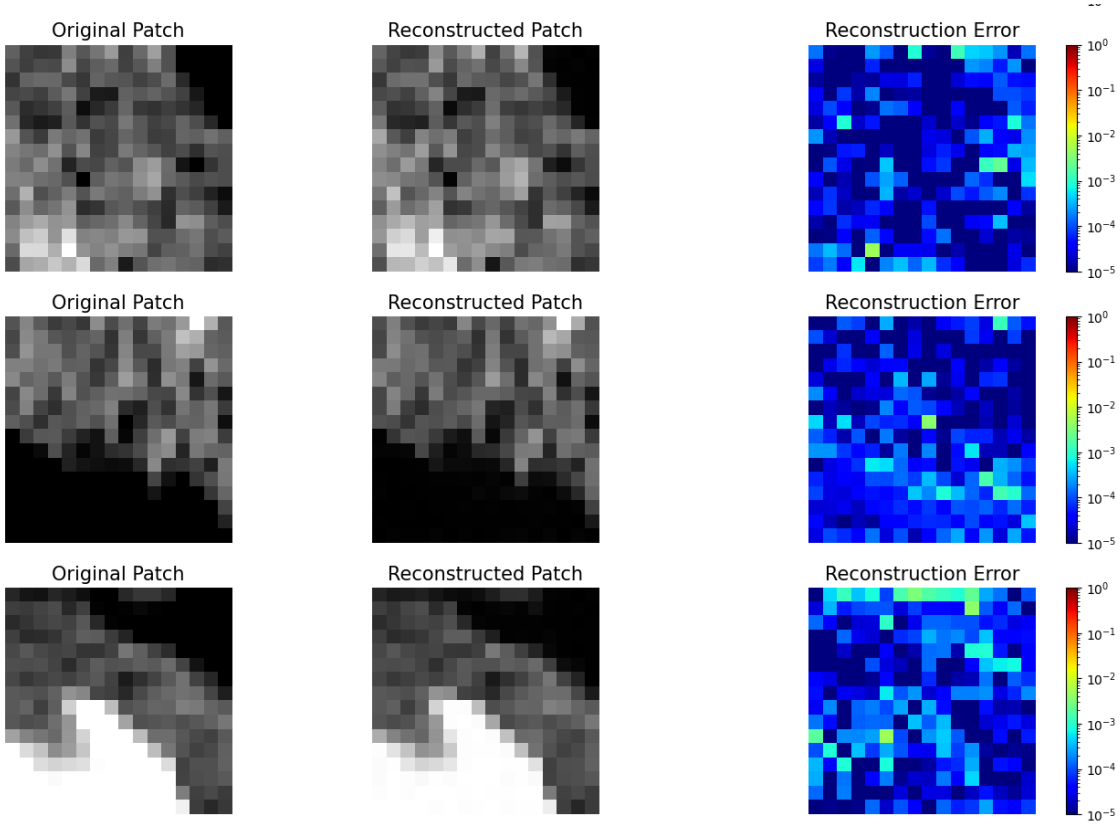


***Figure 50**: Three examples of reconstructed patches by the CAE randomly selected from the test set. Each row contains the 16x16 original input patch (right), the 16x16 reconstructed patch (middle), and the reconstruction error (left) calculated from the scaled pixel intensities.*

4.3.2 Experiment 2: Patch Size 20x20

	MSE
Training	4.367-4
Testing	4.513-4

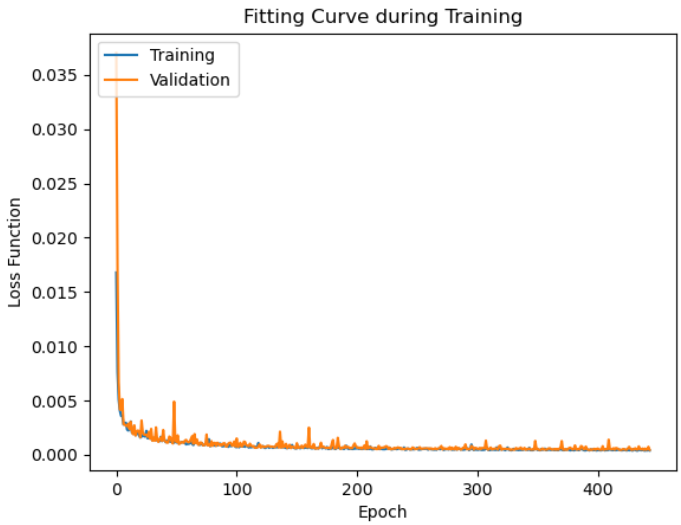


***Figure 51**: Fitting curve for the experiment with a patch size of 20x20.*

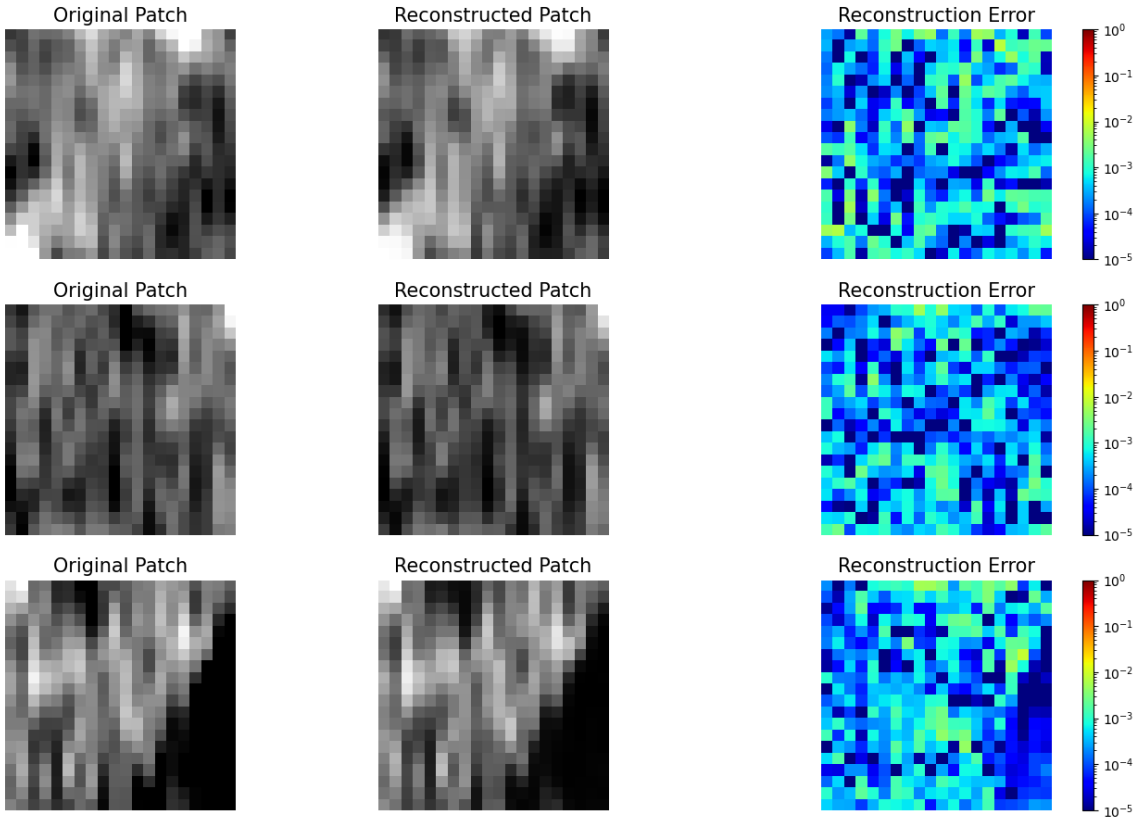


***Figure 52**: Three examples of reconstructed patches by the CAE randomly selected from the test set. Each row contains the 20x20 original input patch (right), the 20x20 reconstructed patch (middle), and the reconstruction error (left) calculated from the scaled pixel intensities.*

4.3.3 Experiment 3: Patch Size 24x24

	MSE
Training	7.235-5
Testing	8.590-5

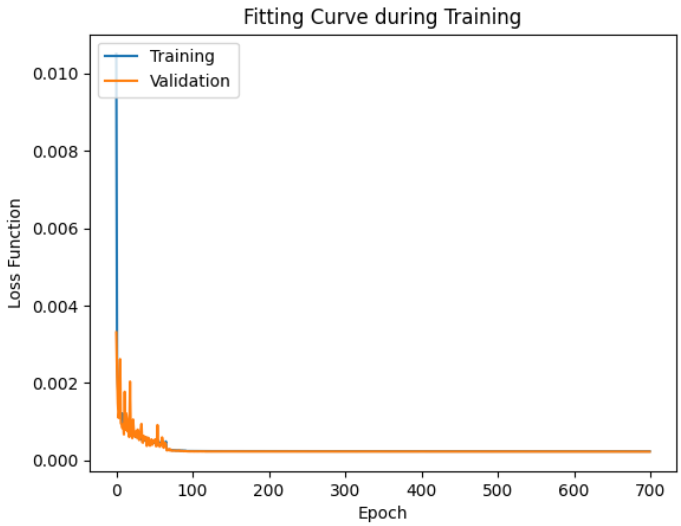


***Figure 53**: Fitting curve for the experiment with a patch size of 24x24*

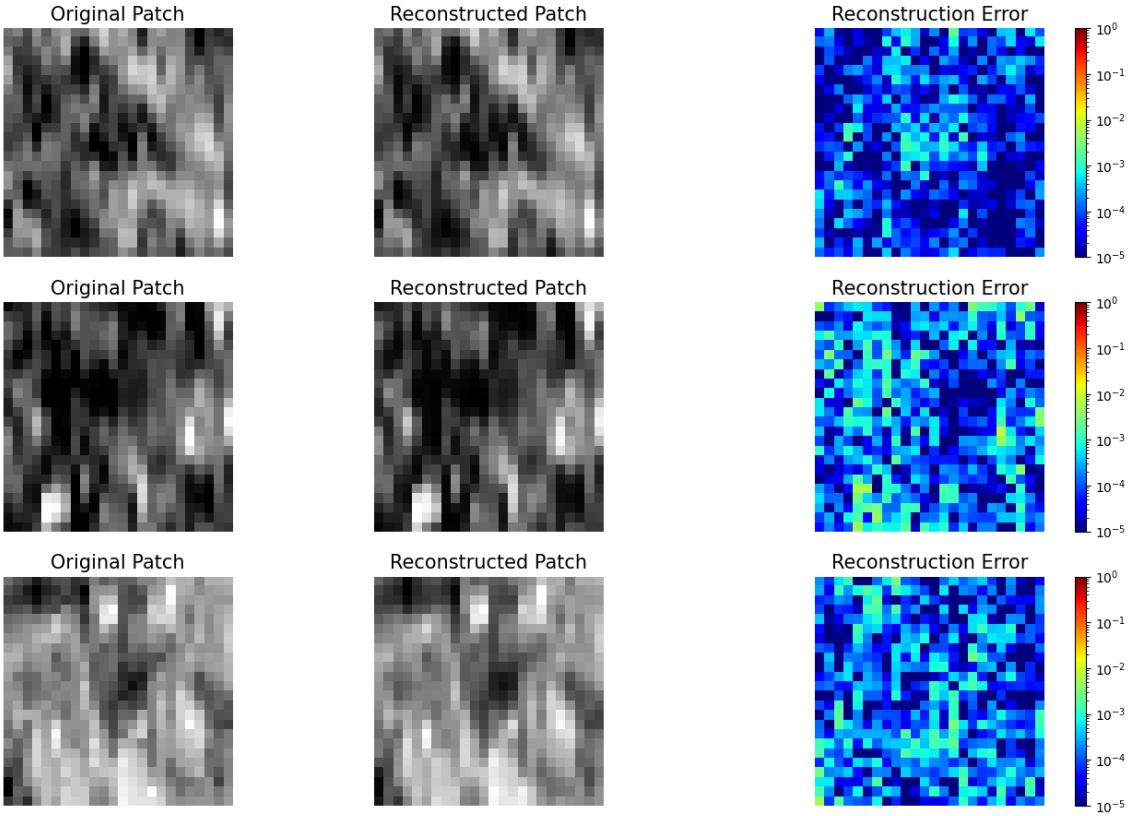


***Figure 54**: Three examples of reconstructed patches by the CAE randomly selected from the test set. Each row contains the 24x24 original input patch (right), the 24x24 reconstructed patch (middle), and the reconstruction error (left) calculated from the scaled pixel intensities.*

