Otto-von-Guericke University Magdeburg

Faculty of Computer Science

Master Thesis

Generation of Synthetic Aortic Dissection Datasets

Author:

Kai Ostendorf

Sep 4, 2023

Advisors:

Prof. Dr. Bernhard Preim
Department of Simulation and Graphics
Otto-von-Guericke University Magdeburg

Dr. Gabriel Mistelbauer
Department of Radiology
Stanford University School of Medicine

Ostendorf, Kai:
Generation of Synthetic Aortic Dissection Datasets
Master Thesis, Otto-von-Guericke University Magdeburg, 2023.

Statement of Authorship

I herewith assure that I wrote the present thesis independently, that the thesis has not been partially or fully submitted as graded academic work, and that I have used no other means than the ones indicated. I have indicated all parts of the work in which sources are used according to their wording or their meaning. I am aware of the fact that violations of copyright can lead to injunctive relief and claims for damages of the author as well as a penalty by the law enforcement agency.

Magdeburg, 04. September, 2023 .....

Abstract

Aortic Dissection is a rare, life-threatening disease affecting the aortic wall layers. Research regarding aortic dissection is crucial to better understand the development and causes of the disease and to develop suitable treatment and recovery plans. Part of this research are in-silico studies and computational fluid dynamics (CFD) simulation that need varying datasets to perform their experiments. These datasets are difficult to obtain because of the disease's rarity and concerns regarding the sharing of patient information between institutions. Synthetically generated datasets can be a way to mitigate this problem and help researchers obtain large databases consisting of varying aortic dissection datasets. To this end, we explore approaches for the generation of aortic dissection datasets: a generative adversarial network (GAN), a statistical shape model (SSM), and two autoencoders in the form of a variational auto encoder (VAE) and adversarial auto encoder (AAE). The GAN uses voxelized input data of aortic dissection point clouds, the SSM uses a parameterized representation of the aorta consisting of centerlines and radii, and the autoencoders receive point clouds of the true and false lumen walls. We conclude that the SSM and autoencoder are best suited for the task of generating aortic dissection datasets with some approach-specific drawbacks that can be improved in the future.

Kurzfassung

Die Aortendissektion ist eine seltene, lebensbedrohliche Erkrankung, die die Wandschichten der Aorta betrifft. Forschung im Bereich der Aortendissektion ist von entscheidender Bedeutung, um die Entstehung und die Ursachen der Krankheit besser zu verstehen und geeignete Behandlungs- und Therapiepläne zu entwickeln. Ein Teil dieser Forschung sind In-silico-Studien und Blutflusssimulationen, die zur Durchführung ihrer Experimente unterschiedliche Datensätze benötigen. Diese Datensätze sind aufgrund der Seltenheit der Krankheit und Bedenken hinsichtlich der Weitergabe von Patientendaten zwischen Institutionen nur schwer zu beschaffen. Synthetisch erzeugte Datensätze bieten eine Möglichkeit dieses Problem zu bewältigen und Forschern große Datenbanken mit Datensätzen zur Aortendissektion zur Verfügung zu stellen. Zu diesem Zweck untersuchen wir folgende Ansätze für die Erstellung künstlicher Aortendissektionsdatensätze: ein GAN, ein SSM und zwei Autocoder in Form eines VAE und AAE. Das GAN verwendet voxelisierte Eingabedaten von Punktwolken einer Aortendissektion, das SSM verwendet eine parametrisierte Darstellung der Aorta, die aus Mittellinien und Radien besteht, und die Autoencoder erhalten Punktwolken der echten und falschen Lumenwände. Unsere Ergebnisse zeigen, dass das SSM und die Autoencoder am besten für die Aufgabe geeignet sind, allerdings mit einigen ansatzspezifischen Nachteilen, die in Zukunft verbessert werden können.

Acknowledgments

We thank Kathrin Bäumler (3D and Quantitative Imaging Laboratory, Department of Radiology, Stanford University) for providing the aortic dissection datasets.

Contents

List of Figures

Acronyms

AAE	adversarial auto encoder
CFD	computational fluid dynamics
cGAN	conditional generative adversarial network
CT	computer tomography
DCGAN	Deep Convolutional Generative Adversarial Network
ELBO	evidence lower bound
GAN	generative adversarial network
GPA	Generalized Procrustes Analysis
ICP	iterative closest points
JSD	Jensen-Shannon Divergence
MPR	multi planar reformation
MRI	magnetic resonance imaging
PC	principal component
PCA	principal component analysis
PCS	physical coordinate space
ReLU	rectified linear unit
SSM	statistical shape model
VAE	variational auto encoder

1. Introduction

Vascular diseases are a large research field in medicine. The causes and effects of various vascular diseases have been researched for decades. With the introduction of computers to the field of medicine new approaches could be utilized to further investigate them. Vascular surface models in combination with CFD were and are an important part of understanding the disease complications and progression in the vascular anatomy. To drive research in these areas a massive amount of data is needed. Using patient data always poses data privacy concerns [89]. To mitigate this problem machine learning models have been utilized in the past years to create synthetic databases tailored to the specific disease.

Currently, no large databases of 3D models of aortic dissection exist that could be used by researchers to develop and test new approaches in the domain of aortic dissection. This problem could be alleviated by creating a database of synthetic aortic dissection data having similar properties as data of real aortic dissection. This thesis aims to investigate the suitability of various generative approaches to create such a synthetic database of 3D aortic dissection models. To our knowledge, no comparable effort has been made to date to incorporate a second flow channel during the generation of aortic geometry. Recent and past approaches mainly focus on the generation of 3D vessel geometry with a single flow channel using a wide range of machine learning approaches and statistical shape modeling.

1.1 Aortic Dissection

Aortic dissection is a life-threatening cardiovascular disease [121]. The rare disease with an incidence of 15 in 100,000 patient years and in-hospital mortality of 39% [66] occurs most often in patients between the ages of 65-75 [55]. It is characterized by the separation of the aortic wall layers into two flow channels called the true and false lumen. The aortic wall is comprised of three layers: the intima facing the bloodstream, the media, and the outer adventitia. The separation is caused by a tear in the intimal layer of the aorta or by bleeding within the aortic wall Fig. 1.1. Complications following the separation can be aortic rupture, pericardial tamponade, and branch vessel malperfusion [15].

Figure 1.1: Aortic dissection is caused by a tear in the intimal layer of the aorta (a) or by bleeding within the aortic wall (b). Reproduced with permission from Springer Nature from [81].

The disease can be classified into type A aortic dissection and type B aortic dissection using the Stanford classification Fig. 1.2. Type A aortic dissection affects the ascending aorta and requires immediate surgical repair while type B aortic dissection affects the descending aorta and is treated using endovascular repair or medical therapy [81].

1.2 Synthetic Databases

In recent history computational models have become an important part of clinical research and practice [65]. Part of these computational models are virtual 3D models of the anatomy. These models are used in many medical applications like in-silico studies, active shape modeling, image analysis, education of patients and students, modeling of anatomical function, disease and outcome prediction, and many more.

Creating surface meshes of a dissected aorta starts with acquiring imaging data. Such data usually comes from computer tomography (CT) data. Subsequently, the aorta is segmented, either manually by specifying cross-sections, or semi-automatically [126]. From the segmentation, a 3D model can then be generated. Surface models of aortic dissections are mainly used to perform CFD simulations to assess the effect of wall and dissection flap deformations on hemodynamic features such as false lumen flow rates, pressure differences, and wall shear stress [15]. Models of aortic dissection are characterized by the complex anatomy of the disease and the aorta itself. These models include important landmarks indicating locations of branching vessels, entry and exit tears, and fenestrations, as well as 3D volumes of the outer vessel wall and the dissection flap, and a centerline for each lumen.

Rare diseases that affect changes in anatomy and physiology often do not have sufficiently large databases of anatomical models that allow researchers to properly develop and test new clinical research prototypes. Synthetic databases can help to overcome this problem by providing a large number of artificially generated test subjects created from a small population of real subjects. Such synthetic databases

Figure 1.2: Following the Stanford classification aortic dissection can be categorized into type A (or DeBakey type I and type II) and type B (or DeBakey type IIIa and type IIIb) aortic dissection. Type A affects the ascending aorta while type B affects the descending aorta. Reproduced with permission from Springer Nature from [81].

are designed to share the same statistical and anatomical properties of the underlying population while having a large number of unique entries.

To find suitable generative models for the generation of synthetic 3D aortic dissection datasets an extensive state-of-the-art review has been conducted. The most important ones are presented in the sections Section 2.2, Section 2.3, and Section 2.4. Based on this review, we decided to investigate the following three different models: a GAN using voxelized input data, a SSM using a centerline-based representation of the aorta, and an autoencoder working on point cloud data. These models are based on approaches used to generate 3D point clouds of the heart [6], surfaces of the aorta [94; 110], and surfaces of blood vessels [25] respectively. In the subsequent paragraphs, we briefly describe each model.

A GAN is a deep neural network that performs a two-player minimax game where one deep neural network called the generator creates synthetic data that imitates the real input data. A second deep neural network called the discriminator, is trained to distinguish between the data created by the generator and the real input data. The generator and discriminator are trained separately using the same loss function which the generator tries to maximize and the discriminator tries to minimize. In our approach, the GAN receives voxel representations of aortas as input and is tasked to produce synthetic voxel representations as output.

A SSM is a statistical approach analyzing the geometrical properties of a set of input shapes [51], [23]. A principal component analysis (PCA) is performed on the covariance matrix of the input shape parameterization relative to a mean shape calculated from the set of input shapes. We chose a centerline-based parameterization encoding the aorta and the true lumen using their respective centerlines and radii.

We investigate two types of autoencoders: a VAE and an AAE. A VAE employs a neural network, called the encoder approximating the underlying posterior distribution of the input data with a variational distribution encoding it in a so-called latent space. A second neural network called the decoder is then tasked to generate new synthetic data by sampling from the latent space. An AAE overcomes some limitations a VAE exhibits by extending the model with a third neural network called the discriminator. Our autoencoder models receive point clouds of aortas with points labeled true and false lumen and are tasked to generate synthetic point clouds of aortas with the labeled lumen.

1.3 Machine Learning

This section gives an overview of machine learning, focussing on subjects and terms that arise later in this thesis. Machine learning falls under the more general term of artificial intelligence, which uses computers to emulate the problem-solving and decision-making capabilities of the human mind. Machine learning models try to imitate how humans learn by steadily improving their precision during each training cycle (epoch). These models are supplied with a large amount of data and tasked to find patterns or to make predictions.

Data for the training of machine learning models is split into training, test, and validation data. Training data is used to train the parameters of a model. Validation data is used to tune the hyperparameters of a model, such as the learning rate of a model or the batch size of the data. Test data is independent of the training data while having the same probability distribution. It is used to check if a model overfits the training data by analyzing its performance. If the model performs better on the training data while the error on the test data is increasing overfitting has taken place.

Machine learning is distinguished into supervised learning, unsupervised learning, and reinforcement learning. Supervised learning uses labeled training data, which is used to train models to predict outcomes or classify data. Unsupervised learning tries to detect patterns or to group data into clusters, without the need for human input. During reinforcement learning a model learns over time to perform the right actions through trial and error and receives rewards if a task has been completed successfully.

A sub-field of machine learning is deep learning. It uses artificial neural networks that try to mimic the human mind using neurons combined into numerous stacked layers. Deep learning is especially focused on finding hidden patterns and learning underlying distributions and connections in the data. Each neuron is governed by its activation function and its assigned weights. A weighted sum is calculated based on the input a neuron receives using its weights. The value of the weighted sum is put through the activation function which determines the output of the neuron.

The training of an artificial neural network is performed in epochs. In each epoch, data is fed through the network's stacked layers to calculate the output. This output is compared to the ground truth using a loss function. This loss is then used by an optimizer to adjust the weights of the model before the next epoch is started. This procedure is performed until the model has converged and the loss is minimal without overfitting the training data.

Neurons can be combined in different ways into layers, resulting in distinct layer types with different functionalities. For tasks such as image analysis or 3D geometry analysis, convolutional layers are used. Pooling layers are used to condense information by reducing the size of the data. Normalization layers normalize the output of previous layers that might be scattered across a large interval.

Loss functions are task-specific functions that calculate the error between the output (predicted values) of the model and the actual values of the ground truth. Depending on the task, various loss functions exist, such as the mean squared error for regression problems, cross-entropy for classification problems, minimax GAN loss for GANs, and the KL divergence for autoencoders. The results of the loss functions or parts inside of loss functions are often log-normalized to make the training more numerically stable avoiding underflow when multiplying very small numbers. Optimizers use the calculated loss to update the parameters of the model, minimize the loss in the next epoch, and improve the model's performance.

The activation function governs a neuron's output. The simpler ones are the binary activation function:

$$f(x) = \begin{cases} 0, & \text{if } x < 1 \\ 1, & \text{otherwise} \end{cases} \tag{1.1}$$

and the linear activation function:

$$f(x) = x. \tag{1.2}$$

Non-linear activation functions enable the stacking of layers. If multiple linear layers are stacked the whole network can be described by combining all layers into one using a single linear activation function. Therefore, the last layer of an artificial neural network consisting of stacked layers of linear or binary activation functions would be a linear or binary function of the input of the first layer. Examples of non-linear activation functions are leaky ReLU:

$$f(x) = \begin{cases} 0, & \text{if } x \geq 1 \\ 0.01 \cdot x, & \text{otherwise} \end{cases} \tag{1.3}$$

or the hyperbolic tangent function:

$$f(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}. \tag{1.4}$$

An optimization algorithm that is commonly used in machine learning is gradient descent. It searches in the direction of the gradient $\nabla F(a_n)$ for the local minimum of a differentiable function:

$$a_{n+1} = a_n - \gamma \nabla F(a_n). \tag{1.5}$$

The optimizer takes steps of size γ in the reverse direction of the gradient and updates the weights of the model after every iteration trying to minimize the loss function.

A loss or criterion often used in combination with GANs is the Wasserstein criterion. It helps to make the training of the generative models more stable and achieve better training results by replacing the discriminator with a critic that uses the Earth Mover's (Wasserstein-1) distance. The divergences GANs minimize can be not continuous regarding the generator's parameters but the Wasserstein-1 distance has the benefit of being continuous everywhere and differentiable almost everywhere [44].

For the evaluation of point cloud-based models, commonly used metrics are the Jensen-Shannon Divergence (JSD), the minimum matching distance, and coverage. The JSD is defined as:

$$JSD(P,Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M), \tag{1.6}$$

where $M = \frac{1}{2}(P+Q)$. D_{KL} is the Kullback-Leibler divergence and P and Q are distributions of points in the reference and generated sets, respectively. P and Q are approximated by voxelizing the point clouds into a 28^3 grid [1]. Coverage (COV) helps to detect mode collapse and is measured as the fraction of point clouds in the reference set S_r being matched to at least one point cloud in the generated set S_g [123]:

$$COV(S_r, S_g) = \frac{|\{\arg\min_{Y \in S_r} D(X,Y) | X \in S_g\}|}{|S_r|}. \tag{1.7}$$

Minimum matching distance (MMD) is a measure of quality for the generated point clouds determined by calculating the similarity between two point cloud sets [123]:

$$MMD(S_g, S_r) = \frac{1}{|S_r|} \sum_{Y \in S_r} \min_{X \in S_g} D(X,Y). \tag{1.8}$$

For both, coverage and minimum matching distance, the Chamfer distance (CD) or Earth Mover's distance (EMD) can be used for $D(\cdot,\cdot)$ and are calculated between two sets of points S_1, S_2 [125]:

$$\begin{aligned} CD(S_1,S_2) &= \frac{1}{2}\left(\frac{1}{|S_1|}\sum_{x\in S_1} min_{y\in S_2}\|x-y\| + \frac{1}{|S_2|}\sum_{y\in S_2} min_{x\in S_1}\|y-x\|\right), \\ EMD(S_1,S_2) &= \min_{\phi: S_1 \to S_2} \sum_{x\in S_1} \frac{1}{2}||x-\phi(x)||_2^2, \end{aligned} \tag{1.9}$$

where ϕ is a bijection.

1.4 Structure of the Thesis

This thesis is structured in the following way: First, a review of the state-of-the-art regarding aortic dissection, GANs, SSMs, autoencoders, and visualization techniques is performed. Next, the methods to generate the synthetic datasets are discussed in detail starting with the GAN, followed by the SSM, and autoencoder. Afterwards, the results and limitations of the approaches are discussed and lastly, we conclude with our recommendation regarding the focus of future work.

2. State-of-the-art

This thesis investigates multiple ways to generate synthetic 3D datasets of aortic dissection. The generative models we focus on are GANs, SSMs, and autoencoders. These models are used in medicine for a multitude of applications such as classification, denoising, image-to-image translation, reconstruction, segmentation, and generation [33]. This section starts with an overview of work done regarding aortic dissection Section 2.1, then covers the three models Section 2.2, Section 2.3, Section 2.4, and closes with the visualization background of aortic dissection Section 2.5.

2.1 Medical Background

This section provides an overview of various research performed on the topic of aortic dissection including epidemiological studies and CFD analysis.

Juraszek et al. [58] review recently developed concepts regarding the diagnosis, classification, and treatment of aortic dissection. The discussed new classifications of aortic dissection based on the Stanford classification are TEM (type, entry, malpferfusion) [104] and STS/SVS (Society of Vascular Surgery/Society of Thoracic Surgeons) [71]. Furthermore, treatment options like the frozen elephant trunk approach and thoracic endovascular repair are reviewed.

Rudenick et al. [97] performed in silico, in vitro, and in vivo studies to investigate true lumen and false lumen hemodynamics in chronic aortic dissection. They focused on pressure and wall shear stress as important hemodynamic parameters. They concluded that the combination of measurements from imaging and computational analysis enables researchers to obtain a more detailed view of these hemodynamic parameters.

Bonfanti et al. [12] developed an in vitro and in silico framework to perform personalized studies of type B aortic dissection incorporating personalized pulsatile flow and dynamic boundary conditions. The framework can be used for intervention planning, the assessments of hemodynamic markers, and support the development of medical devices.

Kim et al. [60] discussed the risk of aortic dissection in ascending aortas that are dilated up to a diameter of 55 mm. They carried out a competing risk analysis to assess the risk factors of aortic dissection and aortic rupture on a dataset of 586 individuals. Their findings indicate a correlation between aortic diameter and the age of patients with dilated aortas.

Harris et al. [49] investigated the mortality of 5131 patients shortly after presenting acute type A aortic dissection. Patients were split into two groups: surgically treatable patients and those who could not be operated on because of various circumstances. The mortality of all patients during the first 48 hours was found to be 5.8%. Among patients who were medically treated the mortality was 23.7%, and 4.4% in patients who received or were supposed to receive surgical treatment. They conclude that their study warrants improvements in the recognition of aortic dissection and the transfer and surgical treatment protocols.

Gudbjartsson et al. [43] reviewed acute type A aortic dissection discussing the various surgical procedures available depending on the presentation and aortic pathology. They determine that short and medium-term outcomes are improving. Zhu et al. [131] investigated the development of type A aortic dissection repair over the last 50 years. Finding that the technical complexity of the procedure has increased and long-term survival after it has improved. Sen et al. [101] study the epidemiology of aortic dissection reviewing its risk factors and presentation in a historical context. Their review includes type A and type B aortic dissection focusing on clinical presentation, risk factors, screening and prevention, diagnosis, and management. They conclude that despite advancements in treatment procedures, morbidity and mortality have only slightly improved.

Braverman et al. [13] examined the clinical features and outcomes of pregnancy-related acute aortic dissection. 29 women who experienced aortic dissection during pregnancy or up to twelve weeks postpartum were studied regarding aortopathy, aortic size, type of aortic dissection, timing of dissection, hypertension, and previous aortic surgery. They found that most aortic dissection occurring during pregnancy is the result of a former undiagnosed aortopathy concluding that monitoring the aorta during pregnancy might lessen the involved risks.

2.2 Generative Adversarial Networks

GANs are one of the first machine learning models focussing on the synthetic creation of 3D datasets. Coming from 2D GANs working on images to create synthetic images of faces or numbers the idea of using a kernel to extract information from them was translated into 3D. 3D input data of various forms was voxelized into a 3D voxelgrid and 3D kernels were used to extract information and create synthetic voxel representations.

The idea and general network structure of a GAN was first proposed by Goodfellow et. al. [39]. The network is based on game theory containing a generator model and a discriminator model. These two models compete in a two-player minimax game to train a generator that can fool the discriminator. In this adversarial process, the generator tries to capture the underlying data distribution of the input data

to generate new synthetic samples with properties based on the input distribution. The discriminator tries to discern if a sample is picked from the real input data or if it is a synthetic sample coming from the generator. The discriminator outputs a confidence value determining how likely the sample is to be created from the generator. During training the accuracy of the discriminator in discerning real from synthetic data is used to compute the loss of the model and update the weights of the generator and discriminator. They evaluated their approach using various 2D datasets including MNIST [27] (handwritten digits), the Toronto Face Database [108] (faces), and CIFAR-10 [64] (color images in ten different classes). After training was completed the generator was able to produce synthetic images based on the original data distribution. They concluded that their proposed model was better or at least competitive in creating synthetic data compared to other state-of-the-art models [8; 9].

Wu et al. [120] proposed a convolutional Deep Belief Network adapting the idea of Deep Belief Networks [53] to 3D voxel data. They represented their input data of 3D geometric shapes as a probability distribution of binary variables on a 3D voxel grid. The model supports object recognition and shape completion from 2.5D depth maps. To train their model the authors created a large database of voxelized CAD models, called ModelNet.

Girdhar et al. [37] propose a generative vector representation that can be used to generate 3D structures in voxel space and to reconstruct 3D models from RGB images. They propose the TL-embedding network for this task which takes voxelized objects and 2D RGB images as input. The voxelized objects are fed into an autoencoder network which condenses them to a 64-dimensional embedding space and reconstructs them back to the voxel representation. The images are processed using a deep convolution neural network which maps them to the same 64-dimensional embedding space. They evaluate their model using average precision as an evaluation metric and compare it to objects that were reconstructed using PCA to access the reconstruction ability of their network. The evaluation of the proposed model shows that the reconstruction ability of the autoencoder outperforms PCA by a large margin and that new objects generated using the autoencoder contain finer details. Furthermore, reconstructing an object from an image gives promising results in capturing details such as reconstructing the right amount of legs of a table.

Wu et al. [119] present a 3D Generative Adversarial Network (3D-GAN) to generate 3D objects and combine it with a VAE [62] (3D-VAE-GAN) to reconstruct a 3D object from a 2D image. The generator of the 3D-GAN consists of multiple 3D convolution layers and produces a $64 \times 64 \times 64$ voxel output from a randomly sampled 200-dimensional latent vector, while the discriminator creates a confidence value deciding if a 3D object is real or synthetic. The structure of the discriminator mirrors that of the generator. To reconstruct 3D objects from 2D images they extend the 3D-GAN with a VAE mapping a 2D image to the latent space using an encoder which consists of multiple 2D convolution layers. Compared to Girdhar et al. [37] and Sharma et al. [102], their approach can generate objects with much higher quality and more fine-grained details. Additionally, their approach outperforms Girdhar et al. [37] in object reconstruction from 2D images.

Baumgartner et al. [5] use Wasserstein GANs [4] to create a feature attribution technique that generates a visual attribution map based on an input image. This attribution map highlights areas in the image that are specific to the class the image belongs to. In the case of images of a brain affected by Alzheimer's disease, the model should learn to understand the different stages of the disease. Wasserstein GANs are characterized by replacing the discriminator with a critic that has no activation function in its final layer. Their training data consists of neuroimaging data of 128x160x112 voxel size which are fed to the map generator network which is based on the U-Net [95]. The critic is modeled as a fully convolutional network that is based on the C3D network [112]. Testing showed that their method produced the most localized disease effect maps compared to other state-of-the-art methods [105; 107; 128].

Fid-Adar et al. [34] train a CNN to classify different types of liver lesions (cysts, metastases, and hemangiomas). To overcome the problem small datasets pose regarding overfitting, they test two approaches to generate more training data: Classic data augmentation (translation, rotation, scaling, flipping, and shearing) and synthetic data augmentation using a GAN. Results are evaluated using 3-fold cross-validation with case separation at the patient level. Classification accuracy increased across all three lesion types using the synthetic data augmentation to train the CNN compared to the classic data augmentation. Furthermore, experts could only distinguish a real from a synthetic lesion 60% of the time.

Rezaei et al. [90] propose a conditional generative adversarial network (cGAN) for the segmentation of brain tumor images. The cGAN consists of a generator that takes 3D MR or CT images and a random vector z as input and outputs a 3D semantic segmentation. This semantic segmentation is fed to the discriminator along with the ground truth, which then outputs a confidence value of whether an object is real or synthetic. Their evaluation determines that their approach is currently not able to outperform ensemble training approaches like [36; 80] that combine multiple networks for the segmentation task.

Danu et al. [25] generate synthetic blood vessel surfaces using GAN and VAE approaches. Input data in the form of surface meshes is voxelized to a size of 128x32x32. The VAE consists of an encoder and a decoder which are both convolutional neural networks. The encoder condenses the voxelized input data to a latent space vector from which the decoder samples to generate new voxelizations. The GAN consists of a generator and a discriminator which are made up of fully convolutional layers. The generator produces new voxelizations from a random input vector and the discriminator tries to distinguish if a voxelization comes from real data or was created by the generator. Their models generate 2D images of vessel-like structures, 3D voxelizations of vessel-like structures, and 3D voxelizations of real anatomical models reconstructed from medical images. During testing on real data, only the GAN produced good results while the VAE failed to converge during the training process.

Rusak et al. [98] generate synthetic MRI volumes using partial volume maps as input for a GAN to enlarge limited data sets for the training of brain segmentation. Partial volume maps distinguish themselves from binary maps by assigning partial affiliation values regarding a single class to a voxel in the image instead of binary maps which assign either a one or a zero. This is beneficial, especially at the borders of tissues where the voxel resolution might not be high enough when only using a

binary map. Their model is based on Pix2Pix [57], which is a cGAN with a generator based on the U-Net [95] and a discriminator based on PatchGAN [57]. The generator contains skip connections between the sampling layers and the discriminator compares patches of images instead of the whole image. Using partial volume maps instead of binary maps results in a higher accuracy of the tissue borders in the synthesized image and even fine changes introduced in the partial volume maps are reflected in the synthetic images.

2.3 Statistical Shape Models

SSMs are used to create statistical models of input data consisting of point locations in space. The statistical model is created using a PCA and is used to approximate shapes based on the calculated principal components (PCs) and eigenvalues. SSMs are used in active shape models and active appearence models which use the approximated shapes to find corresponding shapes in the respective space. The SSM can also be used to just generate new shapes which can be used to populate a synthetic database.

Heimann and Meinzer [51] performed a general review of the techniques required to create and employ 3D SSM in medicine. The review presents the various approaches input data can be represented, how shape models can be reconstructed, how shape correspondence is inferred, how appearance models are built from shape models, and covers how search algorithms use these shape models to find matching anatomy in the input data.

De Bruijne et al. [26] propose various modifications to conventional active shape models of tubular structures. To create the active shape model, an SSM is built from a set of training examples. A training example is described by the coordinates of n landmark points corresponding between shapes. The SSM is extended by modeling the axis and cross-sectional shape deformation separately, resulting in elongated structures being modeled more flexibly. Furthermore, by adding synthetic covariance, a supplementary smooth deformation is introduced, which is decoupled in x, y, and z direction. Lastly, a non-parametric multi-class model replaces the linear one-class gray value model that is usually used in active shape modeling. The shapes are then aligned using Procrustes Analysis using the landmark points to find correspondences and transformed to the tangent space of the mean shape. Performing a PCA on the aligned shapes results in eigenvectors, which represent the modes of shape variation. New shapes can be generated by multiplying each mode with a factor and adding them to the mean shape. Their results show that using their modifications shape approximation errors were reduced and the segmentation accuracy was significantly improved.

Bruse et al. [14] employ SSM for the hierarchical clustering of healthy and pathological aortic arches. 60 aortic arch anatomical models were divided into three subgroups of healthy aortic arches, arches post aortic coarctation repair, and arches post arterial switch operation. Their meshes were aligned using iterative closest points (ICP) and Generalised Procrustes Analysis. The aligned meshes were then used as input for the Deformetrica code framework [29] to calculate a mean shape. Using the framework each of the 60 aortic arches could then be parameterized by

subject-specific transformations of the mean shape using a set of deformation vectors β_i. Therefore, the entire shape information of the population could be combined in a matrix D_{Full} using the deformation vectors of each subject. Performing PCA on this matrix and sorting the eigenvectors according to their variance contribution up to a combined variance of 90% results in the shape loading matrix D_{PCA}. D_{Full} and D_{PCA} were then used as input for the hierarchical clustering.

Duan et al. [28] employ statistical shape modeling as part of their framework to reconstruct a 3D representation of aortic dissection from CT images. The framework consists of image preprocessing, aorta segmentation, aortic dissection extraction, and 3D visualization. Shapes are represented by a set of landmark points which are manually marked by experts in each CT image and are used to build an aorta model library. Shapes in the library are aligned using various affine transformations and a shape model is built by performing a PCA on the shape representation and selecting the first n eigenvectors that contain 72% of the variance. The SSM is then used during the segmentation step to automatically adjust the posture parameters of the initial shape of each layer in the CT images.

Liang et. al. [69] used a machine learning approach including SSM to investigate the relationship between shape features and numerically predicted risk of ascending aortic aneurysm subsequently. 25 cases of ascending aortic aneurysm were used as input for the SSM. The surface of the aorta was reconstructed from the CT images and trimmed at the ascending aorta just distal to the sinotubular junction on the proximal end and at the descending aorta on the distal end. Additionally, all branching vessels were removed. To obtain correspondences between meshes, they developed a remeshing method for converting triangle meshes to quad meshes. The quad meshes were aligned using Generalized Procrustes Analysis (GPA) and a mean shape was calculated using PCA. The first three eigenvectors were selected representing 80.1% of the shape variation and 729 shapes were generated. Finite element analyses were performed on the generated shapes and based on the simulation results machine learning-based rupture risk analysis was carried out.

Marzola et al. [75] propose a framework to automatically detect shape correspondences among 3D models of cranial vaults. Point distribution models of the cranial vault are first aligned using their bounding boxes. Next, the internal and external crusts of the cranial vault are separately aligned using an ICP algorithm. Corresponding points are then found using a k-Nearest Neighbor algorithm. Both crusts are then combined in a single matrix on which the SSM is performed.

Cosentino et al. [24] studied the associations between shape and function in a population of ascending thoracic aortic aneurysms. Aortic surface meshes were extracted from 106 patients separated into two groups according to their aorta morphology (tricuspid aortic valve and bicuspid aortic valve). Additionally, meshes from a control group of 19 patients with non-aneurysmal aorta were obtained. The meshes were cut near the brachiocephalic artery, aligned using ICP, and a mean shape was calculated using GPA. The aligned meshes were then used as input for the PCA. Performing strain and flow analysis and statistical analysis they found that the resulting shape modes are related to biomechanical descriptors concerning shape and function.

Hoeijmakers et al. [54] studied the performance of a SSM-based meta-model combined with CFD simulations to approximate the patient-specific pressure drop across the aortic valve. The SSM is created by first aligning the surface meshes of the aortic valve using GPA and subsequently calculating a mean shape from the aligned meshes. A PCA was then performed to calculate the modes of shape variation. The first three modes, a global scaling parameter, and flow rate were used to parameterize the meta-model training using the Genetic-Aggregation meta-model [99] resulting in a meta-model that could capture the behavior of the pressure drop in the aortic valve.

Thamsen et al. [109] used SSM and a patient cohort of 154 subjects to create a synthetic database of 2652 cases of the aortic arch. The aortic arches were represented using centerlines and radii and aligned by minimizing the sum of squared distances of corresponding points. A mean shape was computed and a PCA was performed on the covariance matrix. The SSM was used to generate a synthetic database of >10000 cases on which CFD simulations were performed. When generating new shapes using the SSM the PCs might be weighted in a way that produces unrealistic non-physiological shapes. To eliminate these non-physiological cases a step wise filtering approach was applied resulting in the final database of 2652 cases. Shapes were discarded when the CFD simulations performed on the generated shapes resulted in urealistic pressure values and shapes where discarded when the stenosis degree was below a certain threshold. Agglomerative hierarchical clustering was performed on the real and synthetic cases identifying three clusters having unique properties but no direct correlation with clinically introduced types of the aortic shape.

Catalano et al. [17] developed a SSM of ascending thoracic aortic aneurysm using a cohort of 106 patients equally split into two groups with bicuspid aortic valve and normal tricuspid aortic valve. The 3D surface models of the ascending thoracic aortic aneurysm excluding branching vessels were aligned using ICP and modes of shape variation were calculated using PCA with the first seven modes capturing 90% of the shape variation. New shapes were generated by multiplying the shape modes with a factor and adding them to the mean shape. Performing CFD simulations on the generated geometry shape modes like aneurysm size and tortuosity could be linked to the flow parameters.

Romero et. al. [94] assessed different strategies to increase the sampling efficiency and control statistical properties of synthetic cohorts of the thoracic aorta. The aortic geometry of 26 cases was represented using a cubic B-spline curve of the centerline with the respective radii following the approach of Romero et. al. [93]. A SSM was built by performing a PCA on the input population. New individuals could then be generated by adding a feature vector containing the modes of shape variation to the mean shape. Three sampling strategies were examined including non-parametric sampling using bootstrapping, parametric sampling using a multivariate Gaussian distribution and a uniform distribution, and a GAN. In total 3000 synthetic shapes were created. The sampling strategies were evaluated using data-driven, clinically-driven, and feature space-driven acceptance criteria. Results showed that bootstrapping and Gaussian sampling generate trustworthy cohorts, uniform sampling produces cohorts with maximum variability, and the GAN sampling achieves the best acceptance efficiency in most acceptance criteria.

Thamsen et. al. [110] generated a database of 2652 virtual cases containing aortic morphometry and hemodynamics using statistical shape modeling to capture the 3D shape variability of patients with coarctation of the aorta. They used a centerline-based shape description of the aorta containing the centerline coordinates including the three side branches of the aortic arch and respective radii. Additionally, two SSMs of velocity inlet profiles were created one for patients with tricuspid aortic valves and one for patients with bicuspid aortic valve defect. Using the three SSMs a database of more than 10000 synthetic cases was created by randomizing the weights of the modes of shape variation using a normal distribution. Next, a stepwise filtering approach was employed removing non-physiological shapes from the database. The shapes were filtered according to the radii, branches intersecting with the aortic arch. CFD simulations were performed on the remaining cases generating a cohort of synthetic models containing morphology and hemodynamic information. 2537 of these models were then used to develop a peak systolic pressure gradient prediction model for the coarctation of the aorta that could outperform the current Bernoulli-based approach.

2.4 Auto Encoders

Autoencoders are machine learning models used to learn encodings of the input data, which can later be reconstructed from the encoded representation. This learned encoding space can also be randomly sampled to create synthetic data following the properties of the input data.

The idea of a VAE was first proposed by Kingma and Welling [62] presenting a stochastic variational inference and learning algorithm that can scale to large datasets. They introduced a Stochastic Gradient Variational Bayes estimator that can be used to efficiently approximate posterior inference in almost any model with continuous latent variables and an Auto-Encoding Variational Bayesian algorithm to further handle independent and identically distributed datasets. The trained posterior inference model can be used for tasks such as recognition. A VAE is derived when a neural network is used for the recognition model. Kingma and Welling [63] also created an in-depth introduction to VAEs with important extensions.

Bello et al. [7] introduced 4Dsurvival a supervised denoising autoencoder to learn latent representations optimized for survival prediction of patients diagnosed with pulmonary hypertension. Image sequences of 302 patients were processed using a fully convolutional network trained on anatomical shape priors creating a dense motion, model which is then used as input for the autoencoder. Beetz et al. [6] proposed a multi-domain VAE, combining the modeling of biventricular anatomy and cardiac electrophysiology. As training data, magnetic resonance images and electrocardiograms (ECG) were acquired from 1300 subjects of the United Kingdom Biobank imaging study [85]. Point clouds at the end of systole (ES) and the end of diastole (ED) were computed from the imaging data and, together with the ECGs, used as input for the VAE. The proposed VAE can generate realistic ECGs, ES point clouds, and ED point clouds when randomly sampling from the latent space distribution.

Achlioptas et al. [1] performed a study of various generative models operating on point cloud data. They developed a deep AutoEncoder network to encode and

decode point cloud data. Using the network various other generative models were tested to compare their performance concerning generalization, fidelity, and diversity. These other models included GANs operating on the raw point cloud data, GANs trained on the latent space of the introduced AutoEncoder network, and Gaussian Mixture Models. As evaluation metrics for the comparison of their unordered point sets, the Earth Mover‘s distance [96] and the Chamfer distance were used. Further evaluation metrics for the explored generative models were the JSD [76], coverage, and minimum matching distance. Their testing showed that for each evaluation metric, the Gaussian Mixture Models trained in the latent space of the AutoEncoder network performed best.

Uy and Lee [113] combine PointNet [86] and NetVlad [3] to recognize large-scale places presented as point clouds acquired using scanners such as LiDAR. PointNet is used to extract a high-dimensional local feature descriptor from the input point cloud which is then used as input for NetVlad. NetVlad then generates a global descriptor for the same point cloud. For the training of their model, the authors introduced two new loss functions, called lazy triplet and lazy quadruplet, that differ from the original triplet and quadruplet losses by using the sum instead of the max operator. Testing showed that their network performs better on the newly introduced losses. Comparing their PointNetVLAD to the original PointNet trained for the place recognition task and a state-of-the-art PointNet trained for object classification on rigid models, their new model could outperform both.

Zamorski et al. [125] present an end-to-end solution in the form of an adversarial autoencoder that can learn a latent space representation from 3D point clouds and generate 3D shapes from the latent space. This is done by extending a deep adversarial autoencoder [74] to take 3D point clouds as input and create 3D output. The feature extraction part of the PointNet [86] is employed as an encoder, with the Earth-Mover distance [96] utilized for the distance metric in the loss function, and the Wasserstein criterion [44] supports the adversarial training. Their model enables simultaneous data generation, feature extraction, clustering, and object interpolation. They compared their model, a 3D adversarial autoencoder, to various other models: a raw point cloud GAN, a latent-space GAN, a Gaussian Mixture Model, and a 3D VAE. To evaluate the generative properties of these models, they employ the Jensen-Shannon Divergence [76], coverage, and minimum matching distance. Their approach scored highest regarding reconstruction capabilities and generative capabilities.

Yang et al. [123] present a framework for the generation of 3D point clouds by learning a two-level hierarchy of distributions. The first level is the distribution of shapes and the second level is the distribution of points given a shape. Their approach can generate new point clouds with an arbitrary number of points. This is achieved by first sampling points from a Gaussian prior and then translating them to their new location in the target shape. The continuous normalizing flow framework [21; 40; 91] is the base model. Chamfer distance and Earth Mover's distance are used to measure similarity between point clouds and JSD, coverage, minimum Matching Distance, and 1-NNA are used to evaluate the generative abilities of the model. Their model outperformed all models and was compared to GAN [1], latent-GAN [1], and PC-GAN [68] regarding all metrics, while having the least amount of parameters to

train. Regarding reconstruction ability, their model outperforms AtlasNet [41] and l-GAN [1], using Earth Mover's distance as an evaluation metric.

Cai et al. [16] generate point clouds by performing stochastic gradient ascent on an unnormalized probability density. They state that a point cloud is a set of samples from a distribution of 3D points. Therefore, the sampling procedure can be seen as moving points from a generic prior distribution to high-likelihood regions of the shape by modeling the gradient of log density [70]. As a result, this model can generate point clouds with an arbitrary number of points. They model two distributions: the distribution of shapes, encoding how shapes of the same family vary between each other, and a mechanism to sample a point cloud from the surface of a shape. To generate new shapes, an autoencoder takes point clouds as input and creates a latent space that is used by a latent-GAN [1] to learn the distribution. Using the generator of the GAN, new point clouds can be sampled. The authors evaluate the reconstruction ability of their method using Chamfer Distance and Earth Mover's distance, and the quality of the generation is evaluated using minimum Matching Distance, coverage, and the 1-NNA classifier. Reconstruction ability is compared against AtlasNet [41], the approach of Achlioptas et al. [1] and PointFlow [123] outperforming all of them using Earth Mover's Distance as a metric. The generation ability of the model is compared to r-GAN [1], GCN-GAN [114], TreeGAN [103], and PointFlow [123], producing visually cleaner shapes.

Chen et al. [20] present a variational autoencoder framework for 3D point cloud generation. The framework contains three parts: an encoder, a flow model, and a decoder. To learn detailed local distance relations, they introduce adaptive-weighted pooling in the encoder of the framework to replace the commonly used max pooling or average pooling. The generated point clouds are evaluated with the minimum matching distance, coverage, Jenson-Shannon divergence [76], and 1-NN classifier accuracy [123]. Earth Mover‘s distance and Chamfer distance are used to evaluate the reconstruction performance. Compared to state-of-the-art approaches like AtlasNet [41], Point-Flow [123], ShapeGF [16], and DiffusionPM [73], their framework performs better or at least competitive.

Anvekar et al. [2] introduce Venatus Geometric Variational Auto Encoder which can capture hierarchical local and global geometric signatures in point clouds. To capture local geometric signatures (e.g., the wings, tail, and engine of an airplane) a newly introduced Geometric Proximity Correlator is used. Global geometry is extracted using variational sampling of the latent space. Compared to multiple other state-of-the-art classification approaches their method reaches competitive accuracy, but does not outperform the best-performing approaches like Point Transformer [127] or PCT [46].

Molnár and Tamás [78] introduced an end-to-end learning approach for Time-of-Flight depth images, to learn the representation of discrete 3D points. Their approach transforms the input of 3D point clouds into 2D space using the compact geometric image representation [42] of the input data. Their network is built from a β-VAE [52] containing the transformation and back-transformation from 3D data to the compact geometric image representation. Chamfer distance was used as a metric to compare the output data against the ground truth reconstruction. Regarding reconstruction

losses, their approach outperforms 3D-AEE [125] when noise is introduced into the data.

Feldman et al. [31] presented VesselVAE a Variational Autoencoder to create synthetic 3D meshes of blood vessels. The network takes as input a binary tree representation of the blood vessel 3D geometry, which is then processed using a Recursive variational Neural Network consisting of an encoder and a decoder. Additionally, the decoder contains a Node Classifier and the Features Decoder Multi-Layer Perceptron. The Node Classifier reconstructs the type of the encoded node (leaf node or internal node with one or two bifurcations). The Features Decoder Multi-Layer Perceptron then reconstructs the attributes of each node (position and radius). Depending on the bifurcation right and left decoder Multi-Layer Perceptrons are employed to decode the next encoded vector in the tree. After the decoding the method of Felkela et. al. [32] is used to generate a mesh from the reconstructed tree which is smoothed and has its resolution increased using the Catmull-Clark subdivision algorithm [18]. To evaluate their approach they analyzed the tortuosity per branch, the vessel centerline total length, the average radius of the tree, and the cosine similarity. The distributions of these metrics are consistent when comparing synthetically generated vessels and real vessels. To qualitatively analyze their approach, they compared their generated meshes to Wolternik et al. [118] and Hamarneh et al. [47] determining that their approach can generate realistic blood vessels with branches that are also modeled realistically.

2.5 Visualization

The visualization of 3D geometries of vascular structures acquired from CT and magnetic resonance imaging (MRI) is a large research field spanning a wide array of rendering and shading techniques. The renderings of meshes generated in this work are based on various past contributions to this field of research. Lawonn et al. [67] investigated illustrative visualization techniques of vascular structures in combination with flow regarding the perception of shape and depth. Rezk-Salma et al. [92] tackled occlusion in the visualization of 3D scalar data using opacity peeling, making outer layers transparent to show the information internal structures provide. Straka et al. [106] proposed a focus-and-context visualization technique combining the visualization of meshes obtained from curved planar reformation and direct volume rendering.

Ostendorf et al. [83] investigated shading styles and rendering techniques for the visualization of aortic dissection meshes. The main problem when rendering aortic dissection meshes is the simultaneous rendering of multiple wall layers and the dissection flap which distinguish aortic dissection. Their work concluded that Fresnel shading is a suitable technique to achieve a focus and context visualization of the wall layers and dissection flap like that used in the ghosted views approach of Glaßer et al. [38]. Shading styles with controllable surface roughness, like Oren-Nayar [82] shading for the ambient lighting and Cook-Torrance [22] shading for the specular lighting, were rated best to render the surfaces of aortic dissection meshes.

This thesis parameterizes the 3D geometry of aortic dissection using voxels as input for the GAN Section 3.2, a centerline-based representation using ellipses as

cross-sections as input for the SSM Section 3.3, or point clouds as input for the VAE Section 3.4. Another possible parameterization is presented by Mistelbauer et al. [77] who implicitly model patient-specific 3D meshes of aortic dissection using elliptic Fourier descriptors.

3. Methodology

In this thesis, we investigate the suitability of a GAN, a SSM, and an autoencoder to generate aortic dissection datasets. The work is motivated by the difficulty of obtaining a large number of datasets to perform research interested in the cause and progression of aortic dissection. This is caused by privacy concerns, the rarity of the disease, and institutions not combining their information and resources. To overcome this problem the generation of synthetic datasets might be a way to create the needed databases. We implemented two machine learning models (GAN, VAE) and an SSM in a Python framework developed for this thesis. The machine learning models are implemented using PyTorch.

The GAN is based on the approach of Wu et al. [119]. Using voxelized point clouds of the aorta, the true lumen, and the false lumen as input, the GAN learns to generate new datasets with similar distributions. This results in the output of synthetic datasets containing voxel representations that have similar statistical properties as the input data while being different enough from the input to be considered unique.

The SSM approach uses a PCA to extract PCs and weights that can be combined to generate new synthetic shapes. Therefore, the approaches of Romero et al. [94] and Thamsen et al. [110] are extended to include a second flow channel. We parameterize the aorta and the true lumen using their respective centerlines and radii.

The autoencoders are inspired by Beetz et al. [6] and Zamorski et al. [125]. Beetz et al. encoded 3D point clouds of the heart captured during systole and diastole together with ECG data in a latent feature space using the encoder of a VAE. The decoder generates realistic virtual point clouds of the heart and the ECG data by sampling from the latent space. We use 3D point clouds of aortic dissection, with points labeled either as true or false lumen to, generate synthetic point clouds of aortic dissection. Additionally, we extend the VAE approach with the AAE introduced by Zamorski et al. to investigate if the VAE benefits from the improvements an AAE provides, such as sharper transitions between shapes in the latent space.

The results of this thesis are limited by the nature of machine learning, which is influenced by the size of the training database, the quality of the training data, and the time spent on the actual training of the models. We aim to lay the groundwork for further research by showing which approaches are promising to be investigated in the future to generate a database of synthetic aortic dissection datasets possibly moving in the direction of implicit modeling [77].

3.1 Data Preprocessing

To perform the various machine learning approaches, a database consisting of ground truth, test, and validation datasets is needed. The medical data sets were acquired at Stanford School of Medicine, and approved by the institutional review board (IRB#41660). From a total of 28 datasets two are removed due to the bad quality of their segmentation. Each dataset consists of three centerlines (aorta, true lumen, false lumen) located in physical coordinate space (PCS), a 3D segmentation mask of the true and false lumen located in the multi planar reformation (MPR), and matrices to transform points from PCS to MPR [116].

The process to extract the MPR representation of the aorta is described by Mistelbauer et al. [77] (Section 3.1) who straighten its centerline and center it at the origin of the MPR space. The centerline is then sampled at equidistant intervals (Figure 3.1a) resulting in n centerline points. Perpendicular to these points, the 3D segmentation mask is resampled. This sampling leads to a new voxel mask consisting of n stacked lumen slices (Figure 3.1i) aligned with the corresponding centerline points (Figure 3.1d). The voxel mask contains unique values, representing the background (0), the true lumen (1) and the false lumen (2).

The provided data is then further processed. First, the voxel mask is separated into point clouds representing the true and false lumen (Figure 3.1b and Figure 3.1c). To generate a point cloud from a voxel mask, we only save the voxels containing the index of the lumen of interest while removing voxels containing the background and the other lumen. This results in a set of n 3D point coordinates located in MPR space that represent the respective lumen:

$$S_{lumen} = \left\{ \begin{matrix} (x_0, y_0, 0) \\ \vdots \\ (x_n, y_n, n) \end{matrix} \right\}. \tag{3.1}$$

Because we are currently operating in MPR space the z-coordinate represents the indices of the sampled centerline points. Therefore, the points of each slice can be assigned to the respective centerline point by comparing the z-coordinate.

Some of the datasets contain errors in the 3D segmentation mask, which result in artifacts of mislabeled lumen data. These artifacts occur with some distance to the lumen of interest which helps in removing them by simply calculating the mean distance d_{mean} of the points in a slice to the corresponding centerline point and removing all points that lie outside d_{max}: $d_{max} = d_{mean} + d_{mean}/3$. The removal distance was found through testing performed on the datasets. We compute this maximum distance for each slice separately, as this local outlier removal leads to better

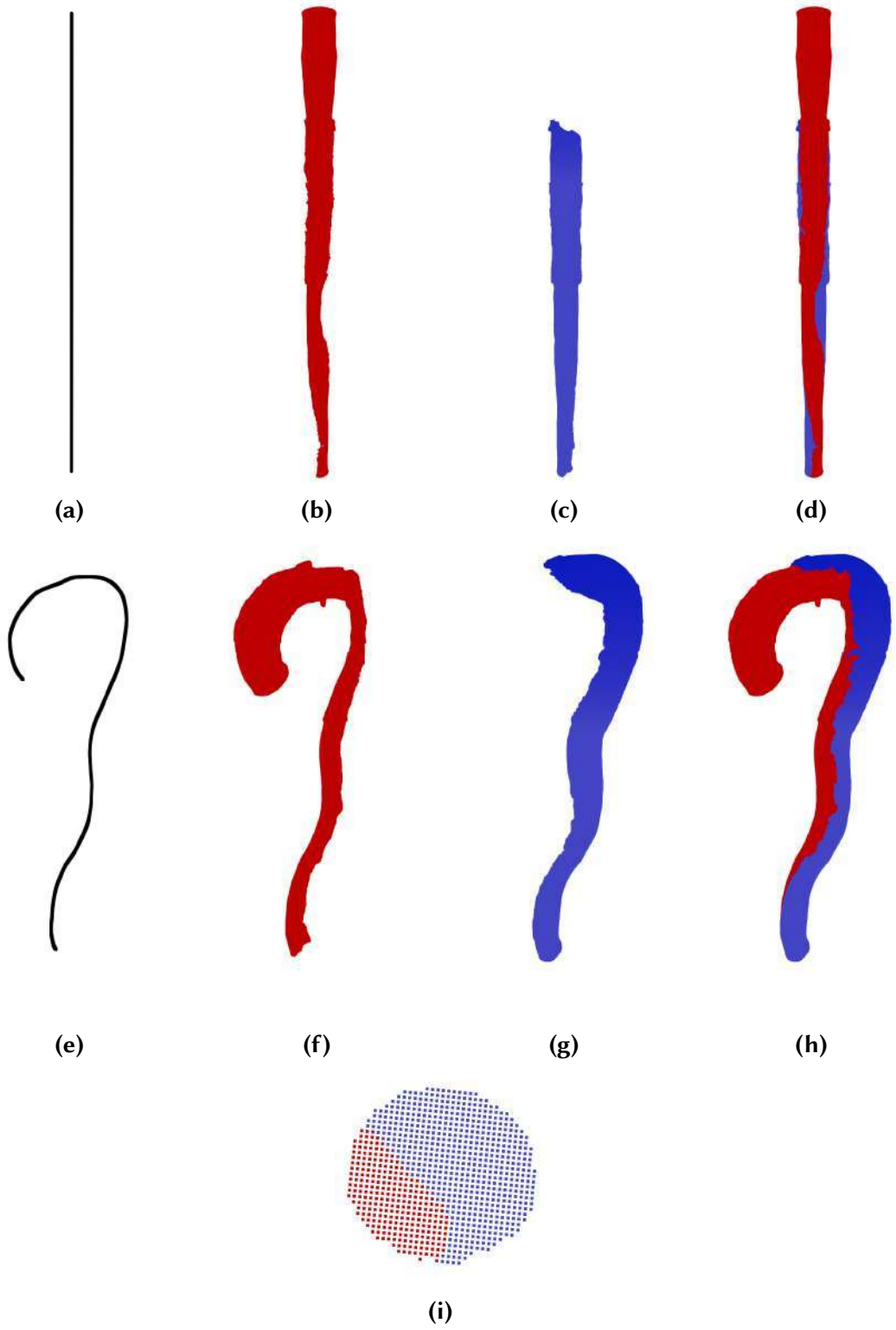


Figure 3.1: Surrounding a centerline transformed to the MPR space (a) are points corresponding to the true lumen in red (b) and false lumen in blue (c). Combining both lumen results in (d). Below each MPR representation the respective PCS transformation of the points is shown. A slice inside of each representation consisting of true and false lumen points is shown in (i). Each slice corresponds to one centerline point.

results than setting a global maximum distance for the entire dataset. Additionally, this approach automates the process of finding the outlier removal distance, which we had to set manually earlier.