4.3.4 Experiment 4: Patch Size 28x28

	MSE
Training	9.180-4
Testing	9.710-4

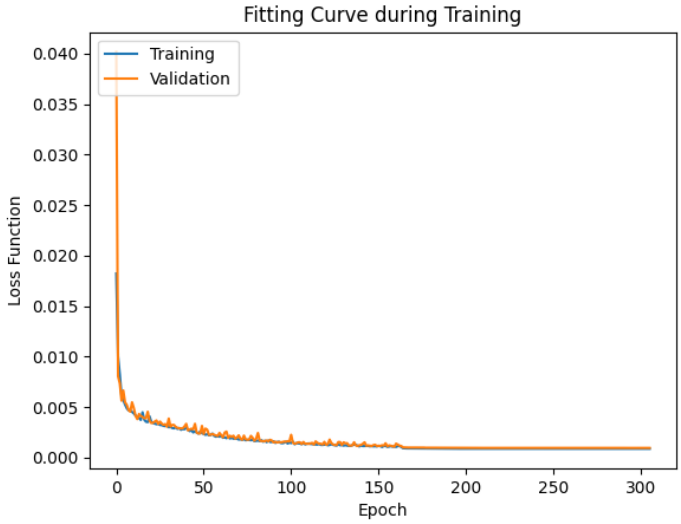


***Figure 55**: Fitting curve for the experiment with a patch size of 28x28.*

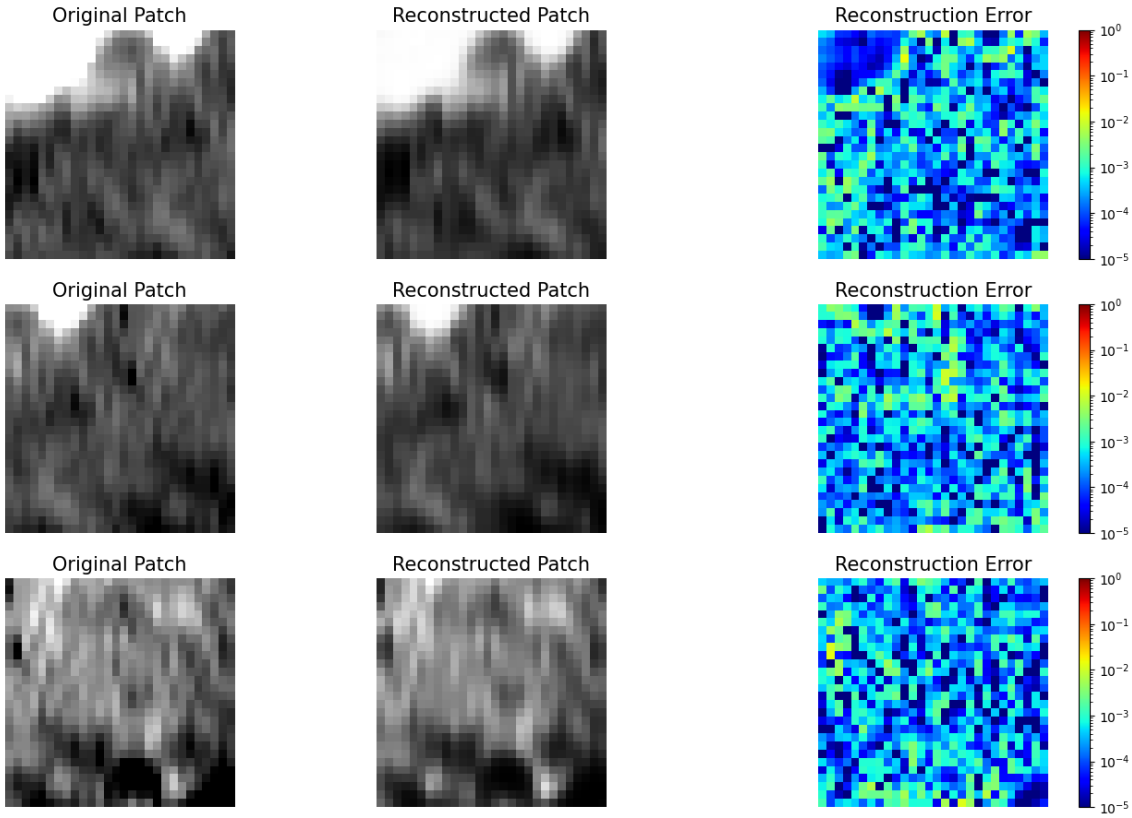


***Figure 56**: Three examples of reconstructed patches by the CAE randomly selected from the test set. Each row contains the 28x28 original input patch (right), the 28x28 reconstructed patch (middle), and the reconstruction error (left) calculated from the scaled pixel intensities.*

4.3.5 Experiment 5: Patch Size 36x36

	MSE
Training	1.109-4
Testing	1.545-4

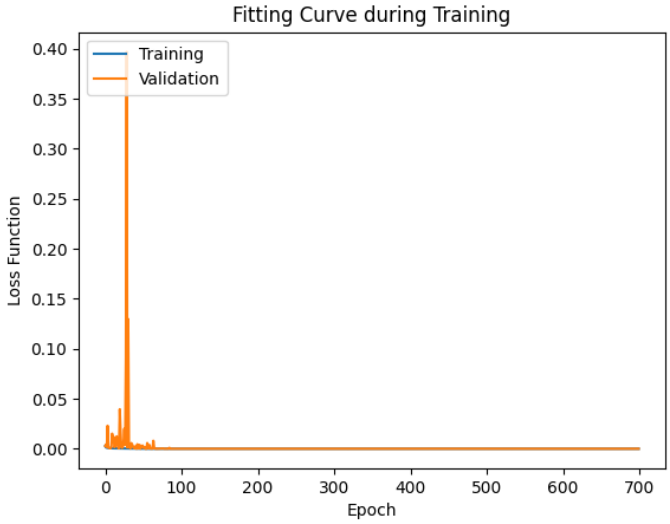


***Figure 57**: Fitting curve for the experiment with a patch size of 36x36.*

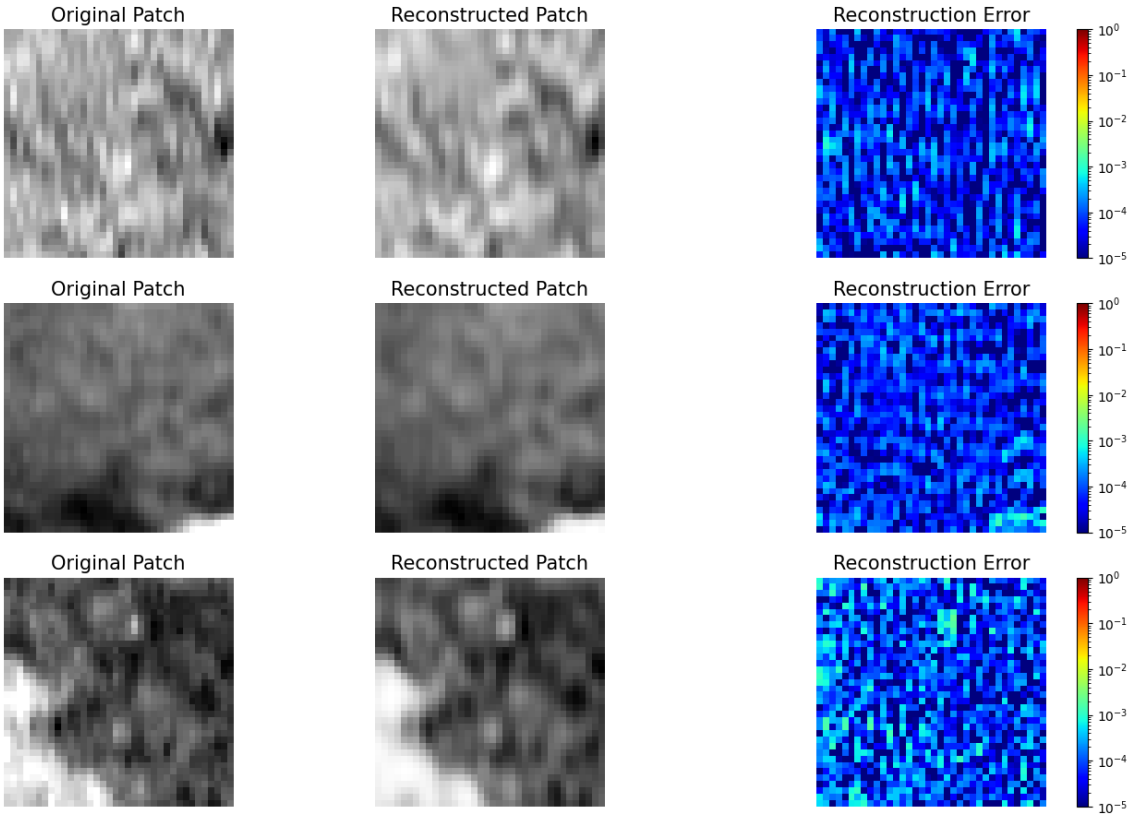


***Figure 58**: Three examples of reconstructed patches by the CAE randomly selected from the test set. Each row contains the 36x36 original input patch (right), the 36x36 reconstructed patch (middle), and the reconstruction error (left) calculated from the scaled pixel intensities.*