We want to directly work on input data located in PCS and generate output data located in PCS, therefore the points located in MPR space $(x_{mpr}, y_{mpr}, z_{mpr})$ have to be transformed to PCS $(x_{pcs}, y_{pcs}, z_{pcs})$. This is done using the provided transformation matrices. Because the points of each slice can be related to their corresponding centerline point each point in the slice can be transformed using the transformation matrix of the corresponding centerline point:

$$\begin{bmatrix} x_{pcs} \\ y_{pcs} \\ z_{pcs} \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ x_c & y_c & z_c & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{mpr} \\ y_{mpr} \\ z_{mpr} \\ 1 \end{bmatrix}. \tag{3.2}$$

The locations of the centerline point location in PCS (Figure 3.1e) are saved in the matrix as x_c, y_c, z_c. a_{11} to a_{34} encode the needed transformations for the true and false lumen points. The transformation from MPR space to PCS is performed for each dataset, resulting in 26 point clouds of true and false lumen (Figure 3.1f and Figure 3.1g) that are then converted to the representations needed for the generative approaches.

3.2 Generative Adversarial Network

As a baseline, we use the approach of Wu et al. [119] who employ a Deep Convolutional Generative Adversarial Network (DCGAN) [88] in their model which in turn is based on a GAN. A GAN is a deep learning model specialized to generate output data that is similarly distributed as the input data. In the example of facial images, a GAN produces similar-looking faces compared to the input data. The network was first proposed by Goodfellow et al. [39], introducing it as a two-player minimax game in which a generator tries to fool a discriminator. The generator G creates synthetic data from a randomized seed which the discriminator D has to distinguish from real data. This is described by the value function $V(D, G)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]. \tag{3.3}$$

$D$is trained to maximize the probability of correctly distinguishing real from synthetic data, where $D(x)$ describes the probability that x came from the real data and $\mathbb{E}_{x \sim p_{data}(x)}$ is the expected value of the real data. G is trained to minimize $\log(1 - D(G(z)))$ where G_z is the random sample produced by the generator from the random noise seed z and $D(G(z))$ is the estimate of the discriminator of the probability of a data sample being real or synthetic. $\mathbb{E}_{z \sim p_z(z)}$ is the expected value over the random noise input to the generator. D and G are both multilayer perceptrons whose weights can be trained.

A DCGAN extends the GAN using 3D convolutional layers and batch normalization in the discriminator and 3D transposed convolutional layers and batch normalization in the generator and rectified linear unit (ReLU) is used as an activation function in the generator, except for the output which uses tanh.

3.2.1 Input Data

To train the DCGAN we need to transform the previously processed datasets (Section 3.1) to a voxel-based representation of size ($64 \times 64 \times 64$). For each dataset, this is done by voxelizing the point cloud data into a ($64 \times 64 \times 64$) dataset, where 1 represents foreground and 0 background. After each dataset is voxelized they are aligned using ICP [11]. ICP is performed as many times as we have input datasets using every dataset as a target and comparing the alignment errors. The best-fitting alignment is then chosen as the input data alignment. Example datasets are shown in Figure 3.2.

3.2.2 Network Architecture

We follow the idea of Wu et al. [119] with some minor adjustments in the setup and training of the model. The generator is set up to receive a 200-dimensional noise vector and tasked to create a voxelized representation of either an aorta, a true lumen, or a false lumen. The noise vector is passed through the generator network consisting of 3D transposed convolutional layers, batch normalization layers, and ReLU layers with a sigmoid layer at the end. The architecture of the generator is shown in Figure 3.3a.

The 3D transposed convolutional layers are used to upsample the size of the input processing the 200-dimensional noise vector into a $64 \times 64 \times 64$ volume. They achieve this by multiplying each input value element-wise by a 3D kernel. The output of each 3D transposed convolutional layer is governed by the number of channels of the input, the kernel size, the padding, and the stride the kernel takes. The kernel sizes (k_s) are ($4 \times 4 \times 4$) with strides (s) of $(1, 2, 2, 2, 2)$ from the first to the fifth layer and padding (p) of size 1 is used. Dilation (d) and output padding (p_{out}) are set to 0. The hyperparameters ensure the correct 3D output size which is governed by the following equation for a cube of size $S \times S \times S$:

$$S_{out} = (S_{in} - 1) \times s - 2 \times p + d \times (k_s - 1) + p_{out} + 1. \tag{3.4}$$

The 3D batch normalization layers normalize the output of the convolutional layers which leads to a faster and more stable training [56]. ReLU is chosen as an activation function because of its good performance on sparse inputs which is ideal for a GAN. The last sigmoid layer is used to produce values between 0 and 1 which is used to set the content of the voxel to either contain data or be empty. From the 200-dimensional noise vector, the output sizes of the different layers are as follows: The first layer outputs a $512 \times 4 \times 4 \times 4$ vector, the second a $256 \times 8 \times 8 \times 8$ vector the third a $128 \times 16 \times 16 \times 16$ vector, the fourth a $64 \times 32 \times 32 \times 32$ vector and the last the final voxelized representation of size $1 \times 64 \times 64 \times 64$.

The discriminator inverts the architecture of the generator and receives either a $1 \times 64 \times 64 \times 64$ voxelization of a real dataset (aorta, true lumen, or false lumen) or a fake voxelization created by the generator. The data is then passed through the discriminator consisting of 3D convolutional layers followed by 3D batch normalization layers and leaky ReLU layers. The 3D convolutional layer is again only followed by a sigmoid layer. The architecture of the generator is shown in Figure 3.3b.

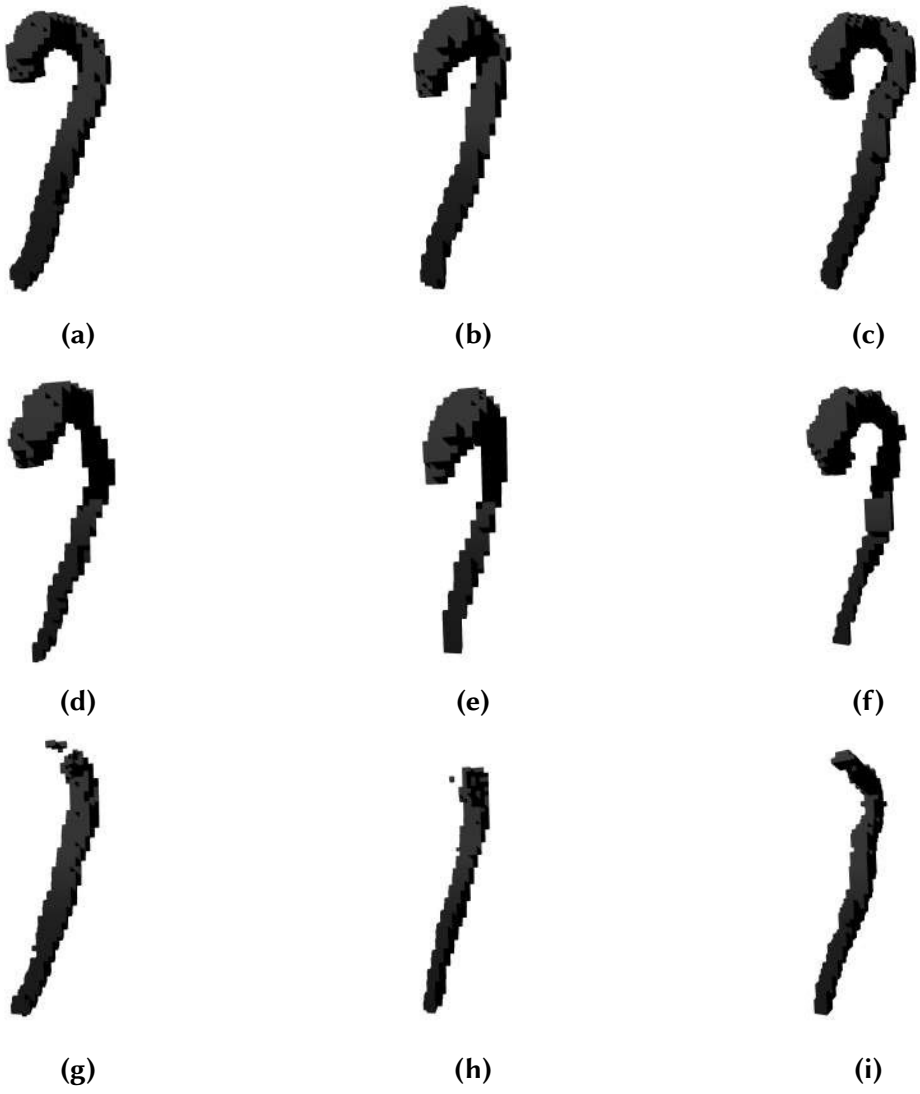


Figure 3.2: Voxelized aortas and lumen (d - i) which are used as input for the GAN approach. A Column presents the aorta (a - c) with the according true lumen (d - f) and the false lumen (g - i).

(b)

Figure 3.3: The generator and discriminator architectures of the GAN. The generator (a) receives a 200-dimensional noise vector as input which is passed through multiple 3D transposed convolutional layers (convT) followed by 3D batch normalization (bn) and ReLU(relu). The last 3D transposed convolutional layer is followed by a sigmoid (sig) layer instead of batch normalization and ReLU. The output of the generator is a synthetic voxelization of either an aorta, a true lumen, or a false lumen. The discriminator (b) receives a voxelization of size $1 \times 64 \times 64 \times 64$ as input which is passed through multiple 3D convolutional layers (conv) followed by 3D batch normalization (bn) and leakyReLU (lrelu). The last 3D convolutional layer is only followed by a sigmoid (sig) layer instead of batch normalization and leakyReLU. The discriminator produces a confidence value whether a voxelization is synthetic or real.

The 3D convolutional layers are used to condense the data to a single value which is the confidence value the discriminator returns to classify a voxelization as either real or synthetic. Just like the 3D transposed convolutional layers the output of 3D convolutional layers is governed by the number of channels of the input, the kernel size, the padding, and the stride of the kernel. The kernel sizes are $(4 \times 4 \times 4)$ with strides of $(2, 2, 2, 2, 1)$ and padding of size 1. LeakyReLU is chosen over ReLU because it fixes the dying ReLU problem. Dying ReLU occurs when a neuron that is part of a ReLU layer becomes inactive and only outputs 0 for any input [72]. This results in the neuron having no further impact on discrimination between real and fake inputs because its weights do not get updated anymore. This is especially problematic when trying to discriminate input [122] which is the case in the discriminator of a GAN. Leaky ReLU helps these inactive neurons to recover back to an active state. The sigmoid layer in the discriminator returns a single value distinguishing if a voxelization was created by the generator or if it was real data. For a $1 \times 64 \times 64 \times 64$ voxelization, the output sizes of the different layers are as follows: The first layer outputs a $64 \times 32 \times 32 \times 32$ vector, the second a $128 \times 16 \times 16 \times 16$ vector the third a $256 \times 8 \times 8 \times 8$ vector, the fourth a $512 \times 4 \times 4 \times 4$ vector and the last a single confidence value between 0 and 1.

3.2.3 Training

The model is trained using the loss function from Equation 3.3 with x being a voxelization of size $1 \times 64 \times 64 \times 64$ and z being the 200-dimensional noise vector. The loss of the generator G_{loss} is dependent on the performance of the discriminator in discerning the synthetic data x_{fake} created by the generator from real data x_{real}:

$$G_{loss} = \log(1 - D(G(z))). \tag{3.5}$$

In each training iteration, the generator is tasked to create synthetic voxelizations for which the discriminator has to output a confidence value. If the discriminator performs poorly and cannot tell if a voxelization is real or synthetic, the output value is around 0.5. In such a case the generator is performing well and the discriminator is underperforming. If the discriminator can accurately discern real from synthetic voxelizations, the discriminator is performing well and the generator is underperforming. The discriminator loss D_{loss} consists of two parts: the accuracy on real data $D(x_{real})$ and synthetic data $D(x_{fake})$:

$$D_{loss} = \log(D(x_{real})) + \log(D(x_{fake})). \tag{3.6}$$

The loss is then used by the optimizer to update the weights of the generator and discriminator to minimize the loss in the next iteration. The training is performed in alternating steps for each batch. First, the discriminator is trained to distinguish the true samples from the fake samples generated by the generator, then the generator is trained to fool the discriminator with the synthetic data samples. As suggested by the original approach of Wu et al. [119], we use the Adam [61] optimizer which is the most common optimizer used in machine learning tasks due to its low memory requirements and fast convergence [50]. We initialize Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e^{-8}$ and set the learning rate of the generator to 0.0025 and that of the discriminator to 0.00001. The discriminator usually learns faster than the generator resulting in bad feedback for the generator. If the discriminator gets too good at differentiating between real from fake data the generator has no basis for what to improve and continues to create fake data that is easily distinguished by the discriminator. For that reason, Wu et al. suggest limiting the weights of the discriminator to only being updated when its accuracy drops below 80% to enable the generator enough time to sufficiently learn to fool the discriminator. Our testing showed that we get the best result when we stop the discriminator training even earlier at 97% accuracy. Furthermore, we use our entire training database of 25 datasets during each epoch.

3.3 Statistical Shape Modelling

SSM uses PCA to generate synthetic shapes from the input data. It is generally performed by calculating a mean shape and relating each input data to this mean shape in a process called registration. Next, PCA is performed on the shape descriptor matrix resulting in the shape modes of the input shapes. The mean shape and the modes of shape variation can then be used to create new synthetic shapes.

3.3.1 Registration and Parameterization

We consider two possible registration approaches: registering each point of the point clouds to other points of the 3D point clouds (pc-to-pc registration) or parameterizing the true and false lumen of the aorta and registering the parameterizations (parameterization-to-parameterization registration). Pc-to-pc registration would result in a more accurate representation of the lumen physiologies than parameterization-to-parameterization registration.

To perform an adequate pc-to-pc registration we select one point cloud as our starting mean shape and then correspond each other point of the other point clouds to the points of our starting mean shape. The starting mean shape is then transformed using generalized Procrustes analysis creating a new shape that minimizes the distances of the points in the other point clouds to the mean shape. This process is repeated until further iterations do not decrease the average distances of corresponding points. Unfortunately, we were not able to adequately determine correspondences between point clouds due to every point cloud having a varying number of points caused by the true and false lumen physiology being very complex and different in shape. This makes establishing 1:1 correspondences impossible. Each point cloud has on average 204000 points for the true lumen and 168000 points for the false lumen. We tried randomly downsampling the point clouds to an even number of points but this resulted in some sections of the physiology being represented by more points than others, which led to worse 1:1 correspondences. Further steps that could be taken to achieve pc-to-pc registration are discussed in Chapter 5.

In the second registration approach, we parameterize the aorta and true lumen with a centerline-based representation using the radii of the cross-section along all curves perpendicular to the centerline curve like recent approaches dealing with statistical shape models of aortas [94; 110; 117]. Each dataset of an aorta consists of the aorta and true lumen centerline points and respective lumen points already located in the MPR space of the centerline of the aorta. To properly parameterize the aortic dissection we only need to use the centerline and radii of the aorta and the true lumen to calculate the SSM. The false lumen can simply be determined by subtracting the true lumen representation from the aorta representation. This approach is also helpful in capturing the mostly concave, half-moon-shaped cross-section of the false lumen which could not be properly represented using only radii of ellipses but would need elliptic Fourier descriptors [77] or comparable approaches which would in turn complicate the SSM.

The centerline points of the aorta and the true lumen are already evenly-spaced along the z-axis of the MPR space along with their respective lumen points perpendicular to the z-axis Section 3.4. Each centerline consists of 431 to 672 centerline points. By considering the centerline points separately we can simplify the problem to two dimensions. For each centerline point, we calculate the ellipse defining the cross-section of the respective lumen at this point. An ellipse can be defined using the following equation:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1, \tag{3.7}$$

with b being the semiminor axis to the semimajor axis a if $b < a$. We calculate a by finding the lumen point $l = [x, y]^T$ in the set of lumen points L that has the maximum distance to the centerline point $c = [x, y]^T$ we are currently interested in:

$$a = \max_i \|c - L_i\|. \tag{3.8}$$

We denote the lumen point with the maximum distance to c as l_a. Additionally, we denote the vector which points from c to l_a as $\vec{v_{cl}}$: To calculate b we need to find the point l_b which is the point with the maximum distance from c when we search along $v_{cl}^{\perp}$. Because we can not be certain to directly hit a point when searching in the positive and negative direction of $v_{cl}^{\perp}$ we consider points in the vicinity of $v_{cl}^{\perp}$. First we calculate $\vec{v_{cl}}$:

$$\vec{v_{cl}} = c - l_a. \tag{3.9}$$

Next, we find a vector $\vec{v_{cl}^{\perp}}$ that lies perpendicular to $\vec{v_{cl}}$ by swapping the x and y coordinates of $\vec{v_{cl}}$ and swapping the sign of the x coordinate:

$$\vec{v_{cl}^{\perp}} = \begin{bmatrix} v_{cl_y} \\ -v_{cl_x} \end{bmatrix}. \tag{3.10}$$

We then define the endpoints e_1 and e_2 of a line containing the vector $\vec{v_{cl}^{\perp}}$ as:

$$\begin{aligned} e_1 &= \vec{v_{cl}^{\perp}} \cdot a + c, \\ e_2 &= -e_1, \end{aligned} \tag{3.11}$$

and can calculate the distance d_l of a lumen point l to this line using the cross product:

$$d_l = (e_2 - e_1) \times (c - e_1). \tag{3.12}$$

This is repeated for every lumen point in the slice. By defining a maximum distance d_{max} from the points in the slice to the line $\vec{v_{cl}^{\perp}}$ we create a subset $Lbmax$ of the points Lb by removing every point that is further away from the line than d_{max}:

$$Lbmax = \{Lb | Lb_i < d_{max}\}. \tag{3.13}$$

The set $Lbmax$ now contains the candidate points for the point l_b which defines the distance b. Lastly, we just need to compute the distance of every point in the set $Lbmax$ to the centerline point and set b as the maximum distance:

$$b = \max_i \|c - Lbmax_i\|. \tag{3.14}$$

This procedure is needed because we cannot guarantee to find a point directly on the line $\vec{v_{cl}^{\perp}}$. Therefore, we have to search in the vicinity of the line for the candidates of b.

We perform this process for the aorta and true lumen centerline and can then represent a single dataset D using the centerline coordinates in PCS of the aorta $c_a = [x_{c_a}, y_{c_a}, z_{c_a}]^T$ and true lumen $ctl = [x_{c_{tl}}, y_{c_{tl}}, z_{c_{tl}}]^T$ and the respective radii of

the aorta a_a, a_b and the true lumen a_{tl}, b_{tl}. This results in a representation of size $N \times (c_a + a_a + b_a + c_{tl} + t_{la} + b_{tl})$ with N being the number of centerline points:

$$D = \begin{bmatrix} (x_{c_a}, y_{c_a}, z_{c_a}, a_a, b_a, x_{ctl}, y_{ctl}, z_{ctl}, a_{tl}, b_{tl})_0 \\ \vdots \\ (x_{ca}, y_{ca}, z_{ca}, a_a, b_a, x_{ctl}, y_{ctl}, z_{ctl}, a_{tl}, b_{tl})_N \end{bmatrix} \tag{3.15}$$

To achieve 1:1 correspondences between the parameterizations of the synthetic dissection datasets, the centerlines need to be represented by the same amount of points. To this end, we use the shortest centerline in our dataset as a baseline and set the number of points of this centerline to be our target for the downsampling. For every dataset, we interpolate the centerline by fitting a cubic spline through the centerline points and sample the curve at evenly-spaced intervals to finally receive centerlines with their respective radii, where each is defined by the same number of centerline points as our shortest centerline. Having established the 1:1 correspondences, the PCA can be performed.

3.3.2 Principal Component Analysis

After creating the representations of the aortic dissection datasets in the form of centerline points and radii for the aorta and the true lumen we calculate a mean shape from the 26 datasets. Every dataset is reshaped from the matrix representation of D (Equation 3.15) with size $N \times 10$ to a row vector $\vec{D}$ with $(N \times 10)$ entries. Using the vector representations of a dataset we can then combine all 26 datasets in a single matrix M:

$$M = \begin{bmatrix} \vec{D}_0 \\ \vdots \\ \vec{D}_{25} \end{bmatrix}. \tag{3.16}$$

The mean shape S_{mean} can then be calculated by:

$$S_{mean} = 1/26 * \sum_{0}^{25} \vec{D}_i \tag{3.17}$$

By subtracting the mean shape S_{mean} from M and scaling each column of the resulting matrix with the standard deviation σ_M of the columns of M we receive the correlation matrix M_{corr}:

$$M_{corr} = (M - S_{mean}) \odot 1/s_M. \tag{3.18}$$

By performing a singular value decomposition on the correlation matrix we receive the PCs (eigenvectors) $\vec{\Lambda}$ and their respective eigenvalues λ. The explained variance v_M of each $\vec{\Lambda}$ can be obtained from λ:

$$v_M = \lambda/|\lambda|. \tag{3.19}$$

We can then sort the principal components according to their explained variance and a synthetic shape S can be reconstructed by multiplying the principal components

$\vec{\Lambda}$ with weights ω randomly sampled in the interval $[-\lambda, \lambda]$ and adding them to the mean shape S_{mean}:

$$S = [\omega_0, \ldots, \omega_i] \begin{bmatrix} \vec{\Lambda}_0 \\ \vdots \\ \vec{\Lambda}_i \end{bmatrix} + S_{mean}. \tag{3.20}$$

We experimented with various bounds for the interval taken from the related work which varied between -3λ and 3λ. Testing showed that the chosen interval of $[-\lambda, \lambda]$ produced the best results for our input data with smaller and larger bounds leading to the generation of too many non-physiological datasets.

3.3.3 Mesh Generation

From a synthetic parameterization of an aortic dissection consisting of the centerlines and radii we then generate surface meshes. For the true lumen wall and the aortic wall, we generate a separate mesh by sampling evenly-spaced points on the radii. This results in the walls now being defined by point clouds which can be used as input for various meshing algorithms such as alpha shapes [30], ball pivoting [10], or screened Poisson surface reconstruction [59]. After testing we determined that the best meshes for our approach can be generated using Poisson surface reconstruction. Examples of the resulting point clouds and generated meshes are shown in Figure 3.4.

3.4 Autoencoders

The third approach we explore is to create synthetic aortic dissection datasets utilizing autoencoders in the form of an AAE [74] and a VAE [61]. Autoencoders are deep learning networks that consist of two main parts: the encoder and the decoder. The encoder creates an encoding of the input data by learning a low-dimensional latent space representation of it. The decoder is trained to reconstruct the input data from the latent space. If the network is sufficiently trained, the latent space can be randomly sampled to generate output data that is similar to the underlying probability distribution of the input data.

Our goal is to have the autoencoders learn the unique properties that define the relations between the true and false lumen of aortic dissection. Then, we can sample the latent space and create new synthetic datasets with realistic true and false lumina. As a baseline, we chose the approaches of Zamorski et al. [125] and Beetz et al. [6]. Zamorski et al. explored the usage of AAEs and VAEs to recreate and randomly sample point clouds of the ShapeNet [19] and ModelNet40 [120] databases, while Beetz et al. explored the creation of synthetic biventricular anatomy and electrocardiograms using a VAE. We combine both approaches by using a slightly modified network architecture from the approach of Zamorski et al. with the loss function introduced by Beetz et al. to create multiclass point clouds. An overview of our approach for both networks is shown in Figure 3.5.

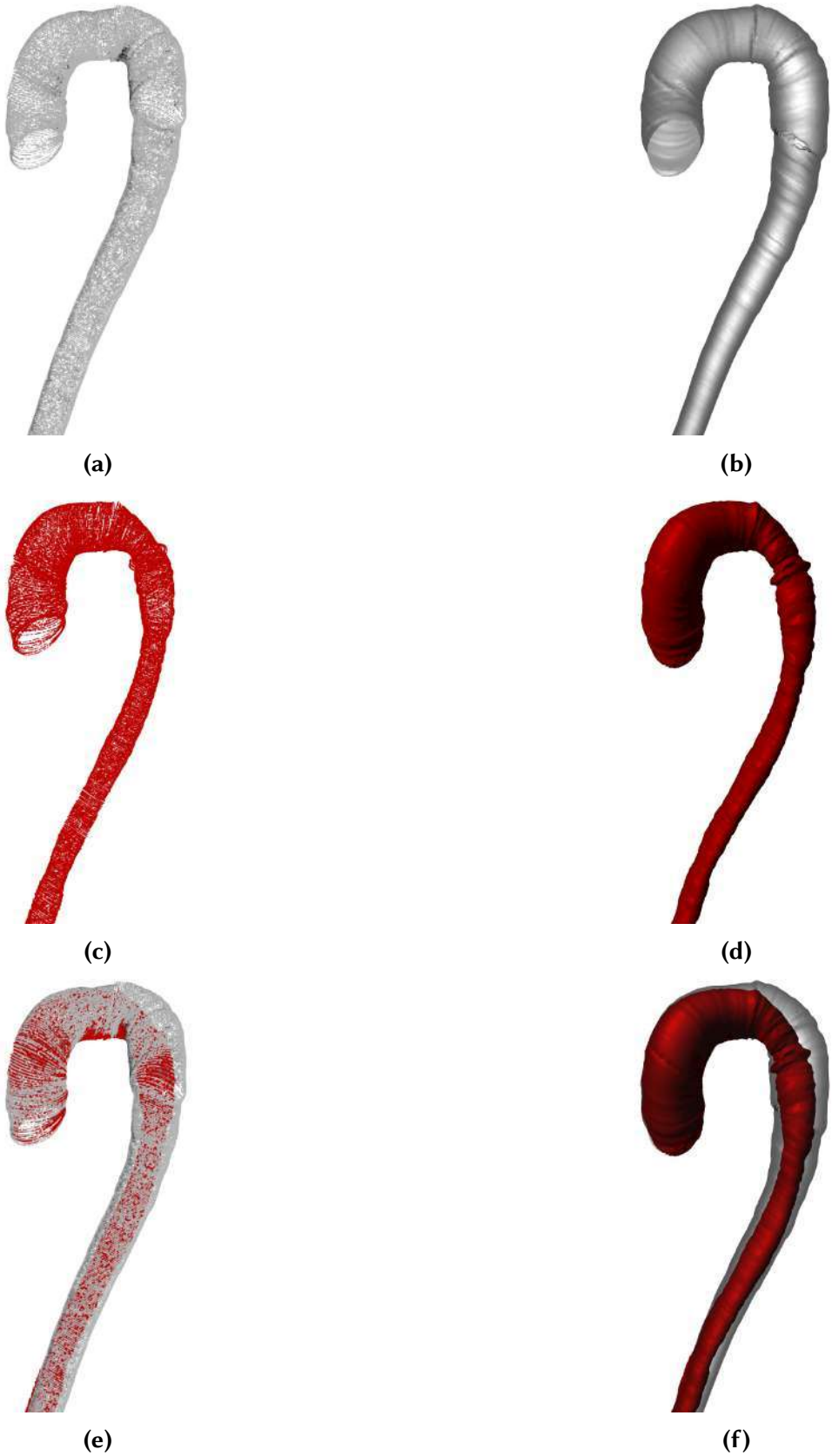


Figure 3.4: From the parameterization of the aorta and the true lumen respective point clouds are generated (a, c). From the point clouds the respective meshes are generated using screened Poisson surface reconstruction (b, d). The complete dataset is obtained by combining both meshes from their respective point clouds (e, f).