4.3.6 Experiment 6: Patch Size 48x48

	MSE
Training	1.411-4
Testing	1.614-4

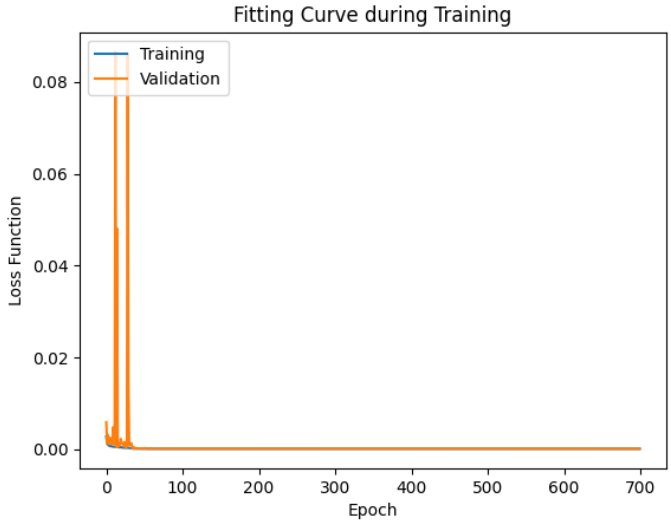


***Figure 59**: Fitting curve for the experiment with a patch size of 48x48.*

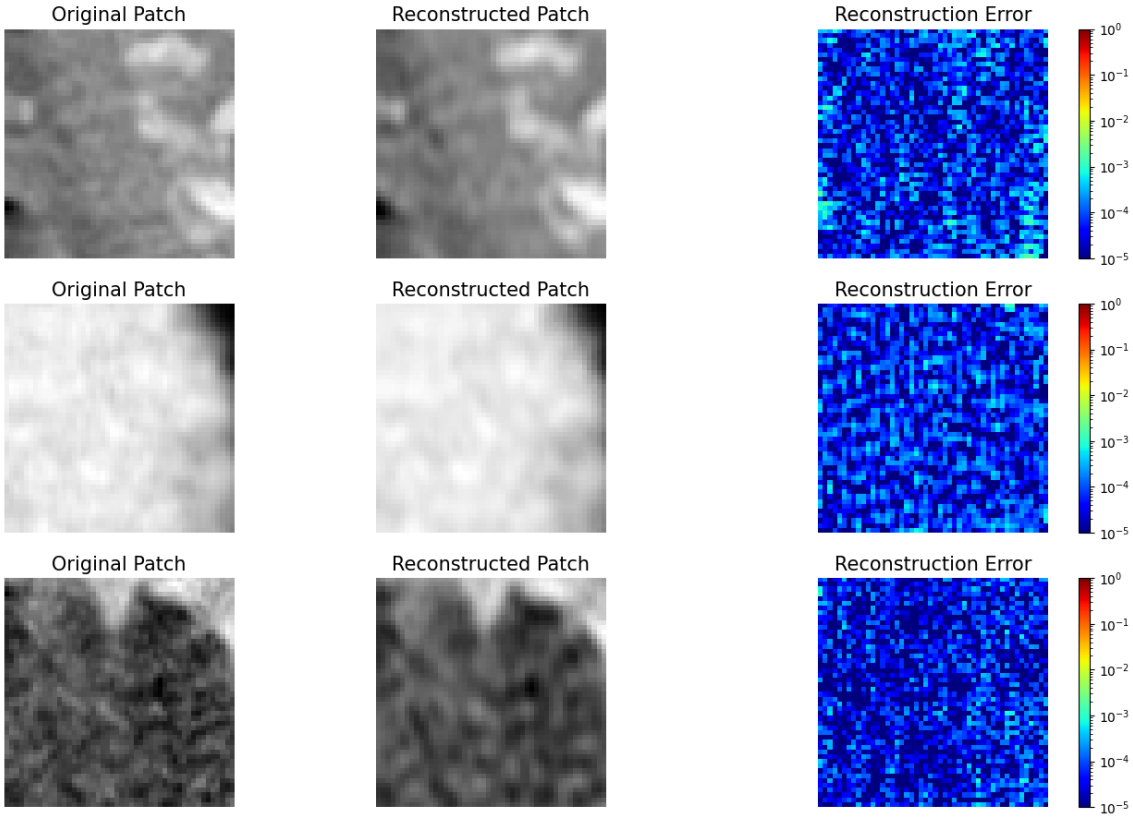


***Figure 60**: Three examples of reconstructed patches by the CAE randomly selected from the test set. Each row contains the 48x48 original input patch (right), the 48x48 reconstructed patch (middle), and the reconstruction error (left) calculated from the scaled pixel intensities.*

4.4 Classification

In this section the final classification results are presented. Features were extracted for each of the CAE models presented in section 0. The two methods presented in subsection 3.3.2 were utilized for building the final feature vector, i.e., *method 1* and *method 2*. All models were evaluated in 10-fold cross-validation experiments, where feature selection was performed for both the entire dataset and the train set. This was done for both the methods used to build the feature vector, where the results for each method are presented in two distinct tables. Additionally, the *method 2* experiments are presented with an average ROC curve for each of the CAE models. In preliminary experiments the two methods used to assign patches to a specific cluster revealed almost identical results (**Figure 25**), and center selection was therefore used to extract features from all the presented models.

Features were selected according to a combination of mutual information statics (MI) and chi-squared statistics (chi2). Firstly, the K-best mutual information features were selected, and subsequently the K-best chi2 features were selected. The K value was found through experimenting. A combination of MI-statistics and chi2-statistics were used to select features for most of the experiments in classification of features extracted via *method 2*. In classification of features extracted via *method 1*, only chi2-statistics were utilized to select the most relevant features. This is shown in the overview of the classification parameters in **Table 9**, where 'All' indicates that all of the features were chosen based on MI-statistics (i.e., only chi2-statistics were utilized to select features.).

Classification was performed via comparison of two methods, which includes Gaussian Process Classifier (GPC) and K-Neighbors Classifier (KNN). For the Gaussian Process Classifier, either Radial Basis Function (RBF) or Dot-Product (DP) were utilized as kernels. For the KNN, the number of neighbors were explored, and the best was selected for each cross-validation experiment. The feature selection parameters for these two classification methods were the same within each cross-validation experiment.

***Table 9**: Overview of the classification parameters for the CAE models. The parameters are given for the two methods used to build the final feature vector, i.e., method 1 (left) and method 2 (right). Additionally, the parameters are fine-tuned according to the feature selection approach, i.e., train vs whole.*

		Method 1				Method 2			
CAE Model	**Feature Selection**	*K-best mut-inf.*	*K-best chi2*	*KNN neigh.*	*GPC kernel*	*K-best mut-inf*	*K-best chi2*	*KNN-Neigh.*	*GPC kernel*
P16	Train	All	25	3	RBF	200	30	4	RBF
	Whole	All	25	3	RBF	200	25	4	RBF
P20	Train	All	300	3	DP	150	30	3	DP
	Whole	All	300	3	DP	150	20	5	DP
P24	Train	All	20	3	DP	150	30	3	DP
	Whole	All	20	3	DP	150	20	5	DP
P28	Train	All	20	3	DP	150	20	3	DP
	Whole	All	20	3	DP	150	20	5	DP

P36	Train	All	25	3	DP	All	20	3	DP
	Whole	All	30	3	DP	All	20	3	DP
P48	Train	All	30	3	DP	All	30	3	DP
	Whole	All	20	3	DP	200	30	4	RBF

4.4.1 Feature Selection

The characterization of the myocardium as a set of features consists of three steps. First the myocardium is clustered into 500 regions by means of a k-means algorithm. In the next step the information from the CCTA image within each cluster is compressed by application of a CAE. In the final step, the information from the encodings within each cluster is combined to a single vector of features representing the myocardium. We evaluated two different methods for performing this last step, where the first approach (*method 1*) represents our understanding of the approach presented in Zreik et al. and where the length of the final vector of features is associated with the number of encodings in the auto encoder (length: 512). However, initial exploration with such an approach resulted in poor classification. As a consequence, we also present an alternative approach (*method 2*). Furthermore, different approaches for reducing the length of the final vector of features through application of methods of feature selection is presented.

The feature selection method uses information from the input and output to select a subset of features which contain the highest amount of information. As in all other aspects of machine learning, the feature selection should be defined only on input-output data from a train set. However, in order to highlight strengths and weaknesses of the final classification and between the methods for generating the vector of features, *method 1* and *method 2*, we present results also when feature selection was based on the entire dataset. Moreover, to evaluate potential dataset bias in the feature-selection, distributions of the amount of information in the selected subset of features are compared with the distribution corresponding to the full length of features. The distribution of the utilized statistics for a selected CAE model for both methods used to define the feature vector are given in the histograms below. In all these figures the train/test split was derived from a random split in the 10-fold cross-validation experiment of the particular model and the feature selection was performed based on information in the train split.

Method 1

Feature selection performed on features extracted via *method 1* was done by utilizing chi2-statistics only. The distributions of the scores are shown for the CAE model with a patch size of 36x36. The number of features selected are the same as used for the final classification of this particular model which was 25 (see **Table 9**).

The blue bins in **Figure 61** represent the distribution of the chi2-scores computed based on the input-output data in the train set, where the 25-best features were selected (i.e., the 25 features with the highest chi2-scores). The orange bins represent the distribution of chi2-scores for these 25 selected features.

The blue bins in **Figure 62** represent the distribution of the chi2-scores based on input-output data of the test set. Moreover, the distribution of the chi2-scores in the 25 selected features (found based on train set above) is highlighted (orange bins).

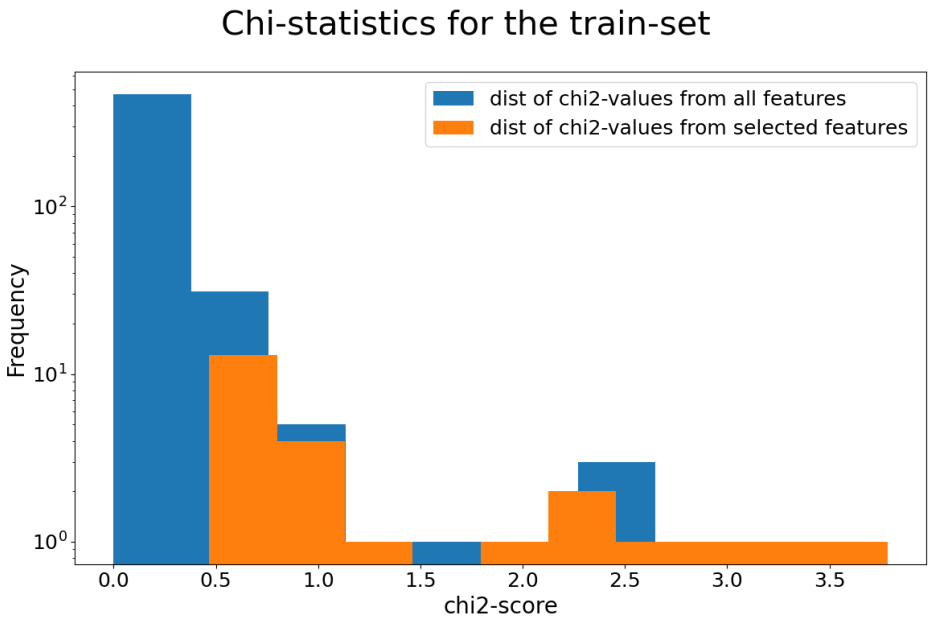


***Figure 61**: Distribution chi2-values for a random k-fold split in the 36x36 patch size experiment. The blue bins represent the distribution of scores computed from the train set, while the orange bins represent the distribution of selected features.*