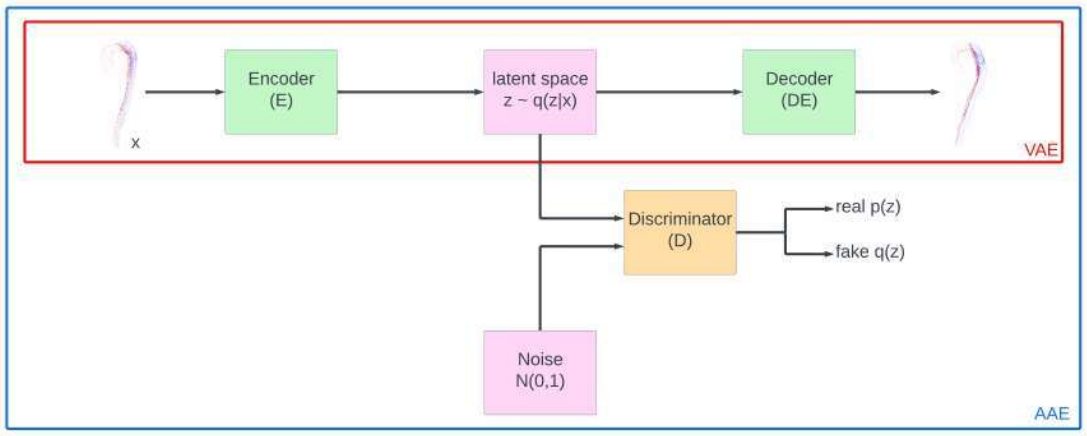


Figure 3.5: Overview of the VAE and AAE network architectures and components shared between both approaches. The VAE only uses the encoder and decoder to construct the latent space z using the Kullback-Leibler divergence. The AAE additionally uses the discriminator to generate the latent space overcoming the limitations the regularization term enforces on the Kullback-Leibler divergence.

3.4.1 Variational Autoencoder

A VAE uses variational inference [35] to create the latent space by approximating the underlying data distribution of the input data. The most important part of the underlying theory of a VAE is the calculation of the lower variational bound.

Using Bayes' theorem, we can calculate the posterior distribution:

$$p_\theta(z|x) = \frac{p_\theta(x,z)p_\theta(z)}{p_\theta(x)}, \tag{3.21}$$

where we have observations x, hidden variables z, and parameters θ governing the distribution. The normalizing constant $p_\theta(x)$ can be computed as:

$$p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)dz. \tag{3.22}$$

The posterior distribution links the data and model but is not easily computable. Therefore, variational inference tries to approximate $p_\theta(z|x)$ using a second distribution $q_\phi(z|x)$ as a proxy for the posterior distribution and fits its parameters ϕ to closely match its performance to that of the target posterior distribution. In the context of VAEs, $q_\phi(z|x)$ can be viewed as the encoder because given an observation x it produces a distribution over the possible values of the latent space z. The decoder is defined by $p_\theta(x|z)$ because using the latent space z produces a distribution over the corresponding values of x.

To calculate the similarity of $q_\phi(z|x)$ and $p_\theta(z|x)$ the Kullback-Leibler divergence D_{KL} is used:

$$D_{KL}(q_\phi(z|x)||p_\theta(z|x)) = \mathbb{E}_{q_\phi}\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]. \tag{3.23}$$

The goal is to minimize $D_{KL}(q_\phi(z|x)||p_\theta(z|x))$ to closely approximate p_θ with q_ϕ. Minimizing the Kullback-Leibler divergence is an intractable problem, therefore, the evidence lower bound (ELBO) is introduced:

$$\begin{aligned}\log p_\theta(x) &= \log \int_z p_\theta(x,z)dz \\ &= \log \int_z p_\theta(x,z)\frac{q_\phi(z)}{q_\phi(z)} \\ &= \mathbb{E}_{q_\phi}\left[\log \frac{p_\theta(x,z)}{q_\phi(z)}\right] \\ &\geq \mathbb{E}_{q_\phi}[\log p(x,z)] - \mathbb{E}_{q_\phi}[\log q_\phi(z)].\end{aligned} \tag{3.24}$$

The variational lower bound of the VAE is obtained by using the ELBO and Kullback-Leibler divergence:

$$L_{\theta,\phi}(x) = \mathbb{E}_{q_\phi}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p_\theta(z)). \tag{3.25}$$

To solve this type of problem, the Monte Carlo estimator is usually used. But when approximating the gradient of the lower bound $L_{\theta,\phi}(x)$ regarding ϕ, the estimator exhibits a very high variance. Therefore, the reparameterization trick is used to express the random variable $\tilde{z} \sim q_\phi(z|x)$ by sampling a noise variable $\epsilon \sim p(\epsilon)$ from a simple distribution $p(\epsilon)$, like the normal distribution $N(0,1)$. This noise variable is then mapped to a more complex distribution $g_\phi(\epsilon, x)$:

$$\tilde{z} = g_\phi(\epsilon, x). \tag{3.26}$$

This distribution can now be used to calculate the gradient:

$$\begin{aligned}\nabla_\phi \mathbb{E}_{q_\phi(z|x)}[f(z)] &= \nabla_\phi \mathbb{E}_{p(\epsilon)}[f(g_\phi(\epsilon,x))] \\ &\approx \frac{1}{L}\sum_{l=1}^{L} f(g_\phi(\epsilon_l, x)),\end{aligned} \tag{3.27}$$

which can be applied to the variational lower bound:

$$\tilde{L}_{\theta,\phi}(x) = -D_{KL}(q_\phi(z|x)||p_\theta(z)) + \frac{1}{L}\sum_{l=1}^{L}\log p_\theta(x, z_l). \tag{3.28}$$

3.4.2 Adversarial Autoencoder

An AAE is a neural network that extends the general idea of an autoencoder by adding a third component to the aforementioned encoder E and decoder DE: the discriminator D. The discriminator in an AAE has a similar task as the discriminator in a GAN (Section 3.2). It distinguishes if a data sample in the latent space comes from real or fake data. AAEs try to solve the limitations VAEs have due to the regularization term requiring specific prior distributions $p_\theta(z)$ for the Kullback-Leibler divergence to be tractable. Additionally, the latent space of an AAE exhibits sharp transitions indicating that the coding space is filled, which is useful when interpolating

on it to generate new samples. An AAE imposes the prior distribution $p_\theta(z)$ on the latent space z by defining an aggregated posterior distribution of $q_\phi(z)$:

$$q_\phi(z) = \int_x q_\phi(z|x)p_\theta(x)dx. \tag{3.29}$$

The AAE uses an adversarial network on top of the latent space to guide $q_\phi(z)$ to match $p_\theta(z)$ and the autoencoder tries to minimize the reconstruction error. The encoder of the autoencoder can be seen as the generator of the adversarial network, which tries to fool the discriminator that the latent space z comes from the true prior distribution $p_\theta(z)$. We choose a Gaussian posterior for the encoder $q_\phi(z|x)$ and use the reparameterization trick (Equation 3.26) to optimize the network.

The training of the adversarial network can be again expressed as a minimax game between the encoder (comparable with the generator in a GAN) and the discriminator with value function $V(E, D)$:

$$\min_E \max_D V(E, D) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(E(z)))]. \tag{3.30}$$

where the encoder and decoder are trained alternately. The loss of the discriminator L_D is defined by:

$$L_D = -V(E, D). \tag{3.31}$$

and the loss of the encoder and decoder $L_{E,DE}$ by:

$$L_{E,DE} = L_{recon} + V(E, D). \tag{3.32}$$

3.4.3 Network Architecture

Following the approach of Zamorski et al. [125] we construct the VAE by parameterizing $q_\phi(z|x)$ with the encoder network E that maps point clouds of aortas to the latent space z. The distribution $q_\phi(z|x)$ is assumed to be normally distributed with: $q_\phi(z|x) = N(\mathbb{E}_\mu(x), \mathbb{E}_\sigma(x))$, with x representing point clouds. Like Beetz et al. [6] and Zamorski et al. [125] we use the PointNet model [86] to generate the same distribution for all possible permutations of the input points in a point cloud set.

PointNet takes unordered sets of points as input and learns to summarize an input point cloud by a sparse set of key points. The network learns several functions $f(x)$ $[f_1, \ldots, f_K]$, where x are the points in a point cloud, to aggregate information from all points learning the properties of the point cloud. $f(x)$ is approximated by a symmetric function $g(h(x))$ where $h(x)$ is a mapping from $\mathbb{R}^N$ to $\mathbb{R}^K$ with $K < N$:

$$f(x_1, \ldots, x_n) \approx g(h(x_1), \ldots, h(x_n)). \tag{3.33}$$

with $f : 2^{\mathbb{R}^N} \to \mathbb{R}$, $h : \mathbb{R}^N \to \mathbb{R}^K$, and $g : \mathbb{R}^K \times \ldots \times \mathbb{R}^K \to \mathbb{R}$. PointNet approximates h using a multi-layer perceptron and g with a single variable function and a max pooling function. We use the output of the max pooling layer to construct the latent space z. Next, the reparameterization trick is applied approximating μ and σ of the normal distribution using two multi-layer perceptrons. This way, the encoder E outputs the latent space z, μ, and σ. The architecture of the encoder is shown in Figure 3.6.

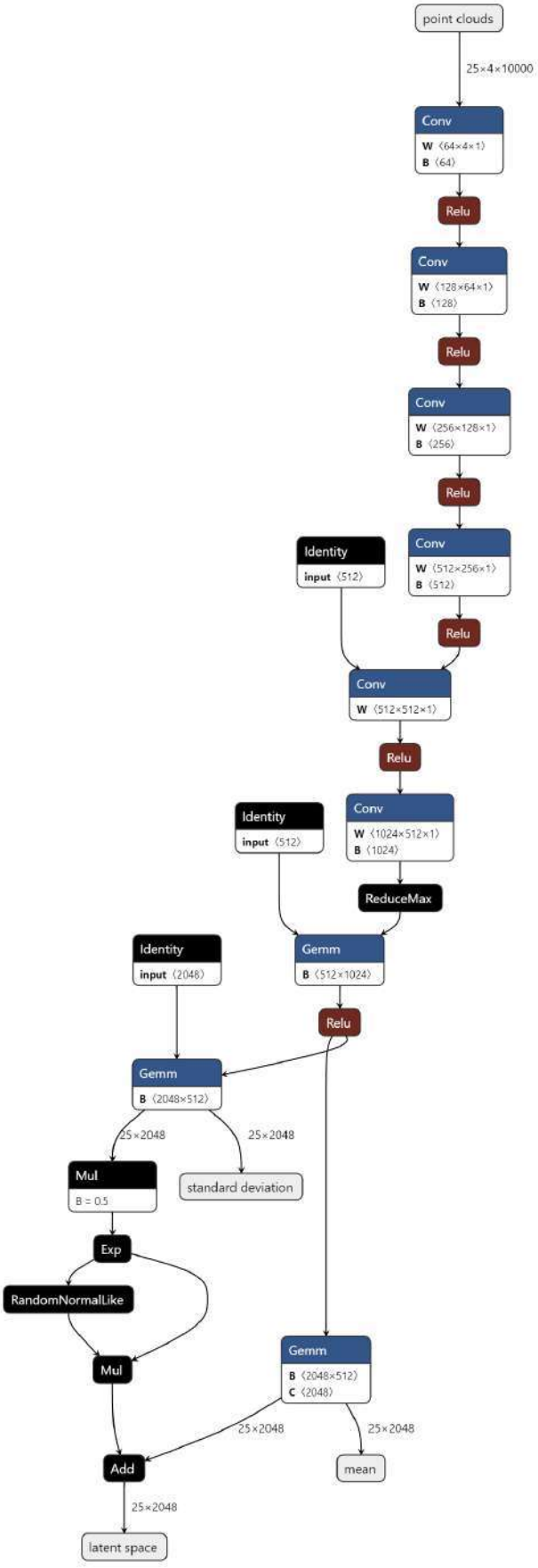


Figure 3.6: The encoder network of the VAE and AAE approach receives 25 point clouds of size 4×10000 points as input with each point containing the 3D coordinates and the classification into true and false lumen. They are passed through a PointNet-like architecture consisting of a multi-layered perceptron build using multiple 1D convolutional layers (Conv) followed by ReLU with a max pooling at the end (ReduceMax). This is followed by linear transformations (Gemm) parameterizing the standard deviation $\sigma(x)$ and mean $\mu(x)$. The mean and standard deviation are then used to approximate the assumed Gaussian distribution of the latent space $z \sim q(z)$ using the reparameterization trick drawing ϵ from a normal distribution (RandomNormalLike). We chose the latent space to be of size 2048. In the AAE the encoder only returns z. In the VAE the encoder additionally returns μ and σ to calculate the Kullback-Leibler divergence L_{KL}.

The decoder DE tries to reconstruct the point clouds x given the latent space representation z. Where each point distribution $p_\theta(x_j|z)$ is modeled using the normal distribution:

$$p_\theta(x_j|z) = N(DE_i(z), 1). \tag{3.34}$$

The decoder consists of six fully connected layers followed by ReLU, except for the last layer, which outputs a point cloud set. Its architecture is shown in figure Figure 3.7.

The encoder and decoder network architectures are the same as in the VAE approach with the addition of the discriminator which consists of multiple fully connected layers followed by ReLU which take as input a representation of the latent space z and output a confidence value if the latent space was generated by the encoder or sampled from a normal distribution. Its architecture is shown in Figure 3.8.

3.4.4 Training

Our goal is to recreate point clouds in which points are classified into true and false lumen points. Therefore, we extend the approach of Zamorski et al. with the loss definition of Beetz et al. and extend the points of the aorta point clouds $\{P_i|i = 1, \ldots, n\}$ to have additional class information: $P_i = (x, y, z, c)$ where c classifies a point as either true or false lumen. The loss function for the VAE network is defined as:

$$L_{total} = L_{KL} + L_{recon}, \tag{3.35}$$

where L_{KL} is the Kullback-Leibler divergence:

$$L_{KL} = D_{KL}(q_\phi(z|x)||p_\theta(z)), \tag{3.36}$$

and L_{recon} is split into a separate loss for the true and false lumen reconstruction:

$$L_{recon} = L_{TL} + L_{FL}. \tag{3.37}$$

The reconstruction losses for the true and false lumen are calculated using the Chamfer distance CD (Equation 1.9), which measures the distance between the reconstructed point cloud S_1 and the ground truth point cloud S_2. The AAE uses the adapted loss function of Equation 3.31 and Equation 3.32 where the reconstruction loss L_{recon} is the same as in the VAE (Equation 3.37).

We use point clouds consisting of 10000 points as input for the network and task it to generate point clouds of 4096 points. The network learns the latent space by trying to reconstruct the input point clouds. The learned latent space representation of the aorta point clouds with classified true and false lumen points then enables us to create new synthetic point clouds with classified true and false lumen points.

3.5 Implementation

To create the various approaches for the generation of synthetic aortic dissection datasets we built a Python framework using PyTorch [84] as our machine learning framework. The machine learning part of our framework was run on Windows Subsystem for Linux 2 under Windows 10 using Python 3.8.10 with PyTorch 2.0.0

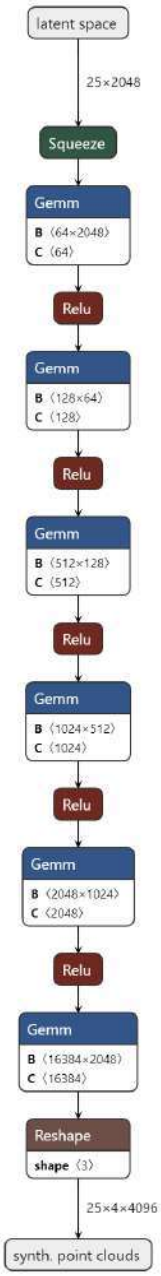


Figure 3.7: The decoder network of the VAE and AAE consists of a multi-layered perceptron built from linear transformation layers followed by ReLU. The decoder receives the latent space generated by the encoder as input and is tasked to create a multi-class point cloud of size 4×4096.

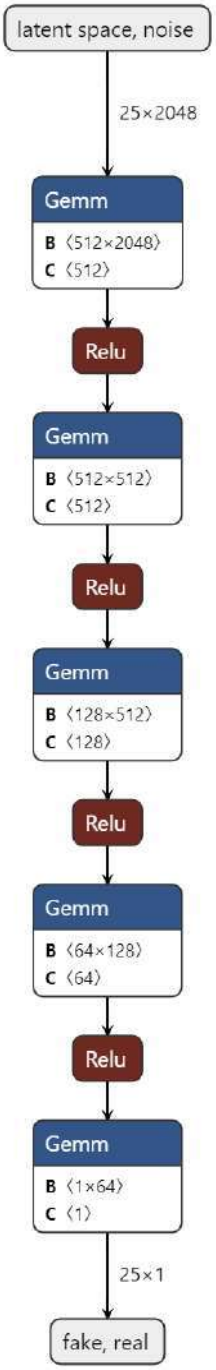


Figure 3.8: The discriminator network is a multi-layered perceptron consisting of multiple linear layers followed by ReLU. It receives either the latent space vector or a noise vector as input and is tasked to output a confidence value of the vector being either fake (latent space vector) or real (noise vector). The discriminator is only used in the AAE.

running CUDA version 10.1.243. We used a system with 16 GB of RAM, an Intel Core i5-10600K 4.1 GHz, and an NVIDIA GeForce RTX 3060Ti with 8 GB VRAM.

We mostly used the voxel and point cloud data structures provided by Open3D (V0.16.0) [130] to handle the preprocessing of our data (Section 3.1) and to convert the synthetic point clouds of our three approaches to meshes. Matrix and vector calculations were performed using Numpy (V1.24.2) [48] and Scipy (V1.10.0) [115].

4. Results and Discussion

This section evaluates the feasibility of the tested approaches and discusses the quality of the generated synthetic aortic dissection datasets. Each of the approaches is evaluated on its own regarding its achieved results in the context of comparable approaches in its domain as well as in the bigger scope of all three approaches. We will show the limitations of each approach and discuss how these can be overcome in the future. As the goal of this thesis is to establish a baseline trying to find suitable approaches for the generation of synthetic aortic dissection datasets, we did not perform an extensive qualitative evaluation questioning multiple domain experts. The approaches still have multiple drawbacks that can be improved that be able to generate more realistic datasets, which would then warrant a more extensive evaluation. We questioned two domain experts one CFD expert and one medical doctor in an open discussion to give us their opinion on the current quality of the generated datasets.

4.1 Generative Adversarial Network

The GAN can generate synthetic voxel representations of size $64 \times 64 \times 64$ from the learned latent space z. The latent space is generated from 26 input datasets which are converted into their respective voxel representations of size $64 \times 64 \times 64$ before being used as input for the network. The network is trained on the representations for the whole aorta, the true lumen, and the false lumen separately. The approach is not able to create representations of the true lumen, false lumen, and the aorta that are dependent on each other. Training the network for 5000 epochs took approximately 8 hours on the employed hardware setup (Section 3.5). The training process of the adversarial training including the generator and discriminator is shown in Figure 4.1.

The generator and discriminator perform the minimax game where both first converge to a loss near 1. During the training, the nature of the minimax game can be observed with one of the two networks gaining the upper hand regarding the training loss and the other one having to catch up in the next epoch. GANs are

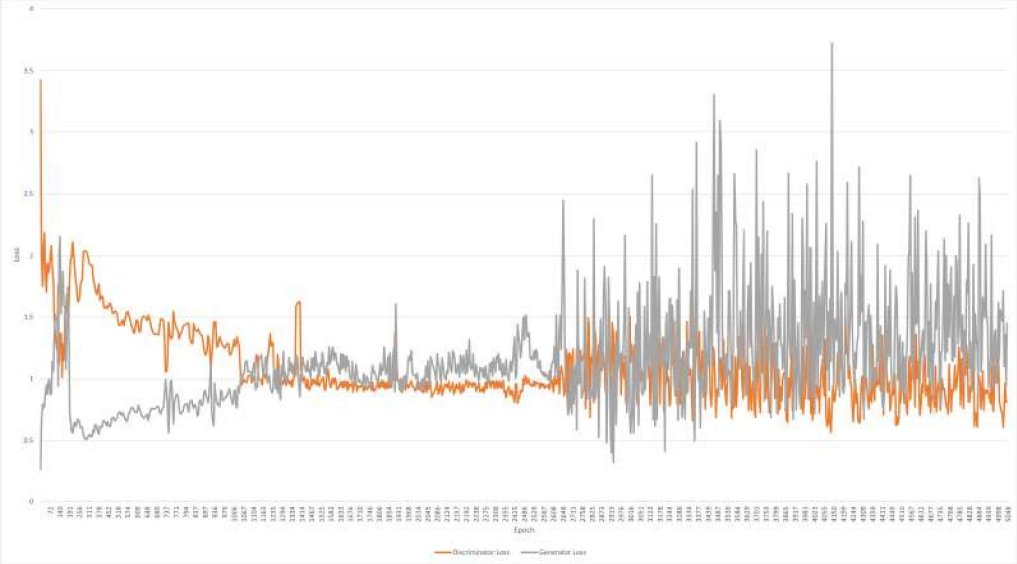


Figure 4.1: Loss of the generator and discriminator of the GAN network competing in the minimax game. The generator and discriminator slowly converge to around a training loss of 1 and keep further training until they reach the max training epoch.

(a) **(b)** **(c)**

Figure 4.2: Even when using the best-performing epoch the GAN can generate unrealistic results: (a) aorta, (b) true lumen, (c) false lumen.

inherently hard to train [79] and a stopping criterion, apart from human intervention is difficult to define [100]. We saved the weights of the GAN every 100 epochs and manually examined the results for the ability of the network to generate distinct but realistic voxel representations of the aorta, the true lumen, and the false lumen. Even when using the weights of the subjectively best-performing epoch (4700) in the network the generator might produce unrealistic results, see Figure 4.2. It is noticeable that the generated voxel representation can be very similar to the input data suggesting overfitting. This is additionally influenced by the small amount of training data for a machine-learning approach. Comparable approaches train on datasets that start at a minimum of 1000 entries per class while we train on 25 entries per class. Additionally, when training on batches with a size of five or smaller, the GAN would collapse [111] and only generate a single voxel representation.

Though the aorta, the true lumen, and the false lumen are trained on the same network with the same parameters, the generated aortas are of much higher quality than the true or false lumen. We suspect that this might be due to the aorta containing the voxel information of both the true and false lumen, effectively

doubling the information. This is the reason why the aorta network performs better on the whole aorta. The results of the networks are shown in Figure 4.3.

We also created a network that can take as input and create $128 \times 128 \times 128$ sized voxel representation. Based on the time to train each epoch, we determined to stop the training early due to time constraints. The training would have taken multiple weeks on our current setup which is unfeasible for the scope of this thesis. Additionally, the achieved quality of the created voxel representations is not sufficient to justify the longer training time. We would have to increase the resolution of the voxel grid by a factor of 100 to sufficiently detail the complex physiology of aortic dissection. Comparisons of the different levels of details are presented in Figure 4.4 To fully capture the segmentation masks of our input datasets we would need to create voxel grids with a size of approximately $150 \times 220 \times 640$ which would result in a rectangular voxel grid with $21,120,000$ voxels. This voxel grid contains eight times more voxels than our currently used voxel grid while being optimized for size. We do not know how well the 3D convolutions perform on rectangular grids. This would have to be investigated in the future. If we use a square grid to capture the segmentation mask the grid size would increase to $640 \times 640 \times 640$ which results in $262,144,000$ voxels per grid being $1,000$ times larger than our currently used grid. Further testing could be performed to find a tradeoff between the resolution and the ability to capture the aortic physiology in sufficient detail. When looking at Figure 4.4 one could argue that Figure 4.4b or Figure 4.4c might already be sufficiently detailed.

4.2 Statistical Shape Modelling

The SSM performs a PCA on the covariance matrix of centerlines for the aorta and true lumen with their respective radii. We compare the PCA combining the true lumen and aorta with a PCA that was only performed on either, the aorta, or the true lumen to get insights into the behavior of the SSM. The PCA was performed using 25 datasets with an additional test data set of 15.

From our 25 training datasets, we extracted 25 PCs. The explained variance for the PCA performed on the aorta combined with the true lumen started from the first PC at 48.7% reaching 90% at 7 PCs. The reconstruction errors on the training data decreased from 15.4% down to 4.6% at 5, reaching 1.5% at 10 PCs. Errors on the test dataset decrease from 20.9% at 1 PC to 9.5 at 5 to below 5% at 13 PCs.

Performing the PCA only on the aorta results in a higher explained variance using less PCs. The first PC encodes 51.9% reaching 90% already at 5 PC. The reconstruction errors on the training data amount to 6.9% at 1 PC and decrease below 1% at 6 PCs. On the test dataset, the errors are 9.6% at 1 PC decreasing below 5% at 4 PCs.