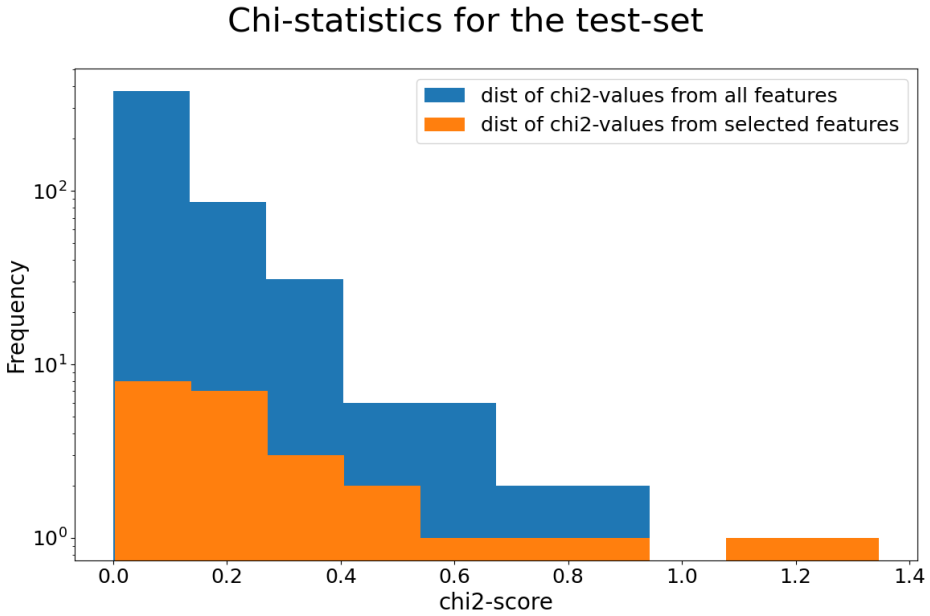


***Figure 62**: Distribution chi2-values for a random k-fold split in the 36x36 patch size experiment. The blue bins represent the distribution of scores computed from the test set, while the orange bins represent the selected features (based on the train set) computed from the test set.*

Method 2

Feature selection performed on features extracted via *method 2* was done by utilizing a combination of MI-statistics and chi2-statistics. The distributions of the scores are shown for the CAE model with a patch size of 20x20. The number of features selected are the same as used for the final classification of this particular model, which is given in **Table 9**.

In **Figure 63**, the blue bins in the histogram to the left represent the distribution of MI-scores computed based on input-output data in the train set, where the 150-best features were chosen. The orange bins in the histogram to the left represent the MI-scores of those particular features. The blue bins in the histogram to the right represent the distribution of chi2-scores of the subset defined by the MI-reduced selection (i.e., 150 features), where the 30-best were chosen. The orange bins in the histogram to the right represent the chi2-scores of those particular features.

In **Figure 64**, the blue bins in the histogram to the left represent the distribution of MI-scores computed based on input-output data in the test set. Moreover, the distribution of the MI-scores of the 150 selected MI features (found from information in the train set) is highlighted in terms of the orange bins in the histogram to the left. The blue bins in the histogram to the right represent the distribution of chi2-scores of the subset defined by the MI-reduced selection (i.e., 150 features). Moreover, the distribution of the 30-selected chi2 features (found from information in the train set) are highlighted in terms of the orange bins.

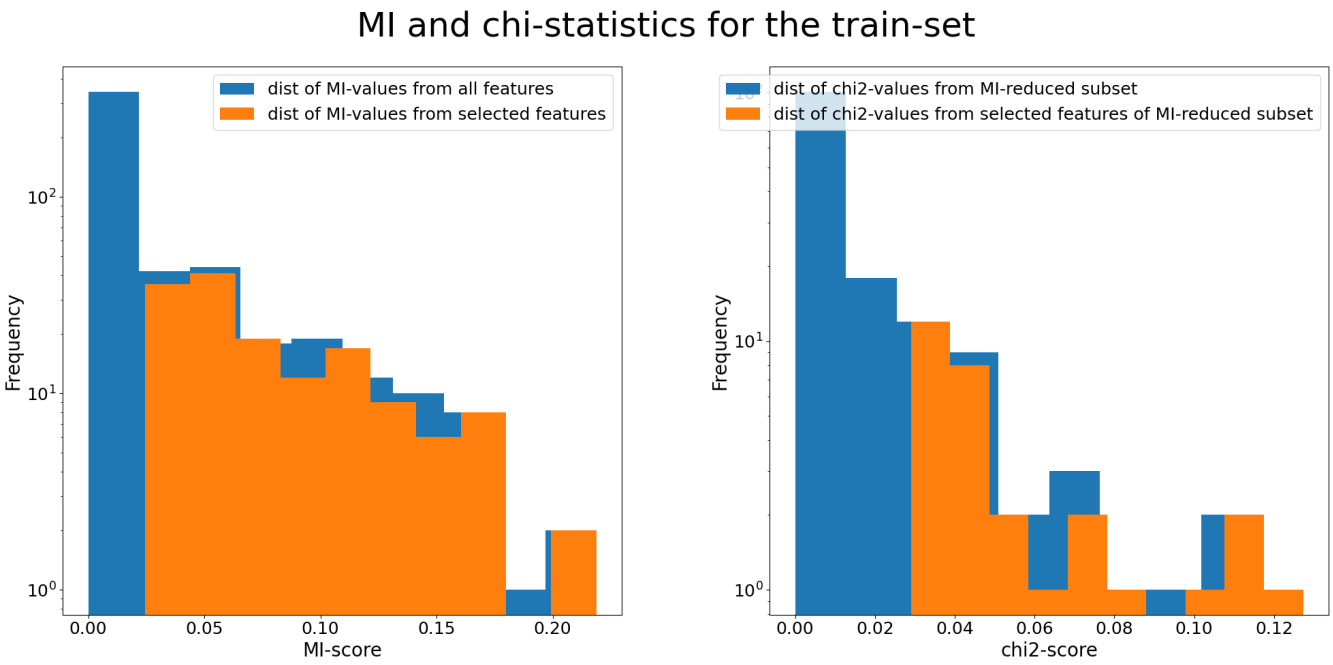


***Figure 63**: Distribution of MI-values (left) and chi2-values (right) for a random k-fold split in the 20x20 patch size experiment. Firstly the 150 best MI-features are selected, and subsequently the 30 best chi2-features are selected from the MI-reduced samples. The blue bins represent the scores computed from the train sett, while the orange bins represent the selected features.*

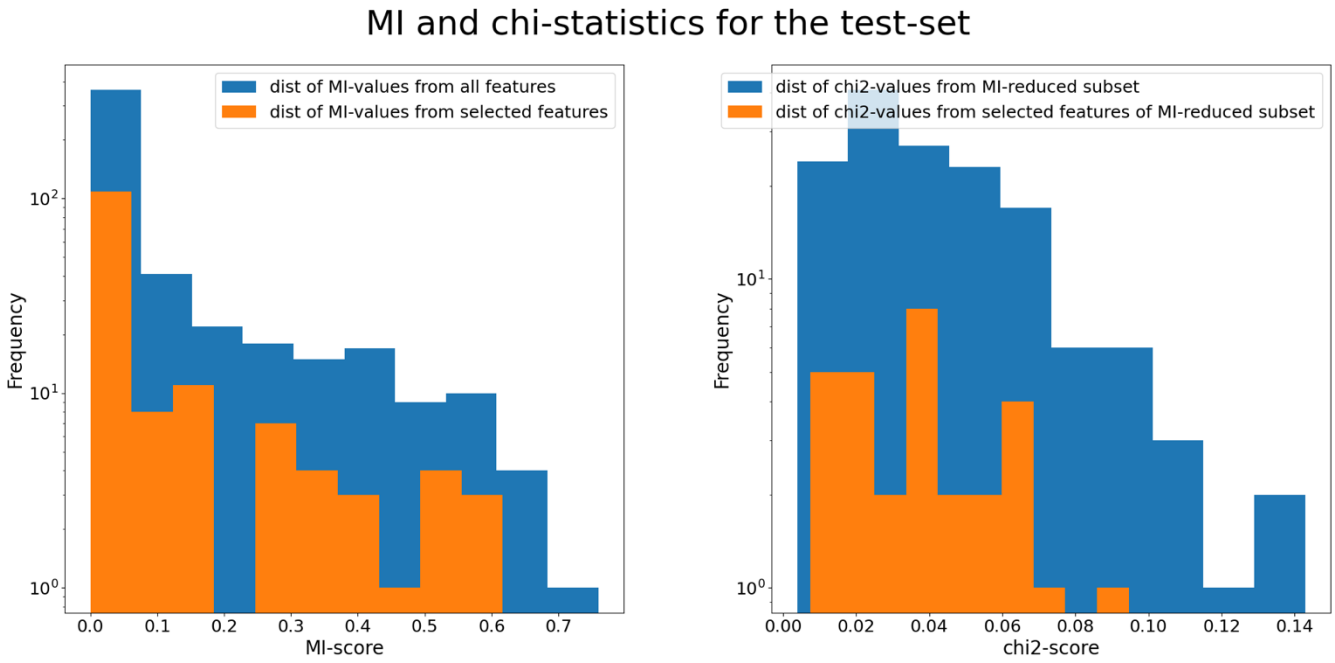


***Figure 64**: Distribution of MI-values (left) and chi2-values (right) for a random k-fold split in the 20x20 patch size experiment. Firstly the 150 best MI-features are selected, and subsequently the 30 best chi2-features are selected from the MI-reduced samples. The blue bins represent the scores computed from the test set, while the orange bins represent the selected features (based on the train set) computed from the test set.*

4.4.2 Results

Method 1

***Table 10**: Overview of the patient classification results using method 1 to build the feature vector. The results are obtained using two different classification methods which include Gaussian Process (GPC) and K-Nearest Neighbors (KNN).*