The PCA performed on only the true lumen encodes less variance in the first PC with 46% reaching 90% at 8 PCs with reconstruction errors on the training data of 8.4% at 1 PC decreasing below 1% at 10 PCs. Errors on the test dataset decrease from 11.3% using 1 PC for the reconstruction to below 5% at 6 PCs. The explained variance of the three PCAs can be seen in Figure 4.5 and the losses on the training and test data are presented in Figure 4.6.

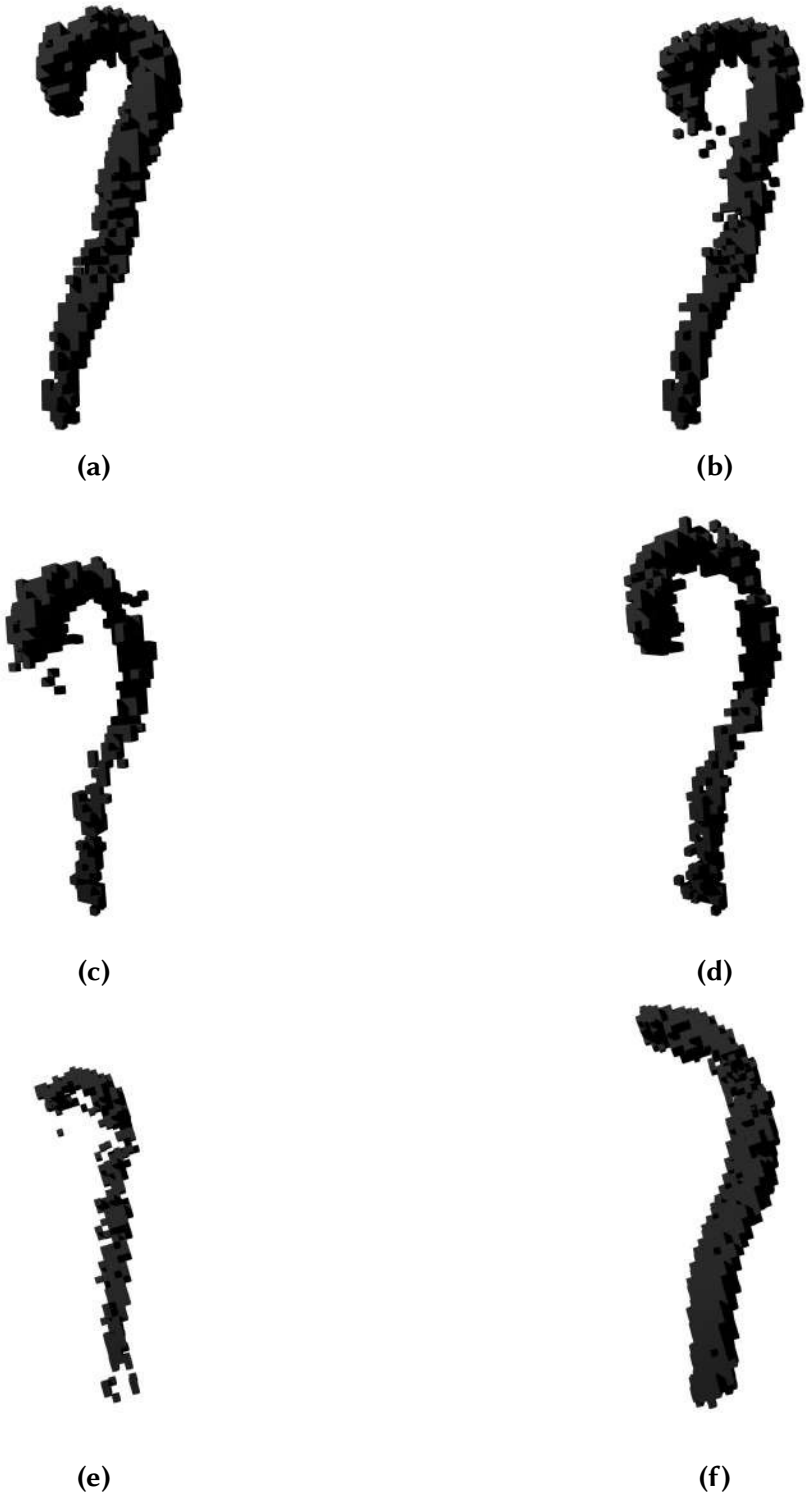


Figure 4.3: Different voxel representations generated by the GAN approach: (a) and (b) show generated aortas, (c) and (d) are generated true lumen, and (e) and (f) are generated false lumen. It has to be noted that true and false lumen can not be generated for a single combined synthetic dissection voxel representation.

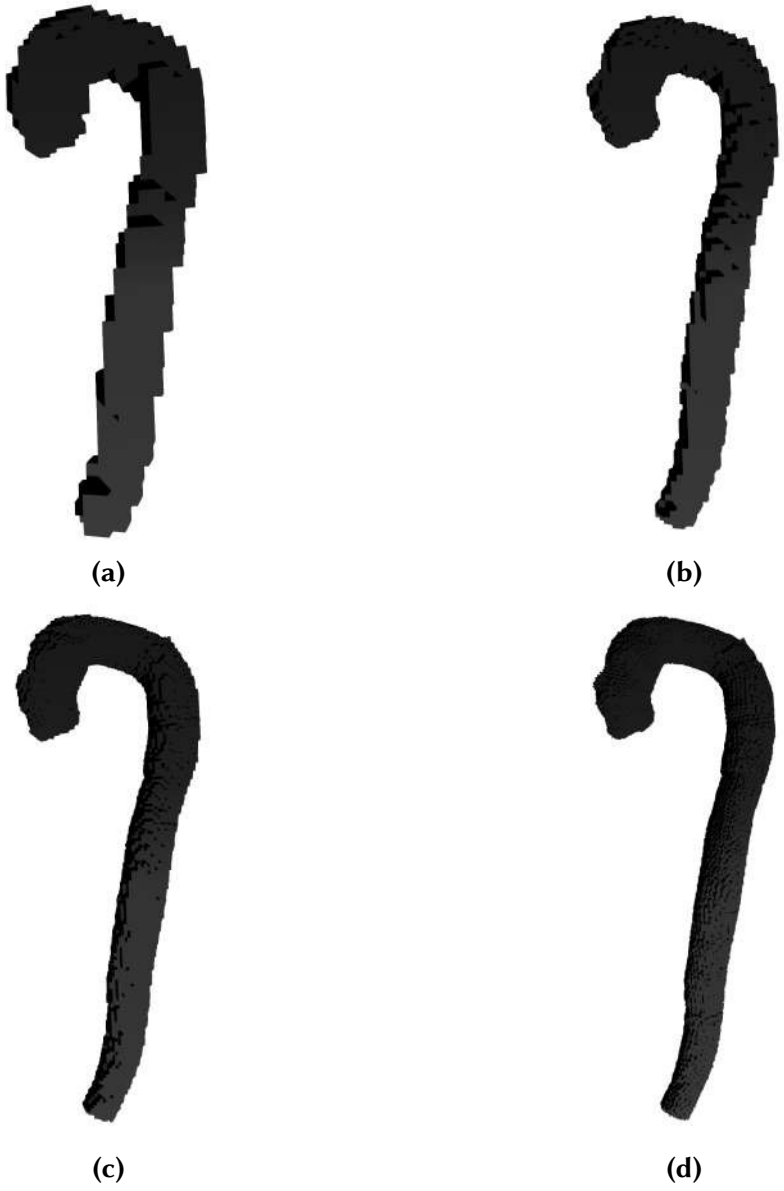


Figure 4.4: The current voxel grid resolution of our input data for the GAN network is shown in (a). Going from (b) over (c) to (d) the resolution is increased exponentially until we reach (d) which is the one that would capture the aortic physiology in its entirety.

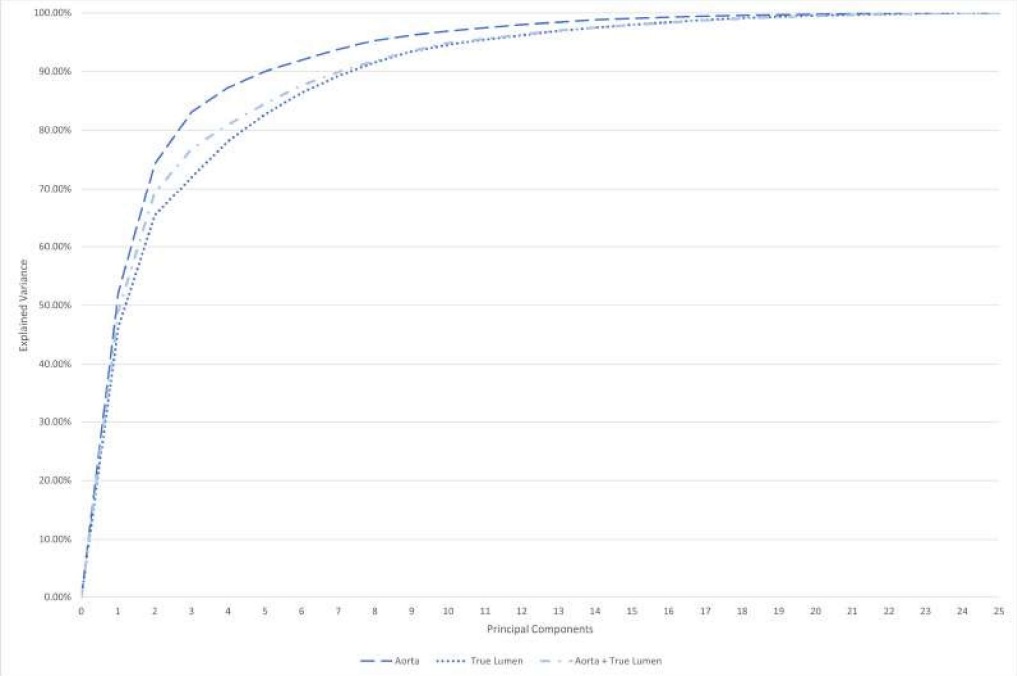


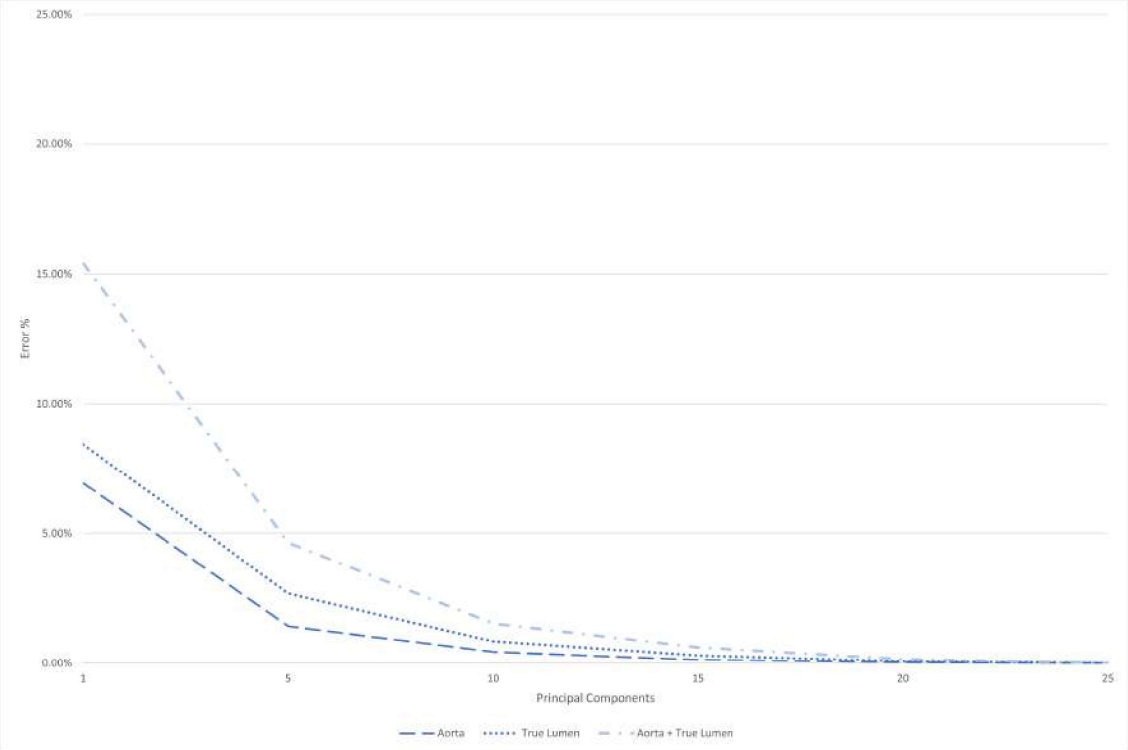
Figure 4.5: The cumulative variance of all 25 principal components of the PCAs performed on the parameterizations of only the aorta, only the true lumen, and the combined approach with true lumen and aorta.

These results suggest that SSM can capture the statistical variance even when combining the encodings for the aorta and the true lumen. The true lumen generally seems to have a bigger influence on the model compared to the aorta, which might be caused by its more complex physiology due to its stronger varying radii and centerline. The performance of the model on the test data suggests good generalizability with it being able to capture the physiological properties of a more general population of the aorta and true lumen.

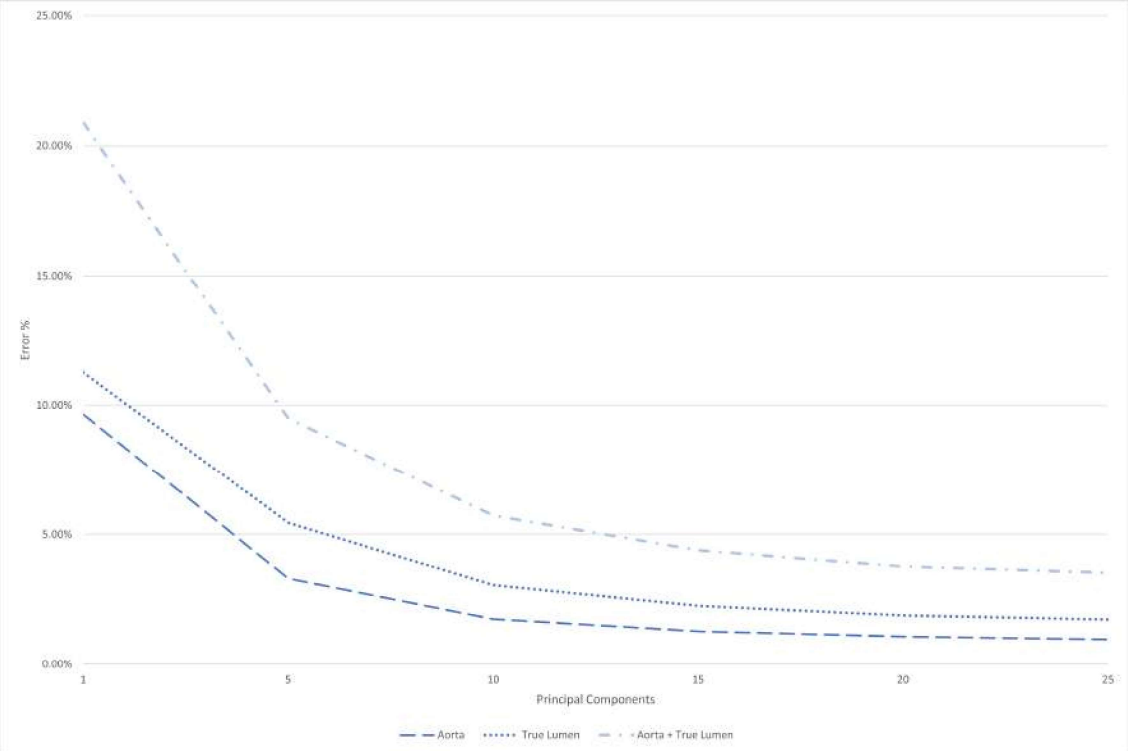
Next, we analyze the PCA combining the aorta and true lumen in more detail regarding centerline and radii information. The reconstruction errors on the training and test data are mostly influenced by the centerline information suggesting that the first PCs mainly contain centerline information. Examining the combined approach, we find that the centerline of the true lumen causes the higher reconstruction error with the radii of both aorta and true lumen only contributing a negligible amount to the error shown in Figure 4.7.

The reconstruction errors on the training data decrease from 6.2% using 1 PC down to below 1% at 6 PCs for the aorta centerline. The reconstruction error for the aorta radii starts at 0.7% plateauing at 10 PC with 0.2%. The true lumen centerline error decreases from 7.3% using 1 PC down to below 1% at 9 PCs. The true lumen reconstruction error starts at 0.8% using 1 PC also plateauing at 10 PCs with an error of 0.02%.

The reconstruction errors on the test data behave similarly, again indicating a good generalizability to the whole population of aortic dissection. The reconstruction error for the aortic centerline decreases from 8.8% using 1 PC down to 1.8% at 10

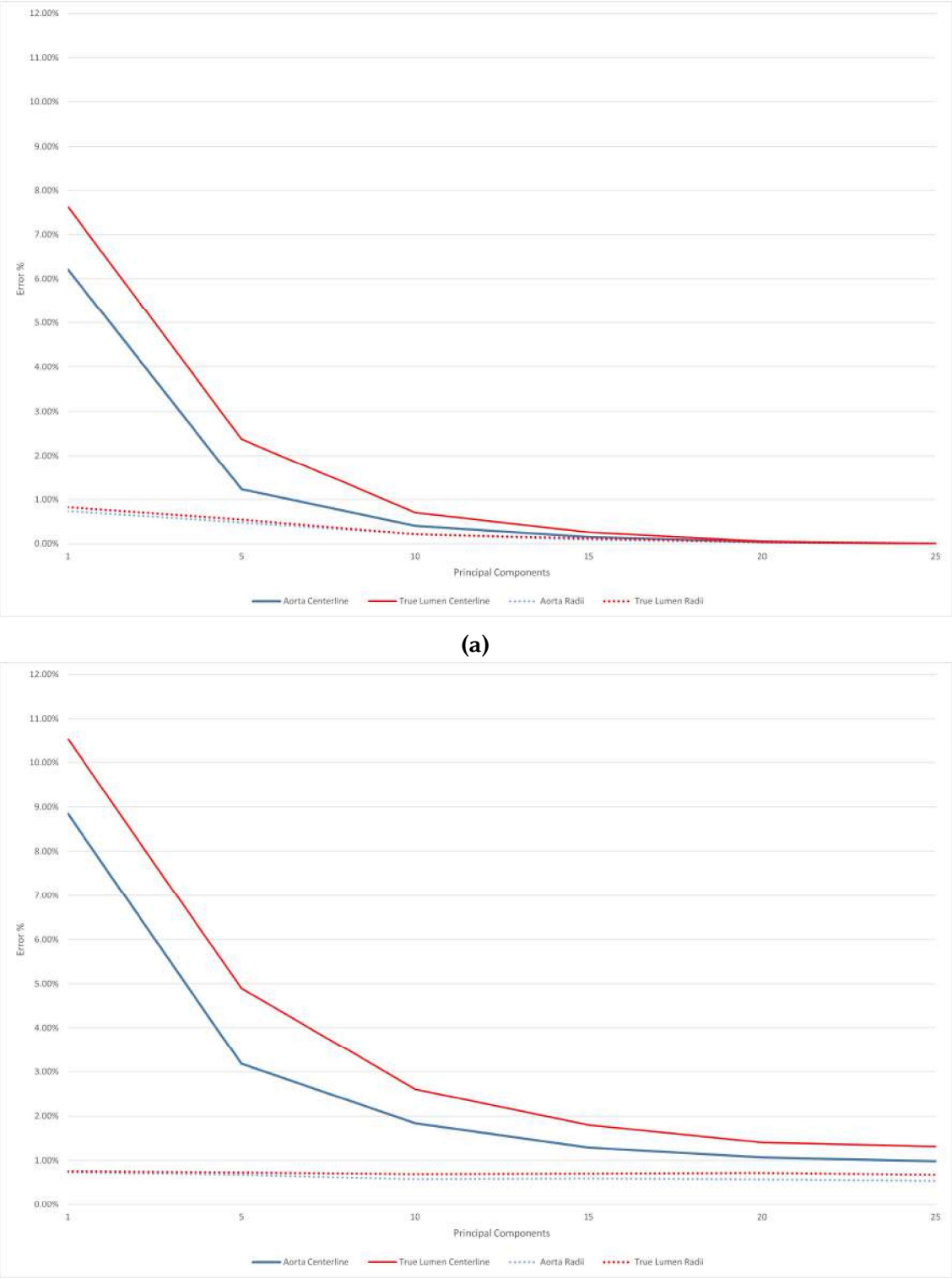


(a)



(b)

Figure 4.6: Comparison between the reconstruction error of the PCAs performed on the aorta, the true lumen, and the combined approach. The performance of the model on the training data is shown in (a) and on the test data in (b).



(a)

(b)

Figure 4.7: Comparison of the reconstruction error of the combined approach considering the centerlines and radii of the aorta and the true lumen in the training data (a) and the test data (b).

PCs. The error for the radii does not change significantly, staying below 1%. The true lumen centerline reconstruction error decreases from 10.5% at 1 PC down to 6.9% at 10 PCs. Interestingly, the reconstruction errors of the radii on the test data do not change noticeably when increasing the PCs. This might indicate that the physiology of aortic dissection is mostly influenced by the centerline information of both the aorta and true lumen. Figure 4.8 and Figure 4.9 show how the meshes created from the centerline and radii representations change when increasing the number of PC to reconstruct the aorta and the true lumen in the combined SSM.

Three synthetic meshes generated from the parameterization are shown in Figure 4.10. By varying the weights of the PCs, new distinct aortic dissection representations can be created. The synthetic shapes vary in centerline length and shape for aorta and true lumen as well as showing varying radii. The true lumen centerline generally follows the aortic centerline but varies sufficiently to create a realistic representation. Overall, this approach manages to create believable representations of aortic dissection. Our two experts agreed with that sentiment but indicated some problems. They concluded that the representation of the aorta looked realistic but the true lumen often tends to protrude through the aortic wall which is unrealistic. Additionally, the true lumen exhibits a much more uneven surface than that of the aorta. Furthermore, the true lumen can be located right in the middle of the aorta surrounded by what would be the false lumen. This case can occur physiologically but is very rare and arises too often when generating new synthetic representations. The weights that are randomly generated within the interval $[-\lambda, \lambda]$ (Section 3.3.2) can also lead to the generation of non-physiological shapes, as demonstrated in Figure 4.11a. Therefore, a validation method to filter non-physiological shapes has to be developed.

To demonstrate the current state of the model when automatically generating synthetic representations and meshes of aortic dissection, we present all meshes of this approach with as little human correction and postprocessing as possible. From the random generation of the representation, over the generation of the mesh from the calculated point clouds, we only intervened manually to select realistic representations, and to remove meshes when the Poisson surface reconstruction failed. A completely synthetic mesh created with further human intervention can be seen in Figure 4.11b. We carefully selected a synthetic representation that has a realistic aorta and true lumen shape, with the true lumen running properly along one side of the aorta. We chose proper parameters for the surface reconstruction and applied multiple postprocessing steps using Laplacian smoothing to improve the uneven surface of the true lumen. This step also helped in keeping the true lumen confined to the aorta. Aside from sophisticated validation methods, this demonstrated that further postprocessing steps could be employed to increase the quality of the generated representations and meshes. Especially, a constraint keeping the true lumen inside the aorta is detrimental to creating realistic dissection shapes.

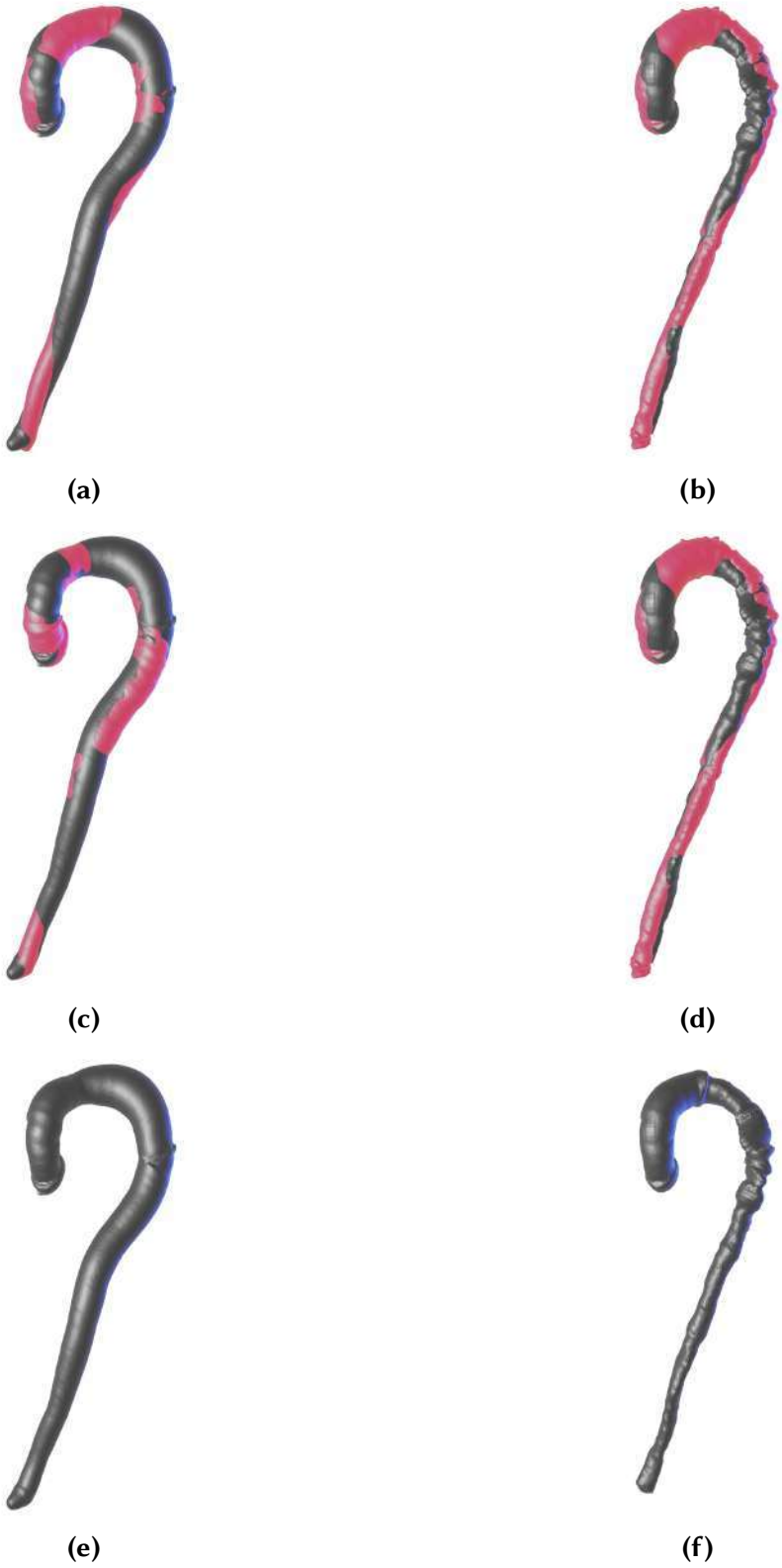


Figure 4.8: The training data reconstruction error of the aorta reconstruction (a, c, e) and the true lumen reconstruction (b, d, f) when using 5 (a, b), 15 (c, d), and 25 (e, f) PCs of the combined approach. The grey shape should be reconstructed with the red shape. When using all PCs, the reconstruction is perfect on the training data, which is to be expected.

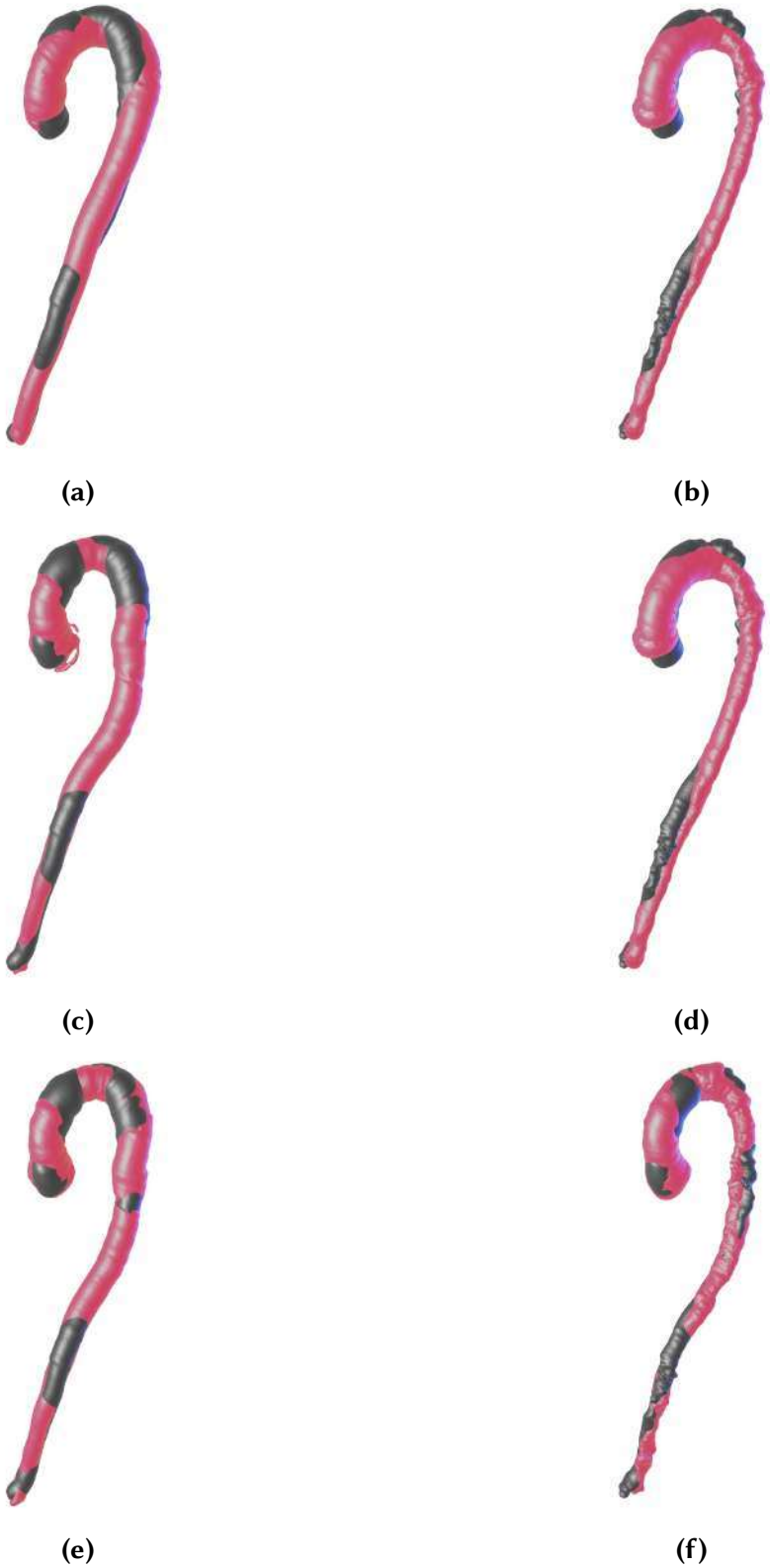


Figure 4.9: The test data reconstruction error of the aorta reconstruction (a, c, e) and the true lumen reconstruction (b, d, f) when using 5 (a, b), 15 (c, d), and 25 (e, f) PCs of the combined approach. The grey shape should be reconstructed with the red shape. Even when using all 25 PCs the reconstruction is not perfect but generalizes well enough to the whole population.

(a) (b) (c)

Figure 4.10: Randomly selected aortas from the SSM approach. The model can generate diverse aortic dissection representations that can be used to create surface meshes of the aorta and true lumen with varying diameters and centerline lengths. The translucent grey surface is the aortic wall and the red surface is the true lumen. Brighter red regions indicate the true lumen protruding through the aortic wall surface.

4.3 Autoencoders

The VAE and AAE were both trained on a training dataset of 25 aortic dissection point clouds with 10000 points labeled as either true or false lumen. Additionally, we used a test dataset of 5 point clouds. The points were only located on the surface of the respective lumen. We evaluate the performance of the autoencoders across all epochs using the JSD to determine the best-performing epoch of both models on the test and training data. Additionally, we examine the models for the development of their reconstruction loss on the test and training data. Each evaluation is performed on the labeled point clouds evaluating if the model can assign the correct lumen to each point and on unlabeled point clouds to compare how the labeling affects the reconstruction.

In the training data, the VAE is plateauing fairly quickly at a reconstruction loss of 0.013 for the true lumen, 0.0142 for the false lumen, and 0.0009 for unlabeled point clouds at 6000 epochs. Continuing the training, the loss experiences some spikes but mostly stays consistent until we terminate the training at 30000 epochs with a reconstruction loss of 0.0138 for the true lumen, 0.0148 for the false lumen, and 0.0012 for unlabeled point clouds. The lowest reconstruction error for the labeled point clouds is reached at 23500 epochs and 18500 epochs for the unlabeled point clouds. The reconstruction loss of the AAE behaves more linearly, consistently decreasing until it reaches its lowest score at epoch 27000 with 0.0196 for the true lumen and 0.0359 for the false lumen. The best epoch for the unlabeled point clouds is epoch

(a) (b)

Figure 4.11: The randomly generated weights can lead to unrealistic shapes where the Aorta and the true lumen are too narrow at the abdominal aorta (a). An aortic dissection model created with multiple manual steps results in a more realistic shape (b). Here the surface reconstruction was manually controlled and multiple smoothing steps were performed to constrain the true lumen to the inside of the aorta. Bright red sections only minimally protrude through the aorta surface.

28000 with a loss of 0.0095. Surprisingly, at the end of the training at 30000 epochs, the training loss spikes. Training the model further might decrease the loss again, like it did in the spikes before. This should be investigated further in the future. The development of the reconstruction loss on the training data for both the VAE and the AAE is shown in Figure 4.12.

The JSD behaves similarly to the reconstruction loss for both models. The VAEs best epoch for the true and false lumen reconstruction is 18500, with a loss of 0.0754 for the true lumen, and 0.0918 for the false lumen. The best-performing epoch of the unlabeled point clouds is 19000, with a loss of 0.049. The AAE is performing best on epoch 28500, with a loss of 0.0992 for the true lumen and 0.1289 for the false lumen. The unlabeled point clouds were reconstructed with the best loss of 0.0531 at epoch 29000. Diagrams plotting the JSD across the epoch are shown in Figure 4.13 and reconstructed point clouds in Figure 4.14.

Even though the performance on the training data indicates that the model is learning the physiology of aortic dissection and can label the true and false lumen points to a sufficiently accurate degree, the generalizability of the model is not supported by its performance on the test data. Generally, the autoencoders have more trouble reconstructing the true lumen compared to the false lumen. This might be because the true lumen has a larger volume and more complex shape compared to the false lumen. The reconstruction loss of the VAE on the test data does not change much when the model is trained for more epochs. The best-performing epoch for the true and false lumen is 500, with a loss of 1.8657 and 1.2112. The unlabeled point clouds are reconstructed best in epoch 500, with a loss of 0.0795. The AAE exhibits the best reconstruction loss for the true and false lumen in epoch 26500 at 1.3298 and 1.318. The best epoch for the unlabeled point clouds is 9000, with a loss of 0.0759. Diagrams for the reconstruction loss on the test data for both models are shown in Figure 4.15.

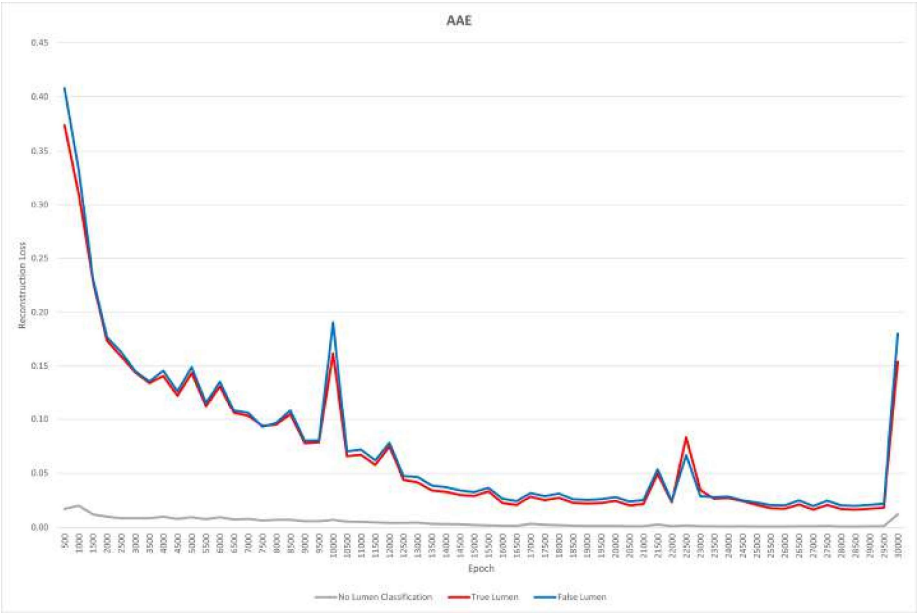
The JSD for the VAE was best on epoch 26000, with a score of 0.5354 for the true lumen, and 0.5618 for the false lumen. The unlabeled point clouds were reconstructed best in epoch 20500, with a score of 0.2398. Using the JSD on the AAE, the best epoch for the true and false lumen point clouds is 6000, with a loss of 0.5311 and 0.546819. unlabeled point clouds were reconstructed best in epoch 12500, with a loss of 0.2553. The development of the JSD for both models is shown in Figure 4.13 and reconstructed point clouds in Figure 4.17.

Even though the performance of the model on the test data indicates a bad generalizability of the model we do not think the theory behind the model is the cause but rather the small amount of training datasets. We only train the autoencoders on a training dataset size of 25 samples, which is a sample size suitable for SSM approaches, but machine learning approaches generally use databases with thousands of datasets. Comparable approaches are usually trained on ShapeNet, which has 1000+ samples per class, and Beetz et al. [6] trained on 850 datasets. A finding supporting the lack of training data is the poor performance of the model on unlabeled test point clouds. The network architecture is very close to the original approach of Zamorski et al. [125] who trained their network on unlabeled point clouds using the ShapeNet and ModelNet which both have 1000+ datasets per class. Additionally, we notice severe overfitting to the training data, supported by the findings that reconstructed point clouds from the test data look like point clouds from the training data.

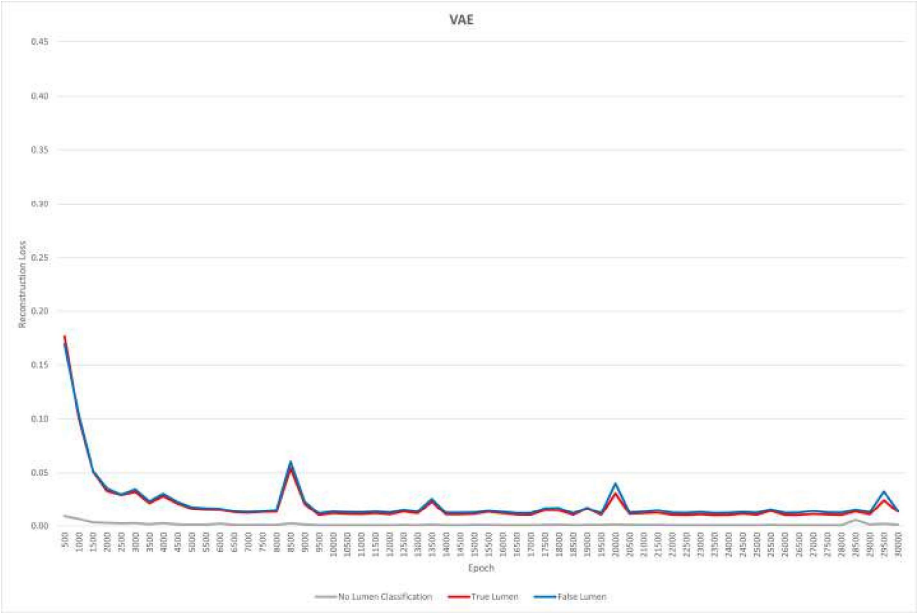
Though the VAE seems to perform better than the AAE regarding the reconstruction of the input point clouds, the generative ability of the AAE seems to be superior. When sampling from the latent space using both models, we notice that the AAE produces realistic samples more often than the VAE. The points generated by the VAE often do not lie on the outer wall of the respective lumen but inside of it and are also often mislabeled. Therefore, we assume that the ability of an AAE to construct a latent space with sharper transition seems to hold for our data. Generated point clouds for the true and false lumen are shown in Figure 4.18.

Only our medical expert evaluated the autoencoder approach. They determined it to produce realistic physiologically correct results. They highlighted the benefit of this approach to directly produce true and false lumen point clouds compared to

the SSM approach. Because we did not post-process the point clouds and showed the direct output of the model, the expert noted typical errors these models can produce: some points were located arbitrarily in space not close to the aorta, or they were not located directly on the wall of the respective lumen. Additionally, the model still mislabels points of the point clouds resulting in single points of either true or false lumen located on the true or false lumen surface. Analogously to the SSM approach, we need to develop a method to validate generated synthetic samples regarding their realism, because these models also sometimes fail to generate realistic shapes; especially regarding the point labeling as can be seen in Figure 4.19.



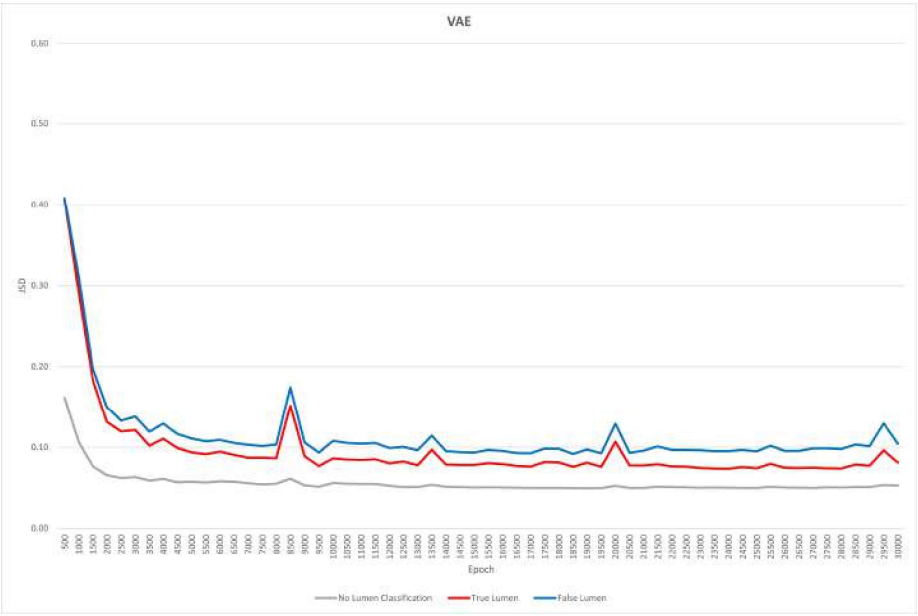
(a)



(b)

Figure 4.12: Reconstruction loss of the VAE (a) and AAE (b) on the training data using the Chamfer distance.

(a)



(b)

Figure 4.13: JSD of the VAE (a) and AAE (b) on the training data.

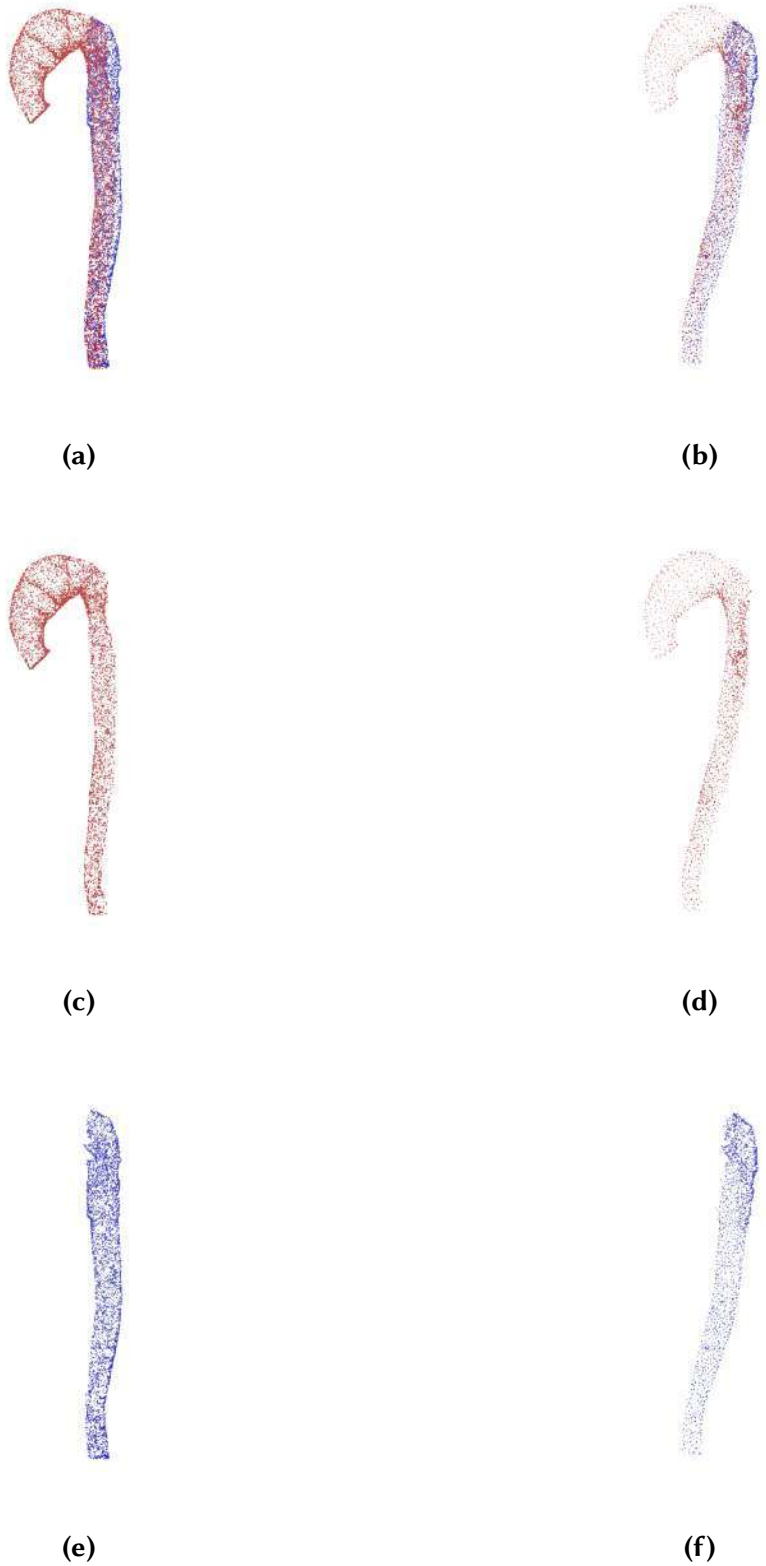
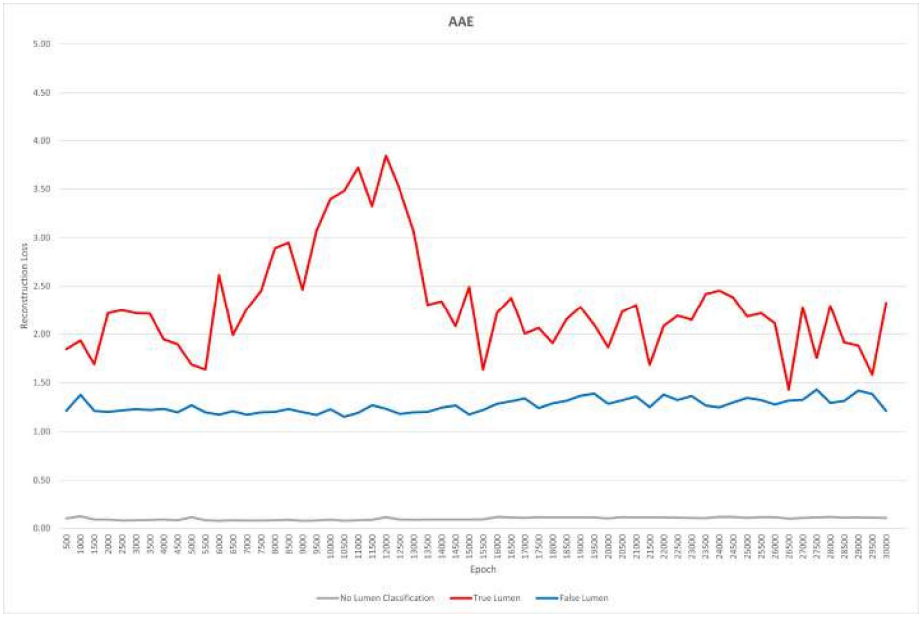
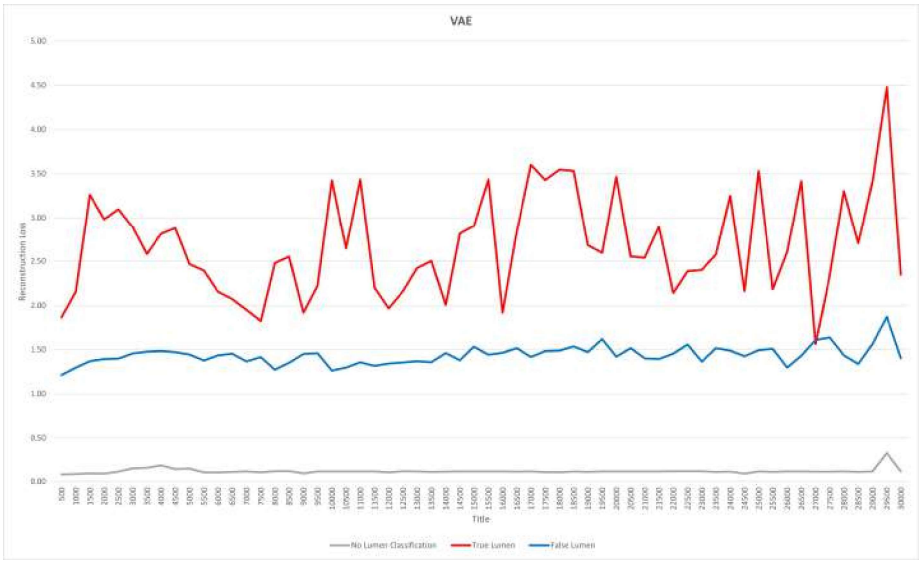


Figure 4.14: Combined point clouds (a, b) of the true (c, d) and false lumen (e, f) reconstructed from the training data.

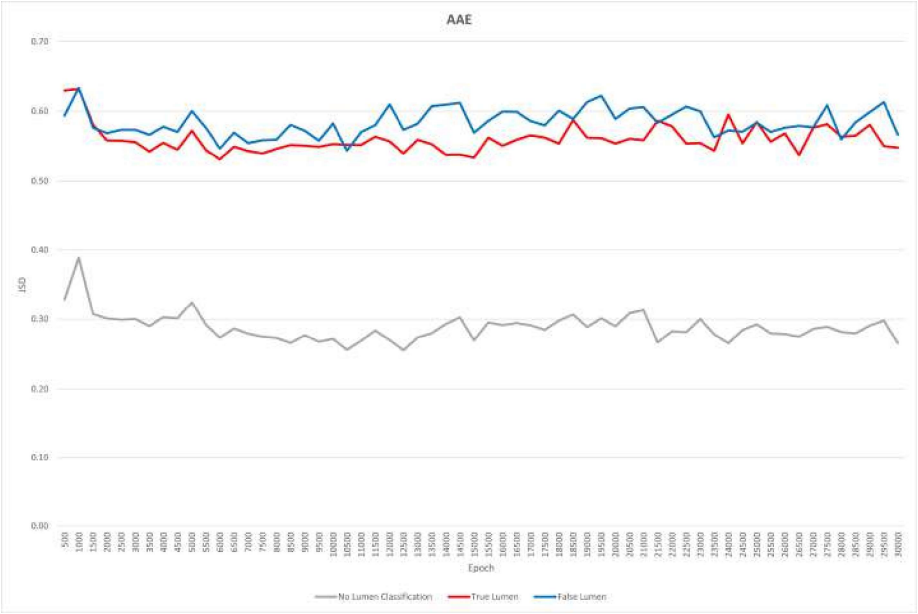


(a)

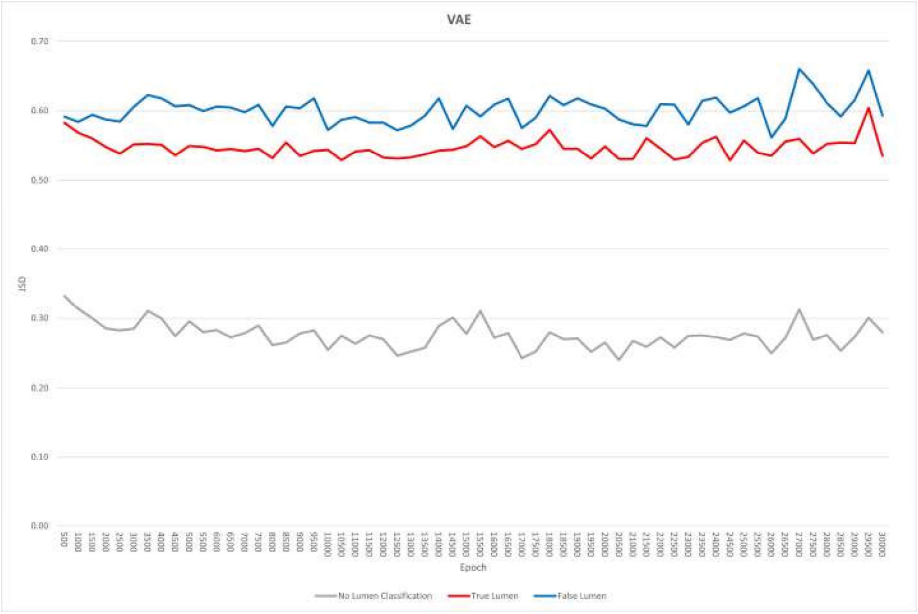


(b)

Figure 4.15: Reconstruction loss of the VAE (a) and AAE (b) on the test data using the chamfer distance.