		FS train set			FS whole dataset		
CAE Model	**Classif. Method**	*AUC*	*Sensitivity*	*Specificity*	*AUC*	*Sensitivity*	*Specificity*
P16	GPC	0.62 (0.23)	0.49 (0.33)	0.64 (0.23)	0.81 (0.18)	0.67 (0.30)	0.81 (0.20)
	KNN	0.51 (0.23)	0.60 (0.31)	0.40 (0.23)	0.68 (0.12)	0.76 (0.28)	0.53 (0.25)
P20	GPC	0.52 (0.24)	0.50 (0.33)	0.54 (0.25)	0.55 (0.26)	0.53 (0.34)	0.57 (0.24)
	KNN	0.57 (0.21)	0.72 (0.23)	0.32 (0.23)	0.54 (0.22)	0.71 (0.30)	0.40 (0.23)
P24	GPC	0.53 (0.23)	0.52 (0.31)	0.53 (0.24)	0.63 (0.25)	0.21 (0.25)	0.83 (0.19)
	KNN	0.47 (0.23)	0.44 (0.37)	0.54 (0.21)	0.47 (0.21)	0.21 (0.21)	0.85 (0.18)
P28	GPC	0.57 (0.21)	0.48 (0.31)	0.62 (0.24)	0.69 (0.21)	0.58 (0.33)	0.70 (0.22)
	KNN	0.54 (0.22)	0.47 (0.32)	0.63 (0.24)	0.65 (0.20)	0.54 (0.31)	0.72 (0.22)
P36	GPC	0.61 (0.23)	0.54 (0.33)	0.62 (0.23)	0.72 (0.23)	0.63 (0.32)	0.70 (0.22)
	KNN	0.61 (0.21)	0.57 (0.32)	0.62 (0.24)	0.65 (0.22)	0.62 (0.30)	0.64 (0.24)
P48	GPC	0.53 (0.24)	0.54 (0.33)	0.50 (0.26)	0.73 (0.21)	0.65 (0.33)	0.61 (0.24)
	KNN	0.63 (0.21)	0.54 (0.30)	0.65 (0.23)	0.68 (0.23)	0.63 (0.33)	0.64 (0.24)

Method 2

***Table 11**: Overview of the patient classification results using method 2 to build the feature vector. The results are obtained using two different classification methods which include Gaussian Process (GPC) and K-Nearest Neighbors (KNN).*

		FS train set			FS whole dataset		
CAE Model	**Classif. Method**	*AUC*	*Sensitivity*	*Specificity*	*AUC*	*Sensitivity*	*Specificity*
P16	GPC	0.70 (0.22)	0.405 (0.31)	0.76 (0.22)	0.94 (0.10)	0.74 (0.28)	0.89 (0.16)
	KNN	0.69 (0.21)	0.602 (0.33)	0.65 (0.25)	0.94 (0.09)	0.86 (0.22)	0.90 (0.15)
P20	GPC	0.69 (0.21)	0.44 (0.30)	0.75 (0.21)	0.84 (0.14)	0.72 (0.28)	0.76 (0.18)
	KNN	0.70 (0.20)	0.69 (0.21)	0.65 (0.24)	0.85 (0.15)	0.78 (0.27)	0.72 (0.27)
P24	GPC	0.69 (0.23)	0.56 (0.32)	0.73 (0.23)	0.95 (0.10)	0.86 (0.23)	0.89 (0.16)
	KNN	0.61 (0.22)	0.40 (0.30)	0.73 (0.23)	0.85 (0.15)	0.64 (0.32)	0.82 (0.19)
P28	GPC	0.69 (0.21)	0.56 (0.33)	0.70 (0.23)	0.86 (0.17)	0.66 (0.30)	0.91 (0.14)
	KNN	0.65 (0.53)	0.54 (0.31)	0.72 (0.22)	0.78 (0.17)	0.33 (0.30)	0.91 (0.14)
P36	GPC	0.64 (0.25)	0.55 (0.32)	0.66 (0.23)	0.88 (0.14)	0.82 (0.26)	0.83 (0.18)
	KNN	0.58 (0.22)	0.56 (0.33)	0.57 (0.26)	0.74 (0.19)	0.74 (0.24)	0.69 (0.14)
P48	GPC	0.68 (0.23)	0.54 (0.33)	0.69 (0.24)	0.95 (0.08)	0.90 (0.19)	0.86 (0.17)
	KNN	0.54 (0.23)	0.52 (0.31)	0.52 (0.25)	0.75 (0.19)	0.78 (0.25)	0.61 (0.25)

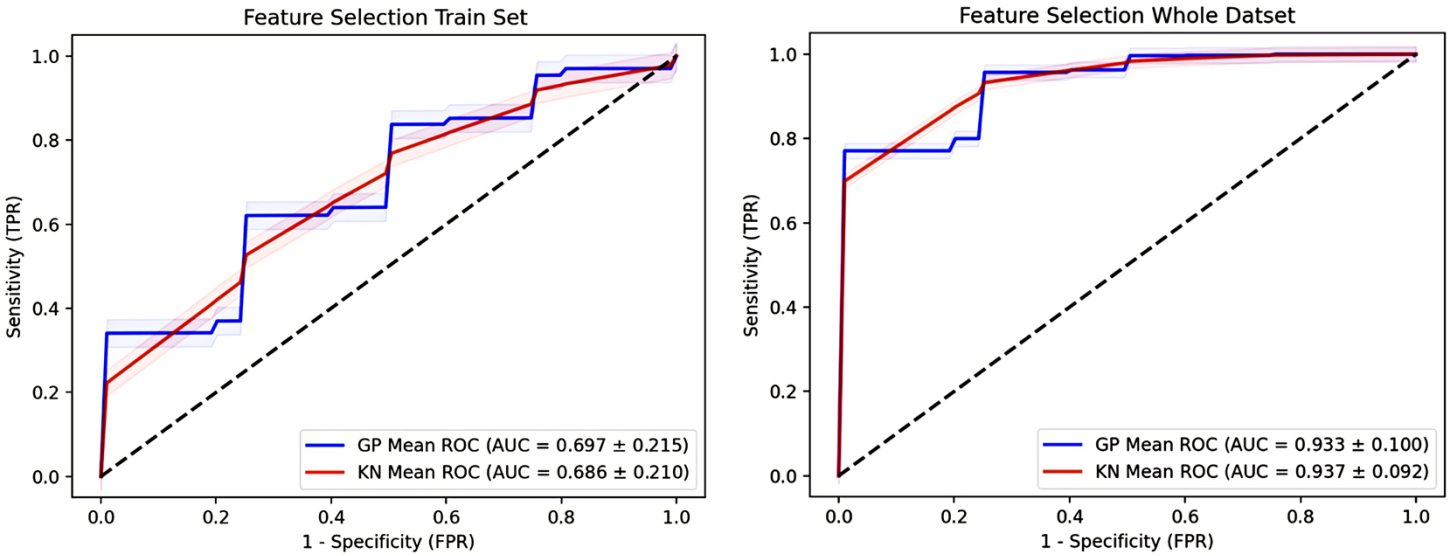


***Figure 65**: Average ROC curves for classification of patients from CAE-model P16 using method 2 to build the feature vector. The FFR cut-off value is set to 0.8 and the shaded area represents a 95 % asymptotic confidence interval of the sensitivity.*

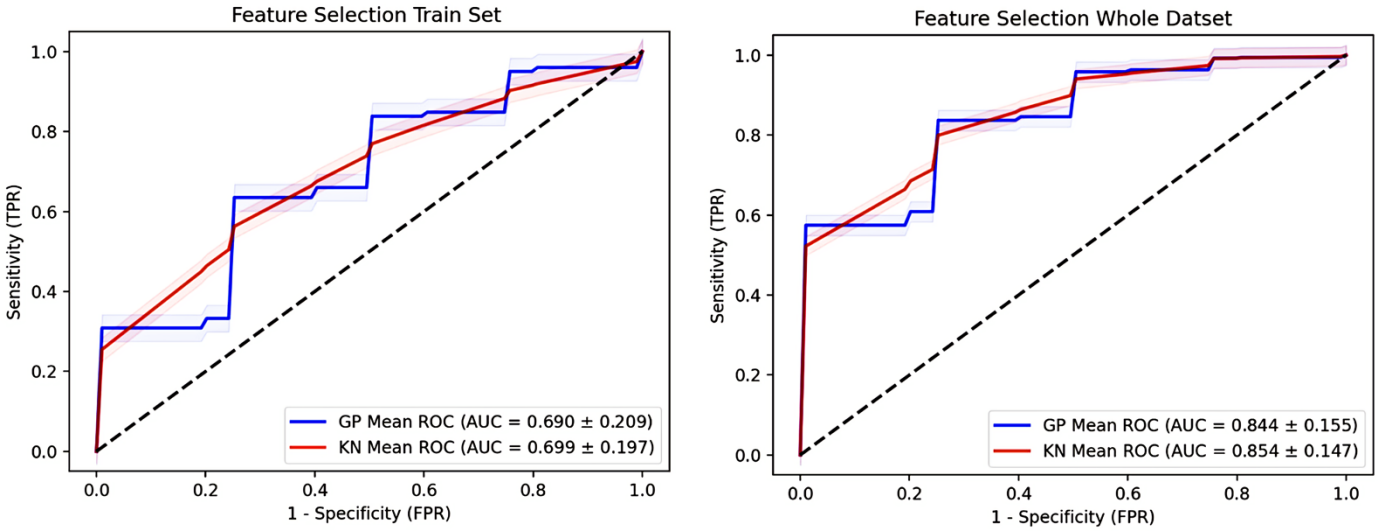


***Figure 66**: Average ROC curves for classification of patients from CAE-model P20 using method 2 to build the feature vector. The FFR cut-off value is set to 0.8 and the shaded area represents a 95 % asymptotic confidence interval of the sensitivity.*

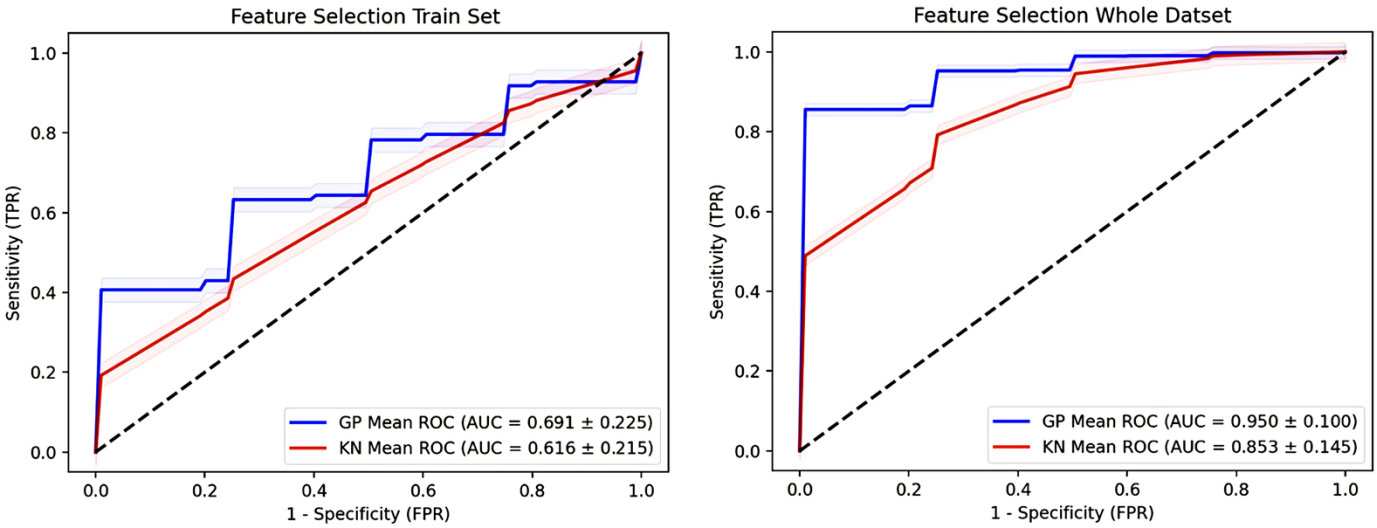


***Figure 67**: Average ROC curves for classification of patients from CAE-model P24 using method 2 to build the feature vector. The FFR cut-off value is set to 0.8 and the shaded area represents a 95 % asymptotic confidence interval of the sensitivity.*