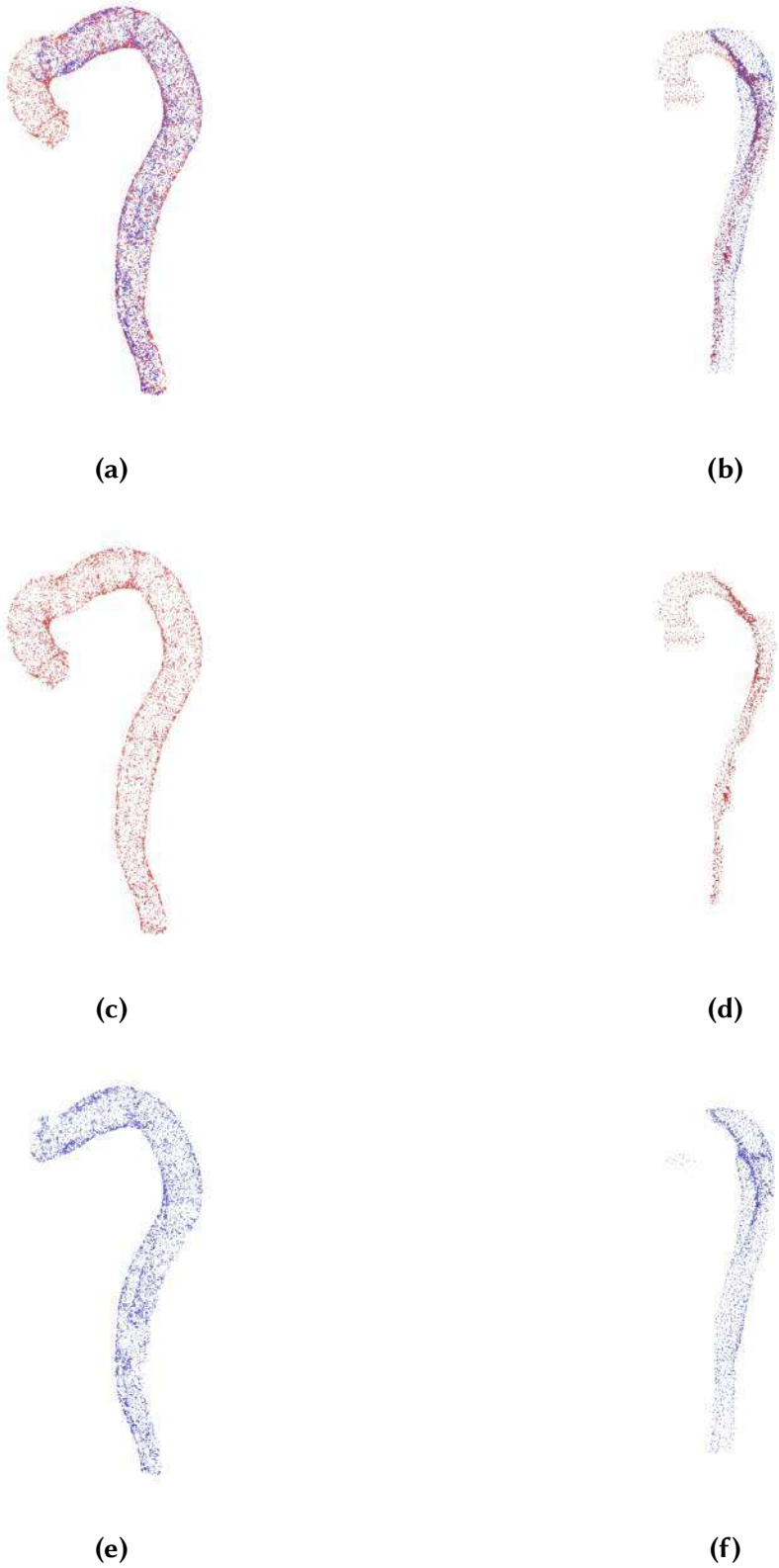


(a)



(b)

Figure 4.16: JSD of the VAE (a) and AAE (b) on the test data.



(a) **(b)**

(c) **(d)**

(e) **(f)**

Figure 4.17: Combined point clouds (a, b) of the true (c, d) and false lumen (e, f) reconstructed from the test data. The model currently fails to properly reconstruct test data point clouds.

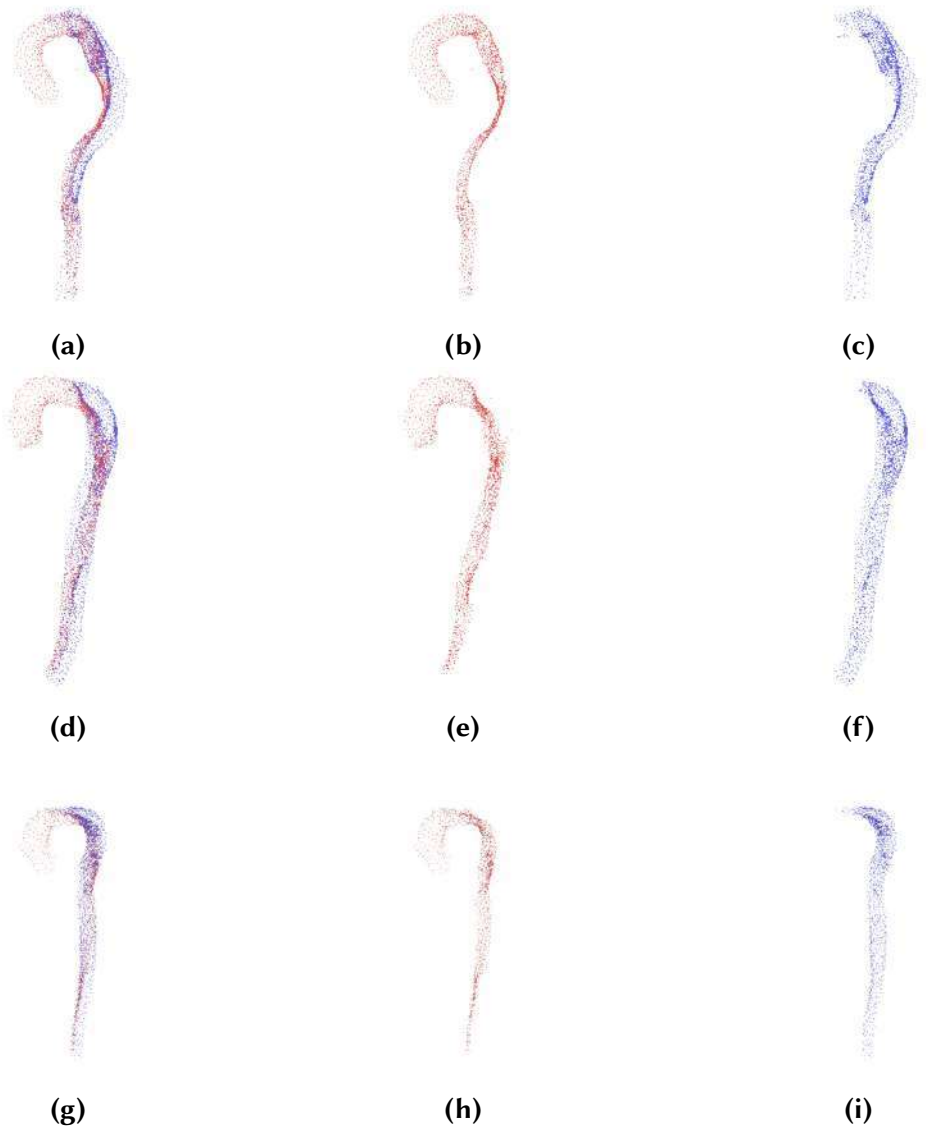


Figure 4.18: Synthetically generated point clouds of aortic dissection (a, d, g) with points labeled as true (b, e, h) and false lumen (c, f, i).

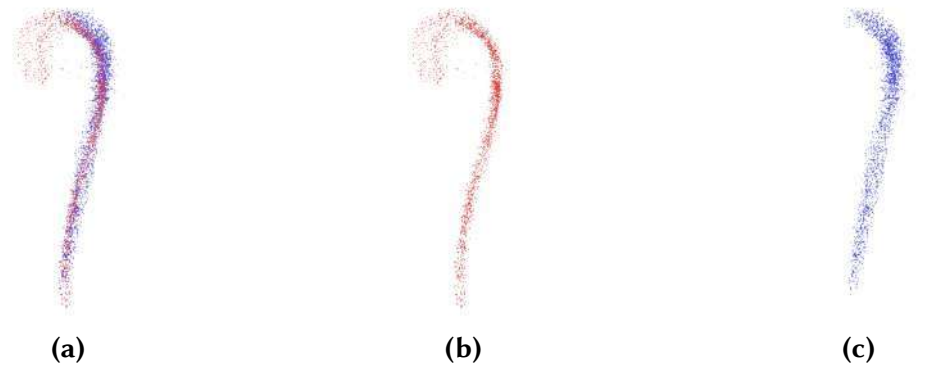


Figure 4.19: Both models might produce unrealistic results with misplaced and mislabeled points: (a) combined point clouds of true (b) and false lumen (c).

5. Conclusion and Future Work

This thesis aimed to set a baseline regarding the creation of a database consisting of synthetic datasets of aortic dissections, including the true and false lumen along the entire length of the aorta, i.e., from the aortic root to the abdominal aorta. Excluded were the branching vessels and locations of entry and exit tears. Three approaches were examined including a GAN, a SSM, and autoencoders such as a VAE and AAE. The SSM and autoencoders are both able to generate realistic aortic dissection datasets with some model-specific drawbacks that have to be improved in the future. The GAN does not seem to be suitable for the creation of aortic dissection datasets.

5.1 Generative Adversarial Network

Though we would not suggest looking further into the GAN approach, several improvements could be made if a GAN is used in some preprocessing or evaluation steps. The resolution limitations of the voxel grid on our hardware and the resulting training times might be improved by using a rectangular voxel grid fitted to the bounding box of the largest aorta in the dataset and not a square one. This would increase the ratio between voxels in the grid that contain the shape information and voxels containing background information reducing the needed computational power. For this approach, the network architecture regarding kernel sizes, voxel grid sizes, strides, and paddings would have to be adapted. We have already mentioned the increased training times when increasing the voxel resolution, but the approach would only be viable when the resolution could be increased to a resolution that could capture the same details as the point cloud approaches. Example voxelizations for higher resolutions are shown in Figure 4.4.

5.2 Statistical Shape model

The SSM is currently the best-suited approach for our small number of training datasets. The approach is already able to create realistic aorta and true lumen representations that can be converted to a surface mesh. The most significant

drawbacks of this approach are that the true lumen wall is protruding through the aortic wall and that the true lumen surface is fairly uneven.

To fix the true lumen protrusion the SSM approach could be extended to constrain the calculated radii and centerlines to the inside of the aorta when generating new shapes. To improve the uneven surface of the true lumen, the radii could either be interpolated between the different centerline points of the lumen or the generated mesh could be smoothed like we manually did in Figure 4.11b. We would suggest first working on improving the parameterization of the aortic dissection by directly working on the centerline and radii and only then using mesh processing algorithms because those need a significant amount of manual intervention to create sufficient meshes. Additionally, we are currently creating the meshes in a cumbersome way by first creating points lying on the circumference of the ellipse defined by the radii and then performing the Poisson surface reconstruction. This should be improved by directly generating a mesh from the parameterization without taking a detour by creating points on the surface. A suitable approach might be that of Zhou et al. [129].

The most important part of automating the generation of a synthetic database would be the automatic evaluation of the generated parameterizations or meshes. Past approaches that generated synthetic databases of aortas use acceptance criteria defined on the clinical biomarkers of the aorta [94] or performed CFD simulations on the generated meshes [117]. Some of our datasets currently contain information regarding the location of specific landmarks, which we can use to create such criteria. Future approaches should also look into including more information into the model: such as the modeling of branching vessels, entry and exit tears, and CFD information. Furthermore, we would like to experiment with performing the PCA on the true and false lumen and combining the generated parameterizations to a coherent representation of aortic dissection. Also, with performing the PCA on the aorta and the false lumen, this time excluding the true lumen. Performing the PCA on the aorta and the false lumen might be beneficial to simplify the model because the false lumen is usually smaller than the true lumen and the true lumen is entirely captured when subtracting the false lumen from the aorta representation.

Looking into point cloud-to-point cloud registration is another promising avenue to achieve a more accurate SSM. Therefore, better sampling strategies for the points have to be investigated to achieve 1:1 correspondences. The available landmark data could also be used to calculate corresponding points. To increase the accuracy of the current parameterization of centerlines and radii the downsampling step could be exchanged for an upsampling step keeping more information of the longer centerlines while interpolating the shorter ones. Possible errors in this upsampling step could be evaluated by comparing it to the current approach.

5.3 Autoencoders

The autoencoder-based approach is currently the most promising approach to generate realistic aortic dissection datasets, because of its ability to generate the aortic dissection point clouds describing the surfaces of the true and false lumen without further processing. The biggest drawback to properly evaluate this approach

is the lack of training data which is not an easy problem to fix. Nevertheless, multiple improvements can be made to the model starting with the preprocessing of the input data. We currently sample the points describing the true and false lumen surfaces randomly using a normal distribution. This random sampling can lead to some areas of a lumen being denser populated by points than others which might hinder the correct learning of the underlying distribution of the aortic dissection database. To overcome this issue we could sample the point clouds using Poisson disk sampling [45] resulting in a more even sampling.

We are currently using PointNet to encode and learn a latent space representation of aortic dissection. In the future, we would like to experiment with the PointNet++ [87] architecture in the encoder of our models. PointNet++ is an advancement of PointNet and might be able to better capture the underlying distribution describing aortic dissection.

The current output of the model are point clouds consisting of 4096 points. To increase the number of points, an upsampling network like FoldingNet [124] could be trained. Such higher-resolution point clouds could then be used to create surface meshes of the true and false lumen.

The model could also be extended by encoding CFD measures such as wall shear stress and pressure, to generate point clouds with points containing that information. Additionally, the input data could be tagged with keywords describing the type of aortic dissection and further information describing the disease. If all of this information gets encoded, a model could be created that allows users to create specific types of aortic dissection depending on the input keywords.

Furthermore, like in the SSM approach, we would need a method to evaluate the generated point clouds regarding their realism. We could adapt the aforementioned acceptance criteria described in Section 5.2 to work with point clouds and use these to evaluate the generated synthetic point clouds. These acceptance criteria might even be directly included in the model, forcing it to only generate realistic point clouds.

Bibliography

[1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning Representations and Generative Models for 3D Point Clouds. *arXiv*, 2017. doi: 10.48550/ARXIV.1707.02392. (cited on Page 6, 14, 15, and 16)

[2] T. Anvekar, R. A. Tabib, D. Hegde, and U. Mudengudi. VG-VAE: A Venatus Geometry Point-Cloud Variational Auto-Encoder. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2022. doi: 10.1109/cvprw56347.2022.00336. (cited on Page 16)

[3] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: CNN Architecture for Weakly Supervised Place Recognition. *arXiv*, 2015. doi: 10.48550/ARXIV.1511.07247. (cited on Page 15)

[4] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *arXiv*, 2017. doi: 10.48550/ARXIV.1701.07875. (cited on Page 10)

[5] C. F. Baumgartner, L. M. Koch, K. C. Tezcan, J. X. Ang, and E. Konukoglu. Visual Feature Attribution Using Wasserstein Gans. *arXiv*, 2017. doi: 10.48550/ARXIV.1711.08998. (cited on Page 10)

[6] M. Beetz, A. Banerjee, and V. Grau. Multi-domain Variational Autoencoders for Combined Modeling of MRI-based Biventricular Anatomy and ECG-based Cardiac Electrophysiology. *Frontiers in Physiology*, 13, 2022. doi: 10.3389/fphys.2022.886723. (cited on Page 3, 14, 19, 30, 34, and 54)

[7] G. A. Bello, T. J. W. Dawes, J. Duan, C. Biffi, A. de Marvao, L. S. G. E. Howard, J. S. R. Gibbs, M. R. Wilkins, S. A. Cook, D. Rueckert, and D. P. O'Regan. Deep-learning Cardiac Motion Analysis for Human Survival Prediction. *Nature Machine Intelligence*, 1(2):95–104, 2019. doi: 10.1038/s42256-019-0019-2. (cited on Page 14)

[8] Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai. Better Mixing via Deep Representations. *arXiv*, 2012. doi: 10.48550/ARXIV.1207.4404. (cited on Page 9)

[9] Y. Bengio, E. Thibodeau-Laufer, and J. Yosinski. Deep Generative Stochastic Networks Trainable by Backprop. *arXiv*, 2013. doi: 10.48550/ARXIV.1306.1091. (cited on Page 9)

[10] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The Ball-pivoting Algorithm for Surface Reconstruction. *IEEE Transactions on*

Visualization and Computer Graphics, 5(4):349–359, 1999. doi: 10.1109/2945.817351. (cited on Page 30)

[11] P. J. Besl and N. D. McKay. A Method for Registration of 3-d Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. doi: 10.1109/34.121791. (cited on Page 23)

[12] M. Bonfanti, G. Franzetti, S. Homer-Vanniasinkam, V. Díaz-Zuccarini, and S. Balabani. A Combined In Vivo, In Vitro, In Silico Approach for Patient-Specific Haemodynamic Studies of Aortic Dissection. *Annals of Biomedical Engineering*, 48(12):2950–2964, 2020. doi: 10.1007/s10439-020-02603-z. (cited on Page 7)

[13] A. C. Braverman, E. Mittauer, K. M. Harris, A. Evangelista, R. E. Pyeritz, D. Brinster, L. Conklin, T. Suzuki, C. Fanola, M. Ouzounian, E. Chen, T. Myrmel, R. Bekeredjian, S. Hutchison, J. Coselli, D. Gilon, P. O'Gara, M. Davis, E. Isselbacher, and K. Eagle. Clinical Features and Outcomes of Pregnancy-Related Acute Aortic Dissection. *JAMA Cardiology*, 2020. doi: 10.1001/jamacardio.2020.4876. (cited on Page 8)

[14] J. L. Bruse, S. Schievano, M. A. Zuluaga, A. Khushnood, K. McLeod, H. N. Ntsinjana, T.-Y. Hsia, M. Sermesant, X. Pennec, and A. M. Taylor. Detecting Clinically Meaningful Shape Clusters in Medical Image Data: Metrics Analysis for Hierarchical Clustering Applied to Healthy and Pathological Aortic Arches. *IEEE Transactions on Biomedical Engineering*, 64(10):2373–2383, 2017. doi: 10.1109/tbme.2017.2655364. (cited on Page 11)

[15] K. Bäumler, V. Vedula, A. M. Sailer, J. Seo, P. Chiu, G. Mistelbauer, F. P. Chan, M. P. Fischbein, A. L. Marsden, and D. Fleischmann. Fluid–structure Interaction Simulations of Patient-specific Aortic Dissection. *Biomechanics and Modeling in Mechanobiology*, 19(5):1607–1628, 2020. doi: 10.1007/s10237-020-01294-8. (cited on Page 1 and 2)

[16] R. Cai, G. Yang, H. Averbuch-Elor, Z. Hao, S. Belongie, N. Snavely, and B. Hariharan. Learning Gradient Fields for Shape Generation. *arXiv*, 2020. doi: 10.48550/ARXIV.2008.06520. (cited on Page 16)

[17] C. Catalano, V. Agnese, G. Gentile, G. M. Raffa, M. Pilato, and S. Pasta. Atlas-Based Evaluation of Hemodynamic in Ascending Thoracic Aortic Aneurysms. *Applied Sciences*, 12(1):394, 2021. doi: 10.3390/app12010394. (cited on Page 13)

[18] E. Catmull and J. Clark. Recursively Generated B-spline Surfaces on Arbitrary Topological Meshes. *Computer-Aided Design*, 10(6):350–355, 1978. doi: 10.1016/0010-4485(78)90110-0. (cited on Page 17)

[19] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv*, 2015. doi: 10.48550/ARXIV.1512.03012. (cited on Page 30)

[20] J. Chen, G. Li, R. Zhang, T. H. Li, and W. Gao. Pointivae: Invertible Variational Autoencoder Framework for 3D Point Cloud Generation. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 3216–3220. IEEE, 2022. doi: 10.1109/icip46576.2022.9897485. (cited on Page 16)

[21] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural Ordinary Differential Equations. *arXiv*, 2018. doi: 10.48550/ARXIV.1806.07366. (cited on Page 15)

[22] R. L. Cook and K. E. Torrance. A Reflectance Model for Computer Graphics. *ACM Transactions on Graphics*, 1(1):7–24, 1982. doi: 10.1145/357290.357293. (cited on Page 17)

[23] T. Cootes. An Introduction to Active Shape Models. *Image Processing and Analysis*, 2000. (cited on Page 4)

[24] F. Cosentino, G. M. Raffa, G. Gentile, V. Agnese, D. Bellavia, M. Pilato, and S. Pasta. Statistical Shape Analysis of Ascending Thoracic Aortic Aneurysm: Correlation between Shape and Biomechanical Descriptors. *Journal of Personalized Medicine*, 10(2):28, 2020. doi: 10.3390/jpm10020028. (cited on Page 12)

[25] M. Danu, C.-I. Nita, A. Vizitiu, C. Suciu, and L. M. Itu. Deep Learning Based Generation of Synthetic Blood Vessel Surfaces. In *2019 23rd International Conference on System Theory, Control and Computing (ICSTCC)*, pages 662–667. IEEE, 2019. doi: 10.1109/icstcc.2019.8885576. (cited on Page 3 and 10)

[26] M. de Bruijne, B. van Ginneken, M. A. Viergever, and W. J. Niessen. Adapting Active Shape Models for 3D Segmentation of Tubular Structures in Medical Images. In *Lecture Notes in Computer Science*, pages 136–147. Springer Berlin Heidelberg, 2003. doi: 10.1007/978-3-540-45087-0_12. (cited on Page 11)

[27] L. Deng. The MNIST Database of Handwritten Digit Images for Machine Learning Research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. (cited on Page 9)

[28] X. Duan, D. Chen, J. Wang, M. Shi, Q. Chen, H. Zhao, R. Zuo, X. Li, and Q. Wang. Visual Three-Dimensional Reconstruction of Aortic Dissection Based on Medical CT Images. *International Journal of Digital Multimedia Broadcasting*, 2017:1–8, 2017. doi: 10.1155/2017/3163759. (cited on Page 12)

[29] S. Durrleman, M. Prastawa, N. Charon, J. R. Korenberg, S. Joshi, G. Gerig, and A. Trouvé. Morphometry of Anatomical Shape Complexes with Dense Deformations and Sparse Parameters. *NeuroImage*, 101:35–49, 2014. doi: 10.1016/j.neuroimage.2014.06.043. (cited on Page 11)

[30] H. Edelsbrunner and E. Mücke. Three-dimensional Alpha Shapes. *arXiv*, 1994. doi: 10.48550/ARXIV.MATH/9410208. (cited on Page 30)

[31] P. Feldman, M. Fainstein, V. Siless, C. Delrieux, and E. Iarussi. VesselVAE: Recursive Variational Autoencoders for 3D Blood Vessel Synthesis. *arXiv*, 2023. doi: 10.48550/ARXIV.2307.03592. (cited on Page 17)

[32] P. Felkel, R. Wegenkittl, and K. Buhler. Surface Models of Tube Trees. In *Proceedings Computer Graphics International*, pages 70–77. IEEE, 2004. doi: 10.1109/cgi.2004.1309194. (cited on Page 17)

[33] A. Ferreira, J. Li, K. L. Pomykala, J. Kleesiek, V. Alves, and J. Egger. GAN-based Generation of Realistic 3D Data: A Systematic Review and Taxonomy. *arXiv*, 2022. doi: 10.48550/ARXIV.2207.01390. (cited on Page 7)

[34] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan. GAN-based Synthetic Medical Image Augmentation for Increased CNN Performance in Liver Lesion Classification. *Neurocomputing*, 321: 321–331, 2018. doi: 10.1016/j.neucom.2018.09.013. (cited on Page 10)

[35] A. Ganguly and S. W. F. Earp. An Introduction to Variational Inference. *arXiv*, 2021. doi: 10.48550/ARXIV.2108.13083. (cited on Page 32)

[36] A. Gholami, S. Subramanian, V. Shenoy, N. Himthani, X. Yue, S. Zhao, P. Jin, G. Biros, and K. Keutzer. A Novel Domain Adaptation Framework for Medical Image Segmentation. *arXiv*, 2018. doi: 10.48550/ARXIV.1810.05732. (cited on Page 10)

[37] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta. Learning a Predictable and Generative Vector Representation for Objects. *arXiv*, 2016. doi: 10.48550/ARXIV.1603.08637. (cited on Page 9)

[38] S. Glasser, K. Lawonn, T. Hoffmann, M. Skalej, and B. Preim. Combined Visualization of Wall Thickness and Wall Shear Stress for the Evaluation of Aneurysms. *IEEE Transactions on Visualization and Computer Graphics*, 20 (12):2506–2515, 2014. doi: 10.1109/tvcg.2014.2346406. (cited on Page 17)