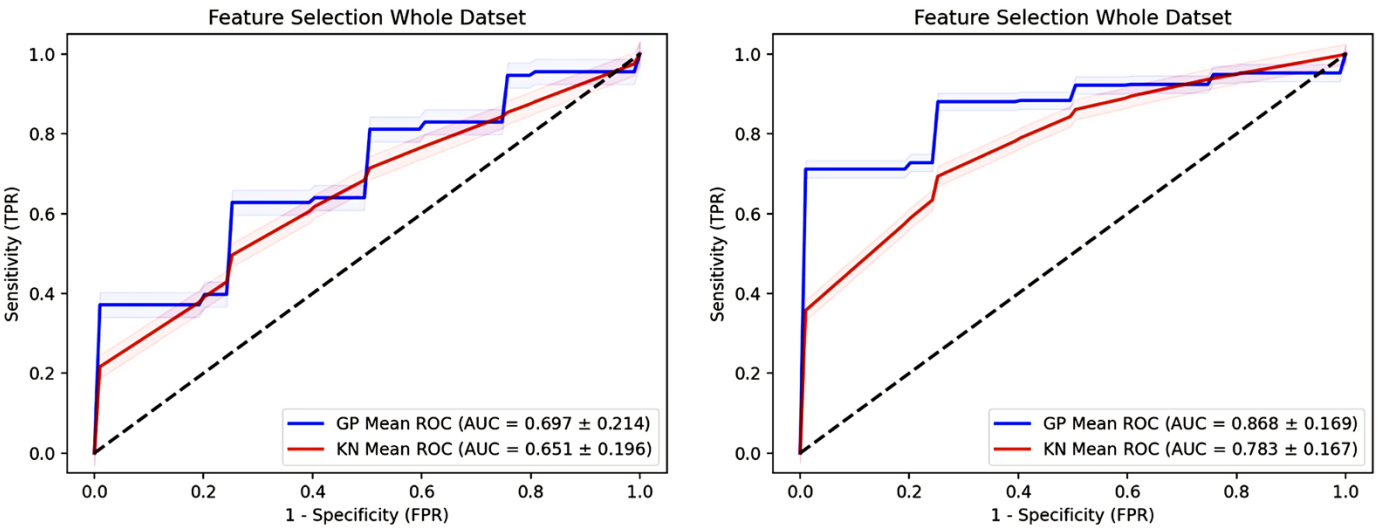


***Figure 68**: Average ROC curves for classification of patients from CAE-model P28 using method 2 to build the feature vector. The FFR cut-off value is set to 0.8 and the shaded area represents a 95 % asymptotic confidence interval of the sensitivity.*

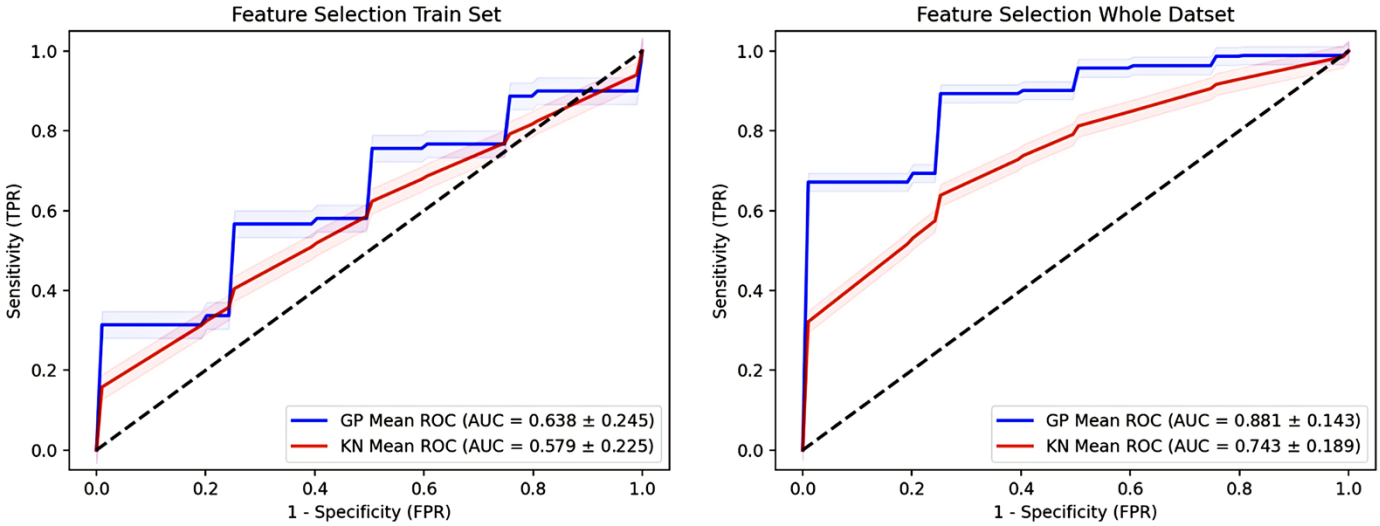


***Figure 69**: Average ROC curves for classification of patients from CAE-model P36 using method 2 to build the feature vector. The FFR cut-off value is set to 0.8 and the shaded area represents a 95 % asymptotic confidence interval of the sensitivity.*

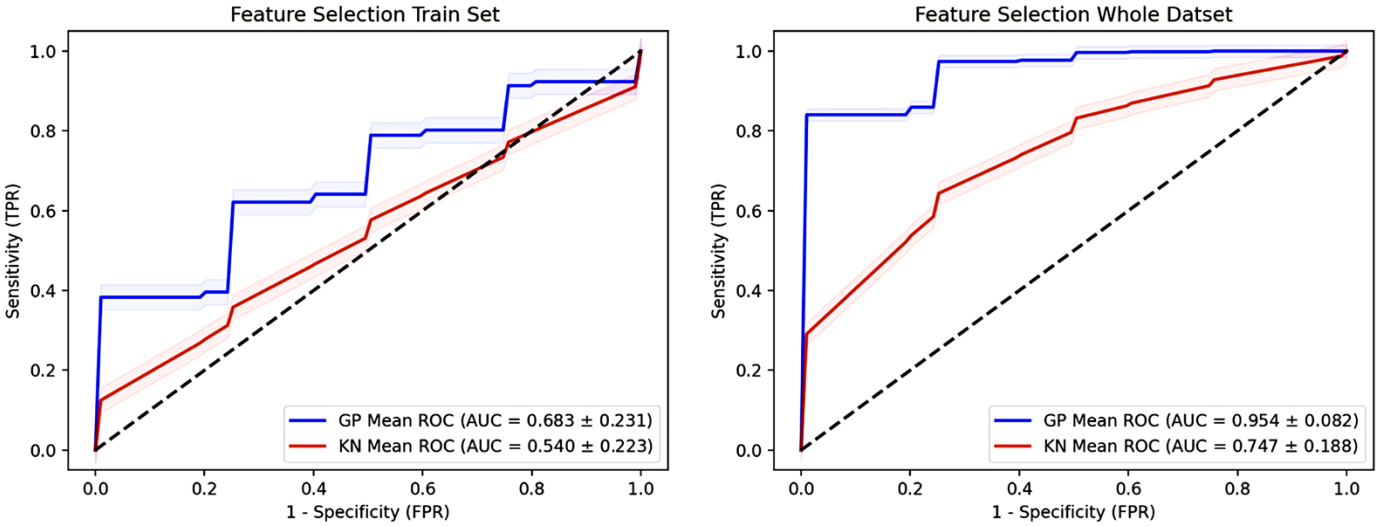


***Figure 70**: Average ROC curves for classification of patients from CAE-model P48 using method 2 to build the feature vector. The FFR cut-off value is set to 0.8 and the shaded area represents a 95 % asymptotic confidence interval of the sensitivity.*

Chapter 5 Discussion

This chapter discusses the various results for each step of the total pipeline presented in Chapter 4. It begins with the CNN automatic segmentation experiments of the LV myocardium, where the effect of CNN architecture and loss function are discussed. Next, the methods used for characterization of the LV myocardium are considered in relation to the final patient classification results.

5.1 Automatic Segmentation

Because of the strong imbalance in the dataset between the two classes (i.e., LV myocardium ~2.9 % and background 97.1 %), pixel accuracy is not a good evaluation metric for this particular segmentation problem. This can be seen through the high accuracy obtained for all experiments. The DSC on the other hand measures the relative overlap between the prediction and ground truth and has the same value for small and large objects. It is therefore reasonable to assume that a high DSC is needed for the clustering results to be accurate. This is also the case for the sensitivity, which measures the proportion of actual positive pixels that were predicted as positive. A lower value for the sensitivity may mean that the actual shape of myocardium isn't captured in the predictions, which again will lead to inaccurate clustering results.

The specificity of myocardium voxels was high for all the experiments (0.996 – 0.997). This can be explained by the class imbalance of the dataset, as the proportion of actual negative voxels (i.e., background voxels) is much higher than actual positive voxels (i.e., myocardium voxels). As a result, the specificity does not provide much information about the actual performance of the different models.

In Zreik et al. an average DSC of 0.914 ± 0.021 was achieved for their CNN-based automatic segmentation of the LV myocardium. Their dataset consisted of 40 samples (manual segmentations) where a 50/50 split was utilized for training and testing. In comparison, the highest average DSC obtained in this thesis was 0.887 which was achieved in the experiment with U-Net Standard and DSL. However, the overall best result for a single model was obtained for the second fold in the experiment with U-Net Standard and Tversky loss, where a DSC of 0.893 was achieved. The highest average sensitivity of myocardium voxels was also obtained for the experiment with U-Net Standard and DSL, which was measured to 0.892. The lowest average DSC of 0.869 was obtained for the experiment with U-Net Dense and TL, with an average sensitivity of 0.881. The single worst prediction was obtained in the U-Net Residual and TL experiment (**Figure 47**), with an DSC of 0.679. Additionally, this experiment also provided the lowest sensitivity of 0.868. These results imply that a more complex architecture does not improve the automatic segmentation results of the LV myocardium. This might be related to the small size of the dataset utilized in this thesis

Furthermore, the validation loss for all the models stopped decreasing at a point between 20-40 epochs, which is shown by the early stopping condition kicking in for all the models. Furthermore, for all the experiments except the U-Net Standard and TL, overfitting was detected. This can be seen from the fitting curves of the different experiments (see **Figure 28**, **Figure 33**, **Figure 38** and **Figure 43**). This indicates that it may be possible to improve the prediction results of the CNN by increasing the size of the dataset.