[39] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. 2014. doi: 10.48550/ARXIV.1406.2661. (cited on Page 8 and 22)

[40] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. Ffjord: Free-form Continuous Dynamics for Scalable Reversible Generative Models. *arXiv*, 2018. doi: 10.48550/ARXIV.1810.01367. (cited on Page 15)

[41] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. Atlasnet: A Papier-mâché Approach to Learning 3D Surface Generation. *arXiv*, 2018. doi: 10.48550/ARXIV.1802.05384. (cited on Page 16)

[42] X. Gu, S. J. Gortler, and H. Hoppe. Geometry Images. *ACM Transactions on Graphics*, 21(3):355–361, 2002. doi: 10.1145/566654.566589. (cited on Page 16)

[43] T. Gudbjartsson, A. Ahlsson, A. Geirsson, J. Gunn, V. Hjortdal, A. Jeppsson, A. Mennander, I. Zindovic, and C. Olsson. Acute Type A Aortic Dissection – a Review. *Scandinavian Cardiovascular Journal*, 54(1):1–13, 2019. doi: 10.1080/14017431.2019.1660401. (cited on Page 8)

[44] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved Training of Wasserstein GANs. *arXiv*, 2017. doi: 10.48550/ARXIV.1704.00028. (cited on Page 6 and 15)

[45] J. Guo, D.-M. Yan, X. Jia, and X. Zhang. Efficient Maximal Poisson-disk Sampling and Remeshing on Surfaces. *Computers & amp; Graphics*, 46:72–79, 2015. doi: 10.1016/j.cag.2014.09.015. (cited on Page 65)

[46] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu. Pct: Point Cloud Transformer. *arXiv*, 2020. doi: 10.48550/ARXIV.2012.09688. (cited on Page 16)

[47] G. Hamarneh and P. Jassi. VascuSynth: Simulating Vascular Trees for Generating Volumetric Image Data with Ground-truth Segmentation and Tree Analysis. *Computerized Medical Imaging and Graphics*, 34(8):605–616, 2010. doi: 10.1016/j.compmedimag.2010.06.002. (cited on Page 17)

[48] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array Programming with NumPy. *Nature*, 585(7825):357–362, 2020. doi: 10.1038/s41586-020-2649-2. (cited on Page 39)

[49] K. M. Harris, C. A. Nienaber, M. D. Peterson, E. M. Woznicki, A. C. Braverman, S. Trimarchi, T. Myrmel, R. Pyeritz, S. Hutchison, C. Strauss, M. P. Ehrlich, T. G. Gleason, A. Korach, D. G. Montgomery, E. M. Isselbacher, and K. A. Eagle. Early Mortality in Type A Acute Aortic Dissection. *JAMA Cardiology*, 7(10):1009, 2022. doi: 10.1001/jamacardio.2022.2718. (cited on Page 8)

[50] E. Hassan, M. Y. Shams, N. A. Hikal, and S. Elmougy. The Effect of Choosing Optimizer Algorithms to Improve Computer Vision Tasks: A Comparative Study. *Multimedia Tools and Applications*, 82(11):16591–16633, 2022. doi: 10.1007/s11042-022-13820-0. (cited on Page 26)

[51] T. Heimann and H.-P. Meinzer. Statistical Shape Models for 3D Medical Image Segmentation: A Review. *Medical Image Analysis*, 13(4):543–563, 2009. doi: 10.1016/j.media.2009.05.004. (cited on Page 4 and 11)

[52] I. Higgins, L. Matthey, A. Pal, C. P. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner. Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *International Conference on Learning Representations*, 2016. (cited on Page 16)

[53] G. E. Hinton, S. Osindero, and Y.-W. Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18(7):1527–1554, 2006. doi: 10.1162/neco.2006.18.7.1527. (cited on Page 9)

[54] M. J. M. M. Hoeijmakers, I. Waechter-Stehle, J. Weese, and F. N. V. de Vosse. Combining Statistical Shape Modeling, CFD, and Meta-modeling to Approximate the Patient-specific Pressure-drop across the Aortic Valve in Real-time. *International Journal for Numerical Methods in Biomedical Engineering*, 36 (10), 2020. doi: 10.1002/cnm.3387. (cited on Page 13)

[55] D. P. J. Howard, A. Banerjee, J. F. Fairhead, J. Perkins, L. E. Silver, and P. M. Rothwell. Population-based Study of Incidence and Outcome of Acute Aortic Dissection and Premorbid Risk Factor Control. *Circulation*, 127(20): 2031–2037, 2013. doi: 10.1161/circulationaha.112.000483. (cited on Page 1)

[56] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv*, 2015. doi: 10.48550/ARXIV.1502.03167. (cited on Page 23)

[57] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image Translation with Conditional Adversarial Networks. *arXiv*, 2016. doi: 10.48550/ARXIV.1611.07004. (cited on Page 11)

[58] A. Juraszek, M. Czerny, and B. Rylski. Update in Aortic Dissection. *Trends in Cardiovascular Medicine*, 32(7):456–461, 2022. doi: 10.1016/j.tcm.2021.08.008. (cited on Page 7)

[59] M. Kazhdan and H. Hoppe. Screened Poisson Surface Reconstruction. *ACM Transactions on Graphics*, 32(3):1–13, 2013. doi: 10.1145/2487228.2487237. (cited on Page 30)

[60] J. B. Kim, M. Spotnitz, M. E. Lindsay, T. E. MacGillivray, E. M. Isselbacher, and T. M. Sundt. Risk of Aortic Dissection in the Moderately dilated Ascending Aorta. *Journal of the American College of Cardiology*, 68(11):1209–1219, 2016. doi: 10.1016/j.jacc.2016.06.025. (cited on Page 8)

[61] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv*, 2014. doi: 10.48550/ARXIV.1412.6980. (cited on Page 26 and 30)

[62] D. P. Kingma and M. Welling. Auto-encoding Variational Bayes. *arXiv*, 2013. doi: 10.48550/ARXIV.1312.6114. (cited on Page 9 and 14)

[63] D. P. Kingma and M. Welling. An Introduction to Variational Autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392, 2019. doi: 10.1561/2200000056. (cited on Page 14)

[64] A. Krizhevsky and G. Hinton. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, Toronto, Ontario, 2009. (cited on Page 9)

[65] P. Lamata, R. Casero, V. Carapella, S. A. Niederer, M. J. Bishop, J. E. Schneider, P. Kohl, and V. Grau. Images As Drivers of Progress in Cardiac Computational Modelling. *Progress in Biophysics and Molecular Biology*, 115 (2-3):198–212, 2014. doi: 10.1016/j.pbiomolbio.2014.08.005. (cited on Page 2)

[66] M. Landenhed, G. Engström, A. Gottsäter, M. P. Caulfield, B. Hedblad, C. Newton-Cheh, O. Melander, and J. G. Smith. Risk Profiles for Aortic Dissection and Ruptured or Surgically Treated Aneurysms: A Prospective Cohort Study. *Journal of the American Heart Association*, 4(1), 2015. doi: 10.1161/jaha.114.001513. (cited on Page 1)

[67] K. Lawonn, I. Viola, B. Preim, and T. Isenberg. A Survey of Surface-based Illustrative Rendering for Visualization. *Computer Graphics Forum*, 37(6): 205–234, 2018. doi: 10.1111/cgf.13322. (cited on Page 17)

[68] C.-L. Li, M. Zaheer, Y. Zhang, B. Poczos, and R. Salakhutdinov. Point Cloud GAN. *arXiv*, 2018. doi: 10.48550/ARXIV.1810.05795. (cited on Page 15)

[69] L. Liang, M. Liu, C. Martin, J. A. Elefteriades, and W. Sun. A Machine Learning Approach to Investigate the Relationship between Shape Features and Numerically Predicted Risk of Ascending Aortic Aneurysm. *Biomechanics and Modeling in Mechanobiology*, 16(5):1519–1533, 2017. doi: 10.1007/s10237-017-0903-9. (cited on Page 12)

[70] Q. Liu, J. D. Lee, and M. I. Jordan. A Kernelized Stein Discrepancy for Goodness-of-fit Tests and Model Evaluation. *arXiv*, 2016. doi: 10.48550/ARXIV.1602.03253. (cited on Page 16)

[71] J. V. Lombardi, G. C. Hughes, J. J. Appoo, J. E. Bavaria, A. W. Beck, R. P. Cambria, K. Charlton-Ouw, M. H. Eslami, K. M. Kim, B. G. Leshnower, T. Maldonado, T. B. Reece, and G. J. Wang. Society for Vascular Surgery (SVS) and Society of Thoracic Surgeons (STS) Reporting Standards for Type B Aortic Dissections. *Journal of Vascular Surgery*, 71(3):723–747, 2020. doi: 10.1016/j.jvs.2019.11.013. (cited on Page 7)

[72] L. Lu. Dying ReLU and Initialization: Theory and Numerical Examples. *Communications in Computational Physics*, 28(5):1671–1706, 2020. doi: 10.4208/cicp.oa-2020-0165. (cited on Page 25)

[73] S. Luo and W. Hu. Diffusion Probabilistic Models for 3D Point Cloud Generation. *arXiv*, 2021. doi: 10.48550/ARXIV.2103.01458. (cited on Page 16)

[74] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial Autoencoders. *arXiv*, 2015. doi: 10.48550/ARXIV.1511.05644. (cited on Page 15 and 30)

[75] A. Marzola, M. Servi, and Y. Volpe. A Reliable Procedure for the Construction of a Statistical Shape Model of the Cranial Vault. In *Lecture Notes in Mechanical Engineering*, pages 788–800. Springer International Publishing, 2019. doi: 10.1007/978-3-030-31154-4_67. (cited on Page 12)

[76] M. L. Menéndez, J. A. Pardo, L. Pardo, and M. C. Pardo. The Jensen-Shannon Divergence. *Journal of the Franklin Institute*, 334(2):307–318, 1997. doi: 10.1016/s0016-0032(96)00063-4. (cited on Page 15 and 16)

[77] G. Mistelbauer, C. Rössl, K. Bäumler, B. Preim, and D. Fleischmann. Implicit Modeling of Patient-specific Aortic Dissections with Elliptic Fourier Descriptors. *Computer Graphics Forum*, 40(3):423–434, 2021. doi: 10.1111/cgf.14318. (cited on Page 18, 20, and 27)

[78] S. Molnár and L. Tamás. Representation Learning For point Clouds With variational Autoencoders. In *Lecture Notes in Computer Science*, pages 727–737. Springer Nature Switzerland, 2023. doi: 10.1007/978-3-031-25075-0_49. (cited on Page 16)

[79] O. Mounjid and X. Guo. Convergence of Gans Training: A Game and Stochastic Control Methodology. *arXiv*, 2021. doi: 10.48550/ARXIV.2112.00222. (cited on Page 42)

[80] A. Myronenko. 3D MRI Brain Tumor Segmentation Using Autoencoder Regularization. *arXiv*, 2018. doi: 10.48550/ARXIV.1810.11654. (cited on Page 10)

[81] C. A. Nienaber, R. E. Clough, N. Sakalihasan, T. Suzuki, R. Gibbs, F. Mussa, M. P. Jenkins, M. M. Thompson, A. Evangelista, J. S. M. Yeh, N. Cheshire, U. Rosendahl, and J. Pepper. Aortic Dissection. *Nature Reviews Disease Primers*, 2(1), 2016. doi: 10.1038/nrdp.2016.53. (cited on Page 2 and 3)

[82] M. Oren and S. K. Nayar. Generalization of the Lambertian Model and Implications for Machine Vision. *International Journal of Computer Vision*, 14(3):227–251, 1995. doi: 10.1007/bf01679684. (cited on Page 17)

[83] K. Ostendorf, D. Mastrodicasa, K. Bäumler, M. Codari, V. Turner, M. J. Willemink, D. Fleischmann, B. Preim, and G. Mistelbauer. Shading Style Assessment for Vessel Wall and Lumen Visualization. In S. Oeltze-Jafra, N. N. Smit, B. Sommer, K. Nieselt, and T. Schultz, editors, *Eurographics Workshop on Visual Computing for Biology and Medicine*. The Eurographics Association, 2021. ISBN 978-3-03868-140-3. doi: 10.2312/vcbm.20211350. (cited on Page 17)

[84] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An Imperative Style, High-performance Deep Learning Library. *arXiv*, 2019. doi: 10.48550/ARXIV.1912.01703. (cited on Page 36)

[85] S. E. Petersen, P. M. Matthews, F. Bamberg, D. A. Bluemke, J. M. Francis, M. G. Friedrich, P. Leeson, E. Nagel, S. Plein, F. E. Rademakers, A. A. Young, S. Garratt, T. Peakman, J. Sellors, R. Collins, and S. Neubauer. Imaging in Population Science: Cardiovascular Magnetic Resonance in 100,000 Participants of UK Biobank - Rationale, Challenges and Approaches. *Journal of Cardiovascular Magnetic Resonance*, 15(46), 2013. doi: 10.1186/1532-429 x-15-46. (cited on Page 14)

[86] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation. *arXiv*, 2016. doi: 10.48550/ARX IV.1612.00593. (cited on Page 15 and 34)

[87] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *arXiv*, 2017. doi: 10.48550/ARXIV.1706.02413. (cited on Page 65)

[88] A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv*, 2015. doi: 10.48550/ARXIV.1511.06434. (cited on Page 22)

[89] J.-F. Rajotte, R. Bergen, D. L. Buckeridge, K. E. Emam, R. Ng, and E. Strome. Synthetic Data As an Enabler for Machine Learning Applications in Medicine. *iScience*, 25(11):105331, 2022. doi: 10.1016/j.isci.2022.105331. (cited on Page 1)

[90] M. Rezaei, H. Yang, and C. Meinel. Voxel-GAN: Adversarial Framework for Learning Imbalanced Brain Tumor Segmentation. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pages 321–333. Springer International Publishing, 2019. doi: 10.1007/978-3-030-11726-9_29. (cited on Page 10)

[91] D. J. Rezende and S. Mohamed. Variational Inference with Normalizing Flows. *arXiv*, 2015. doi: 10.48550/ARXIV.1505.05770. (cited on Page 15)

[92] C. Rezk-Salama and A. Kolb. Opacity Peeling for Direct Volume Rendering. *Computer Graphics Forum*, 25(3):597–606, 2006. doi: 10.1111/j.1467-8659.2006.00979.x. (cited on Page 17)

[93] P. Romero, S. Santos, R. Sebastian, F. Martinez-Gil, D. Serra, P. Calvillo, A. Rodrıguez, R. M. Puig, L. Martı-Bonmatı, A. Alberich-Bayarri, et al. Reconstruction of the Aorta Geometry Using Canal Surfaces. In *Proceedings of the International Conference on Computational and Mathematical Biomedical Engineering*, 2019. (cited on Page 13)

[94] P. Romero, M. Lozano, F. Martínez-Gil, D. Serra, R. Sebastián, P. Lamata, and I. García-Fernández. Clinically-driven Virtual Patient Cohorts Generation: An Application to Aorta. *Frontiers in Physiology*, 12:713118, 2021. doi: 10.3389/fphys.2021.713118. (cited on Page 3, 13, 19, 27, and 64)

[95] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv*, 2015. doi: 10.48550/ARXIV.1505.04597. (cited on Page 10 and 11)

[96] Y. Rubner. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000. doi: 10.1023/a:1026543900054. (cited on Page 15)

[97] P. A. Rudenick, M. Bordone, B. H. Bijnens, E. Soudah, E. Oñate, D. Garcia-Dorado, and A. Evangelista. A Multi-method Approach Towards Understanding the Pathophysiology of Aortic Dissections – the Complementary Role of In-silico, In-vitro and In-vivo Information. In *Statistical Atlases and Computational Models of the Heart*, pages 114–123. Springer Berlin Heidelberg, 2010. doi: 10.1007/978-3-642-15835-3_12. (cited on Page 7)

[98] F. Rusak, R. S. Cruz, P. Bourgeat, C. Fookes, J. Fripp, A. Bradley, and O. Salvado. 3D Brain MRI GAN-based Synthesis Conditioned on Partial Volume Maps. In *Simulation and Synthesis in Medical Imaging*, pages 11–20. Springer International Publishing, 2020. doi: 10.1007/978-3-030-59520-3_2. (cited on Page 10)

[99] M. B. Salem and L. Tomaso. Automatic Selection for General Surrogate Models. *Structural and Multidisciplinary Optimization*, 58(2):719–734, 2018. doi: 10.1007/s00158-018-1925-3. (cited on Page 13)

[100] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved Techniques for Training GANs. *arXiv*, 2016. doi: 10.48550/ARXIV.1606.03498. (cited on Page 42)

[101] I. Sen, Y. M. Erben, C. Franco-Mesa, and R. R. DeMartino. Epidemiology of Aortic Dissection. *Seminars in Vascular Surgery*, 34(1):10–17, Mar. 2021. doi: 10.1053/j.semvascsurg.2021.02.003. (cited on Page 8)

[102] A. Sharma, O. Grau, and M. Fritz. VConv-DAE: Deep Volumetric Shape Learning without Object Labels. *arXiv*, 2016. doi: 10.48550/ARXIV.1604.03755. (cited on Page 9)

[103] D. W. Shu, S. W. Park, and J. Kwon. 3D Point Cloud Generative Adversarial Network Based on Tree Structured Graph Convolutions. *arXiv*, 2019. doi: 10.48550/ARXIV.1905.06292. (cited on Page 16)

[104] H.-H. Sievers, B. Rylski, M. Czerny, A. L. M. Baier, M. Kreibich, M. Siepe, and F. Beyersdorf. Aortic Dissection Reconsidered: Type, Entry Site, Malperfusion Classification Adding Clarity and Enabling Outcome Prediction. *Interactive CardioVascular and Thoracic Surgery*, 30(3):451–457, 2019. doi: 10.1093/icvts/ivz281. (cited on Page 7)

[105] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for Simplicity: The All Convolutional Net. *arXiv*, 2014. doi: 10.48550/ARXIV.1412.6806. (cited on Page 10)

[106] M. Straka, M. Cervenansky, A. L. Cruz, A. Kochl, M. Sramek, E. Groller, and D. Fleischmann. The VesselGlyph: Focus and Context Visualization in CT-Angiography. In *IEEE Visualization*, pages 385–392, 2004. doi: 10.1109/visual.2004.104. (cited on Page 17)

[107] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic Attribution for Deep Networks. *arXiv*, 2017. doi: 10.48550/ARXIV.1703.01365. (cited on Page 10)

[108] J. Susskind, A. Anderson, and G. E. Hinton. The Toronto Face Dataset. Technical report, Technical Report UTML TR 2010-001, U. Toronto, 2010. (cited on Page 9)

[109] B. Thamsen, P. Yevtushenko, L. Gundelwein, H. Lamecker, T. Kühne, and L. Goubergrits. Unsupervised Learning and Statistical Shape Modeling of the Morphometry and Hemodynamics of Coarctation of the Aorta. In *Medical*

Image Computing and Computer Assisted Intervention, pages 776–785. 2020. doi: 10.1007/978-3-030-59719-1_75. (cited on Page 13)

[110] B. Thamsen, P. Yevtushenko, L. Gundelwein, A. A. A. Setio, H. Lamecker, M. Kelm, M. Schafstedde, T. Heimann, T. Kuehne, and L. Goubergrits. Synthetic Database of Aortic Morphometry and Hemodynamics: Overcoming Medical Imaging Data Availability. *IEEE Transactions on Medical Imaging*, 40(5):1438–1449, 2021. doi: 10.1109/tmi.2021.3057496. (cited on Page 3, 14, 19, and 27)

[111] H. Thanh-Tung and T. Tran. On Catastrophic Forgetting and Mode Collapse in Generative Adversarial Networks. *arXiv*, 2018. doi: 10.48550/ARXIV.1807.04015. (cited on Page 42)

[112] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning Spatiotemporal Features with 3D Convolutional Networks. *arXiv*, 2014. doi: 10.48550/ARXIV.1412.0767. (cited on Page 10)

[113] M. A. Uy and G. H. Lee. PointNetVLAD: Deep Point Cloud Based Retrieval for Large-scale Place Recognition. *arXiv*, 2018. doi: 10.48550/ARXIV.1804.03492. (cited on Page 15)

[114] D. Valsesia, G. Fracastoro, and E. Magli. Learning Localized Generative Models for 3D Point Clouds Via Graph Convolution. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJeXSo09FQ. (cited on Page 16)

[115] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, A. Vijaykumar, A. P. Bardelli, A. Rothberg, A. Hilboll, A. Kloeckner, A. Scopatz, A. Lee, A. Rokem, C. N. Woods, C. Fulton, C. Masson, C. Häggström, C. Fitzgerald, D. A. Nicholson, D. R. Hagen, D. V. Pasechnik, E. Olivetti, E. Martin, E. Wieser, F. Silva, F. Lenders, F. Wilhelm, G. Young, G. A. Price, G.-L. Ingold, G. E. Allen, G. R. Lee, H. Audren, I. Probst, J. P. Dietrich, J. Silterra, J. T. Webber, J. Slavič, J. Nothman, J. Buchner, J. Kulick, J. L. Schönberger, J. V. de Miranda Cardoso, J. Reimer, J. Harrington, J. L. C. Rodríguez, J. Nunez-Iglesias, J. Kuczynski, K. Tritz, M. Thoma, M. Newville, M. Kümmerer, M. Bolingbroke, M. Tartre, M. Pak, N. J. Smith, N. Nowaczyk, N. Shebanov, O. Pavlyk, P. A. Brodtkorb, P. Lee, R. T. McGibbon, R. Feldbauer, S. Lewis, S. Tygier, S. Sievert, S. Vigna, S. Peterson, S. More, T. Pudlik, T. Oshima, T. J. Pingel, T. P. Robitaille, T. Spura, T. R. Jones, T. Cera, T. Leslie, T. Zito, T. Krauss, U. Upadhyay, Y. O. Halchenko, and Y. Vázquez-Baeza. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17(3):261–272, 2020. doi: 10.1038/s41592-019-0686-2. (cited on Page 39)

[116] W. Wang, B. Jüttler, D. Zheng, and Y. Liu. Computation of Rotation Minimizing Frames. *ACM Transactions on Graphics*, 27(1):1–18, 2008. doi: 10.1145/1330511.1330513. (cited on Page 20)

[117] H. Wiputra, S. Matsumoto, J. E. Wagenseil, A. C. Braverman, R. K. Voeller, and V. H. Barocas. Statistical Shape Representation of the Thoracic Aorta: Accounting for Major Branches of the Aortic Arch. *Computer Methods in Biomechanics and Biomedical Engineering*, pages 1–15, 2022. doi: 10.1080/10255842.2022.2128672. (cited on Page 27 and 64)

[118] J. M. Wolterink, T. Leiner, and I. Isgum. Blood Vessel Geometry Synthesis Using Generative Adversarial Networks. *arXiv*, 2018. doi: 10.48550/ARXIV.1804.04381. (cited on Page 17)

[119] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum. Learning a Probabilistic Latent Space of Object Shapes Via 3D Generative-adversarial Modeling. *arXiv*, 2016. doi: 10.48550/ARXIV.1610.07584. (cited on Page 9, 19, 22, 23, and 26)

[120] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D Shapenets: A Deep Representation for Volumetric Shapes. *arXiv*, 2014. doi: 10.48550/ARXIV.1406.5670. (cited on Page 9 and 30)

[121] M. Wundram, V. Falk, J.-J. Eulert-Grehn, H. Herbst, J. Thurau, B. A. Leidel, E. Göncz, W. Bauer, H. Habazettl, and S. D. Kurz. Incidence of Acute Type A Aortic Dissection in Emergency Departments. *Scientific Reports*, 10(7434), 2020. doi: 10.1038/s41598-020-64299-4. (cited on Page 1)

[122] B. Xu, N. Wang, T. Chen, and M. Li. Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv*, 2015. doi: 10.48550/ARXIV.1505.00853. (cited on Page 25)

[123] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan. Pointflow: 3D Point Cloud Generation with Continuous Normalizing Flows. *arXiv*, 2019. doi: 10.48550/ARXIV.1906.12320. (cited on Page 6, 15, and 16)

[124] Y. Yang, C. Feng, Y. Shen, and D. Tian. Foldingnet: Point Cloud Auto-Encoder Via Deep Grid Deformation. *arXiv*, 2017. doi: 10.48550/ARXIV.1712.07262. (cited on Page 65)

[125] M. Zamorski, M. Zięba, P. Klukowski, R. Nowak, K. Kurach, W. Stokowiec, and T. Trzciński. Adversarial Autoencoders for Compact Representations of 3D Point Clouds. *arXiv*, 2018. doi: 10.48550/ARXIV.1811.07605. (cited on Page 6, 15, 17, 19, 30, 34, and 54)

[126] J. Zhang, J. Liu, S. Wei, D. Chen, J. Xiong, and F. Gao. Semi-Supervised Aortic Dissections Segmentation: A Time-dependent Weighted Feedback Fusion Framework. *Computerized Medical Imaging and Graphics*, 106:102219, 2023. doi: 10.1016/j.compmedimag.2023.102219. (cited on Page 2)

[127] H. Zhao, L. Jiang, J. Jia, P. Torr, and V. Koltun. Point Transformer. *arXiv*, 2020. doi: 10.48550/ARXIV.2012.09164. (cited on Page 16)

[128] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. *arXiv*, 2015. doi: 10.48550/ARXIV.1512.04150. (cited on Page 10)

[129] H. Zhou, J. K. Min, and G. Xiong. Implicit Tubular Surface Generation Guided by Centerline. *arXiv*, 2016. doi: 10.48550/ARXIV.1606.03014. (cited on Page 64)

[130] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A Modern Library for 3D Data Processing. *arXiv*, 2018. doi: 10.48550/ARXIV.1801.09847. (cited on Page 39)

[131] Y. Zhu, B. Lingala, M. Baiocchi, J. J. Tao, V. T. Arana, J. W. Khoo, K. M. Williams, A. A.-R. Traboulsi, H. C. Hammond, A. M. Lee, W. Hiesinger, J. Boyd, P. E. Oyer, E. B. Stinson, B. A. Reitz, R. S. Mitchell, D. C. Miller, M. P. Fischbein, and Y. J. Woo. Type A Aortic Dissection—Experience Over 5 Decades. *Journal of the American College of Cardiology*, 76(14):1703–1713, 2020. doi: 10.1016/j.jacc.2020.07.061. (cited on Page 8)