In Zreik et al. the automatic segmentation was performed in a two-step CNN model. Firstly, the rough shape of the myocardium was detected, and then smoothing was applied on this shape. This method might be more suited for this specific segmentation problem and may explain the overall higher prediction results. However, in Zreik et al. a larger dataset of 40 samples was utilized. Therefore, by increasing the size of the dataset, it may be possible to improve the results and get closer to what has been achieved in Zreik et al. utilizing the method proposed in this thesis. The lower performance of the network (compared to Zreik et. al) could also be an effect of differing image quality or segmentations. The resolution of the CCTA images utilized in this thesis is low compared to images used in the latest work in the medical image segmentation field [8][9][10], which was revealed by the small mean voxel spacing. As a consequence, this might support the need for a "multi-center" study to validate machine learning based analysis of medical imaging.

To sum up, it is reasonable to conclude that fitting on full-size images through extensive data-augmentation is efficient when the resolution of the images is low. Furthermore, no performance gain was detected from introducing a more complex architecture. However, it is worth mentioning that this may be a result of the low resolution of the images and/or the limited size of the dataset.

5.2 Myocardium Characterization

In this section, the methods used for characterization of the LV myocardium are discussed. This includes the CAE, k-means clustering, and the two methods used to build the feature vector, i.e., *method 1* and *method 2*.

5.2.1 CAE

A number of CAE models were trained for each of the patch sizes in preliminary experiments, where different parameters were explored (i.e., architecture, clipping, normalization etc.). The one that provided the best classification results was presented for each patch size. For the smallest patch-sizes, i.e., 16x16 – 24x24, upsampling and downsampling via strided convolutions gave the best classification results. For the two largest patch sizes of 36x36 and 48x48, the best classification results were provided by a combination of downsampling via max-pooling and upsampling via strided convolutions.

For the 28x28 patch size experiment, the best results were obtained for a CAE with two convolutional layers, where upsampling and downsampling was performed via two max-pooling layers and two upsampling layers. Why this approach gave the best results for this

specific patch size is hard to say but can be explained by the weaknesses of *method 2* for building the patient feature vector. This is further discussed in subsection 5.2.3.

For the larger patch-sizes, the CAE was not able to fully reproduce the input-patches when strided convolutions were used for both upsampling and downsampling. This was shown through a relatively high MSE error (i.e., ~ 0.001) and by visual inspection of the reconstructed patches produced by the CAE. This can be explained by the fact that strided convolutions are learnt, as opposed to max-pooling and upsampling which are fixed procedures. For larger patch sizes, it is reasonable to believe that it is harder for the CAE to learn these procedures. These results are not included in the thesis, as only the best model for each patch size was chosen.

Pixel intensity clipping based on the percentiles of intensities of myocardium voxels from the manual segmentations was tested for all the patch-sizes. However, the CAE was only able to reproduce its input-patches for the smaller patch sizes, i.e., 16x16 – 28x28, for such a clipping range. In particular a range of (0, 275) was utilized, including ~ 95 % of the myocardium voxels (**Figure 12**). Experiments both without clipping and with wider clipping ranges than those presented here were performed for the smaller patch sizes, but this resulted in lower classification results in the final step of this thesis. However, the MSE for these models was lower compared to the models presented. This can be explained by the fact that the MSE is measured based on the scaled pixel intensities. This occurred even though the MSE for the CAE-patches were lower for models without clipping. However, this needs to be seen in relation with the fact that the MSE value was calculated based on normalized HU-values (i.e., intensities were normalized by the difference between the maximum and minimum HU value within the scan). Hence, application of different ranges of clipping will have an implicit effect on the MSE-value, where wider clipping ranges will decrease the MSE, since the absolute value of differences between normalized myocardium voxel intensities will be smaller. The same effect can also be seen through visual inspection the CAE results from the two largest patch sizes (**Figure 58** and **Figure 60**) compared to the results from the three smallest patch sizes (**Figure 50**, **Figure 52** and **Figure 54**). For all those experiments the MSE is about the same (i.e., ~ 0.0001), but for the two largest patch sizes where a wider clipping range is utilized the reconstructed patches look blurrier and contain less structural details of the LV myocardium tissue.

5.2.2 Clustering

The k-means algorithm uses random seed initialization to split the LV myocardium into 500 spatially connected clusters, resulting in clusters that have a random order if we compare the obtained clusters across different patients. The numeration of the clusters comparing two or more patients will then be different (i.e., cluster nr. 10 of one patient will not be the same as cluster nr. 10 for another patient). This is a weakness of using k-means to divide the LV myocardium into regions, especially for the features extracted via *method 2*, which is further explained in subsection 5.2.3. Furthermore, as the algorithm is initialized via random seeds,

the clustering results for a specific patient is not deterministic if the experiment is performed multiple times.

5.2.3 Feature Extraction

We evaluated two different methods for performing the feature extraction, where the first approach (*method 1)* represents our understanding of the approach presented in Zreik et. al and where the length of the final vector of features is associated with the number of encodings in the CAE (length: 512). This method uses the clusters to find the highest standard deviation of a single encoding from the vector of encodings obtained by the CAE, which should make it independent of the ordering of the clusters.

We were not able to reproduce the results for FFR-classification obtained in Zreik et. al (AUC ~ 0.74) by directly applying their proposed method. In fact, when the full length of the feature vector was used in classification, we were not able to achieve AUC values much higher than 0.5 (results not shown here). Since in this approach the length of the feature vector is much longer (512) than the number of patients/labels (66) in this thesis, we evaluated methods for reducing the length of the feature-vector through feature selection. However, by studying the histogram in **Figure 62**, generalization of feature selection on an unseen test set seems problematic. This might indicate that the features are somehow dependent on the ordering of the clusters. This could also be a reason for the low classification performance obtained for features extracted via *method 1* shown in **Table 10**.

In the second method (*method 2)* the length of the final vector of features is associated with the number of clusters (length: 500). This method extracts the maximum standard deviation of the whole vector of encodings of the CAE (length: 512) within each cluster to define the vector of features. This method provided the overall best classification results, both when features were selected from the train set and when feature selection was based on the entire dataset. This suggests that a high standard deviation of the 512 encodings produced by the CAE implies presence of abnormal tissue. However, as was also the case for *method 1,* poor classification was obtained when the full-length feature vector from *method 2* was used for classification. Hence application of methods for feature selection was necessary. Furthermore, for *method 2* the order of the features is random, i.e., there is no logical connection between the individual features for the different patients. This is a result of the weaknesses of how the k-means clustering algorithm divides the LV myocardium into regions, which is explained in subsection 5.2.2. Nevertheless, a varying amount of information was measured for the individual features via mutual information statistics and chi squared scores (see **Figure 63** and **Figure 64**). This indicates that good classification results are feasible if relevant features are extracted for the individual patients. This claim is strengthened by the high classification metrics obtained when feature selection was performed on the entire dataset. Furthermore, this is also an indication that it is the association between the features that are important, and not the ordering and value of the specific features. Still, as there is no relationship between the features for the different patients, a pre-prediction of the relevant features for a population based on information from another population will not be possible. Seen in the perspective of

using the train set to reduce the number of features, this will not be anything but a random selection from the original set of features.

The high classification performance of *method 2* when feature selection was performed on the entire dataset might also be a result of other mechanisms that come into play. Considering that the test set was included to extract the most relevant features, it is possible that the selected features might map the input to the target uniquely, without specifically saying something about the significance of the stenosis. In that case, it is possible that the performed feature selection generates a synthetic correlation between the input and target. This is possible as the number of features is relatively high compared to the number of patients.

The overall results obtained for *method 1* were lower compared to *method 2.* This indicates that the features extracted by means of *method 1* contains less information about the "unhealthiness" of the LV myocardium tissue compared to *method 2*. This is also shown in the histograms visualizing the distribution of MI-values and chi2-values. While *method 1* (**Figure 61** and **Figure 62**) revealed to contain some features with higher chi2-values compared to *method 2* (**Figure 63** and **Figure 64**), the distribution of features with chi2-values greater than zero was much higher for *method 2*. It is also important to mention that this comparison is done to the chi2-values of *method* 2-features which is firstly reduced by means of MI-statistics. This reduction was not performed for the features extracted via *method 1*, as the MI-statistics revealed to contain a very low amount of information for this method. Furthermore, the better results obtained via *method 2* strengthens the claim that all the encodings of the CAE might be telling something about the significance of a stenosis. This is the case as this method uses all the encodings of the CAE to compute the features, whereas *method 1* assumes that some of the 512 encodings contain more information than others.

In Zreik et al. however, good classification performance was obtained for *method 1* compared to the results obtained in this thesis. This could be a consequence of the smaller sample size used in this thesis. In Zreik et al. a sample size of 126 patients was used for the classification, compared to our sample size of 66 patients. Furthermore, the method used in Zreik et al. for characterization of the LV myocardium, i.e., how they used the clusters and CAE encodings to obtain the feature vector, is not explicitly explained. The lower results obtained in this thesis (for *method 1)* may therefore also be a consequence of a misinterpretation of the method used in Zreik et al.

5.2.4 Classification

Multiple classification methods were tested in preliminary experiments, including SVM and Deep Learning. In Zreik et al. SVM was utilized. However, both GPC and KNN outperformed this method, and were therefore used instead. GPC and KNN revealed overall very similar results for all experiments. However, a noticeable difference was revealed for features extracted via *method 2*. For feature selection on the train set, GPC seemed to outperform KNN for the larger patch sizes (i.e., 24x24 – 48x48). For the two smallest patch sizes (i.e., 20x20 and 16x16), best results were obtained for KNN. For feature selection

performed on the entire dataset, similar performance was revealed for the two smallest patch sizes. For the largest patch sizes, best results were obtained from using GPC.

For the features extracted via *method 1*, the performance of the two classification methods seemed more or less random in relation to the patch size when features were selected from the train set. For feature selection performed on the entire dataset on the other hand, GPC outperformed KNN in all experiments for this method.

All the classification results were obtained via 10-fold cross-validation experiments. Smaller k-values were tested in preliminary experiments, but with lower classification results. This is reasonable, as a smaller k-value increases the number of patients in each test-fold, which again means that there are fewer samples used for training each of the k different models. Furthermore, in the experiments where features were selected from the train set, smaller k-values provided very low classification performance. As a smaller k-value means that there are fewer samples available to train each model, it also means that there are fewer samples to select features from. However, using a larger number of folds also reduces the number of samples available for testing of each model. This may lead to predictions where all the samples are classified as positive or negative and may explain the high standard deviation of the classification scores. A 10-fold cross-validation was also utilized for evaluation in Zreik et al. with a much lower standard deviation for the classification metrics. This might be a consequence of several factors, including the difference in available data. Another reason could be the randomness associated with the second method used to define the feature vector (i.e., *method 2)*. It might be reasonable to assume that the negative impact of those weaknesses is strengthened when feature selection is performed (both on the train set and the entire dataset), which could also explain the variable classification performance between the different folds.

Chapter 6 Conclusion and Future Work

In this thesis, a pipeline for prediction of the significance of coronary artery stenosis has been developed. A number of experiments have been completed for the different steps of the pipeline, where both strengths and weaknesses were detected.

For the first step of the pipeline, i.e., the automatic segmentation of the LV myocardium, results comparable with state-of-the-art were achieved. The best results were obtained with the U-Net Standard and DSL with an average DSC of 0.877 across all three folds. The single best model was obtained from for the second fold in the experiment with U-Net Standard and TL, where a DSC of 0.893 were achieved. In Zreik et al. a DSC of 0.91 was reported using a larger dataset compared to the dataset used in this thesis.

For the next two steps, which consider the characterization of the segmented LV myocardium and the final the patient classification, we were not able to reproduce the results presented by Zreik et. al (AUC ~ 0.74) by applying (our understanding of) their method directly (i.e., *method 1*). However, high classification performance was revealed for *method 2* when features were selected from the entire dataset (AUC of ~ 0.90). Furthermore, reasonably good performance was achieved when features were selected from the train set (AUC of ~ 0.70). These results are a good indication that the CAE encodings of myocardium patches might contain information about the presence of abnormal tissue. However, the method used to collect these features is a considerable weakness of this step. This applies particularly to the clustering method, where the numeration of clusters is different for each patient. To improve this step, a new method for dividing the LV myocardium into consistent regions across different patients is required. If such a method is developed, the results presented in this thesis indicate that strong classification performance is achievable. Furthermore, as this will lead to consistency in the ordering features across different patients, the method will not only be applicable to the population utilized, but also to a new population of patients.

In future work, it may be possible to use the automatic segmentations of the LV myocardium and the encodings of the CAE to somehow get boundary conditions for simulations of coronary blood and see how it unfolds in relation to the stenosis. In that case, local features that correlate with how much blood flow a region of the myocardium is exposed to, or how much ability it has to dilate its blood vessels to increase flow when needed, need to be extracted.

Bibliography

[1] Dariush Mozaffarian, Emelia J. Benjamin, Alan S. Go, Donna K. Arnett, Michael J. Blaha, Mary Cushman, Sandeep R. Das, Sarah de Ferranti, Jean-Pierre Després, Heather J. Fullerton et al. «Heart Disease and Stroke Statistics—2016 Update.».

[2] Nico H. J. Pijls, William F. Fearon, William F. Fearon et al. «Fractional Flow Reserve Versus Angiographyfor Guiding Percutaneous Coronary Interventionin Patients With Multivessel Coronary Artery Disease.» 2010.

[3] Nico H.J. Pijls, MD, et al. «Experimental Basis of Determining Maximum Coronary, Myocardial, and Collateral Blood Flow by Pressure Measurements for Assessing Functional Stenosis Severit Before and After Percutaneous Transluminal Coronary Angioplasty.» 1993.

[4] Majd Zreika, Nikolas Lessmanna , Robbert W. van Hamersvelt, Jelmer M. Wolterinka, Michiel Voskuil , Max A. Viergever, Tim Leiner, Ivana Išguma. «Zreik et. al.: Deep learning analysis of the myocardium in coronary CT angiography for identification of patients with functionally significant coronary artery stenosis.» 2018.

[5] Koh JS, Koo BK, Kim JH, Yang HM, Park KW, Kang HJ, Kim HS, Oh BH, Park YB. Relationship between fractional flow reserve and angiographic and intravascular ultrasound parameters in ostial lesions: major epicardial vessel versus side branch ostial lesions. JACC Cardiovasc Interv. 2012 Apr;5(4):409-15. doi: 10.1016/j.jcin.2012.01.013. PMID: 22516397.

[6] Meijboom WB, Meijs MF, Schuijf JD, Cramer MJ, Mollet NR, van Mieghem CA, Nieman K, van Werkhoven JM, Pundziute G, Weustink AC, de Vos AM, Pugliese F, Rensing B, Jukema JW, Bax JJ, Prokop M, Doevendans PA, Hunink MG, Krestin GP, de Feyter PJ. Diagnostic accuracy of 64-slice computed tomography coronary angiography: a prospective, multicenter, multivendor study. J Am Coll Cardiol. 2008 Dec 16;52(25): 2135-44. doi: 10.1016/j.jacc.2008.08.058. PMID: 19095130.

[7] Müller LO, Fossan FE, Bråten AT, Jørgensen A, Wiseth R, Hellevik LR. «Impact of baseline coronary flow and its distribution on fractional flow reserve prediction.» 2019.

[8] Müller, Dominik & Soto-Rey, Iñaki & Kramer, Frank. (2020). Automated Chest CT Image Segmentation of COVID-19 Lung Infection based on 3D U-Net.

[9] Zhang Zhengxin, Liu Qingjie, Wang Yunhong. 2018. Road Extraction by Deep Residual U-Net. IEEE Geoscience and Remote Sensing Letters. doi:10.1109/LGRS.2018.2802944.

[10] Kolařík, Martin & Burget, Radim & Uher, Vaclav & Riha, Kamil & Dutta, Malay.(2019). Optimized High Resolution 3D Dense-U-Net Network for Brain and Spine Segmentation. Applied Sciences. 9. 404. 10.3390/app9030404.

[11] Myocard. [Internet] Available at: https://sml.snl.no/myokard

[12] Figure 1. Niccoli G, Montone RA, Sabato V, Crea F. Role of Allergic Inflammatory Cells in Coronary Artery Disease. Circulation. 2018 Oct 16;138(16):1736-1748. doi: 10.1161/CIRCULATIONAHA.118.035400. PMID: 30354461.

[13] Dr Henry Knipe, D. C. M. M., 2019. www.radiopaedia.org. Available at: https://radiopaedia.org/articles/nifti-file-format

[14] Woods R. Gonzalez, R. Digital Image Processing. Pearson Education International, 2010.

[15] Michael A Nielsen. Neural networks and deep learning, volume 25. Determination press USA, 2015
[16] Bengio Y. Goodfellow, I. and A. Courville. Deep Learning. MIT Press, 2016
[17] Kingma, Diederik & Ba, Jimmy. (2014). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations. doi: arXiv:1412.6980
[18] Clevert, Djork-Arné & Unterthiner, Thomas & Hochreiter, Sepp. (2016). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). doi: arXiv:1511.07289
[19] Glorot, X., Bordes, A. & Bengio, Y.. (2011). Deep Sparse Rectifier Neural Networks. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, in Proceedings of Machine Learning Research 15:315-323 Available from http://proceedings.mlr.press/v15/glorot11a.html.
[20] Nyúl LG, Udupa JK. On standardizing the MR image intensity scale. Magn Reson Med. 1999 Dec;42(6):1072-81. doi: 10.1002/(sici)1522-2594(199912)42:6<1072::aid-mrm11>3.0.co;2-m. PMID: 10571928.
[21] S. Roy, A. Carass and J. L. Prince, "Patch based intensity normalization of brain MR images," 2013 IEEE 10th International Symposium on Biomedical Imaging, San Francisco, CA, 2013, pp. 342-345, doi: 10.1109/ISBI.2013.6556482.
[22] L. Perez, J. Wang, The Effectiveness of Data Augmentation in Image Classification using Deep Learning, (2017). doi: arXiv:1712.04621
[23] Isensee F, Petersen J, Klein A, Zimmerer D, Jaeger PF, Kohl S, et al. nnU-Net: self-adapting framework for U-Net-based medical image segmentation. 2018. doi: http://arxiv.org/abs/1809.10486.
[24] Isensee F, Maier-Hein KH. An attempt at beating the 3D U-Net. 2019;1–7. doi: http://arxiv.org/abs/1908.02182.
[25] Heller N, Sathianathen N, Kalapara A, Walczak E, Moore K, Kaluzniak H, et al. The KiTS19 challenge data: 300 kidney tumor cases with clinical context, CT semantic segmentations, and surgical outcomes. 2019. doi: http://arxiv.org/abs/1904.00445
[26] Figure 3. K-fold cross-validation method. Ren, Qiubing & Li, Mingchao & Han, Shuai. (2019). Tectonic discrimination of olivine in basalt using data mining techniques based on major elements: a comparative study from multiple perspectives. Big Earth Data. 3. 1-18. 10.1080/20964471.2019.1572452.
[27] Max Kuhn, Kjell Johnson. Applied Predictive Modeling, 2013.
[28] Srivastava, Nitish & Hinton, Geoffrey & Krizhevsky, Alex & Sutskever, Ilya & Salakhutdinov, Ruslan. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 15. 1929-1958.
[29] Shen, Xu & Tian, Xinmei & He, Anfeng & Sun, Shaoyan & Tao, Dacheng. (2016). Transform-Invariant Convolutional Neural Networks for Image Classification and Search. 1345-1354. 10.1145/2964284.2964316. doi: arXiv:1912.01447.
[30] Springenberg, Jost & Dosovitskiy, Alexey & Brox, Thomas & Riedmiller, Martin. (2014). Striving for Simplicity: The All Convolutional Net. doi: arXiv:1412.6806
[31] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. doi: arXiv:1603.07285, 2016.

[32] Jordan, J., 2018. www.jeremyjordan.me. [Internett]. Available at: https://www.jeremyjordan.me/semantic-segmentation/
[33] Litjens G, Kooi T, Bejnordi BE, Setio AAA, Ciompi F, Ghafoorian M, van der Laak JAWM, van Ginneken B, Sánchez CI. A survey on deep learning in medical image analysis. Med Image Anal. 2017 Dec;42:60-88. doi: 10.1016/j.media.2017.07.005. Epub 2017 Jul 26. PMID: 28778026.
[34] Ronneberger O, Philipp Fischer, Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. Lect Notes Comput Sci (Including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 2015;9351:234–41. doi:10.1007/978-3-319- 24574-4
[35] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.
[36] Seyed SSM, Erdogmus D, Gholipour A. Tversky loss function for image segmentation using 3D fully convolutional deep networks 2017. doi: arXiv:1706.05721.
[37] Chi-squared algorithm [Internet]. Available at: https://github.com/scikit-learn/scikit-learn/blob/647fcb1ac13abd8c2eb2554d526f4ad41fee6778/sklearn/feature_selection/_univariate_selection.py.
[38] Witten, I. H., Frank, E.,, Hall, M. A. (2011). Data Mining: Practical Machine Learning Tools and Techniques. Amsterdam: Morgan Kaufmann. ISBN: 978-0-12-374856-0
[39] Ian Witten Eibe Frank Mark Hall Christopher Pal. Data Mining: Practical Machine Learning Tools and Techniques, 4th edition) 2016
[40] Ross, Brian. (2014). Mutual Information between Discrete and Continuous Data Sets. PloS one. 9. e87357. 10.1371/journal.pone.0087357.
[41] Robust Scaler [Internet]. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html
[42] Rasmussen, Carl Edward, and Christopher K. I. Williams. Gaussian Processes for Machine Learning. Adaptive Computation and Machine Learning Series. MIT Press, 2006
[43] Contributors, W. C., 2017. File:BodyPlanes.jpg. [Internett]. Available at: https://commons.wikimedia.org/w/index.php?title=File:BodyPlanes.jpg&oldid=266754236
[44] Magnus Själander, Magnus Jahre, Gunnar Tufte, and Nico Reissmann, "EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure", arXiv:cs.DC/1912.05848, 2019.